A Priori Approaches to Vehicle Routing in Theory and Practice

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Vorgelegt von

Luise Puhlmann

aus

Darmstadt

Bonn, Februar 2026

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

Gutachter/Betreuer:	Professor Dr. Jens Vygen
Gutachter:	Professor Dr. Stephan Held
Tag der Promotion:	13. April 2026
Erscheinungsjahr:	2026

Acknowledgments

First and foremost, I would like to thank my advisor Prof. Dr. Jens Vygen for his support, his encouragement, and the excellent supervision. Working with him is always a real pleasure and I learned a lot from discussions with him.

I also want to thank my other co-authors (besides Prof. Dr. Jens Vygen) – working with them was equally enjoyable: Dr. Jannis Blauth, Manuel Christalla, Dr. Meike Neuwohner, Niklas Schlomberg, and Prof. Dr. Vera Traub.

Special thanks go to Dr. Dirk Müller who is always available for help with the BonnTour code and coding in general.

Additionally, I want to thank my former colleague Dr. Jannis Blauth and my current colleague Daniel Ebert for the nice working atmosphere in our shared office. Thanks also go to the student assistants who complete our BonnTour team with which it was always fun to work.

Moreover, I want to thank Niklas Schlomberg, Dr. Dirk Müller, and Prof. Dr. Jens Vygen for proofreading earlier version of this thesis and giving helpful feedback.

I also wish to thank our cooperation partner Greenplan who put a lot of work into ensuring that BonnTour is successfully used in practice. In particular, I want to mention Dr. Clemens Beckmann, Florian Merget, and the mathematical team of Greenplan.

Last but not least, thanks go to all my other colleagues at the institute. Not least because of them, I have always enjoyed going to the institute over the past years.

Contents

Chapter 1

Introduction

This thesis is devoted to combinatorial optimization problems that arise in logistics. With an increasing demand for delivering goods all over the world, finding efficient solutions for these problems is a major economic interest. In particular, we deal with those problems that need to be solved before having complete information about the exact setting: It can be helpful to pre-compute data in order to obtain high-quality solutions faster as soon as the input is complete.

We approach these kinds of problems from two directions. In the first part, we look at theoretical problems that arise in the context of logistical tasks. They are generalizations of the well-known *Traveling Salesperson Problem*, short TSP. The TSP asks for a cheapest *tour* visiting all addresses from a given input set V and returning to the start point. To this end, a cost function $c\colon V^2 \to \mathbb{R}_{\geq 0}$ specifies how expensive it is to travel from one address to another. We now consider a generalization called the *A Priori TSP*. There, we are additionally given *activation probabilities* $p\colon V \to [0,1]$. The task is to compute a tour visiting all customers in V, but the objective function is to minimize the expected cost of the tour when cut short to the *active customers*. In our model, each customer $v \in V$ is active independently with probability $p(v)$. The A Priori TSP contains the TSP (when choosing $p \equiv 1$) which is NP-hard. Therefore the A Priori TSP is NP-hard, too, and there is no algorithm with polynomial running time that computes an optimum solution unless P = NP. We hence consider *approximation algorithms*. An approximation algorithm with *approximation ratio* (or *approximation guarantee*) $\alpha > 1$ computes a solution with cost at most $\alpha \cdot \mathrm{OPT}$ in polynomial time. Here, OPT denotes the cost of an optimum solution. The goal is to find approximation algorithms with the smallest possible approximation ratio. If the cost function c can be chosen arbitrarily, there are no approximation algorithms for the TSP. Usually, one therefore considers the TSP with *triangle inequality*, i.e., $c(u,w) \leq c(u,v) + c(v,w)$ for $u, v, w \in V$. We distinguish the *Symmetric* TSP and the *Asymmetric* TSP: In the symmetric case, we demand $c(v,w) = c(w,v)$ for all $v, w \in V$; the asymmetric case comes without this assumption. This notion is also transferred to the A Priori TSP.

In Chapter 3, which is based on joint work with Jannis Blauth, Meike Neuwohner, and Jens Vygen ([BNPV25]), we improve the best known guarantees for the Symmetric A Priori TSP: We investigate the sampling algorithms introduced in previous work ([GGLS08] and especially [ST08]) and examine the question what guarantees can be obtained by a sampling approach, even when taking an optimal sample, and the question which probability distribution yields the best results when sampling randomly. Thereby, we obtain a randomized approximation algorithm for symmetric A Priori TSP with approximation ratio less than 3.1, improving upon the previously best known ratio of 3.5. Furthermore, we obtain a deterministic polynomial-time algorithm with approximation guarantee less than 5.9, improving upon the previously best known ratio

of 6.5. We also give an example of an instance family providing lower bounds for our analysis, complementing a lower bound already found in [Puh21].

Chapter 4, which is based on joint work with Manuel Christalla and Vera Traub ([CPT26]), is devoted to the Asymmetric A Priori TSP. There, we give the first non-trivial approximation algorithm, obtaining an approximation guarantee of $\mathcal{O}(\sqrt{|V|})$. The approach for this approximation algorithm uses ideas from the algorithms used in the symmetric version. Moreover, we provide an algorithm with quasi-polynomial running time that computes a solution with a poly-logarithmic approximation guarantee. In order to achieve this, we provide a series of polynomial-time reductions. We first introduce a novel generalization of the Asymmetric TSP that we call Hop-ATSP and to which we then reduce Asymmetric A Priori TSP. Then we obtain structured instances by using directed low-diameter decompositions ([BNW22; BFHL25; Li25]). Next, we reduce solving Hop-ATSP on these structured instances to a covering problem for which we then give an approximation algorithm that runs in quasi-polynomial time. A more detailed outline for Chapter 3 and Chapter 4 is given in Section 2.1.

The second part of this thesis is about solving tour planning problems in practice. At the Research Institute for Discrete Mathematics of the University of Bonn, we have developed a software package called "BonnTour" in cooperation with our industry partner Greenplan (see `www.greenplan.de`). BonnTour is capable of finding good solutions for real-world tour planning problems with numerous constraints and is successfully used in practice by partners relying on Greenplan. An overview over the main techniques used in BonnTour can be found in [BHMSTTV24] and in [Bla24].

In this thesis, we take a closer look at some pre-processing steps performed in BonnTour, namely Contraction Hierarchies (Chapter 6) as well as Address Mapping and Shared Parking (Chapter 7). Contraction Hierarchies are computed from the road graph before we even know the addresses that need to be visited. As soon as we have the information about where these addresses are, the Contraction Hierarchies enable us to quickly answer the question how long it takes to get from one address to another. The concept of Contraction Hierarchies was introduced by [GSSD08]; a first version within BonnTour was implemented by Rathke ([Rat21]). In this thesis, we present improvements made to the implementation of Contraction Hierarchies in BonnTour, enabling us to use them in practice on a daily basis: In particular, we take a closer look at approximating the arrival time functions associated with each edge of the Contraction Hierarchy, thus making their construction faster and less memory-intensive. We also show how to add new vertices into an existing Contraction Hierarchy without destroying its relevant structure:

Contraction Hierarchies work with a total order on the vertices (called *vertex importance*) and add shortcut edges to guarantee that in order to find fastest paths, it suffices to consider so-called *up-down paths*, that is paths which first only go upwards in the total order (i.e., vertices becoming more important), and then only go downwards (i.e., vertices becoming less important). For solving specific instances, vertices that represent the addresses need to be inserted into pre-computed Contraction Hierarchies. Using a careful definition of *vertex insertion* and *edge subdivision*, we show that it is possible not to destroy the up-down paths between previously existing vertices while also guaranteeing that the fastest paths from or to the new vertex can be found by only considering up-down paths. We also extend these techniques to the insertion of multiple new vertices.

In Address Mapping and Shared Parking, we find good parking positions before having computed the tours. In particular, we do not yet have the information from which direction we arrive at an address and where we go next. However, in order to compute good tours later, it is crucial to already have a good estimate of where a suitable place to park near the address is. We conclude the thesis with the presentation of experimental results in Chapter 8, giving empirical proof that the techniques discussed in the previous chapters lead to good outcomes.

Part I

Approximation Algorithms for A Priori Versions of the TSP

Chapter 2

Introduction to A Priori TSP

This chapter is based on joint work with Jannis Blauth, Meike Neuwohner, and Jens Vygen ([BNPV25]) and with Manuel Christalla and Vera Traub ([CPT26]).

2.1 Outline and contributions

The Traveling Salesperson Problem (TSP) is among the most famous and well-studied problems in combinatorial optimization. An instance of the Asymmetric TSP consists of a finite set V of *customers* or *vertices* and distances/costs $c\colon V \times V \to \mathbb{R}_{\geq 0}$ satisfying the triangle inequality, i.e., $c(x,z) \leq c(x,y) + c(y,z)$ for all $x, y, z \in V$. In the Symmetric TSP, we additionally demand $c(x,y) = c(y,x)$ for all $x, y \in V$. The task is to compute a shortest tour on V, that is a cycle with vertex set V minimizing the sum of the edge costs.

While it is easy to achieve constant-factor approximations for the Symmetric TSP [RSL77] (improved in [Chr76; Ser78; KKO23]), for a long time only logarithmic approximation factors were known for the Asymmetric TSP [FGM82; Blä08; KLSS05; FS07; AGMGS17] (see also [TV25]). The first constant-factor approximations for the more general Asymmetric TSP have been discovered relatively recently [STV20] (improved in [TV22; TV25]).

In this work, we focus on stochastic versions of both the Symmetric and the Asymmetric TSP, where each vertex $v \in V$ has an activation probability $p(v)$. We write $A \sim p$ to indicate that we sample a random set $A \subseteq V$ of active vertices from p, i.e., every vertex belongs to A with probability $p(v)$ and the sampling of different vertices is independent. For a tour T on V, the tour $T[A]$ on A results from T by short-cutting to the set A of active vertices, i.e., skipping all visits of vertices outside of A. See Figure 2.1.

Many algorithms for stochastic discrete optimization problems sample a sub-instance, solve the resulting deterministic problem (often by some approximation algorithm), and extend this solution to the original instance [EGRS10; GGLS08; GKPR07; GKR03; GPRS04; ST08]. The (Symmetric) A Priori TSP is a nice and well-studied example for this approach. In Chapter 3 (which is based on [BNPV25]), we address the following questions: What guarantee can we obtain by such an approach, even if we take an optimal sample? If we sample randomly, according to which distribution? What guarantee can we obtain by a deterministic polynomial-time algorithm?

The task in the A Priori TSP (with independent activation) is, given a (semi-) metric space (V, c) and activation probabilities $p\colon V \to [0,1]$, to design a TSP tour T (visiting all of V) *before* knowing which customers will be active. *After* we know which customers are active we can cut

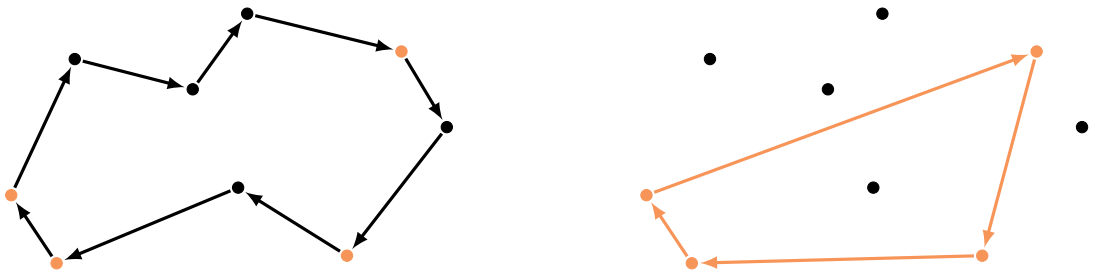


Figure 2.1: Left: a tour T through a set of vertices in an asymmetric instance; right: the tour $T[A]$ resulting from cutting T short to the set A of orange vertices.

the tour T short by skipping the inactive customers. The goal is to minimize the expected cost of the resulting tour (visiting the active customers).

Note that computing an optimum a priori tour is APX-hard as the metric TSP is APX-hard [PY93], which is the special case where all activation probabilities are 1. We therefore study approximation algorithms. A *ρ-approximation algorithm* for the A Priori TSP is a polynomial-time algorithm that computes a tour of expected cost at most $\rho \cdot \mathrm{OPT}$ for any given instance, where OPT denotes the expected cost of an optimum a priori tour.

Shmoys and Talwar [ST08] devised a randomized 4-approximation algorithm and a deterministic 8-approximation algorithm. A randomized constant-factor approximation algorithm was discovered independently by Garg, Gupta, Leonardi and Sankowski [GGLS08]. The randomized Shmoys–Talwar algorithm easily improves to a 3.5-approximation by using the Christofides–Serdyukov algorithm instead of the double tree algorithm as a subroutine for TSP (as noted by [ES18]), and slightly better using the new Karlin–Klein–Oveis Gharan algorithm [KKO24]. The deterministic algorithm was improved to a 6.5-approximation by van Zuylen [Zuy11]; a slight improvement of this guarantee follows from the recent deterministic version of the Karlin–Klein–Oveis Gharan algorithm [KKO23].

All known approximation algorithms for the A Priori TSP are of the following type. Select a nonempty subset S of customers and find a TSP tour for S (the *master tour*). Connect each other customer $v \in V \setminus S$ with a pair of parallel edges to a nearest point $\mu(v)$ in the master tour. We call this a *master route solution*. Once we know the set of active customers, we pay for the entire master tour (pretending to visit also its inactive customers!) and pay $2c(\mu(v), v)$ for each active customer v outside S to cover the round trip visiting v from $\mu(v)$. See Fig. 2.2 for an example. Of course, we could cut the resulting tour shorter (we visit some inactive customers, and we visit some customers several times), but we will not account for this possible gain (unless fewer than two customers are active). Instead we define the *master route cost* for a master route solution on S as

$$\mathrm{MR}(S) \;\coloneqq\; \mathbb{E}_{A\sim p}\left[\mathbb{1}_{|A|\geq 2}\cdot\left(\mathrm{TSP}(S,c)+2\cdot\sum_{v\in A}c(v,S)\right)\right]. \tag{2.1}$$

Here $c(v,S)=\min\{c(v,s):s\in S\}$ denotes the distance between v and a nearest customer in S, and $\mathrm{TSP}(S,c)$ denotes the length of an optimum TSP tour for S.

For instances that have a *depot*, i.e. $d \in V$ with $p(d) = 1$, the Shmoys–Talwar algorithm [ST08] includes a customer v into S with probability $p(v)$: the sampling probability is exactly the activation probability. Although this is natural and allows for a simple analysis that we summarize in Section 3.1.1 (assuming a depot), we show that this is not optimal. Decreasing the

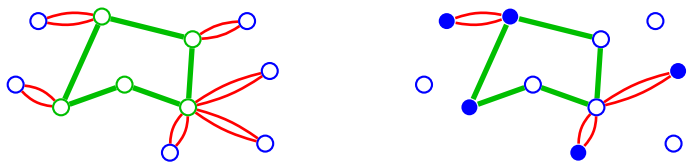


Figure 2.2: Left: A master route solution with a master tour (green, thick) and connections of the other customers to that master tour (red, curved). Right: After knowing which customers are active (filled), the master route solution reduces to a tour visiting all of the master tour and the other active customers.

probability of including a customer into the master tour improves the approximation guarantee. To be more precise, in Section 3.2 and Section 3.3, we analyze the following *sampling algorithm* for A Priori TSP instances with depot. Let $f\colon (0,1] \to [0,1]$ with $f(1) = 1$.

(i) Sample a subset $S \subseteq V$ by including every customer v independently with probability $f(p(v))$.

(ii) Call an α-approximation algorithm for (metric) TSP in order to compute a TSP tour for S, which serves as master tour.

(iii) Connect every customer outside S to the nearest customer in S by a pair of parallel edges.

For a given instance this algorithm has expected approximation ratio at most

$$\frac{1}{\mathrm{OPT}} \cdot \mathbb{E}_{S \sim f \circ p}\left[\alpha \cdot \mathrm{TSP}(S,c) + 2 \cdot \sum_{v \in V} p(v) \cdot c(v,S)\right] \tag{2.2}$$

(where $\frac{0}{0} \coloneqq 1$). Shmoys and Talwar [ST08] used the identity function $f(p) = p$ and also showed how to extend their algorithm to the general case, i.e. instances that do not necessarily contain a depot. We give a more general reduction, proving that it is sufficient to only consider instances with depot (also in the Asymmetric A Priori TSP), in Section 2.3.1.

In Section 3.2, we devise a linear program depending on f whose optimal solution is an upper bound on the approximation ratio of the sampling algorithm when using the function f. Using $f\colon p \mapsto 1 - (1-p)^\sigma$ with $\sigma = 0.663$, we obtain a sampling algorithm with approximation guarantee less than 3.1. In Section 3.3, we provide a lower bound for the approximation ratio of the sampling algorithm of 3.049, showing that our choice of f is almost best possible.

Moreover, we introduce the notion of the *master route ratio* which measures the gap between the cost of an optimum master route solution and an overall optimum solution to the A Priori TSP: It is defined to be the supremum of

$$\frac{\min\{\mathrm{MR}(S) : \emptyset \neq S \subseteq V\}}{\mathrm{OPT}} \tag{2.3}$$

taken over all A Priori TSP instances (where $\frac{0}{0} \coloneqq 1$). By the analysis of the Shmoys–Talwar algorithm ([ST08]), it is at most 3. The analysis of the deterministic algorithm devised by van Zuylen ([Zuy11]) depends on the value of the master route ratio. In Section 3.4, by an analysis similar to our approach in Section 3.2, we give a better upper bound of 2.6 for the master route ratio. This immediately improves the bound on the approximation ratio of van Zuylen's algorithm to less than 5.9. Our upper bound on the master route ratio comes close to the lower bound of 2.541 proven in [Puh21] and [BNPV25].

In contrast to the symmetric special case, to the best of our knowledge, no nontrivial approximation algorithms for the Asymmetric A Priori TSP are known. In Asymmetric A Priori TSP (with independent activation) we are given a set V of customers, an activation probability $p(v) \in [0,1]$ for each customer $v \in V$, and a distance/cost function $c\colon V \times V \to \mathbb{R}_{\geq 0}$ satisfying the triangle inequality. Again, the task is to compute a tour T on V minimizing the expected cost

$$\mathbb{E}_{A\sim p}\Big[c\big(T[A]\big)\Big] \tag{2.4}$$

after short-cutting T to the random set A of active vertices.

The approximation factor compares the objective value (2.4) of the computed tour to the optimal objective value, i.e., to the expected cost

$$\mathrm{OPT}(V,c,p) \coloneqq \min_{T \text{ tour on } V} \mathbb{E}_{A\sim p}\Big[c\big(T[A]\big)\Big]$$

of an optimal a priori tour after short-cutting. Many of the above-mentioned algorithms for Symmetric A Priori TSP have been analyzed with respect to the expected length of the best a posteriori tour, i.e., with respect to $\mathbb{E}_{A\sim p}\big[\mathrm{TSP}(A,c)\big]$, where $\mathrm{TSP}(A,c)$ denotes the length of a shortest tour on the vertex set A. This is the expected length of the in hindsight optimal tour, i.e., the best tour one could achieve when knowing the set A upfront instead of knowing merely the activation probabilities. It provides a lower bound on the optimal value $\mathrm{OPT}(V,c,p)$. With such an analysis it is impossible to achieve approximation factors below the *adaptivity gap*

$$\sup\left\{\frac{\mathrm{OPT}(V,c,p)}{\mathbb{E}_{A\sim p}\big[\mathrm{TSP(A,c)}\big]} : (V,c,p) \text{ instance of A Priori TSP}\right\}. \tag{2.5}$$

In Chapter 3, we actually analyze our sampling algorithm for (Symmetric) A Priori TSP with respect to the cost $\mathrm{OPT}(V,c,p)$ of the optimal a priori tour and not with respect to the optimal a posteriori tour. However, we do not achieve an approximation factor below the adaptivity gap of the Symmetric A Priori TSP, because [ST08] implies that the adaptivity gap of Symmetric A Priori TSP is at most 3.

In Chapter 4 (which is based on [CPT26]), we give an $\mathcal{O}(\sqrt{n})$-approximation algorithm for Asymmetric A Priori TSP that is inspired by the master route solution approach. Our analysis however only compares against the expected length of the best a posteriori tour. Furthermore, we prove a polynomial lower bound on the adaptivity gap for Asymmetric A Priori TSP. Moreover, we show that a poly-logarithmic approximation ratio, and hence an approximation ratio below the adaptivity gap, can be achieved by a deterministic algorithm with quasi-polynomial running time.

To obtain such an algorithm, we provide a series of polynomial-time reductions. First we reduce to a novel generalization of the Asymmetric Traveling Salesperson Problem, called Hop-ATSP. Next, we use directed low-diameter decompositions to obtain structured instances, for which we then provide a reduction to a covering problem. Eventually, we obtain a polynomial-time reduction of Asymmetric A Priori TSP to a problem of finding a path in an acyclic digraph minimizing a particular objective function, for which we give an $\mathcal{O}(\log n)$-approximation algorithm in quasi-polynomial time.

For our approaches to both the Symmetric and the Asymmetric A Priori TSP, we need a reduction to instances that contain a depot (i.e. $d \in V$ with $p(d) = 1$), and a reduction to instances where all customers (except for maybe the depot) have the same (tiny) activation probability. We will perform this pre-processing of A Priori TSP instances in Section 2.3.

2.2 Further related work

The TSP has also been studied under the aspect of robust optimization, where the set of customers that need to be visited is known in advance, but the edge lengths are chosen probabilistically or even adversarially [GMP23; THP14]. The a priori optimization problem where the set of customers is chosen adversarially is known as Universal TSP [JLNRS05; GGLS08; SS08; GKSS10; BCK11; GHR06; HKL06]. The probabilistic version that we consider was introduced by Jaillet [Jai85] and Bertsimas [Ber88]. Since then, various aspects of the problem have been investigated, including the asymptotic behavior of random instances [Ber88; BJO90; BFB03; Jai85; Jai88], online variants [GGLS08], or exact algorithms [AKB17]. Approximation algorithms and lower bounds have also been studied for general probability distributions, like the black-box model where nothing is known about the probability distribution [SS08; GKSS10], or the scenario model [EIJS18]. In the scenario model, we are given a list of subsets of V (called *scenarios*), and a probability for each scenario, thus differing from the independent activation model considered in this thesis. [EIJS18] examine the case with only a constant number of scenarios, the case where each scenario contains all but a constant number of vertices, and the case where the scenarios are nested subsets of V, and give constant-factor approximations for these restricted cases of the A Priori TSP in the scenario model.

Besides TSP, many other problems have been studied in an a priori setting, including for example Vehicle Routing problems, Set Cover, Steiner Tree, Traveling Repairman, and Network Design [ES18; EGRS10; AGLW17; GGLS08; GKPR07; GKR03; GPRS04; FMSSSSZ14; EIJS18; FS20; NGN20; GGLMSS13; GKL24; HKLR05].

Previous approaches to design a linear program that yields the approximation ratio of a certain algorithm for some optimization problem (e.g., [GK98; JMMSV03]) typically required an infinite family of linear programs and could not obtain a bound for general instances by just solving a single linear program.

2.3 Pre-processing A Priori TSP instances

For both the Symmetric and the Asymmetric A Priori TSP, we will need statements that allow us to restrict our attention to instances that have a *depot*, i.e., a vertex $d \in V$ with $p(d) = 1$, or to instances where all activation probabilities are tiny and the same for all vertices (except for maybe the depot). The goal of this section is therefore to provide suitable lemmata and theorems on which we can rely in the further analyses. In particular, we will show in Section 2.3.1 that instances where the total activation probability is small can be solved within a sufficiently small factor of the optimum. In instances where the total activation probability is large, we will see that it does not hurt much to declare one vertex to be the depot. We will therefore obtain the following statements:

Theorem 2.1. *Let $\varepsilon > 0$ and $\rho \geq 3$ be constants. If there exists a (randomized) polynomial-time ρ-approximation algorithm for instances (V, c, p) of the Symmetric A Priori TSP that have a depot, then there is a (randomized) polynomial-time $(\rho + \varepsilon)$-approximation algorithm for general instances of the Symmetric A Priori TSP.*

Theorem 2.2. *Let $\alpha \colon \mathbb{N} \to \mathbb{R}_{\geq 1}$. Suppose there is a polynomial-time algorithm that, given an instance (V', c', p') of Asymmetric A Priori TSP with Depot, returns an a priori tour T' on V' that satisfies*

$$\mathbb{E}_{A' \sim p'} \left[c'(T'[A']) \right] \leq \alpha(n') \cdot \mathbb{E}_{A' \sim p'} \left[\mathrm{TSP}(A', c') \right], \tag{2.6}$$

then there is a polynomial-time algorithm that, given a general instance (V, c, p) of Asymmetric A Priori TSP, returns a tour T on V that satisfies

$$\mathbb{E}_{A\sim p}\left[c(T[A])\right] \leq 5 \cdot \alpha(n) \cdot \mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A, c)\right], \tag{2.7}$$

where $n' \coloneqq |V'|$ and $n \coloneqq |V|$.

In Section 2.3.2, we show how to transform the instances such that all vertices have the same tiny activation probability. The idea is to replace each vertex with a set of co-located copies that all have a smaller activation probability such that the probability that any of these copies is active resembles the activation probability of the original vertex. For the Asymmetric A Priori TSP, we will show:

Proposition 2.3. *Given an instance (V, c, p) of Asymmetric A Priori TSP where $p_{\min} := \min_{v\in V} p(v) \geq \varepsilon > 0$, we can transform it in polynomial time (in $|V|$ and $1/\varepsilon$) into an instance (V', c', p') of Asymmetric A Priori TSP such that*

(i) $p'(v) = \varepsilon/2$ *for any* $v \in V'$,

(ii) $\mathrm{OPT}(V', c', p') \leq \mathrm{OPT}(V, c, p)$,

(iii) any tour T' in (V', c', p') can be converted into a tour T in (V, c, p) in polynomial time such that $\mathbb{E}_{A\sim p'}\left[c'(T'[A])\right] \geq \frac{1}{4}\mathbb{E}_{A\sim p}\left[c(T[A])\right]$, and

(iv) $|V| \leq |V'| \leq 4|V|/\varepsilon$.

For our algorithm for the symmetric special case of A Priori TSP, we cannot afford to lose a factor of 4. We therefore have to be more careful and conduct more technical arguments. However, we do not need to explicitly transform our instances for our algorithm, which allows us to introduce potentially superpolynomially many copies. We will work with the following notion:

Definition 2.4. *Let $\varepsilon > 0$. An instance (V, c, p) of A Priori TSP is called* ε-normalized *if the instance contains a depot $d \in V$ (with $p(d) = 1$), and $p(v) = \varepsilon$ for all $v \in V \setminus \{d\}$.*

The goal is to prove the following lemma:

Lemma 2.5. *Let $(\varepsilon_i)_{i\in\mathbb{N}} \in (0, 1]^{\mathbb{N}}$ with $\lim_{i\to\infty} \varepsilon_i = 0$. Let $\mathcal{I}$ be the class of all ε-normalized instances with $\varepsilon = \varepsilon_i$ for some $i \in \mathbb{N}$. Then*

(i) The master route ratio (see (2.3)) restricted to instances in $\mathcal{I}$ is the same as the master route ratio restricted to all instances with depot.

(ii) Let $\sigma \in (0, 1)$. Every upper bound on (2.2) for $f(p) = \sigma p \ \forall p \in (0, 1)$ for all instances in $\mathcal{I}$ implies the same upper bound on (2.2) for $f(p) = 1 - (1 - p)^\sigma$ for arbitrary instances with depot.

2.3.1 Introducing a depot

In this section, we show that assuming that the instance contains a *depot*, i.e., an element $d \in V$ with $p(d) = 1$, is no severe restriction – neither in the symmetric version of A Priori TSP nor in the Asymmetric A Priori TSP. We remark that although we can condition on the event that at least two customers are active (because otherwise the cost is zero regardless of the a priori tour), we cannot simply guess these a priori and declare one of them to be the depot. The previous works on Symmetric A Priori TSP, [ST08] and [Zuy11], provided ad hoc proofs that *their* algorithms extend to the general case although they were first formulated for the special case with depot. Theorem 2.1 yields a general reduction for the symmetric case, losing only an

arbitrarily small constant; in the general reduction for the asymmetric case (Theorem 2.2), we lose a constant factor.

Additionally, we also prove that when comparing against the expected cost of an a posteriori optimum tour through the active vertices, we also only lose a constant factor by reducing to instances with a depot. This statement will be useful for proving Theorem 4.1, giving an upper bound on the adaptivity gap.

The high-level idea for the proofs of Theorem 2.1 and Theorem 2.2 is the same and consists of two parts: First we show that for instances whose total activation probability is larger than some constant, it does not harm much to declare one customer to be the depot. This can be done for both the symmetric and the asymmetric version at once, see Lemma 2.6. The second part is to find sufficiently good approximations of the optimum solution for instances where the total activation probability is bounded by a constant. There, for the symmetric version of A Priori TSP, we heavily exploit the symmetric structure of the instance (Lemma 2.11). We thus need a different statement for the asymmetric version, but since it is acceptable to lose constant factors there, the 2-approximation for instances with $\sum_{v \in V} p(v) \leq \frac{1}{2}$ that we provide in Lemma 2.8 is sufficient.

We start with the following lemma that we will apply in both the symmetric and the asymmetric version of A Priori TSP (which is possible since the symmetric version is a special case of Asymmetric A Priori TSP):

Lemma 2.6. *For all instances (V, c, p) of Asymmetric A Priori TSP and all tours S on V, there exists a vertex $d \in V$ such that the instance (V, c, p') of Asymmetric A Priori TSP with Depot defined by*

$$p'(v) = \begin{cases} 1 & v = d \\ p(v) & \text{else,} \end{cases}$$

satisfies

$$\mathbb{E}_{A \sim p} \left[c\left(T[A]\right) \right] \leq \mathbb{E}_{A \sim p'} \left[c\left(T[A]\right) \right] \tag{2.8}$$

for all tours T on V and

$$\mathbb{E}_{A \sim p'} \left[c(S[A]) \right] \leq \left(1 + \frac{2}{p(V)} \right) \cdot \mathbb{E}_{A \sim p} \left[c(S[A]) \right], \tag{2.9}$$

where $p(V) := \sum_{v \in V} p(v)$.

If we apply Lemma 2.6 for S being an optimal a priori tour and we assume $p(V)$ to be lower bounded by some positive constant, then it indeed implies the desired reduction because we can guess the depot d by enumerating all possibilities.

We need a similar lemma in order to prove Theorem 4.1 (and thus proving an upper bound on the adaptivity gap of Asymmetric A Priori TSP): To this end, a statement that there exists a vertex $d \in V$ for which the cost of an a posteriori optimum tour through the active vertices does not increase too much when declaring d to be a depot is necessary:

Lemma 2.7. *For all instances (V, c, p) of Asymmetric A Priori TSP, there exists a vertex $d \in V$ such that the instance (V, c, p') of Asymmetric A Priori TSP with Depot defined by*

$$p'(v) = \begin{cases} 1 & v = d \\ p(v) & \text{else,} \end{cases}$$

satisfies

$$\mathbb{E}_{A \sim p} \left[c\left(T[A]\right) \right] \leq \mathbb{E}_{A \sim p'} \left[c\left(T[A]\right) \right] \tag{2.10}$$

for all tours T on V and

$$\mathbb{E}_{A\sim p'}\left[\mathrm{TSP}(A,c)\right] \leq 2\left(1+\frac{1}{p(V)}\right)\mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A,c)\right]. \tag{2.11}$$

Proof of Lemma 2.6 and Lemma 2.7. For a fixed $v \in V$, we consider the modified activation probabilities that result from declaring v to be a depot:

$$p^{v=d}(u) \coloneqq \begin{cases} 1 & v = u, \\ p(u) & \text{else}, \end{cases}$$

for all $u \in V$. For any family of tours $(T_A)_{A\subseteq V}$ where T_A is a tour on A for each $A \subseteq V$, we have

$$\begin{aligned}
&\mathbb{E}_{A\sim p}\left[c\left(T_{A\cup\{v\}}\right)\right] \\
&= \sum_{U\subseteq V} \mathbb{P}_{A\sim p}[A=U]\cdot c(T_{U\cup\{v\}}) \\
&= \sum_{U\subseteq V: v\in U} \left(\mathbb{P}_{A\sim p}[A=U\setminus\{v\}] + \mathbb{P}_{A\sim p}[A=U]\right)\cdot c(T_U) \\
&= \sum_{U\subseteq V: v\in U} \prod_{u\in U\setminus\{v\}} p(u)\cdot \prod_{w\in V\setminus U}(1-p(w))\cdot c(T_U) \\
&= \sum_{U\subseteq V: v\in U} \mathbb{P}_{A\sim p^{v=d}}[A=U]\cdot c(T_U) \\
&= \mathbb{E}_{A\sim p^{v=d}}[c(T_A)].
\end{aligned}$$

This implies (2.8) and (2.10), since by the triangle inequality

$$\mathbb{E}_{A\sim p}\left[c\left(T[A]\right)\right] \leq \mathbb{E}_{A\sim p}\left[c\left(T[A\cup\{v\}]\right)\right] = \mathbb{E}_{A\sim p^{v=d}}\left[c\left(T[A]\right)\right],$$

for any tour T on V. Note that this holds independently of the choice of v. We will now see that there is also a choice of d satisfying (2.9), and a choice of d satisfying (2.11).

For $v \in V$ and a tour T on some subset $A \subseteq V$ with $v \in A$ define $D_v(T)$ to be the sum of the cost of the edges entering and leaving v in T.

Finding d that satisifies (2.9): We can bound the expected cost of a tour S on V with respect to $p^{v=d}$ as follows:

$$\begin{aligned}
&\mathbb{E}_{A\sim p^{v=d}}[c(S[A])] \\
&= \sum_{U\subseteq V} \mathbb{P}_{A\sim p}[A=U]\cdot c(S[U\cup\{v\}]) \\
&\leq \sum_{U\subseteq V} \mathbb{P}_{A\sim p}[A=U]\cdot c(S[U]) + \sum_{\substack{U\subseteq V \\ U\setminus\{v\}\neq\emptyset}} \mathbb{P}_{A\sim p}[A=U]\cdot D_v(S[U\cup\{v\}]).
\end{aligned}$$

Note that this bound depends on our choice of $v \in V$. We now take the weighted average of these bounds, weighting the bound for choosing a specific $v \in V$ as depot with factor $\frac{p(v)}{p(V)}$. This allows us to conclude that there is a choice of $v \in V$ for declaring v as depot for which the expected cost of a tour S on V with respect to $p^{v=d}$ is at most

$$\sum_{U\subseteq V} \mathbb{P}_{A\sim p}[A=U]\cdot c(S[U]) + \sum_{v\in V}\frac{p(v)}{p(V)}\cdot \sum_{\substack{U\subseteq V \\ U\setminus\{v\}\neq\emptyset}} \mathbb{P}_{A\sim p}[A=U]\cdot D_v(S[U\cup\{v\}]). \tag{2.12}$$

Given $u \neq w \in U$, we denote the set of customers (other than u and w) that we visit when traversing S from u to w by $S_{(u,w)}$. Moreover, for a tour T on $U \subseteq V$ and a vertex $u \in U$, denote by u_T^+ the successor of u in T and by u_T^- the predecessor of u in T. Now for any ordered pair $(u, w) \in V^2$, the edge (u, w) is part of the edges whose cost constitute $D_v(S[U \cup \{v\}])$ if and only if $w = v$ and $u = w^-_{S[U\cup\{v\}]}$ or if $u = v$ and $w = u^+_{S[U\cup\{v\}]}$. Therefore the coefficient of $c(u, w)$ in the right-hand summand of (2.12) is

$$\begin{aligned}
&\frac{p(w)}{p(V)} \cdot \mathbb{P}_{A\sim p}[u = w^-_{S[U\cup\{v\}]}] + \frac{p(u)}{p(V)} \cdot \mathbb{P}_{A\sim p}[w = u^+_{S[U\cup\{v\}]}] \\
&= \frac{2}{p(V)} \cdot p(u) \cdot p(w) \cdot \prod_{v\in S_{(u,w)}} (1 - p(v)) \\
&= \frac{2}{p(V)} \cdot \mathbb{P}_{A\sim p}[u \in A,\, w = u^+_{S[A]}]
\end{aligned}$$

Summing over all ordered pairs (u, w) yields a bound of $\frac{2}{p(V)} \cdot \mathbb{E}_{A\sim p}[c(S[A])]$. Plugging this into (2.12), we obtain that there is a choice of v that we can declare as depot such that the expected cost of a tour S on V with respect to $p^{v=d}$ is at most

$$\left(1 + \frac{2}{p(V)}\right) \cdot \mathbb{E}_{A\sim p}[c(S[A])]$$

as required.

Finding d that satisfies (2.11): For $A \subseteq V$ we denote by T^*_A an optimal ATSP tour on A with respect to c.

Any tour T on a subset $A \subseteq V \setminus \{v\}$ can be extended to a tour on $A \cup \{v\}$ for some $v \in V$ by adding the edges $(x, v), (v, y)$ and $E(T_{[y,x]})$, where $x = \arg\min_{x\in A\setminus\{v\}} c(x, v)$, $y = \arg\min_{y\in A\setminus\{v\}} c(v, y)$ and $T_{[y,x]}$ denotes the y-x-subpath on T. The resulting edge set is connected and Eulerian and thus can be cut short to an ATSP tour on $A \cup \{v\}$. Since $c(E(T_{[y,x]})) \leq c(T)$, we get for all $A \subseteq V, v \in V$:

$$c\left(T^*_{A\cup\{v\}}\right) \leq 2c\left(T^*_A\right) + D_v(T^*_{A\cup\{v\}}). \tag{2.13}$$

Furthermore, we have $\sum_{v\in V} \mathbb{1}_{v\in A} \cdot D_v(T^*_{A\cup\{v\}}) = \sum_{v\in A} D_v(T^*_A) = 2c(T^*_A)$. This holds for every choice of A, hence we can apply expectation to both sides of this equality yielding

$$\mathbb{E}_{A\sim p}\left[\sum_{v\in V} \mathbb{1}_{v\in A} \cdot D_v(T^*_{A\cup\{v\}})\right] = 2\mathbb{E}_{A\sim p}\left[c(T^*_A)\right].$$

Moreover, we can write

$$\sum_{v\in V} p(v) \cdot \mathbb{E}_{A\sim p}\left[D_v(T^*_{A\cup\{v\}})\right] = \mathbb{E}_{A\sim p}\left[\sum_{v\in V} \mathbb{1}_{v\in A} \cdot D_v(T^*_{A\cup\{v\}})\right]$$

because $\mathbb{1}_{v\in A}$ and $D_v(T^*_{A\cup\{v\}})$ are independent random variables since the definition of $D_v(T^*_{A\cup\{v\}})$ does not depend on whether A contains v. In particular, this means that there exists $\hat{v} \in V$ such that

$$\mathbb{E}_{A\sim p}\left[D_{\hat{v}}(T^*_{A\cup\{\hat{v}\}})\right] \leq \frac{2\mathbb{E}_{A\sim p}\left[c(T^*_A)\right]}{p(V)}. \tag{2.14}$$

We will choose $\hat{v}$ as the vertex minimizing this expression. This proves (2.11) since

$$\begin{aligned}
\mathbb{E}_{A\sim p^{\hat{v}=d}}\left[\mathrm{TSP}(A,c)\right] &= \mathbb{E}_{A\sim p^{\hat{v}=d}}\left[c\left(T_A^*\right)\right] \\
&= \mathbb{E}_{A\sim p}\left[c\left(T^*_{A\cup\{\hat{v}\}}\right)\right] \\
&\overset{(2.13)}{\leq} 2\mathbb{E}_{A\sim p}\left[c\left(T_A^*\right)\right] + \mathbb{E}_{A\sim p}\left[D_{\hat{v}}(T^*_{A\cup\{\hat{v}\}})\right] \\
&\overset{(2.14)}{\leq} 2(1+1/p(V))\mathbb{E}_{A\sim p}\left[c\left(T_A^*\right)\right] \\
&= 2(1+1/p(V))\mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A,c)\right].
\end{aligned}$$

□

Thus, if we are given an instance of Asymmetric A Priori TSP where $p(V) \geq k$ for some constant k, we can guess an optimal depot vertex d in polynomial time by trying all vertices $v \in V$ and choosing the best. We then increase the activation probability of d to 1, solve the resulting instance, and view the tour T we obtain as a solution to our original instance. Note that it is easy to compute the exact expected cost of T deterministically in $\mathcal{O}(|V|^3)$ time (and thus choose the best v to declare as depot) via

$$\mathbb{E}_{A\sim p}[c(T[A])] \;=\; \sum_{(u,w)\in V^2} p(u)\cdot p(w)\cdot\left(\prod_{t\in T_{(u,w)}}(1-p(t))\right)\cdot c(u,w), \tag{2.15}$$

where we again write $T_{(u,w)}$ for the set of customers (other than u and w) that we visit when traversing T from u to w. If $k = \frac{1}{2}$, we lose at most a factor of $1+\frac{2}{p(V)} \leq 5$ in the approximation ratio in this reduction. This factor is sufficiently small to prove Theorem 2.2.

In order to prove Theorem 2.1, we cannot choose k as small as $\frac{1}{2}$. Instead, we will show that, for given constants $\varepsilon > 0$ and $\rho \geq 3$, it suffices to consider the case where $p(V) \geq \frac{2\rho}{\varepsilon}$. This way, we only lose at most a factor of $1+\frac{\varepsilon}{\rho}$ when restricting to instances that have a depot (cf. Lemma 2.11). Since this approach relies heavily on the symmetric structure of an instance, we use Lemma 2.8 for the general asymmetric case instead.

For showing how to deal with instances with a small expected number of active customers (i.e., where this expected number is bounded by a constant), it is crucial to take into account that the cost of an a priori tour cut short to fewer than two customers is zero: For example, if $c(v,w) = 1$ for all $v, w \in V$ with $v \neq w$ and $p(v) = \frac{\varepsilon}{n}$ for all $v \in V$ (where $n = |V|$ and $\varepsilon > 0$ tends to zero), then $\mathrm{OPT} \approx \varepsilon^2$ but the standard analysis of any master route solution (even if consisting only of a single point) yields cost at least roughly 2ε.

In order to prove Theorem 2.2, we take advantage of the following lemma:

Lemma 2.8. *Let (V,c,p) be an instance of Asymmetric A Priori TSP. If $p(V) \leq \frac{1}{2}$, then any tour on V has expected cost at most $2\mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A,c)\right]$.*

Proof. Let T be any tour on V. If less than 2 vertices are active, the cost of T cut short to the active vertices is zero. Otherwise let $v_1,\ldots,v_k$ be the active vertices in this order on T. By the triangle inequality this tour costs at most $\sum_{i=1}^{k}(c(s,v_i)+c(v_i,s))$ for any $s \in V$. This gives an upper bound on the expected cost for each $s \in V$. Taking the average over all these bounds

weighted by the activation probability of s yields

$$\begin{aligned}\mathbb{E}_{A\sim p}[c(T[A])] &\leq \sum_{s\in V}\frac{p(s)}{p(V)}\cdot\left(\sum_{v\in V}\mathbb{P}_{A\sim p}\left[|A|\geq 2 \text{ and } v\in A\right]\cdot(c(s,v)+c(v,s))\right)\\ &= \sum_{s\in V}\frac{p(s)}{p(V)}\cdot\left(\sum_{v\in V}p(v)\cdot\mathbb{P}_{A\sim p}\left[|A|\geq 2 \mid v\in A\right]\cdot(c(s,v)+c(v,s))\right)\\ &\leq \sum_{s\in V}\sum_{v\in V}p(s)p(v)(c(s,v)+c(v,s)),\end{aligned}$$

where the last inequality follows from the fact that

$$\mathbb{P}_{A\sim p}\left[|A|\geq 2 \mid v\in A\right]=\mathbb{P}_{A\sim p}\left[|A\setminus\{v\}|\geq 1\right]\leq\mathbb{E}_{A\sim p}\left[|A\setminus\{v\}|\right]=p(V\setminus\{v\})$$

by Markov's inequality.

On the other hand, we can bound

$$\begin{aligned}\mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A,c)\right] &\geq \sum_{x,y\in V}\mathbb{P}_{A\sim p}\left[A=\{x,y\}\right]\cdot(c(x,y)+c(y,x))\\ &= \sum_{x,y\in V}p(x)p(y)\cdot\mathbb{P}_{A\sim p}\left[A\subseteq\{x,y\}\right]\cdot(c(x,y)+c(y,x))\\ &\geq \sum_{x,y\in V}p(x)p(y)\cdot\mathbb{P}_{A\sim p}\left[A=\emptyset\right]\cdot(c(x,y)+c(y,x)).\end{aligned}$$

Now, by Markov's inequality,

$$\mathbb{P}_{A\sim p}\left[A=\emptyset\right]=1-\mathbb{P}_{A\sim p}\left[|A|\geq 1\right]\geq 1-\mathbb{E}_{A\sim p}\left[|A|\right]=1-p(V)\geq\frac{1}{2}.$$

Therefore we conclude

$$\mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A,c)\right]\geq\sum_{x,y\in V}\frac{1}{2}p(x)p(y)\left(c(x,y)+c(y,x)\right)\geq\frac{1}{2}\cdot\mathbb{E}_{A\sim p}[c(T[A])].$$

□

Combining this with Lemma 2.7 leads us to the following conclusion:

Corollary 2.9. *Let $\alpha\colon\mathbb{N}\to\mathbb{R}_{\geq 1}$. Suppose there is a polynomial-time algorithm that, given an instance (V',c',p') of Asymmetric A Priori TSP with Depot, returns an a priori tour T' on V' that satisfies*

$$\mathbb{E}_{A'\sim p'}\left[c'(T'[A'])\right]\leq\alpha(n')\cdot\mathbb{E}_{A'\sim p'}\left[\mathrm{TSP}(A',c')\right], \tag{2.16}$$

then there is a polynomial-time algorithm that, given a general instance (V,c,p) of Asymmetric A Priori TSP, returns a tour T on V that satisfies

$$\mathbb{E}_{A\sim p}\left[c(T[A])\right]\leq 6\cdot\alpha(n)\cdot\mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A,c)\right], \tag{2.17}$$

where $n':=|V'|$ and $n:=|V|$.

Proof. Let (V, c, p) be a general instance of Asymmetric A Priori TSP. If $p(V) = \sum_{v \in V} p(v) \leq 1/2$, then, by Lemma 2.8, any tour tour T on V satisfies $\mathbb{E}_{A \sim p}[c(T[A])] \leq 2\mathbb{E}_{A \sim p}[\mathrm{TSP}(A, c)]$.

Otherwise, for every $v \in V$, we obtain an instance of Asymmetric A Priori TSP (V, c, p^v) with depot v by

$$p^v(u) := \begin{cases} 1 & u = v, \\ p(u) & \text{else.} \end{cases}$$

For each such instance, we compute an priori tour T_v that satisfies (2.16), and return the one minimizing $\mathbb{E}_{A \sim p}[c(T_v[A])]$.

Applying Lemma 2.7 to (V, c, p) implies that there exists a vertex $d \in V$ such that (2.10) and (2.11) of Lemma 2.7 are satisfied. Since $p(V) \geq 1/2$, this yields

$$\begin{aligned} \min_{v \in V} \mathbb{E}_{A \sim p}[c(T_v[A])] &\leq \mathbb{E}_{A \sim p}[c(T_d[A])] \\ &\overset{(2.10)}{\leq} \mathbb{E}_{A \sim p^d}[c(T_d[A])] \\ &\overset{(2.16)}{\leq} \alpha(n) \cdot \mathbb{E}_{A \sim p^d}[\mathrm{TSP}(A, c)] \\ &\overset{(2.11)}{\leq} \alpha(n) \cdot 2\left(1 + \frac{1}{p(V)}\right) \cdot \mathbb{E}_{A \sim p}[\mathrm{TSP}(A, c)] \\ &\leq 6 \cdot \alpha(n) \cdot \mathbb{E}_{A \sim p}[\mathrm{TSP}(A, c)]. \end{aligned}$$

□

Using Lemma 2.6 instead of Lemma 2.7 proves Theorem 2.2.

While losing constant factors in the analysis of the algorithm for Asymmetric A Priori TSP is acceptable since we only show a poly-logarithmic approximation guarantee anyways, we cannot afford to lose so much in the symmetric special case. Fortunately, using the fact that the cost is zero if fewer than two customers are active, Shmoys and Talwar [ST08] show that the a priori tour resulting from an optimum master route solution is *always* at most a factor 3 worse than an optimum a priori tour. The other good news is that a near-optimal master route solution can be found in polynomial time if the expected number of active customers is bounded by a constant, which has some similarities to a PTAS found by Eisenbrand, Grandoni, Rothvoß and Schäfer [EGRS10] for connected facility location in a bounded case. We first need the following fact on the sum of independent Bernoulli random variables:

Lemma 2.10. *Let $\varepsilon > 0$ and $k > 0$ be constants. Then there is a constant $N = N(k, \varepsilon)$ such that the following holds. Let $X_1, \ldots, X_n$ be independent Bernoulli variables and $X = \sum_{i=1}^n X_i$ with $\mathbb{P}[X \geq 2] > 0$ and $\mathbb{E}[X] \leq k$. Then $\mathbb{P}[X > N \mid X \geq 2] \leq \varepsilon$.*

Proof. Let $p_i := \mathbb{P}[X_i = 1]$ for $i = 1, \ldots, n$. We have $\sum_{i=1}^n p_i = \mathbb{E}[X] \leq k$.

Let $S := \{i \in \{1, \ldots, n\} : p_i \leq \frac{1}{2}\}$ be the indices of the Bernoulli variables with small success probability. Let $X' = \sum_{i \in S} X_i$. We always have

$$X - X' \;\leq\; \left|\left\{i \in \{1, \ldots, n\} : p_i > \frac{1}{2}\right\}\right| \;<\; 2k.$$

To bound the probability that X' is large, we compute, for $m > 2$,

$$\begin{aligned}
\mathbb{P}[X' = m] &= \sum_{T \in \binom{S}{m}} \prod_{j \in T} p_j \cdot \prod_{j \in S \setminus T} (1 - p_j) \\
&= \frac{1}{m} \sum_{i \in S} p_i \cdot \sum_{T \in \binom{S \setminus \{i\}}{m-1}} \prod_{j \in T} p_j \cdot \prod_{j \in S \setminus (T \cup \{i\})} (1 - p_j) \\
&= \frac{1}{m} \sum_{i \in S} \frac{p_i}{1 - p_i} \cdot \sum_{T \in \binom{S \setminus \{i\}}{m-1}} \prod_{j \in T} p_j \cdot \prod_{j \in S \setminus T} (1 - p_j) \\
&\leq \frac{1}{m} \sum_{i \in S} \frac{p_i}{1 - p_i} \cdot \sum_{T \in \binom{S}{m-1}} \prod_{j \in T} p_j \cdot \prod_{j \in S \setminus T} (1 - p_j) \\
&= \frac{1}{m} \sum_{i \in S} \frac{p_i}{1 - p_i} \cdot \mathbb{P}[X' = m - 1] \\
&\leq \frac{1}{m} \sum_{i \in S} 2p_i \cdot \mathbb{P}[X' = m - 1] \\
&\leq \frac{2k}{m} \cdot \mathbb{P}[X' = m - 1],
\end{aligned}$$

where we used $p_i \leq \frac{1}{2}$ for $i \in S$ in the second inequality. An iterative application yields

$$\mathbb{P}[X' = m] \leq \frac{2 \cdot (2k)^{m-2}}{m!} \cdot \mathbb{P}[X' = 2].$$

Hence, for any integer $\ell \geq 2ek$,

$$\begin{aligned}
\mathbb{P}[X \geq 2k + \ell] &\leq \mathbb{P}[X' \geq \ell + 1] \leq \sum_{m=\ell+1}^{\infty} \frac{2 \cdot (2k)^{m-2}}{m!} \cdot \mathbb{P}[X' = 2] \\
&\leq \frac{2}{\ell} \cdot \mathbb{P}[X' = 2] \leq \frac{2}{\ell} \cdot \mathbb{P}[X \geq 2],
\end{aligned}$$

where the third inequality follows (with Stirling's formula) from

$$\begin{aligned}
\sum_{m=\ell+1}^{\infty} \frac{(2k)^{m-2}}{m!} &\leq \sum_{m=\ell+1}^{\infty} \frac{(\frac{\ell}{e})^{m-2}}{m!} \leq \frac{e(\frac{\ell}{e})^{\ell}}{\ell(\ell+1)\ell!} \cdot \sum_{i=0}^{\infty} e^{-i} \\
&= \frac{e^2}{(e-1)\ell(\ell+1)} \cdot \frac{(\frac{\ell}{e})^{\ell}}{\ell!} \leq \frac{e^2}{\sqrt{2\pi\ell}(e-1)\ell(\ell+1)} \leq \frac{1}{\ell}
\end{aligned}$$

for $\ell \geq 1$. Setting $\ell = \lceil \max\{2ek, \frac{2}{\varepsilon}\} \rceil$ and $N = 2k + \ell$ finishes the proof. □

Now we are ready to prove the following:

Lemma 2.11. *Let $k > 0$ and $\varepsilon > 0$ be constants. Then we can find an a priori tour with expected cost at most $(3 + \varepsilon) \cdot \mathrm{OPT}$ for any given (symmetric) A Priori TSP instance (V, c, p) with $\sum_{v \in V} p(v) \leq k$ in polynomial time.*

Proof. Shmoys and Talwar [ST08] showed that if we randomly sample a subset $S \subseteq V$ with at least two elements, where S is chosen with probability $(\mathbb{P}_{A \sim p}[A = S])/(\mathbb{P}_{A \sim p}[|A| \geq 2])$, then any

a priori tour T_S corresponding to the master route solution with an optimum TSP tour for S as master tour has expected cost at most $3 \cdot \mathrm{OPT}$.

Now we exploit that $\sum_{v \in V} p(v) \leq k$. We may assume $\varepsilon \leq 1$. By Lemma 2.10 there is a constant $N = N(k, \frac{\varepsilon}{4})$ such that, for the randomly chosen S, the probability that $|S| > N$ is at most $\frac{\varepsilon}{4}$. Hence for at least one set S with $2 \leq |S| \leq N$ we have

$$\mathbb{E}_{A \sim p}\left[c(T_S[A])\right] \;\leq\; \frac{3}{1 - \frac{\varepsilon}{4}} \cdot \mathrm{OPT} \;\leq\; (3 + \varepsilon) \cdot \mathrm{OPT}.$$

Since N is a constant, we can find such a set S and an optimum TSP tour for S by complete enumeration. □

The proof of Theorem 2.1 follows (similarly to the proof of Corollary 2.9) from combining Lemma 2.11 and Lemma 2.6 when choosing the constant k in Lemma 2.11 as $\frac{2\rho}{\varepsilon}$.

2.3.2 Reducing to (almost) uniform activation probabilities

For describing our algorithms for both the symmetric version of A Priori TSP and the Asymmetric A Priori TSP, it will be convenient to only consider instances where all vertices (except for maybe a depot) have the same activation probability. The high-level idea of our reduction is rather simple: we replace each vertex by a set of co-located copies, each with the same low activation probability, such that for each vertex the probability that at least one of its copies is active is sufficiently similar to the probability that the original vertex is active. We start with the respective reduction for the Asymmetric A Priori TSP, i.e. the proof of Proposition 2.3. Since we only aim for an algorithm with a poly-logarithmic approximation guarantee, we are fine with losing constant factors in this reduction.

In order to prove Lemma 2.5, the lemma that we will need for the our algorithm for the symmetric case, we need to be more careful in order not to incur any error. As we will see, it requires some technical care to prove Lemma 2.5 formally which we will do in the end of this subsection.

When replacing each vertex $v \in V$ by so many co-located copies with the new uniform activation probability that the probability that at least one of these copies is active is approximately the same as the original activation probability $p(v)$, it will in general not be exactly equal to $p(v)$. To prove that this is not a problem, we show that changing the activation probabilities of all vertices by at most a small factor does not have a large impact on the expected cost. First, we show that decreasing activation probabilities cannot lead to a larger expected cost.

Lemma 2.12. *Let (V, c, p) and (V, c, p') be two instances of Asymmetric A Priori TSP on the same vertex set V and the same cost function c and $p'(v) \leq p(v)$ for all $v \in V$. Let T be a tour on V. Then the expected cost of T with respect to p' is at most as large as the cost of T with respect to p, i.e.*

$$\mathbb{E}_{A \sim p'}\left[c(T[A])\right] \leq \mathbb{E}_{A \sim p}\left[c(T[A])\right].$$

Proof. We prove this for the case where there is exactly one vertex $\hat{v}$ with $p'(\hat{v}) < p(\hat{v})$. The lemma then follows by induction. Let $V = V_1 \cup V_2$ where we set $V_1 = \{A \subseteq V : \hat{v} \in A\}$ and $V_2 = \{A \subseteq V : \hat{v} \notin A\}$. Then $r \colon V_1 \to V_2, A \mapsto A \setminus \{\hat{v}\}$ is a bijection. Note that for any $U \in V_1$,

we have

$$\begin{aligned}\mathbb{P}_{A\sim p}\left[A = U \text{ or } A = r(U)\right] &= \prod_{v\in r(U)} p(v) \cdot \prod_{v\notin U} (1-p(v)) \\ &= \prod_{v\in r(U)} p'(v) \cdot \prod_{v\notin U} (1-p'(v)) \\ &= \mathbb{P}_{A\sim p'}\left[A = U \text{ or } A = r(U)\right].\end{aligned}$$

The triangle inequality gives us $c(T[r(U)]) \leq c(T[U])$ and therefore

$$\begin{aligned}\mathbb{E}_{A\sim p'}\left[c(T[A])\right] &= \sum_{U\subseteq V} \mathbb{P}_{A\sim p'}\left[A = U\right] \cdot c(T[U]) \\ &= \sum_{U\in V_1} \mathbb{P}_{A\sim p'}\left[A = U \text{ or } A = r(U)\right] \cdot \left(p'(\hat{v}) \cdot c(T[U]) + (1-p'(\hat{v})) \cdot c(T[r(U)])\right) \\ &= \sum_{U\in V_1} \mathbb{P}_{A\sim p'}\left[A = U \text{ or } A = r(U)\right] \cdot \left(c(T[r(U)]) + p'(\hat{v}) \cdot (c(T[U]) - c(T[r(U)]))\right) \\ &\leq \sum_{U\in V_1} \mathbb{P}_{A\sim p}\left[A = U \text{ or } A = r(U)\right] \cdot \left(c(T[r(U)]) + p(\hat{v}) \cdot (c(T[U]) - c(T[r(U)]))\right) \\ &= \sum_{U\subseteq V} \mathbb{P}_{A\sim p}\left[A = U\right] \cdot c(T[U]) \\ &= \mathbb{E}_{A\sim p}\left[c(T[A])\right].\end{aligned}$$

□

Next, we show that when decreasing each activation probability by a factor of at most β, the expected tour cost can decrease at most by a factor β^2.

Lemma 2.13. *Let (V, c, p) and (V, c, p') be two instances of Asymmetric A Priori TSP on the same vertex set V and the same cost function c and $\beta \cdot p(v) \leq p'(v) \leq p(v)$ for all $v \in V$ (where $0 \leq \beta \leq 1$). Let T be a tour on V. Then the expected cost of T with respect to p' is not much smaller than the cost of T with respect to p, in particular*

$$\mathbb{E}_{A\sim p'}\left[c(T[A])\right] \geq \beta^2 \mathbb{E}_{A\sim p}\left[c(T[A])\right].$$

Proof. First, consider the case where T is a cycle. Given two vertices $s, t \in V$, let $V_{(s,t)}$ denote the set of vertices that lie (strictly) between s and t on T. The probability that we have to pay for the edge (s, t) after cutting T short to the active vertices drawn randomly w.r.t. p' is

$$p'(s) \cdot p'(t) \cdot \prod_{v\in V_{(s,t)}} (1-p'(v)) \geq \beta^2 \cdot p(s) \cdot p(t) \cdot \prod_{v\in V_{(s,t)}} (1-p(v)).$$

Thus

$$\begin{aligned}\mathbb{E}_{A\sim p'}\left[c(T[A])\right] &= \sum_{s,t\in V} c(s,t) \cdot p'(s) \cdot p'(t) \cdot \prod_{v\in V_{(s,t)}} (1-p'(v)) \\ &\geq \beta^2 \cdot \sum_{s,t\in V} c(s,t) \cdot p(s) \cdot p(t) \cdot \prod_{v\in V_{(s,t)}} (1-p(v)) \\ &= \beta^2 \cdot \mathbb{E}_{A\sim p}\left[c(T[A])\right].\end{aligned}$$

The case where T visits vertices multiple times can be treated analogously. □

We are finally able to show Proposition 2.3 which we restate here for convenience. Later we will apply it for $\varepsilon = \frac{1}{n}$.

Proposition 2.3. *Given an instance (V, c, p) of Asymmetric A Priori TSP where $p_{\min} := \min_{v \in V} p(v) \geq \varepsilon > 0$, we can transform it in polynomial time (in $|V|$ and $1/\varepsilon$) into an instance (V', c', p') of Asymmetric A Priori TSP such that*

(i) $p'(v) = \varepsilon/2$ *for any* $v \in V'$,

(ii) $\mathrm{OPT}(V', c', p') \leq \mathrm{OPT}(V, c, p)$,

(iii) any tour T' in (V', c', p') can be converted into a tour T in (V, c, p) in polynomial time such that $\mathbb{E}_{A \sim p'}[c'(T'[A])] \geq \frac{1}{4}\mathbb{E}_{A \sim p}[c(T[A])]$, and

(iv) $|V| \leq |V'| \leq 4|V|/\varepsilon$.

Proof. We construct (V', c', p') from (V, c, p) by replacing each $v \in V$ by a set of vertices $C(v)$ (which we will also call *cluster*), i.e. we set $V' = \bigcup_{v \in V} C(v)$. Each $C(v)$ will contain at least one vertex. Let $\pi \colon V' \to V$ be the projection mapping a vertex $v' \in V'$ to the original vertex $v \in V$ such that $v' \in C(v)$. Define c' by setting $c'(v', w') = c(\pi(v'), \pi(w'))$ for $v', w' \in V'$. Set $p'(v') = \varepsilon/2$ for all $v' \in V'$. For each $v \in V$ we choose the number of vertices in $C(v)$ as

$$k_v \coloneqq \left\lceil \log_{1-\varepsilon/2}\left(1 - \frac{p(v)}{2}\right)\right\rceil.$$

We claim that this satisfies

$$p(v) \geq \mathbb{P}_{A \sim p'}[A \cap C(v) \neq \emptyset] \geq \frac{1}{2} \cdot p(v). \tag{2.18}$$

Since we can write $\mathbb{P}_{A \sim p'}[A \cap C(v) \neq \emptyset] = 1 - (1 - \varepsilon/2)^{k_v}$, the second inequality is clear by our choice of k_v. For the first inequality, we may assume that $p(v) < 1$. Since $p(v) \geq \varepsilon$, we have $(1 - \varepsilon/2) \cdot (1 - \frac{p(v)}{2}) \geq (1 - p(v))$. This implies

$$k_v \leq \log_{1-\varepsilon/2}\left(1 - \frac{p(v)}{2}\right) + 1 \leq \log_{1-\varepsilon/2}(1 - p(v)),$$

and hence $1 - (1 - \varepsilon/2)^{k_v} \leq p(v)$, proving the first inequality of (2.18).

Note that $\log_{1-\varepsilon/2}\left(1 - \frac{p(v)}{2}\right) \geq 1$. Moreover, since $(1 - \varepsilon/2)^{2/\varepsilon} \leq \frac{1}{e} \leq \frac{1}{2} \leq 1 - \frac{p(v)}{2}$, we have $\log_{1-\varepsilon/2}(1 - \frac{p(v)}{2}) \leq 2/\varepsilon$. We conclude that $1 \leq k_v \leq 4/\varepsilon$ and therefore

$$|V| \leq |V'| \leq \frac{4|V|}{\varepsilon}.$$

Note that (i) is satisfied by construction.

For (ii), consider an optimum a priori tour T^* for (V, c, p). Let the tour T' on V' arise from T^* by visiting the clusters $C(v)$ in the same order as T^* visits the vertices $v \in V$, and visit vertices inside each cluster consecutively, but in an arbitrary order. Edges inside $C(v)$ have cost 0 and we have $\mathbb{P}_{A \sim p'}[C(v) \cap A \neq \emptyset] \leq p(v)$ by (2.18). Thus Lemma 2.12 implies that $\mathbb{E}_{A \sim p'}[c'(T'[A])] \leq \mathbb{E}_{A \sim p}[c(T^*[A])]$ which yields (ii).

Similarly, if a tour T' in (V', c', p') visits all vertices of each cluster consecutively, it corresponds to a tour T in (V, c, p) with $\mathbb{E}_{A \sim p'}[c'(T'[A])] \geq \frac{1}{4}\mathbb{E}_{A \sim p}[c(T[A])]$ by Lemma 2.13. In order to prove (iii) it thus only remains to show that any tour T' in (V', c', p') can be transformed into a tour T'' in (V', c', p') that visits all vertices of the same cluster $C(v)$ consecutively in polynomial

time without increasing the cost. To this end let T' visit a vertex $v_1 \in C(v)$, then follow a path P, then visit $v_2 \in C(v)$, and follow a path Q before returning to v_1. Construct T_1 out of T' by visiting v_2 immediately after v_1 thus concatenating P and Q without visiting v_2 in between, and T_2 out of T' by visiting v_1 immediately after v_2 (thus concatenating Q and P without visiting v_1 in between). Let $A \subseteq V'$ be the set of active vertices. If $v_1 \in A$ then $c'(T_1[A]) \leq c'(T'[A])$ by the triangle inequality because we get rid off the detour to v_2 between P and Q. If $v_2 \in A$, but $v_1 \notin A$, then $T_1[A]$ visits v_2 followed by the concatenation of $P[A]$ and $Q[A]$ before returning to v_2 while $T'[A]$ visits v_2 followed by the concatenation of $Q[A]$ and $P[A]$ in this order. If the expected cost of the path resulting from visiting Q after P is larger than the one of the path resulting from visiting first Q, then P, T_1 is thus more expensive in expectation than T'. But then T_2 is cheaper than T': If $v_2 \in A$ then we easily get $c'(T_2[A]) \leq c'(T'[A])$. If $v_1 \in A$, but $v_2 \notin A$, then $T_2[A]$ visits first $Q[A]$, then $P[A]$ while $T'[A]$ visits $Q[A]$ after $P[A]$. Therefore at least one of T_1 and T_2 has expected cost at most the expected cost of T'. By induction we can hence rearrange T' such that it visits all vertices of each cluster consecutively without increasing the cost. Note that this rearrangement can be done in polynomial time because in each step we can compute the expected cost of attaching Q after P resp. P after Q and decide for the better in polynomial time. □

This concludes our reduction to uniform activation probabilities with which we will work in Chapter 4. In order to show Lemma 2.5, i.e. the statement we need in Chapter 3, we first prove a few auxiliary statements that will be useful later.

Proposition 2.14. *Let $0 < x < y < 1$. Then*

$$\frac{x}{y} - x \;<\; \frac{\ln(1-x)}{\ln(1-y)} \;<\; \frac{x}{y}.$$

Proof. For the first inequality, we calculate

$$-\ln(1-x) \;=\; \int_{1-x}^{1} \frac{1}{t}\, dt \;>\; x \quad \text{and} \quad -\ln(1-y) \;=\; \int_{1-y}^{1} \frac{1}{t}\, dt \;<\; \frac{y}{1-y}.$$

For the second inequality, we compute

$$\begin{aligned} -\ln(1-y) \;&=\; \int_{1-y}^{1} \frac{1}{t}\, dt > \int_{1-y}^{1-x} \frac{1}{1-x}\, dt + \int_{1-x}^{1} \frac{1}{t}\, dt \\ &=\; \frac{y-x}{1-x} + \int_{1-x}^{1} \frac{1}{t}\, dt > \frac{y-x}{x} \cdot \int_{1-x}^{1} \frac{1}{t}\, dt + \int_{1-x}^{1} \frac{1}{t}\, dt \\ &=\; \frac{y}{x} \cdot \int_{1-x}^{1} \frac{1}{t}\, dt = -\frac{y}{x} \cdot \ln(1-x). \end{aligned}$$

□

Lemma 2.15. *Let $\sigma \in (0,1)$, $\lambda > 0$ and $p \in (0,1]$. Then there is $\varepsilon_p \in (0,1)$ with the following property: For every $\varepsilon \in (0, \varepsilon_p]$ there is $k \in \mathbb{N}$ such that*

(i) $1 - (1-\varepsilon)^k \leq p \leq \varepsilon k$,

(ii) $(1-p)^\sigma \leq (1-\sigma\varepsilon)^k$,

(iii) $1 - (1-p)^\sigma \leq (1+\lambda) \cdot (1 - (1-\sigma\varepsilon)^k)$,

(iv) $(1-\varepsilon)^k \leq (1+\lambda) \cdot (1-p)$ *if* $p < 1$, *and* $(1-\varepsilon)^k \leq \lambda$ *if* $p = 1$.

Proof. For $\varepsilon \in (0,1)$ choose

$$k \;\coloneqq\; \begin{cases} \left\lfloor \frac{\ln(1-p)}{\ln(1-\varepsilon)} \right\rfloor & \text{if } p < 1 \\ \left\lceil \max\left\{ \frac{1}{\varepsilon}, \frac{\ln(\frac{\lambda}{1+\lambda})}{\ln(1-\sigma\varepsilon)} \right\} \right\rceil & \text{if } p = 1 \end{cases}.$$

We first handle the easier case $p = 1$, in which the inequalities hold for any fixed $\varepsilon \in (0,1)$. Indeed, the first inequality in (i) and (ii) follow directly from $p = 1$. Moreover, the second inequality in (i) is implied by our choice of k. Finally, the definition of k yields

$$(1-\varepsilon)^k \le (1-\sigma\varepsilon)^k \le \frac{\lambda}{1+\lambda} < \lambda,$$

which implies (iii) and (iv).

Now, we deal with the more interesting case $p < 1$. Applying Proposition 2.14 with $x = \frac{p}{2}$ and $y = p$ yields

$$-\ln(1-p) \;>\; -2\ln(1-\tfrac{p}{2}) \;>\; 0.$$

Thus, there is $\varepsilon_1 > 0$ such that

$$-\ln(1-p) \;\ge\; (1+\varepsilon_1)\cdot(-2)\cdot\ln(1-\tfrac{p}{2}).$$

As $p > 0$, we have $(1-p)^\sigma < 1$, so there is $\kappa > 0$ with

$$1-(1+\kappa)\cdot(1-p)^\sigma \;\ge\; (1+\lambda)^{-1}\cdot(1-(1-p)^\sigma). \tag{2.19}$$

Pick $\varepsilon_2 > 0$ such that

$$(1-\sigma\varepsilon_2)^{-1}\cdot(1-p)^{-\sigma\varepsilon_2} \;\le\; 1+\kappa.$$

Finally, let $\varepsilon_p \coloneqq \min\{p\cdot\varepsilon_1, \varepsilon_2, \frac{p}{3}, \frac{\lambda}{1+\lambda}\}$. Now let $0 < \varepsilon \le \varepsilon_p$. Then we have

$$\frac{\ln(1-p)}{\ln(1-\varepsilon)} \;\ge\; \frac{p}{\varepsilon}\cdot\left(1+\frac{\varepsilon}{p}\right) \tag{2.20}$$

because

$$\left(1+\frac{\varepsilon}{p}\right)\cdot\frac{-\ln(1-\varepsilon)}{\varepsilon} \;\le\; (1+\varepsilon_1)\cdot\frac{-\ln(1-\varepsilon)}{\varepsilon} \;<\; (1+\varepsilon_1)\cdot\frac{-\ln(1-\frac{p}{2})}{\frac{p}{2}} \le \frac{-\ln(1-p)}{p},$$

where we used $\varepsilon_p \le p\cdot\varepsilon_1$ in the first inequality, Proposition 2.14 with $x = \varepsilon \le \frac{p}{3} < \frac{p}{2} = y$ in the second inequality, and the definition of ε_1 in the third inequality. By definition of ε_p and ε_2, we also have

$$(1-\sigma\varepsilon)^{-1}\cdot(1-p)^{-\sigma\varepsilon} \;\le\; 1+\kappa. \tag{2.21}$$

To prove (i), we compute

$$\begin{aligned} 1-(1-\varepsilon)^k \;\le\; 1-(1-\varepsilon)^{\frac{\ln(1-p)}{\ln(1-\varepsilon)}} \;=\; p \;=\; \varepsilon\cdot\left(\frac{p}{\varepsilon}\cdot\left(1+\frac{\varepsilon}{p}\right)-1\right) \\ \overset{(2.20)}{\le}\; \varepsilon\cdot\left(\frac{\ln(1-p)}{\ln(1-\varepsilon)}-1\right) \;\le\; \varepsilon\cdot k. \end{aligned}$$

To prove (ii), we calculate

$$(1-\sigma\varepsilon)^k \;\geq\; (1-\sigma\varepsilon)^{\frac{\ln(1-p)}{\ln(1-\varepsilon)}} \;=\; (1-p)^{\frac{\ln(1-\sigma\varepsilon)}{\ln(1-\varepsilon)}} \;\geq\; (1-p)^{\frac{\sigma\varepsilon}{\varepsilon}} = (1-p)^{\sigma}.$$

Here, the last inequality follows since $1-p \in [0,1)$ and $\frac{\ln(1-\sigma\varepsilon)}{\ln(1-\varepsilon)} < \frac{\sigma\cdot\varepsilon}{\varepsilon}$ by Proposition 2.14.

To prove (iii), we compute

$$\begin{aligned}(1-\sigma\varepsilon)^k &\leq (1-\sigma\varepsilon)^{-1}\cdot(1-\sigma\varepsilon)^{\frac{\ln(1-p)}{\ln(1-\varepsilon)}}\\ &= (1-\sigma\varepsilon)^{-1}\cdot(1-p)^{\frac{\ln(1-\sigma\varepsilon)}{\ln(1-\varepsilon)}}\\ &\leq (1-\sigma\varepsilon)^{-1}\cdot(1-p)^{\sigma\cdot(1-\varepsilon)}\\ &\overset{(2.21)}{\leq} (1+\kappa)\cdot(1-p)^{\sigma},\end{aligned}$$

where the second inequality follows from Proposition 2.14 with $x=\sigma\varepsilon$ and $y=\varepsilon$. Hence,

$$1-(1-\sigma\varepsilon)^k \;\geq\; 1-(1+\kappa)\cdot(1-p)^{\sigma} \;\overset{(2.19)}{\geq}\; (1+\lambda)^{-1}\cdot(1-(1-p)^{\sigma}).$$

Finally, to prove (iv), we calculate

$$(1+\lambda)^{-1}\cdot(1-\varepsilon)^k \;=\; \left(1-\frac{\lambda}{1+\lambda}\right)\cdot(1-\varepsilon)^k \;\leq\; (1-\varepsilon)^{k+1} \;\leq\; (1-\varepsilon)^{\frac{\ln(1-p)}{\ln(1-\varepsilon)}} \;=\; 1-p.$$

□

Lemma 2.16. *Let (V,c,p) be an instance of A Priori TSP with depot d and let $d \in S \subseteq V$. Then the master route cost of the master route solution on S (cf. (2.1)) is*

$$\mathrm{MR}(S) \;=\; \left(1-\prod_{v\in V\setminus\{d\}}(1-p(v))\right)\cdot \mathrm{TSP}(S,c) + 2\cdot\sum_{v\in V\setminus\{d\}} p(v)\cdot c(v,S).$$

Proof. By definition, we have

$$\begin{aligned}\mathrm{MR}(S) &= \mathbb{E}_{A\sim p}\left[\mathbb{1}_{|A|\geq 2}\cdot\left(\mathrm{TSP}(S,c)+2\cdot\sum_{v\in A}c(v,S)\right)\right]\\ &= \mathbb{P}_{A\sim p}[|A|\geq 2]\cdot\mathrm{TSP}(S,c) + 2\cdot\mathbb{E}_{A\sim p}\left[\mathbb{1}_{|A|\geq 2}\cdot\sum_{v\in A}c(v,S)\right].\end{aligned}$$

As the depot is always active, we have $|A|<2$ if and only if every customer in $V\setminus\{d\}$ is inactive, which happens with probability $\prod_{v\in V\setminus\{d\}}(1-p(v))$. Hence,

$$\mathbb{P}_{A\sim p}[|A|\geq 2] \;=\; 1-\prod_{v\in V\setminus\{d\}}(1-p(v)).$$

We further observe that

$$\mathbb{E}_{A\sim p}\left[\mathbb{1}_{|A|\geq 2}\cdot\sum_{v\in A}c(v,S)\right] \;=\; \mathbb{E}_{A\sim p}\left[\sum_{v\in A\setminus\{d\}}c(v,S)\right] = \sum_{v\in V\setminus\{d\}} p(v)\cdot c(v,S),$$

where the first equality follows from the facts that $\sum_{v\in A}c(v,S)=\sum_{v\in A\setminus\{d\}}c(v,S)$ since $d\in S$ and that this sum can only be nonzero if $|A|\geq 2$ because d is always active. □

Now we are ready to prove Lemma 2.5 which we restate here for convenience:

Lemma 2.5. *Let $(\varepsilon_i)_{i\in\mathbb{N}} \in (0,1]^{\mathbb{N}}$ with $\lim_{i\to\infty} \varepsilon_i = 0$. Let $\mathcal{I}$ be the class of all ε-normalized instances with $\varepsilon = \varepsilon_i$ for some $i \in \mathbb{N}$. Then*

(i) The master route ratio (see (2.3)) restricted to instances in $\mathcal{I}$ is the same as the master route ratio restricted to all instances with depot.

(ii) Let $\sigma \in (0,1)$. Every upper bound on (2.2) for $f(p) = \sigma p \ \forall p \in (0,1)$ for all instances in $\mathcal{I}$ implies the same upper bound on (2.2) for $f(p) = 1-(1-p)^\sigma$ for arbitrary instances with depot.

Proof. Let (V,c,p) be an instance of A Priori TSP with depot d. We show that for every $\delta > 0$, there exist $i \in \mathbb{N}$ and an ε_i-normalized instance (V',c',p') with depot d such that

(i) $\mathrm{OPT}(V',c',p') \le \mathrm{OPT}(V,c,p)$;

(ii) $(1+\delta)\cdot \min\limits_{S'\subseteq V': d\in S'} \mathrm{MR}(S') \ge \min\limits_{S\subseteq V: d\in S} \mathrm{MR}(S)$;

(iii) Let $f(q) \coloneqq 1-(1-q)^\sigma$ and $f'(q) \coloneqq \sigma\cdot q$ for all $q \in [0,1)$, and $f(1) \coloneqq f'(1) \coloneqq 1$. Then

$$\begin{aligned}(1+\delta)\cdot \mathbb{E}_{S\sim f'\circ p'}\left[\alpha\cdot \mathrm{TSP}(S,c') + 2\cdot \sum_{v\in V'} p(v)\cdot c'(v,S)\right]\\ \ge \mathbb{E}_{S\sim f\circ p}\left[\alpha\cdot \mathrm{TSP}(S,c) + 2\cdot \sum_{v\in V} p(v)\cdot c(v,S)\right].\end{aligned}$$

Note that this immediately gives Lemma 2.5. Let $n \coloneqq |V|$ and pick $\lambda > 0$ such that $1-\lambda \ge (1+\delta)^{-1}$, $(1+\lambda)^{n-1} \le 1+\delta$ and

$$(1+\delta)\cdot\left(1-(1+\lambda)^{n-1}\cdot \prod_{v\in V\setminus\{d\}}(1-p(v))\right) \ \ge\ 1-\prod_{v\in V\setminus\{d\}}(1-p(v)). \tag{2.22}$$

Observe that this is possible because $\prod_{v\in V\setminus\{d\}}(1-p(v)) < 1$ since $p(v) > 0$ for all $v \in V$, meaning that (2.22) is a strict inequality for $\lambda = 0$. Apply Lemma 2.15 to obtain constants $\varepsilon_{p(v)} > 0$ for all $v \in V$. Pick i such that $\varepsilon_i \le \min_{v\in V} \varepsilon_{p(v)}$. Let (V',c',p') result from (V,c,p) by keeping the depot d with $p(d) = 1$ unchanged, and replacing each other $v \in V$ by k_v copies of v, each with activation probability ε_i, where k_v is chosen as the number k in Lemma 2.15 for $p = p(v)$ and $\varepsilon = \varepsilon_i$. Denote the projection of V' onto V by π. Then

$$\begin{aligned}\mathrm{TSP}(S',c') &= \mathrm{TSP}(\pi(S'),c) \qquad \text{and}\\ c'(v',S') &= c(\pi(v'),\pi(S')) \qquad \text{for all } S'\subseteq V', v'\in V'.\end{aligned} \tag{2.23}$$

First, we show (i). Let T^* be an optimum a priori tour for (V,c,p) and let T' arise from T^* by replacing each $v \in V\setminus\{d\}$ by its k_v copies. Then T' is a feasible solution for (V',c',p'). Now, sample $S' \subseteq V'$ according to the activation probabilities p'. We always have $c'(T'[S']) = c(T^*[\pi(S')])$. Define $\bar{p}(d) \coloneqq 1$ and $\bar{p}(v) \coloneqq 1-(1-\varepsilon_i)^{k_v}$ for each $v \in V\setminus\{d\}$. Note that $\bar{p}(v) \le p(v)$ by Lemma 2.15 (i). Moreover, for $v \in V$, the probability that $v \in \pi(S')$ equals $\bar{p}(v)$. Hence, the expected cost of T' equals the expected cost of T^* for the instance $(V,c,\bar{p})$. Thus, Lemma 2.12 allows us to conclude that $\mathrm{OPT}(V',c',p') \le \mathrm{OPT}(V,c,p)$.

Next we show (ii). Let $S' \subseteq V'$ with $d \in S'$ and let $S \coloneqq \pi(S')$. Note that $\mathrm{TSP}(S, c) = \mathrm{TSP}(S', c')$. We claim that

$$(1+\delta)^{-1} \cdot \left(1 - \prod_{v \in V \setminus \{d\}} (1 - p(v))\right) \;\leq\; 1 - \prod_{v' \in V' \setminus \{d\}} (1 - p'(v')). \tag{2.24}$$

Indeed, if there is $w \in V \setminus \{d\}$ with $p(w) = 1$, then Lemma 2.15 (iv) tells us that

$$\begin{aligned} 1 \;-\; \prod_{v' \in V' \setminus \{d\}} (1 - p'(v')) \;&\geq\; 1 - (1-\varepsilon_i)^{k_w} \;\geq\; 1 - \lambda \;\geq\; (1+\delta)^{-1} \\ &=\; (1+\delta)^{-1} \cdot \left(1 - \prod_{v \in V \setminus \{d\}} (1 - p(v))\right). \end{aligned}$$

On the other hand, if $p(v) < 1$ for all $v \in V \setminus \{d\}$, then Lemma 2.15 (iv) and (2.22) tell us that

$$\begin{aligned} (1+\delta)^{-1} \cdot \left(1 - \prod_{v \in V \setminus \{d\}} (1 - p(v))\right) \;&\leq\; 1 - \prod_{v \in V \setminus \{d\}} (1+\lambda) \cdot (1 - p(v)) \\ &\leq\; 1 - \prod_{v \in V \setminus \{d\}} (1-\varepsilon_i)^{k_v} \\ &=\; 1 - \prod_{v' \in V' \setminus \{d\}} (1 - p'(v')). \end{aligned}$$

By Lemma 2.16 and Lemma 2.15 (i), we obtain

$$\begin{aligned} \mathrm{MR}(S) \;&=\; \left(1 - \prod_{v \in V \setminus \{d\}} (1 - p(v))\right) \cdot \mathrm{TSP}(S, c) + 2 \cdot \sum_{v \in V \setminus \{d\}} p(v) \cdot c(v, S) \\ &\overset{(2.24)}{\leq} (1+\delta) \cdot \left(1 - \prod_{v' \in V' \setminus \{d\}} (1 - p'(v'))\right) \cdot \mathrm{TSP}(S', c') + 2 \cdot \sum_{v \in V \setminus \{d\}} p(v) \cdot c(v, S) \\ &\leq\; (1+\delta) \cdot \left(1 - \prod_{v' \in V' \setminus \{d\}} (1 - p'(v'))\right) \cdot \mathrm{TSP}(S', c') + 2 \cdot \sum_{v \in V \setminus \{d\}} k_v \cdot \varepsilon_i \cdot c(v, S) \\ &=\; (1+\delta) \cdot \left(1 - \prod_{v' \in V' \setminus \{d\}} (1 - p'(v'))\right) \cdot \mathrm{TSP}(S', c') + 2 \cdot \sum_{v' \in V' \setminus \{d\}} p'(v') \cdot c'(v', S') \\ &=\; (1+\delta) \cdot \mathrm{MR}(S'). \end{aligned}$$

Thus, $(1+\delta) \cdot \mathrm{MR}(S') \geq \mathrm{MR}(S)$.

Finally, we bound the sampling costs as in (iii). For every $U \subseteq V$ with $d \in U$, we have

$$\begin{aligned}
\mathbb{P}_{S\sim f\circ p}[S = U] &= \prod_{v\in U\setminus\{d\}} (1-(1-p(v))^\sigma) \cdot \prod_{v\in V\setminus U} (1-p(v))^\sigma \\
&\leq \prod_{v\in U\setminus\{d\}} (1+\lambda)\cdot(1-(1-\sigma\cdot\varepsilon_i)^{k_v}) \cdot \prod_{v\in V\setminus U} (1-\sigma\cdot\varepsilon_i)^{k_v} \\
&\leq (1+\lambda)^{n-1} \cdot \mathbb{P}_{S'\sim f'\circ p'}[\pi(S') = U] \\
&\leq (1+\delta)\cdot \mathbb{P}_{S'\sim f'\circ p'}[\pi(S') = U],
\end{aligned}$$

where we used Lemma 2.15 (ii) and (iii) in the first inequality. We use this to compare the expected costs of the master tours:

$$\begin{aligned}
\alpha\cdot\mathbb{E}_{S\sim f\circ p}[\mathrm{TSP}(S,c)] &= \alpha\cdot\sum_{U\subseteq V} \mathbb{P}_{S\sim f\circ p}[S=U]\cdot\mathrm{TSP}(U,c) \\
&\leq (1+\delta)\cdot\alpha\cdot\sum_{U\subseteq V} \mathbb{P}_{S'\sim f'\circ p'}[\pi(S')=U]\cdot\mathrm{TSP}(U,c) \\
&\overset{(2.23)}{=} (1+\delta)\cdot\alpha\cdot\sum_{U'\subseteq V'} \mathbb{P}_{S'\sim f'\circ p'}[S'=U']\cdot\mathrm{TSP}(U',c') \\
&= (1+\delta)\cdot\alpha\cdot\mathbb{E}_{S'\sim f'\circ p'}[\mathrm{TSP}(S',c')].
\end{aligned}$$

For the connection costs, we compute a bound of

$$\begin{aligned}
\mathbb{E}_{S\sim f\circ p}\left[\sum_{v\in V} 2\cdot p(v)\cdot c(v,S)\right] &= \sum_{U\subseteq V} \mathbb{P}_{S\sim f\circ p}[S=U]\cdot\sum_{v\in V} 2\cdot p(v)\cdot c(v,U) \\
&\leq (1+\delta)\cdot\sum_{U\subseteq V} \mathbb{P}_{S'\sim f'\circ p'}[\pi(S')=U]\cdot\sum_{v\in V} 2\cdot p(v)\cdot c(v,U) \\
&\leq (1+\delta)\cdot\sum_{U\subseteq V} \mathbb{P}_{S'\sim f'\circ p'}[\pi(S')=U]\cdot\sum_{v\in V} 2\cdot\varepsilon_i\cdot k_v\cdot c(v,U) \\
&\overset{(2.23)}{=} (1+\delta)\cdot\sum_{U'\subseteq V'} \mathbb{P}_{S'\sim f'\circ p'}[S'=U']\cdot\sum_{v\in V'} 2\cdot p'(v)\cdot c'(v,U') \\
&= (1+\delta)\cdot\mathbb{E}_{S'\sim f'\circ p'}\left[\sum_{v\in V'} 2\cdot p'(v)\cdot c'(v,S')\right]
\end{aligned}$$

where we used Lemma 2.15 (i) in the second inequality. This proves (iii). □

Chapter 3

Improved Guarantees for the A Priori TSP

This is joint work with Jannis Blauth, Meike Neuwohner, and Jens Vygen ([BNPV25]).

3.1 Introduction

3.1.1 Motivating questions

We start by reviewing the randomized algorithm by Shmoys and Talwar [ST08]. If fewer than two customers are active, any a priori tour can be cut short to a single point, resulting in cost zero. The algorithm by Shmoys and Talwar [ST08] selects each customer v independently into S with probability $p(v)$: exactly the activation probability. Assuming that the resulting set S is nonempty, there exists an associated master route solution with expected cost at most

$$\mathrm{MR}(S) \;\coloneqq\; \mathbb{E}_{A\sim p}\left[\mathbb{1}_{|A|\geq 2}\cdot\left(\mathrm{TSP}(S,c)+2\cdot\sum_{v\in A}c(v,S)\right)\right].$$

Here $\mathrm{TSP}(S,c)$ denotes the length of an optimum TSP tour for S, and $c(v,S)=\min\{c(v,s) : s\in S\}$ denotes the distance between v and a nearest customer in S (which is zero if $v\in S$); moreover, $\mathbb{E}_{A\sim p}$ denotes the expectation when the set A of active customers is sampled with respect to the given activation probabilities. Later on, $\mathbb{P}_{A\sim p}$ is used analogously. We multiply with $\mathbb{1}_{|A|\geq 2}$ (which is 1 if $|A|\geq 2$ and 0 otherwise) because the cost of the solution is zero if fewer than two customers are active.

If there is a customer d with $p(d)=1$ (a *depot*), then S always contains d and we can bound

$$\mathrm{MR}(S) \;\leq\; \mathrm{TSP}(S,c)+2\cdot\mathbb{E}_{A\sim p}\left[\sum_{v\in A\setminus\{d\}}c(v,S\setminus\{v\})\right].$$

Note that the above upper bound also accounts for connecting active customers in S to the nearest other customer in S, which is not necessary but will allow the following. Taking the expectation over the random choice of S, an upper bound on the expected cost of that master

route solution is

$$\mathbb{E}_{S\sim p}[\mathrm{MR}(S)] \;\le\; \mathbb{E}_{S\sim p}[\mathrm{TSP}(S,c)] + 2\cdot\mathbb{E}_{S\sim p}\left[\sum_{v\in S\setminus\{d\}} c(v, S\setminus\{v\})\right]$$

as the probability distributions to choose S and A are identical and the vertices are sampled independently. Since $\sum_{v\in S\setminus\{d\}} c(v, S\setminus\{v\}) \le \mathrm{TSP}(S,c)$ for all S and $\mathbb{E}_{S\sim p}[\mathrm{TSP}(S,c)] \le \mathrm{OPT}$, where OPT again denotes the expected cost of an optimum a priori tour, this yields

$$\mathbb{E}_{S\sim p}[\mathrm{MR}(S)] \;\le\; 3\cdot\mathrm{OPT}. \tag{3.1}$$

The work of Shmoys and Talwar [ST08] implies that (3.1) also holds when there is no depot and when we take the conditional expectation under the condition that $|S| \ge 2$ (see also [Zuy11]). The Shmoys–Talwar algorithm cannot find an optimum TSP tour for S but uses the double tree algorithm with approximation guarantee 2. As noted by [ES18], one can as well use the Christofides–Serdyukov algorithm with approximation guarantee $\frac{3}{2}$, or in fact any α-approximation algorithm for TSP. Then the expected cost of the resulting master route solution is at most $(\alpha+2)\cdot\mathrm{OPT}$.

This motivates the following questions:

(i) Is it optimal to sample S with exactly the activation probabilities (which is crucially used in the above analysis), or can we improve on the factor $\alpha+2$ by sampling fewer or more?

(ii) How bad can the best master route solution be? We will call this the *master route ratio*: by the Shmoys–Talwar analysis, it is at most 3.

(iii) Can we obtain an approximation guarantee equal to the master route ratio by a master route solution based on random sampling, assuming that we can find optimum TSP tours? What is the best we can achieve with a $\frac{3}{2}$-approximation algorithm for TSP?

(iv) Can we obtain a better deterministic algorithm without a better TSP algorithm?

We give almost complete answers to all these questions.

3.1.2 Our results

The possibility that we sample the empty set or that no customer is active causes significant complications. The previous works [ST08] and [Zuy11] gave ad hoc proofs that *their* algorithms (which are also formulated with a depot) extend to the general case. In Section 2.3.1, we have given a general reduction, losing only an arbitrarily small constant:

Fortunately, instances in which the expected number of active customers is small can be solved easily with an approximation factor $3+\varepsilon$ (for any $\varepsilon > 0$; similar to [EGRS10]; cf. Lemma 2.11), and hence much better than the known guarantees. As we have seen in Section 2.3.1, for instances with a large expected number of active customers, we can assume that there is a customer d that is always active (cf. Theorem 2.1). So we assume this henceforth and call d the depot.

The Shmoys-Talwar algorithm [ST08] includes a customer v into S with probability $p(v)$: the sampling probability is exactly the activation probability. Although this is natural and allows for the simple analysis in Section 3.1.1 (assuming a depot), we show that this is not optimal. Decreasing the probability of including a customer into the master tour improves the approximation guarantee. In Section 3.2 and Section 3.3, we analyze the following *sampling algorithm* (that we already introduced in 2.1) for A Priori TSP instances with depot. Let $f\colon (0,1] \to [0,1]$ with $f(1) = 1$.

(i) Sample a subset $S \subseteq V$ by including every customer v independently with probability $f(p(v))$.

(ii) Call an α-approximation algorithm for (metric) TSP in order to compute a TSP tour for S, which serves as master tour.

(iii) Connect every customer outside S to the nearest customer in S by a pair of parallel edges.

For a given instance this algorithm has expected approximation ratio at most

$$\frac{1}{\mathrm{OPT}} \cdot \mathbb{E}_{S \sim f \circ p} \left[\alpha \cdot \mathrm{TSP}(S, c) + 2 \cdot \sum_{v \in V} p(v) \cdot c(v, S) \right] \tag{3.2}$$

(where $\frac{0}{0} \coloneqq 1$). Shmoys and Talwar [ST08] used the identity function $f(p) = p$. It is easy to construct examples where sampling less or more is better. For example, if $c(v, w) = 1$ for all $v, w \in V$ with $v \neq w$ (and all activation probabilities except for the depot are tiny), it is best to include only the depot in the master tour.[1] On the other hand, if $V = \{v_0, \ldots, v_{n-1}\}$ and $c(v_i, v_j) = \min\{j - i, n + i - j\}$ for $i < j$ (i.e., (V, c) is the metric closure of a cycle), the more we sample, the better. However, even if we choose f depending on the instance, there is a limit on what we can achieve:

Theorem 3.1. *No matter how f is chosen, even depending on the instance in an arbitrary way, the sampling algorithm has no better approximation ratio than*

- 2.655 *even if it computes an optimum TSP tour on the sampled customers;*
- 3.049 *assuming that we never compute a TSP tour on the sampled customers of cost less than* 1.4999 *times the cost of an optimum tour.*

See Section 3.3 for the proof and Fig. 3.5 for a lower bound for different values of α. We do not have a matching upper bound, but we come close. For $\alpha = 1.5$ we prove (in Section 3.2):

Theorem 3.2. *For $\alpha = 1.5$ and $f(p) = 1 - (1-p)^\sigma$ with $\sigma = 0.663$, the sampling algorithm for A Priori TSP instances with depot has approximation guarantee less than* 3.1.

Fig. 3.1 shows this function f. Together with Theorem 2.1 this immediately implies one of our main results:

Corollary 3.3. *There is a randomized* 3.1*-approximation algorithm for A Priori TSP.* □

We conjecture that the bounds in Theorem 3.1 are actually attained by the sampling algorithm with $f(p) = 1-(1-p)^\sigma$, independent of the instance, where σ is a positive constant that depends on α only. See Comment 3.16 for details.

Having explored the limits of the random sampling approach, one might ask what is the limit of choosing an *optimal* master route solution. By van Zuylen's work [Zuy11], the answer to this question is the key to obtain a better deterministic approximation algorithm (see Theorem 3.7). We use the following definition from [Puh21] and [BNPV25]:

Definition 3.4 (master route ratio)**.** *The* master route ratio *is defined to be the supremum of*

$$\frac{\min\{\mathrm{MR}(S) : \emptyset \neq S \subseteq V\}}{\mathrm{OPT}}$$

taken over all A Priori TSP instances (where $\frac{0}{0} \coloneqq 1$).

[1] The expected cost of the master route solution with only the depot d in the master route is $\sum_{v \in V \setminus \{d\}} 2p(v)$, while the expected cost of an optimum a priori tour is at least $\sum_{v \in V \setminus \{d\}} p(v)$. Thus, this yields an approximation ratio of 2 instead of 3.

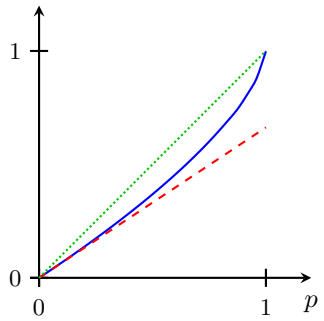


Figure 3.1: The function $p \mapsto 1 - (1-p)^\sigma$ with $\sigma = 0.663$ (blue, solid) defines the sampling probability in Theorem 3.2, which is always at most the identity function (green, dotted), and for small p approximately equal to $p \mapsto \sigma \cdot p$ (red, dashed).

Sometimes we will consider a master route ratio restricted to a certain class of instances. It is very easy to see that the master route ratio (even when restricted to instances with a depot) is at least 2 (for example, if $c(v,w) = 1$ for all $v, w \in V$ with $v \neq w$ and all activation probabilities except for the depot are tiny). By the Shmoys–Talwar analysis, it is at most 3. We will show in Section 3.4:

Theorem 3.5. *The master route ratio for A Priori TSP instances with depot is less than* 2.6.

[Puh21] and [BNPV25] also gave a family of instances for which the master route ratio converges to at least $\frac{1}{1-e^{-1/2}}$, thus providing a lower bound for the master route ratio:

Theorem 3.6. *The master route ratio is at least* $\frac{1}{1-e^{-\frac{1}{2}}} > 2.541$.

The idea is to consider the complete graph on $n+1$ vertices $d, v_1, \ldots, v_n$ where the distance between any two vertices is 1. We then replace each vertex v_i $(1 \leq i \leq n)$ by a vertex set $V_i := \{v_{i,1}, \ldots, v_{i,m}\}$, defining distances as $c(d, v_{i,j}) = 1$ and

$$c(v_{i,j}, v_{i',j'}) \;=\; \begin{cases} 1 & i \neq i' \\ 0 & i = i' \end{cases}.$$

The activation probabilities are given by $p(v_{i,j}) = \frac{1}{2m}$ and $p(d) = 1$. The master route ratio of this type of instance converges to at least

$$\frac{n}{n(1-e^{-\frac{1}{2}})+2}$$

for $m \to \infty$, which itself converges to $\frac{1}{1-e^{-\frac{1}{2}}}$ as $n \to \infty$. We conjecture that the master route ratio is exactly $\frac{1}{1-e^{-1/2}}$.

As van Zuylen's [Zuy11] analysis reveals, her algorithm is a $(2+\alpha\rho)$-approximation algorithm if the master route ratio is ρ and we have an algorithm for TSP that guarantees to produce a tour of cost at most α times the value of the subtour relaxation:

Theorem 3.7. *Let ρ denote the master route ratio for A Priori TSP instances with depot. Suppose we have an algorithm for (metric) TSP that always computes a tour of cost at most α times the value of the subtour elimination LP. Then there is a deterministic $(2+\alpha\rho)$-approximation algorithm for A Priori TSP instances with depot.*

While this theorem is essentially due to van Zuylen in [Zuy11], she did not formally define the master route ratio. A proof can be found in [Puh21] and [BNPV25].

So our new upper bound on the master route ratio immediately implies a better guarantee (combining Theorems 2.1, 3.5, and 3.7 with $\alpha = \frac{3}{2}$ ([Wol80]) or $\alpha = 1.5 - 10^{-36}$ ([KKO23])):

Corollary 3.8. *There is a deterministic 5.9-approximation algorithm for A Priori TSP.* □

3.1.3 Our techniques

The lower bound (Theorem 3.1) is obtained by analyzing a simple example instance family. The main technical difficulty is in proving the upper bounds.

To prove Theorem 3.2 and Theorem 3.5, it will be convenient to only consider instances in which all customers (except the depot) have the same tiny activation probability, i.e. so called *normalized* instances. Recall the definition of ε-normalized instances:

Definition 2.4. *Let $\varepsilon > 0$. An instance (V, c, p) of A Priori TSP is called* ε-normalized *if the instance contains a depot $d \in V$ (with $p(d) = 1$), and $p(v) = \varepsilon$ for all $v \in V \setminus \{d\}$.*

As we have seen in Section 2.3, it suffices to consider ε-normalized instances (cf. Lemma 2.5) which we will do from now on.

On a high level, our proofs of Theorem 3.2 and Theorem 3.5 are similar. In both cases we will design a linear program that encodes the metric c by variables and minimizes the expected cost of an optimum a priori tour subject to (a relaxation of) the constraint that the expected cost of the output of the sampling algorithm is at least 1 (for Theorem 3.2) or the expected cost of any master route solution is at least 1 (for Theorem 3.5), respectively. Then the reciprocals of the LP values yield the desired upper bounds.

However, this approach has to overcome several obstacles. First, it is not obvious how to encode the metric c by finitely many variables, given that we need to consider arbitrary instance sizes. We do this by fixing an optimum a priori tour T^* (a cyclic order of the customers) and carefully aggregating distances of customer pairs with the same number of hops in between on T^*. Of course we exploit the structure of normalized instances.

In the end, we will (almost) ignore variables that correspond to a very large number of hops (where it is very unlikely that none of the customers "in between" is active). These variables have negligible impact because the probability that these edges occur decreases exponentially with increasing number of hops on T^*, whereas the average length of these edges can only grow linearly due to the triangle inequality.

The next idea is to consider certain structured solutions only. Rather than connecting a customer v that is not in the master tour to the *nearest* customer $\mu(v)$ in the master tour, we consider only two possible members of the master tour: we traverse T^* from v in each of the two possible directions, and consider the first customer that we meet and that is contained in our master tour. None of these two may be a nearest one in the master tour, but we still obtain an upper bound. For bounding the master route ratio, we will in addition only consider master tours whose customers are equidistantly distributed on T^* (except for the depot).

In this way, we obtain an optimization problem for a fixed uniform activation probability p (i.e., for p-normalized instances). However, we must let $p \to 0$ according to Lemma 2.5 and hence need a description that is independent of p. This is another major obstacle. To overcome it, we use a second level of aggregation (buckets, rounding the number of hops to integer multiples of, say, $\frac{1}{100p}$). However, this causes several difficulties. In the case of the sampling algorithm, describing the expected cost of the output of the sampling algorithm in terms of the buckets is nontrivial. In case of the master route ratio, the same holds for master route solutions and actually requires a third level of aggregation (bucket intervals).

In the end, we obtain (in both cases) a single, relatively compact, linear program that yields an upper bound for all instance sizes and all activation probabilities from a sequence that converges to zero. We solve the dual LP numerically and just need to check feasibility to prove the desired upper bounds.

3.2 Upper bound on the approximation ratio of random sampling

In this section we will prove Theorem 3.2. As mentioned earlier, we will design a single linear program such that the reciprocal of its optimum value is an upper bound on the approximation ratio of the sampling algorithm for a certain class of normalized instances. For this sake, let $\beta, b_0 > 0$ be constants that we will choose later. We will consider ε-normalized instances where ε is of the form $\varepsilon = \frac{\beta}{b}$ for some odd integer $b \geq b_0$. The meaning of these constants will become clear in Section 3.2.2. For such instances we will obtain an upper bound on the approximation ratio of the sampling algorithm, when sampling each customer with probability σp for $\sigma = 0.663$ (in addition to the depot). Combined with Lemma 2.5, this immediately yields the same upper bound on the approximation guarantee of the sampling algorithm that samples each customer v with probability $1 - (1 - p(v))^\sigma$ for arbitrary A Priori TSP instances with depot.

3.2.1 An optimization problem to bound the approximation ratio

In this section, we first describe an upper bound for all p-normalized instances (for a fixed uniform activation probability p) by a single optimization problem. We will consider the algorithm that samples each customer with probability σp. Let T^* be a fixed optimum a priori tour, with customers appearing in the order $v_0, v_1, \ldots, v_{n-1}$; here v_0 denotes the depot. Let $v_i := v_0$ for $i < 0$ or $i > n - 1$. For $k \in \mathbb{Z}_{\geq 1}$ we define

$$C_k \;:=\; p^2 \cdot \sum_{j \in \mathbb{Z}} c(v_j, v_{j+k}).$$

Observe that only finitely many summands are nonzero. See Figure 3.2 for an example. Since c is a metric, the numbers C_k are nonnegative and satisfy the triangle inequality, that is, for all $i, j \geq 1$

$$C_{i+j} \;\leq\; C_i + C_j. \tag{3.3}$$

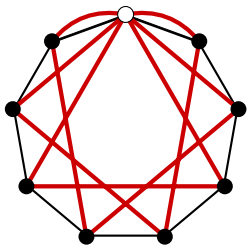


Figure 3.2: The depot v_0 is the white circle at the top; the tour T^* is drawn in black. Adding up the costs of the edges marked in red gives $\frac{C_3}{p^2}$.

Moreover, we can express the expected cost of T^* in terms of the C_i.

Proposition 3.9. *The expected cost of T^* is exactly*

$$\sum_{i=1}^{\infty}(1-p)^{i-1}\cdot C_i. \tag{3.4}$$

Proof. Let $1 \le i \le n-2$ and $1 \le j \le n-i-1$. Then v_j and v_{j+i} are consecutive active customers with probability $p^2\cdot(1-p)^{i-1}$; note that the cost of the edge $\{v_j, v_{j+i}\}$ is counted with exactly the same coefficient in (3.4). Moreover, for $1 \le j \le n-1$, v_j is the first active customer after the depot with probability $p\cdot(1-p)^{j-1}$, and the cost of the edge $\{v_0, v_j\}$ is counted $\sum_{i=j}^{\infty} p^2\cdot(1-p)^{i-1} = p\cdot(1-p)^{j-1}$ times in (3.4). By symmetry, the terms also match for the last active customer before the depot. □

We now consider the master route solution resulting from sampling each customer with probability σp (in addition to the depot). Let α again denote the approximation guarantee of the TSP algorithm that we use. We will now show that the expected cost of this master route solution is at most

$$\sigma^2\sum_{k=1}^{\infty}(1-\sigma p)^{k-1}\cdot\left(\alpha\cdot C_k + 2p\cdot\sum_{i=1}^{k-1}\min\left\{C_i,\, C_{k-i}\right\}\right). \tag{3.5}$$

By the same argumentation as in the proof of Proposition 3.9, the master tour has expected cost at most

$$\alpha\cdot\mathbb{E}_{S\sim q}[c(T^*[S])] \;=\; \alpha\cdot\sigma^2\cdot\sum_{k=1}^{\infty}(1-\sigma p)^{k-1}\cdot C_k,$$

where $q(v) = \sigma p$ for all $v \in V\setminus\{d\}$ and $q(d) = 1$.

Next we bound the expected cost of connecting the active customers to the master tour. Instead of connecting v to the nearest customer in the master tour, we consider only two options: the first sampled customer that we meet when traversing T^* from v in either direction. Note that sampling v_0 with probability 1 is equivalent to sampling each v_j with $j \le 0$ and $j \ge n$ with probability σp. Now, for $j \in \mathbb{Z}$ and $k \ge 2$, the probability that v_j and v_{j+k} are sampled, but none of the intermediate customers is, equals $(\sigma p)^2\cdot(1-\sigma p)^{k-1}$. In this case, the total expected cost of connecting the intermediate active customers can be bounded by $2p\cdot\sum_{i=1}^{k-1}\min\{c(v_j, v_{j+i}), c(v_{j+i}, v_{j+k})\}$. Thus we can bound the expected cost of connecting all active customers to the master tour by

$$\begin{aligned}
&\sum_{k=2}^{\infty}\sigma^2p^2\cdot(1-\sigma p)^{k-1}\cdot\sum_{j\in\mathbb{Z}}2p\cdot\sum_{i=1}^{k-1}\cdot\min\left\{c(v_j, v_{j+i}),\, c(v_{j+i}, v_{j+k})\right\}\\
&\le\; 2\sigma^2p^3\cdot\sum_{k=1}^{\infty}(1-\sigma p)^{k-1}\cdot\sum_{i=1}^{k-1}\min\left\{\sum_{j\in\mathbb{Z}}c(v_j, v_{j+i}),\, \sum_{j\in\mathbb{Z}}c(v_{j+i}, v_{j+k})\right\}\\
&=\; 2\sigma^2p^3\cdot\sum_{k=1}^{\infty}(1-\sigma p)^{k-1}\cdot\sum_{i=1}^{k-1}\min\left\{\sum_{j\in\mathbb{Z}}c(v_j, v_{j+i}),\, \sum_{j\in\mathbb{Z}}c(v_j, v_{j+(k-i)})\right\}\\
&=\; 2\sigma^2p\cdot\sum_{k=1}^{\infty}(1-\sigma p)^{k-1}\cdot\sum_{i=1}^{k-1}\min\left\{C_i,\, C_{k-i}\right\}.
\end{aligned}$$

We conclude that the ratio of (3.5) to (3.4) is an upper bound on the approximation guarantee of the sampling algorithm for that instance. Note that the number of customers appears neither

in (3.4) nor in (3.5). In other words, minimizing (3.4) subject to the constraints that (3.5) is equal to 1 and the C_i are nonnegative and satisfy the triangle inequality (3.3) yields the reciprocal of an upper bound on the approximation guarantee of the sampling algorithm on all p-normalized instances. We arrive at the following optimization problem:

$$\min \sum_{i=1}^{\infty} (1-p)^{i-1} \cdot C_i \qquad \text{(Sampling-OP)}$$

$$\begin{aligned} \text{subject to} \qquad C_i &\geq 0 && \text{for } i \in \mathbb{N} && (3.6)\\ C_i + C_j &\geq C_{i+j} && \text{for } i,j \in \mathbb{N} && (3.7)\\ \sum_{k=1}^{\infty} (1-\sigma p)^{k-1} \cdot \left(\alpha \cdot C_k + 2p \cdot \sum_{i=1}^{k-1} \min\left\{C_i,\, C_{k-i}\right\} \right) &\geq \sigma^{-2}. && && (3.8) \end{aligned}$$

Note that in (3.8) we only require that (3.5) is at least 1 instead of exactly 1. This does not change the infimum because we can always scale all the C_i's. We have proved:

Lemma 3.10. *Let $0 < p < 1$. The reciprocal of the value of* (Sampling-OP) *is an upper bound on the approximation guarantee for the sampling algorithm with $f(p) = \sigma p$ for all p-normalized instances.* □

3.2.2 Obtaining a single linear program

Note that we have an infinite set of optimization problems (one for each choice of p), and, in view of Lemma 2.5, we have to consider the limit for $p \to 0$.

In the following, we require that p is of the form $p = \frac{\beta}{b}$ for some odd integer $b \geq b_0$. Note that $p \to 0$ as $b \to \infty$. In order to obtain a single optimization problem for all such values of p, we put subsequent C_i's into buckets of size b. More precisely, we define buckets

$$B_i \;\coloneqq \sum_{j=\max\{1, ib-\frac{b-1}{2}\}}^{ib+\frac{b-1}{2}} C_j \qquad (3.9)$$

for $i \geq 0$. In the following, we show that we can use the constraints in (Sampling-OP) to generate (slightly relaxed) constraints that only depend on these buckets. First, we note that the buckets are chosen such that they still satisfy the triangle inequality.

Proposition 3.11. *For all $i, j \geq 1$,*

$$B_{i+j} \;\leq\; B_i + B_j.$$

Proof. Indeed, using (3.7), as illustrated in Fig. 3.3,

$$\begin{aligned} B_{i+j} \;&=\; \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} C_{(i+j)b+k} \;=\; \sum_{k=0}^{\frac{b-1}{2}} C_{(i+j)b-\frac{b-1}{2}+2k} + \sum_{k=1}^{\frac{b-1}{2}} C_{(i+j)b-\frac{b-1}{2}+2k-1} \\ &\leq\; \sum_{k=0}^{\frac{b-1}{2}} \left(C_{ib-\frac{b-1}{2}+k} + C_{jb+k} \right) + \sum_{k=1}^{\frac{b-1}{2}} \left(C_{ib+k} + C_{jb-\frac{b-1}{2}+k-1} \right) \\ &=\; \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} C_{ib+k} + \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} C_{jb+k} \;=\; B_i + B_j. \end{aligned}$$

□

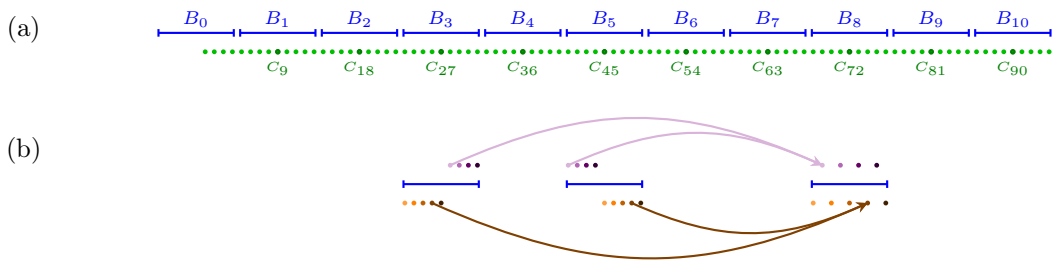


Figure 3.3: (a): The green dots stand for $C_1, C_2, \ldots$, and the centers of the buckets (C_{ib} for $i \geq 1$) are highlighted. Here the bucket size is $b = 9$, and the blue intervals show the buckets $B_0, B_1, B_2, \ldots$. (b): Combining the triangle inequalities for the C_i's leads to triangle inequalities for the B_i's; here shown for $B_i = B_3$ and $B_j = B_5$: We add up all triangle inequalities for C_k from B_3 and C_ℓ from B_5 where C_k and C_ℓ have the same color; illustrated with $C_{26} + C_{48} \leq C_{74}$ and $C_{28} + C_{41} \leq C_{69}$.

Next we aim for an upper bound on the left-hand side of (3.8) that only depends on the buckets. First we show:

Lemma 3.12.

$$\sum_{k=1}^{\infty}(1-\sigma p)^{k-1} \cdot \sum_{i=1}^{k-1} \min\left\{C_i,\, C_{k-i}\right\} \;\leq\; b \cdot \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} e^{-(i+j-1)\cdot\sigma b p} \cdot \min\{B_i, B_j\}.$$

Proof. For $i \in \mathbb{Z}_{\geq 0}$, let $I_i = \{\max\{1, ib - \frac{b-1}{2}\}, \ldots, ib + \frac{b-1}{2}\}$ be the set of indices in the i-th bucket. Then

$$\begin{aligned}
&\sum_{k=1}^{\infty}(1-\sigma p)^{k-1} \cdot \sum_{i=1}^{k-1} \min\left\{C_i,\, C_{k-i}\right\} \\
&= \sum_{k=1}^{\infty}\sum_{\ell=1}^{\infty}(1-\sigma p)^{k+\ell-1} \cdot \min\left\{C_k,\, C_\ell\right\} \\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k\in I_i}\sum_{\ell\in I_j}(1-\sigma p)^{k+\ell-1} \cdot \min\{C_k, C_\ell\} \\
&\leq \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}(1-\sigma p)^{(i+j-1)\cdot b} \cdot \sum_{k\in I_i}\sum_{\ell\in I_j} \min\{C_k, C_\ell\} \\
&\leq \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} e^{-(i+j-1)\cdot\sigma b p} \cdot \sum_{k\in I_i}\sum_{\ell\in I_j} \min\{C_k, C_\ell\}.
\end{aligned}$$

In the last inequality we used $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$. Now,

$$\begin{aligned}
\sum_{k\in I_i}\sum_{\ell\in I_j} \min\{C_k, C_\ell\} \;&\leq\; \min\left\{\sum_{k\in I_i}\sum_{\ell\in I_j} C_k, \sum_{k\in I_i}\sum_{\ell\in I_j} C_\ell\right\} \\
&= \min\left\{|I_j| \cdot B_i, |I_i| \cdot B_j\right\} \\
&\leq b \cdot \min\{B_i, B_j\}.
\end{aligned}$$

□

Using Lemma 3.12 and $\beta = bp$, the left-hand side of (3.8) can be upper bounded by

$$\begin{aligned}
&\alpha \cdot \sum_{k=1}^{\infty}(1-\sigma p)^{k-1} \cdot C_k \;+\; 2bp \cdot \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} e^{-(i+j-1)\cdot\sigma bp} \cdot \min\{B_i, B_j\} \\
&\leq\; \alpha \cdot \sum_{k=0}^{\infty}(1-\sigma p)^{\max\{0,kb-\frac{b-1}{2}-1\}} \cdot B_k \;+\; 2bp \cdot \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} e^{-(i+j-1)\cdot\sigma bp} \cdot \min\{B_i, B_j\} \\
&\leq\; \alpha \cdot \sum_{k=0}^{\infty} e^{-(k-1)\cdot\sigma\beta} \cdot B_k \;+\; 2\beta \cdot \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} e^{-(i+j-1)\cdot\sigma\beta} \cdot \min\{B_i, B_j\}. \qquad (3.10)
\end{aligned}$$

The last inequality follows from $(1-\sigma p)^{kb-\frac{b-1}{2}-1} \leq (1-\sigma p)^{kb-b}$ and $1+x \leq e^x$ for all $x \in \mathbb{R}$. Note that we still sum over infinitely many variables. Hence, in order to get a finite linear program, we aim for an upper bound on (3.10) that only depends on the buckets B_i with $i \leq N$ for some integer N that we will choose later. For this, we use the triangle inequality (Proposition 3.11) to bound the terms depending on buckets B_i with $i > N$ by some term depending on $B_1, \ldots, B_N$ only. For large N this will result in a negligible error as the coefficients in (3.10) decrease exponentially. In Section 3.2.4 we prove the following bound on the error term:

Lemma 3.13. *Let* $\delta_1 \coloneqq \frac{4\beta}{e^{N\sigma\beta}(e^{\sigma\beta}-1)}$ *and* $\delta_2 \coloneqq \left(\alpha + \frac{2\beta}{e^{N\sigma\beta}(e^{\sigma\beta}-1)}\right) \cdot \frac{e^{-N\sigma\beta}}{(1-e^{-\sigma\beta})^2} \cdot (1+N-e^{-\sigma\beta}N)$. *Then*

$$\begin{aligned}
&\alpha \cdot \sum_{k=N+1}^{\infty} e^{-(k-1)\cdot\sigma\beta} \cdot B_k \;+\; 2\beta \cdot \sum_{i,j\in\mathbb{Z}_{\geq 0}:\max\{i,j\}>N} e^{-(i+j-1)\cdot\sigma\beta} \cdot \min\{B_i, B_j\} \\
&\leq\; \delta_1 \cdot \sum_{k=0}^{N} e^{-(k-1)\cdot\sigma\beta} \cdot B_k + \delta_2 \cdot B_1.
\end{aligned}$$

Therefore, we get a lower bound on (Sampling-OP) by minimizing $\sum_{i=1}^{\infty}(1-p)^{i-1} \cdot C_i$ subject to (3.9) and $B_i \geq 0$ for $i \geq 0$, $B_{i+j} \leq B_i + B_j$ for $i, j \geq 1$ with $i + j \leq N$, and

$$(\alpha+\delta_1) \cdot \sum_{k=0}^{N} e^{-(k-1)\cdot\sigma\beta} \cdot B_k + 4\beta \cdot \sum_{j=0}^{N}\sum_{i=0}^{j} e^{-(i+j-1)\cdot\sigma\beta} \cdot \min\{B_i, B_j\} + \delta_2 \cdot B_1 \geq \sigma^{-2}. \qquad (3.11)$$

Note that the objective still contains infinitely many variables and depends on p. The first problem can easily be resolved by bounding

$$\sum_{i=1}^{\infty}(1-p)^{i-1} \cdot C_i \;\geq\; \sum_{i=0}^{\infty}(1-p)^{bi+\frac{b-1}{2}-1} \cdot B_i \;\geq\; \sum_{i=0}^{N}(1-p)^{(i+\frac{1}{2})b} \cdot B_i. \qquad (3.12)$$

It remains to get rid of the dependence on b and p (recall that $p = \frac{\beta}{b}$). To this end, we exploit that $\lim_{b\to\infty}(1-\frac{\beta}{b})^{(i+\frac{1}{2})b} = e^{-(i+\frac{1}{2})\beta}$ for all $i = 0, \ldots, N$, and that by Lemma 2.5, we can choose b_0 arbitrarily large. This will allow us to conclude that we can replace the objective by $\sum_{i=0}^{N} e^{-(i+\frac{1}{2})\beta} \cdot B_i$ and still obtain an upper bound (see the proof of Lemma 3.14 for the technical details). Putting everything together, we arrive at the following LP.

$$\min \sum_{i=0}^{N} e^{-(i+\frac{1}{2})\beta} \cdot B_i \qquad \text{(Sampling-LP)}$$

$$\text{subject to} \quad (\alpha+\delta_1) \cdot \sum_{k=0}^{N} e^{-(k-1)\cdot\sigma\beta} \cdot B_k + \delta_2 \cdot B_1 \; + 4\beta \cdot \sum_{j=0}^{N} \sum_{i=0}^{j} e^{-(i+j-1)\cdot\sigma\beta} \cdot M_{i,j} \;\geq\; \sigma^{-2} \qquad (3.13)$$

$$B_i + B_j \;\geq\; B_{i+j} \qquad \text{for } 1 \leq i \leq j \leq N \text{ with } i+j \leq N \qquad (3.14)$$

$$B_i \;\geq\; M_{i,j} \qquad \text{for } 0 \leq i \leq j \leq N \qquad (3.15)$$

$$B_j \;\geq\; M_{i,j} \qquad \text{for } 0 \leq i \leq j \leq N \qquad (3.16)$$

$$B, M \;\geq\; 0. \qquad (3.17)$$

Recall that δ_1 and δ_2 were defined in Lemma 3.13. We conclude:

Lemma 3.14. *Let N be an integer and $\beta > 0$. The reciprocal of the optimum value of* (Sampling-LP) *is an upper bound on the approximation guarantee of the sampling algorithm for $f(p) = 1-(1-p)^\sigma$ (using an α-approximation algorithm for TSP), for all A Priori TSP instances with depot.*

Proof. We compare the value of (Sampling-LP) to the value of (Sampling-OP). We showed above that for any feasible solution C to (Sampling-OP) we obtain a feasible solution (B, M) to (Sampling-LP) via (3.9) and $M_{i,j} = \min\{B_i, B_j\}$.

Fix $\delta > 0$. Then there exists $b_0 \in \mathbb{N}$ such that for all odd integers $b \geq b_0$

$$\sum_{i=0}^{N} e^{-(i+\frac{1}{2})\beta} \cdot B_i \;\leq\; (1+\delta) \cdot \sum_{i=0}^{N} \left(1-\tfrac{\beta}{b}\right)^{(i+\frac{1}{2})b} \cdot B_i \overset{(3.12)}{\leq} (1+\delta) \cdot \sum_{i=0}^{\infty} \left(1-\tfrac{\beta}{b}\right)^{i-1} \cdot C_i.$$

Thus the value of (Sampling-LP) is at most $(1+\delta)$ times the value of (Sampling-OP) with $p = \frac{\beta}{b}$ for all odd integers $b \geq b_0$. Hence, by Lemma 3.10, $(1+\delta)$ times the reciprocal of the optimum value of (Sampling-LP) is an upper bound on (3.2) for all $\frac{\beta}{b}$-normalized instances for all odd integers $b \geq b_0$. By Lemma 2.5, the same bound then holds for all instances with depot. Since this bound holds for all $\delta > 0$, it also holds for $\delta = 0$. □

3.2.3 The dual LP

In order to obtain a lower bound on the optimum value of (Sampling-LP), we provide a *feasible solution* to the *dual* linear program. For the dual LP, we introduce variables $x_{i,j}$ for the inequalities of type (3.14), variables $v_{i,j}$ and $w_{i,j}$ for the inequalities of type (3.15) and (3.16), respectively, and a variable y for inequality (3.13). Using these variables, the dual LP looks as follows:

$$\max \; \sigma^{-2} \cdot y \qquad \text{(Dual-Sampling-LP)}$$

$$\text{subject to} \quad 4\beta \cdot e^{-(i+j-1)\cdot\sigma\beta} \cdot y \;\leq\; v_{i,j} + w_{i,j} \quad \text{for } 0 \leq i \leq j \leq N \qquad (3.18)$$

$$(\alpha+\delta_1) \cdot e^{-(k-1)\cdot\sigma\beta} \cdot y + \sum_{j=k}^{N} v_{k,j} + \sum_{j=0}^{k} w_{j,k} + \mathbb{1}_{k=1} \cdot \delta_2 \cdot y$$

$$+\mathbb{1}_{k>0} \cdot \left(\sum_{i=1}^{\min\{k,N-k\}} x_{i,k} + \sum_{i=k}^{N-k} x_{k,i} - \sum_{\substack{1 \leq i \leq j \leq N, \\ i+j=k}} x_{i,j} \right) \;\leq\; e^{-(k+\frac{1}{2})\beta} \qquad \text{for } 0 \leq k \leq N \qquad (3.19)$$

$$x, y, v, w \;\geq\; 0. \qquad (3.20)$$

Corollary 3.15. *Let N be an integer and $\beta > 0$. For any feasible solution (x, y, v, w) to* (Dual-Sampling-LP)*, σ^2/y is an upper bound on the approximation ratio of the sampling algorithm with $f(p) = 1 - (1-p)^\sigma$ restricted to A Priori TSP instances with depot.* □

We have computed a dual solution using Gurobi 10.0.1 with $\alpha = 1.5$, $\beta = \frac{1}{100}$, $N = 2500$, and $\sigma = 0.663$, yielding an upper bound of 3.094 and thus proving Theorem 3.2. The dual solution and a Python script that verifies that this is a feasible solution to (Dual-Sampling-LP) can be found at `https://doi.org/10.60507/FK2/JCUIRI`. For $\alpha = 1$, we get an upper bound of 2.694.

Comment 3.16. Solving (Sampling-LP) with the same values for α, β, N, and σ yields an A Priori TSP instance of the same shape as the example provided in Section 3.3. Hence we conjecture that the upper bound given by (Dual-Sampling-LP) converges to the lower bound given in Theorem 3.1 for $\beta \to 0$ and $N\beta \to \infty$.

3.2.4 Bounding the error term (Proof of Lemma 3.13)

We first prove the following auxiliary lemma:

Lemma 3.17. *Let $n \in \mathbb{N}$ and $q \in (0,1)$. Then*

$$\sum_{k=n+1}^{\infty} k \cdot q^{k-1} \;=\; \frac{q^n}{(1-q)^2} \cdot (1 + n - qn). \tag{3.21}$$

Proof. By induction on n. For $n = 0$, the statement is equivalent to the well-known formula

$$\sum_{k=1}^{\infty} k \cdot (1-q) \cdot q^{k-1} \;=\; \frac{1}{1-q}$$

for the expected value of a geometrically distributed random variable. Next, assume that (3.21) holds for some $n \in \mathbb{N}$. Then

$$\sum_{k=n+2}^{\infty} k \cdot q^{k-1} \;=\; \sum_{k=n+1}^{\infty} k \cdot q^{k-1} - (n+1) \cdot q^n \overset{(3.21)}{=} \frac{q^n}{(1-q)^2} \cdot (1 + n - qn) - (n+1) \cdot q^n$$
$$= \frac{q^n}{(1-q)^2} \cdot (1 + n - qn - (n+1) \cdot (1-q)^2) \;=\; \frac{q^{n+1}}{(1-q)^2} \cdot (n + 2 - q(n+1)),$$

which is (3.21) for $n+1$. □

Now we are ready to prove Lemma 3.13:

Proof of Lemma 3.13. We compute

$$2\beta \cdot \sum_{i,j \in \mathbb{Z}_{\geq 0} : \max\{i,j\} > N} e^{-(i+j-1)\cdot\sigma\beta} \cdot \min\{B_i, B_j\}$$
$$\leq\; 4\beta \cdot \sum_{i=0}^{N} \sum_{j=N+1}^{\infty} e^{-(i+j-1)\cdot\sigma\beta} \cdot B_i \;+\; 2\beta \cdot \sum_{i=N+1}^{\infty} \sum_{j=N+1}^{\infty} e^{-(i+j-1)\cdot\sigma\beta} \cdot B_i$$
$$=\; 4\beta \cdot \sum_{i=0}^{N} e^{-(i-1)\cdot\sigma\beta} \cdot B_i \cdot \sum_{j=N+1}^{\infty} e^{-j\sigma\beta} \;+\; 2\beta \cdot \sum_{i=N+1}^{\infty} e^{-(i-1)\cdot\sigma\beta} \cdot B_i \cdot \sum_{j=N+1}^{\infty} e^{-j\sigma\beta}$$
$$=\; \delta_1 \cdot \sum_{i=0}^{N} e^{-(i-1)\cdot\sigma\beta} \cdot B_i \;+\; \frac{2\beta}{e^{N\sigma\beta}(e^{\sigma\beta} - 1)} \cdot \sum_{i=N+1}^{\infty} e^{-(i-1)\cdot\sigma\beta} \cdot B_i$$

Bounding $B_i \leq i \cdot B_1$ for $i > N$ by using to the triangle inequality (Proposition 3.11), we obtain

$$\begin{aligned}
&\alpha \cdot \sum_{k=N+1}^{\infty} e^{-(k-1)\cdot\sigma\beta} \cdot B_k \;+\; 2\beta \cdot \sum_{i,j\in\mathbb{Z}_{\geq 0}:\max\{i,j\}>N} e^{-(i+j-1)\cdot\sigma\beta} \cdot \min\{B_i, B_j\} \\
&\leq\; \delta_1 \cdot \sum_{i=0}^{N} e^{-(i-1)\cdot\sigma\beta} \cdot B_i + \left(\alpha + \frac{2\beta}{e^{N\sigma\beta}(e^{\sigma\beta}-1)}\right) \cdot \sum_{k=N+1}^{\infty} e^{-(k-1)\cdot\sigma\beta} \cdot B_k \\
&\leq\; \delta_1 \cdot \sum_{i=0}^{N} e^{-(i-1)\cdot\sigma\beta} \cdot B_i + \left(\alpha + \frac{2\beta}{e^{N\sigma\beta}(e^{\sigma\beta}-1)}\right) \cdot \sum_{k=N+1}^{\infty} e^{-(k-1)\cdot\sigma\beta} \cdot k \cdot B_1 \\
&=\; \delta_1 \cdot \sum_{i=0}^{N} e^{-(i-1)\cdot\sigma\beta} \cdot B_i + \delta_2 \cdot B_1,
\end{aligned}$$

where we used Lemma 3.17 in the final equality with $n = N$ and $q = e^{-\sigma\beta}$. □

3.3 Lower bound on the approximation ratio of the sampling algorithm

In this section we prove Theorem 3.1. We will provide a family of instances for which the sampling algorithm that we described in Section 3.1.2 has no better approximation ratio than 2.655, no matter how we choose f, and even when assuming that we can compute optimal TSP tours on the sampled customers. Using the currently best approximation guarantee for metric TSP leads to a ratio of more than 3.049 (again, even if we try all f).

In the proof we exploit that each instance in the family that we describe has uniform activation probability, i.e., $p(v) = p$ for each $v \in V$, where p is a small positive number. Hence, it suffices to consider functions f with $f(1) = 1$ and $f(p) = \sigma p$ for some $\sigma \in \left[0, \frac{1}{p}\right]$.

Let $\gamma \in [1, 2]$ be a parameter chosen later, depending only on the approximation ratio α for TSP. Let $0 < \varepsilon \ll 1$. We will choose $p > 0$ (very small) and $n \in \mathbb{N}$ (very large), depending on γ and ε only (cf. Lemma 3.18). We then consider a p-normalized instance of the A Priori TSP with $V = \{v_0, \cdots, v_{n-1}\}$ with v_0 being the depot. Then for $0 \leq i < j \leq n-1$ with $k = \min\{j - i, i + n - j\}$ define distances $c(v_i, v_j) = \frac{c_k}{n}$, where (cf. Fig. 3.4)

$$c_k \;:=\; \begin{cases} \frac{\gamma}{p} & \text{if } k \leq \frac{\gamma}{p} \\ k & \text{otherwise.} \end{cases}$$

Note that they form a metric. We will show the claimed lower bounds for this set of instances.

We choose p and n such that the following properties hold that we will use later:

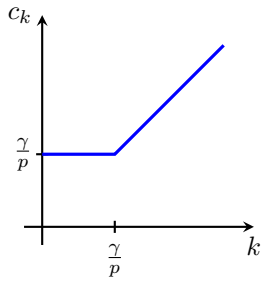
Lemma 3.18. *For every $\gamma \in [1, 2]$ and every $\varepsilon \in (0, \frac{1}{4})$, there are $p > 0$ and $n \in \mathbb{N}$ such that $\frac{\gamma}{p}$ is integral and*

(i) $p \leq (1 - 4\varepsilon) \cdot \varepsilon^2$,

(ii) $(1 - \varepsilon p) \cdot (1 - xp)^{\frac{y}{p}} \geq e^{-xy} - \varepsilon$ *for all* $x \in \left[0, \frac{1}{p}\right]$ *and* $y \in \left[1, \frac{2}{\varepsilon}\right]$,

(iii) $n \geq \frac{4}{\varepsilon p} + \frac{3}{\varepsilon}$,

(iv) $(1 + x) \cdot e^{-p\varepsilon^2 x} \leq \varepsilon - \frac{1}{n}$ *for all* $x \geq \varepsilon^2 pn - 3$.



α	γ	best choice for σ	lower bound on approx. ratio of the sampling algorithm
1	1.623	≈ 0.623	> 2.655
$\frac{4}{3}$	1.383	≈ 0.651	> 2.914
1.4999	1.291	≈ 0.663	> 3.049

Figure 3.4: Left: The distance of v_i and v_j for $0 \leq i < j \leq n-1$ is $\frac{c_k}{n}$, depending on $k = \min\{j-i, i+n-j\}$. The figure shows the dependence of c_k on k. Right: The table shows, for a given approximation ratio α that our black box TSP approximation algorithm achieves, how we should choose γ such that the sampling algorithm performs worst possible even if we choose σ best possible.

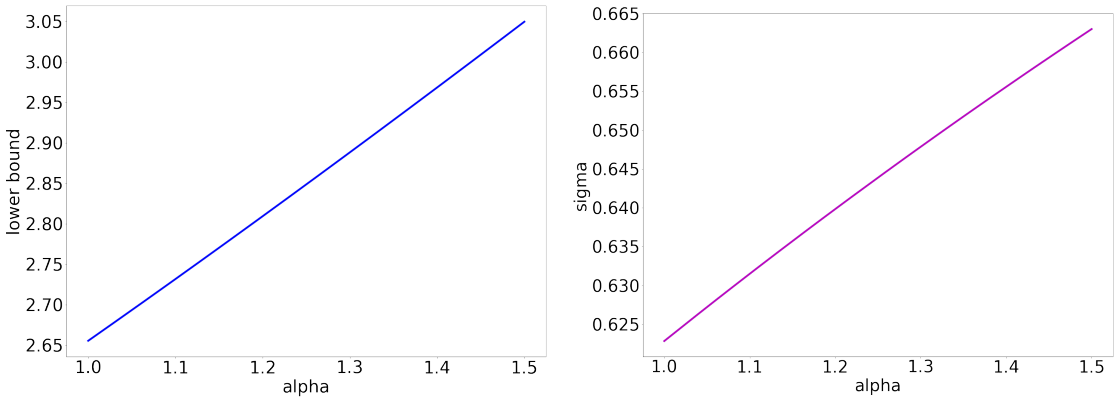


Figure 3.5: Left: The lower bound on the approximation ratio of the sampling algorithm resulting from our lower bound example, depending on the approximation ratio α that our black box TSP approximation algorithm achieves. Right: The plot shows the best choice for σ (again depending on α).

Proof. Fix $\gamma \in [1,2]$ and $\varepsilon \in (0, \frac{1}{4})$. As $\lim_{t\to 0}(1-t)^{-\frac{2\ln(\varepsilon)}{\varepsilon}} = 1$, pick $\delta \in (0,1)$ with

$$1 - \frac{\varepsilon}{2} \;\leq\; (1-\delta)^{-\frac{2\ln(\varepsilon)}{\varepsilon}}. \tag{3.22}$$

Moreover, as $\lim_{t\to 0}(1-t)^{\frac{1}{t}} = e^{-1}$, we can pick $p \in (0, \frac{1}{2})$ such that (i) holds, $\frac{\gamma}{p}$ is integral and such that

$$\forall t \in (0, -\ln(\varepsilon)\cdot p] : \; (1-t)^{\frac{1}{t}} \;\geq\; (1-\delta)\cdot e^{-1}.$$

In particular,

$$\forall x \in (0, -\ln(\varepsilon)] : \; (1-xp)^{\frac{1}{xp}} \;\geq\; (1-\delta)\cdot e^{-1}. \tag{3.23}$$

We show that (ii) holds. First of all, if $x > -\ln(\varepsilon)$, then the right-hand-side of (ii) is negative, whereas the left-hand-side is nonnegative, so we are done in this case. Next, assume that $x \leq$

$-\ln(\varepsilon)$ holds. If $x = 0$, then (ii) is equivalent to $1 - \varepsilon p \geq 1 - \varepsilon$. For $x > 0$, we calculate

$$\begin{aligned}
(1-\varepsilon p)\cdot(1-xp)^{\frac{y}{p}} &= (1-xp)^{\frac{y}{p}} - \varepsilon p\cdot(1-xp)^{\frac{y}{p}} && \mid p \in \left(0, \frac{1}{2}\right) \\
&\geq \left((1-xp)^{\frac{1}{xp}}\right)^{xy} - \frac{\varepsilon}{2} && \mid (3.23) \\
&\geq (1-\delta)^{xy}\cdot e^{-xy} - \frac{\varepsilon}{2} && \mid x \leq -\ln(\varepsilon) \\
&\geq (1-\delta)^{-\frac{2\ln(\varepsilon)}{\varepsilon}}\cdot e^{-xy} - \frac{\varepsilon}{2} && \mid (3.22) \\
&\geq \left(1-\frac{\varepsilon}{2}\right)\cdot e^{-xy} - \frac{\varepsilon}{2} \\
&\geq e^{-xy} - \varepsilon.
\end{aligned}$$

Finally, as $\lim_{x\to\infty}(1+x)\cdot e^{-p\varepsilon^2 x} = 0$, we may choose n subject to (iii) and (iv). □

To prove Theorem 3.1 we use the following auxiliary lemma multiple times:

Lemma 3.19. *For $\beta \in \left(0, \frac{1}{p}\right]$ and $k \in \mathbb{N}_{\geq \frac{\gamma}{p}}$ we have*

$$\sum_{i=1}^{k}(\beta p)^2\cdot(1-\beta p)^{i-1}\cdot c_i = \beta\gamma + (1-\beta p)^{\frac{\gamma}{p}} - (1+\beta pk)(1-\beta p)^k, \tag{3.24}$$

where $0^0 := 1$.

Proof. The case $\beta p = 1$ is straightforward. Next, assume $\beta \in (0, \frac{1}{p})$. For $k = \frac{\gamma}{p}$, we have

$$\begin{aligned}
&\sum_{i=1}^{\frac{\gamma}{p}}(\beta p)^2\cdot(1-\beta p)^{i-1}\cdot c_i = \sum_{i=1}^{\frac{\gamma}{p}}(\beta p)^2\cdot(1-\beta p)^{i-1}\cdot\frac{\gamma}{p} = (\beta\gamma)\cdot\sum_{i=1}^{\frac{\gamma}{p}}(\beta p)\cdot(1-\beta p)^{i-1} \\
&= (\beta\gamma)\cdot\left(1-(1-\beta p)^{\frac{\gamma}{p}}\right) = \beta\gamma + (1-\beta p)^{\frac{\gamma}{p}} - \left(1+\beta p\cdot\frac{\gamma}{p}\right)(1-\beta p)^{\frac{\gamma}{p}}.
\end{aligned}$$

For the case $k \geq \frac{\gamma}{p} + 1$, we use Lemma 3.17 to compute

$$\begin{aligned}
&\sum_{i=1}^{k}(\beta p)^2\cdot(1-\beta p)^{i-1}\cdot c_i \\
&= \sum_{i=1}^{\frac{\gamma}{p}}(\beta p)^2\cdot(1-\beta p)^{i-1}\cdot c_i + (\beta p)^2\cdot\left(\sum_{i=\frac{\gamma}{p}+1}^{\infty} i\cdot(1-\beta p)^{i-1} - \sum_{i=k+1}^{\infty} i\cdot(1-\beta p)^{i-1}\right) \\
&= (\beta\gamma)\cdot\left(1-(1-\beta p)^{\frac{\gamma}{p}}\right) + (1-\beta p)^{\frac{\gamma}{p}}\cdot\left(1+\frac{\gamma}{p}\beta p\right) - (1-\beta p)^k\cdot(1+k\beta p) \\
&= \beta\gamma + (1-\beta p)^{\frac{\gamma}{p}} - (1+\beta pk)(1-\beta p)^k.
\end{aligned}$$

□

Now we proceed to the main part of the proof of Theorem 3.1. First, we aim for an upper bound on the expected cost of an optimum a priori tour:

Lemma 3.20. *The optimum a priori tour for the considered instance has expected cost at most $(1+\varepsilon)\cdot(\gamma + e^{-\gamma})$.*

Proof. Consider the a priori tour T^* that visits $v_0, \dots, v_{n-1}$ in this order. This a priori tour has expected cost

$$\begin{aligned}
&\sum_{i=1}^{n-2} \sum_{j=1}^{n-1-i} p^2 \cdot (1-p)^{i-1} \cdot c(v_j, v_{j+i}) + \sum_{i=1}^{n-1} p \cdot (1-p)^{i-1} \cdot \big(c(v_0, v_i) + c(v_{n-i}, v_0)\big) \\
&\leq \sum_{i=1}^{n-2} p^2 \cdot (1-p)^{i-1} \cdot c_i + 2 \sum_{i=1}^{n-1} p \cdot (1-p)^{i-1} \cdot \frac{c_i}{n} \\
&\leq (1+\varepsilon) \cdot \sum_{i=1}^{n-1} p^2 \cdot (1-p)^{i-1} \cdot c_i \\
&\leq (1+\varepsilon) \cdot (\gamma + (1-p)^{\frac{\gamma}{p}}) \\
&\leq (1+\varepsilon) \cdot (\gamma + e^{-\gamma}),
\end{aligned}$$

where we used $\frac{2}{n} \leq \varepsilon p$ (which follows from Lemma 3.18 (iii)) in the second inequality and Lemma 3.19 for $\beta = 1$ and $k = n-1$ in the third inequality. □

Second, we aim for a lower bound on the expected cost of the master route solution: We define $q\colon V \to [0,1]$ by setting $q(v) \coloneqq \sigma p$ for $v \in V \setminus \{v_0\}$ and $q(v_0) \coloneqq 1$.

Lemma 3.21. *The expected cost of the master route solution when sampling each customer $v \in V$ with probability $q(v)$ is at least*

$$\min\left\{ \frac{e^{-2}}{\varepsilon} - 1,\ (1-\varepsilon)^3 \cdot (1-4\varepsilon) \cdot \left(\alpha(\sigma\gamma + e^{-\sigma\gamma}) + 2\gamma + \frac{1}{\sigma} e^{-2\sigma\gamma} \right) \right\}.$$

Proof. We distinguish several cases, depending on how large σ is.
Case 1: $\sigma \leq \varepsilon$.

In this case, we consider the connection cost only. For each v_i with $\lceil \frac{1}{\varepsilon p} \rceil \leq i \leq n - \lceil \frac{1}{\varepsilon p} \rceil$, the probability that no sampled vertex is fewer than $\lceil \frac{1}{\varepsilon p} \rceil$ hops on T^* away from v_i is at least

$$(1-\sigma p)^{2\lceil \frac{1}{\varepsilon p} \rceil - 1} \ \geq\ (1-\varepsilon p)^{1+\frac{2}{\varepsilon p}} \ \geq\ e^{-2} - \varepsilon,$$

where we used Lemma 3.18 (ii) with $x = \varepsilon$ and $y = \frac{2}{\varepsilon}$. In this event, if v_i is active, we have to pay connection cost at least $2 \cdot \frac{1}{n} \lceil \frac{1}{\varepsilon p} \rceil$. Hence the total connection cost is at least

$$(e^{-2} - \varepsilon) \cdot \left(n + 1 - 2 \left\lceil \frac{1}{\varepsilon p} \right\rceil \right) \cdot \frac{2p}{n} \left\lceil \frac{1}{\varepsilon p} \right\rceil \ \geq\ (e^{-2} - \varepsilon) \cdot \left(n - 1 - \frac{2}{\varepsilon p} \right) \cdot \frac{2}{\varepsilon n} \ \geq\ \frac{e^{-2}}{\varepsilon} - 1,$$

where we used that $n \geq \frac{4}{\varepsilon p} + 2$ by Lemma 3.18 (iii) in the last inequality.
Case 2: $\sigma > \varepsilon$. Let S denote the set of sampled customers, and, if $|S| \geq 2$, let $e_{\max}[S]$ be a longest edge of the tour $T^*[S]$ that we get from T^* by skipping the customers that were not sampled. Note that S and $e_{\max}[S]$ are random variables that depend on the sampling.

We want to compute a lower bound on the expected cost of the master tour. It is not always true that $T^*[S]$ is an optimum TSP tour for S, but almost. Namely, if $|S| = 1$, the statement is true, and otherwise, we claim that $c(T^*[S]) - c(e_{\max}[S])$ is a lower bound on the cost of any TSP tour for S. This follows from

Claim 1: For every $\{v_0\} \subsetneq S \subseteq V$, $T^*[S] \setminus \{e_{\max}[S]\}$ is a min-cost spanning tree in (S, c).

Proof: To prove this, we may assume (by the cyclic symmetry of c) that $S = \{v_i : i \in I\}$ and $e_{\max}[S] = \{v_{\min I}, v_{\max I}\}$; then let $i, \ell \in I$ with $i < \ell$, and let $\{j, k\}$ be any edge on the path from

i to ℓ in the tree $T^*[S]\setminus\{e_{\max}[S]\}$ (i.e., $i \le j < k \le \ell$ and $j,k \in I$). We show $c(v_i,v_\ell) \ge c(v_j,v_k)$, which implies optimality of the spanning tree $T^*[S]\setminus\{e_{\max}[S]\}$ (see, e.g., Theorem 6.3 in [KV18]). Indeed, $c(v_i,v_\ell) \ge \frac{1}{n}\max\{\frac{\gamma}{p},\, \min\{\ell-i,\, n+i-\ell\}\} \ge \frac{1}{n}\max\{\frac{\gamma}{p},\, \min\{k-j,\, n+\min I-\max I\}\} \ge \min\{c(v_j,v_k), c(e_{\max}[I])\} \ge c(v_j,v_k)$. This concludes the proof of Claim 1. ■

Next we bound the expected cost of $e_{\max}[S]$, or, to be more precise,

$$\sum_{\{v_0\}\subsetneq U\subseteq V} \mathbb{P}_{S\sim q}[S=U]\cdot c(e_{max}[U]).$$

The probability that $T^*[S]$ contains an edge of cost at least $\varepsilon+\frac{1}{n}$ is at most

$$\sum_{j=0}^{n-1}(1-\sigma p)^{\left(\varepsilon+\frac{1}{n}\right)\cdot n-1} \;=\; n\cdot(1-\sigma p)^{\varepsilon n} \;\le\; n\cdot e^{-\sigma p\varepsilon n} \;\le\; n\cdot e^{-p\varepsilon^2 n} \;\le\; \varepsilon-\frac{1}{n}$$

by Lemma 3.18 (iv) (when choosing $x=n$). As every edge costs less than 1, the expected length of $e_{\max}[S]$ is less than 2ε.

Therefore, using Claim 1, the expected cost of the optimum master tour for S is at least $c(T^*[S])-2\varepsilon$, and the expectation cannot increase if we sample every customer, including the depot, with probability σp. Hence, writing $v_i = v_{i-n}$ for $i = n,\dots,2n-2$ we get as lower bound

$$\begin{aligned}
\mathbb{E}_{S\sim q}\left[\mathrm{TSP}(S,c)\right] &\ge \mathbb{E}_{S\sim q}\left[c(T^*[S])\right]-2\varepsilon\\
&\ge \sum_{i=1}^{n-1}\sum_{j=0}^{n-1}(\sigma p)^2\cdot(1-\sigma p)^{i-1}\cdot c(v_j,v_{j+i})-2\varepsilon\\
&= \sum_{i=1}^{n-1}(\sigma p)^2\cdot(1-\sigma p)^{i-1}\cdot c_i-2\varepsilon\\
&= \left(\sigma\gamma+(1-\sigma p)^{\frac{\gamma}{p}}-(1+\sigma p(n-1))(1-\sigma p)^{n-1}\right)-2\varepsilon,
\end{aligned}$$

where we used Lemma 3.19 for $k=n-1$ and $\beta=\sigma$ (recall $\varepsilon<\sigma\le\frac{1}{p}$) in the last equation. Using $\sigma\ge\varepsilon$, we can apply Lemma 3.18 (iv) with $x=\sigma(n-1)$ and obtain

$$(1+\sigma p(n-1))\cdot(1-\sigma p)^{n-1} \;\le\; (1+\sigma(n-1))\cdot e^{-\sigma p(n-1)} \;\le\; (1+\sigma(n-1))\cdot e^{-p\varepsilon^2\sigma(n-1)} \;\le\; \varepsilon.$$

This yields

$$\begin{aligned}
\mathbb{E}_{S\sim q}\left[\mathrm{TSP}(S,c)\right] &\ge \sigma\gamma+(1-\sigma p)^{\frac{\gamma}{p}}-3\varepsilon\\
&\ge \sigma\gamma+e^{-\sigma\gamma}-4\varepsilon\\
&\ge (1-4\varepsilon)\cdot(\sigma\gamma+e^{-\sigma\gamma}) \qquad (3.25)
\end{aligned}$$

using Lemma 3.18 (ii) for $x=\sigma$ and $y=\gamma$ in the second inequality and $x+e^{-x}\ge 1$ for all $x\in\mathbb{R}$ in the last inequality.

Case 2a: $\sigma \ge \frac{\varepsilon}{p}$.

Then we get that (3.25) is at least

$$(1-4\varepsilon)\cdot\frac{\gamma\varepsilon}{p} \;\ge\; \frac{1}{\varepsilon} \;\ge\; \frac{e^{-2}}{\varepsilon}-1,$$

where we used $\gamma\ge 1$ and Lemma 3.18 (i) in the first inequality.

Case 2b: $\varepsilon < \sigma < \frac{\varepsilon}{p}$.

We obtain a lower bound on the connection costs as follows: Let $N := \left\lfloor \frac{\varepsilon n - 1}{2} \right\rfloor$. Note that $N \geq \frac{\gamma}{p}$ since $n \geq \frac{2\gamma}{\varepsilon p} + \frac{3}{\varepsilon}$ by Lemma 3.18 (iii). We only consider vertices v_i with $N+1 \leq i \leq n-1-N$ and connect them to a sampled customer that is at most N hops away on T^*, if such a customer exists. Otherwise, we do not connect them at all. This yields a lower bound for the expected cost of connecting the active customers to the master tour of

$$
\begin{aligned}
&\sum_{j=N+1}^{n-1-N} \sum_{i=1}^{N} 2p \cdot (1-\sigma p)^{2i-1} \cdot \mathbb{P}_{S\sim q}\left[S \cap \{v_{j-i}, v_{j+i}\} \neq \emptyset\right] \cdot c(v_j, v_{j+i}) \\
&= \frac{n-1-2N}{n} \cdot \sum_{i=1}^{N} 2p \cdot (1-\sigma p)^{2i-1} \cdot \sigma p(2-\sigma p) \cdot c_i && \mid N = \left\lfloor \frac{\varepsilon n - 1}{2} \right\rfloor \\
&\geq (1-\varepsilon) \cdot \sum_{i=1}^{N} 2p \cdot (1-\sigma p)^{2i-1} \cdot \sigma p(2-\sigma p) \cdot c_i \\
&\geq (1-\varepsilon) \cdot (2-\sigma p) \cdot (1-\sigma p) \cdot \frac{1}{2\sigma} \sum_{i=1}^{N} (2\sigma p)^2 \cdot (1-2\sigma p)^{i-1} \cdot c_i && \mid \sigma p \leq \varepsilon \\
&\geq (1-\varepsilon)^3 \cdot \frac{1}{\sigma} \cdot \sum_{i=1}^{N} (2\sigma p)^2 \cdot (1-2\sigma p)^{i-1} \cdot c_i && \mid \text{Lemma 3.19} \\
&= (1-\varepsilon)^3 \cdot \frac{1}{\sigma} \cdot \left(2\sigma\gamma + (1-2\sigma p)^{\frac{\gamma}{p}} - (1+2\sigma p N) \cdot (1-2\sigma p)^N\right) && \mid \text{Lemma 3.18 (ii)} \\
&\geq (1-\varepsilon)^3 \cdot \frac{1}{\sigma} \cdot \left(2\sigma\gamma + e^{-2\sigma\gamma} - \varepsilon - (1+2\sigma p N) \cdot (1-2\sigma p)^N\right) \\
&\geq (1-\varepsilon)^3 \cdot \frac{1}{\sigma} \cdot \left(2\sigma\gamma + e^{-2\sigma\gamma} - 2\varepsilon\right) \\
&\geq (1-\varepsilon)^3 \cdot (1-2\varepsilon) \cdot \frac{1}{\sigma} \cdot \left(2\sigma\gamma + e^{-2\sigma\gamma}\right).
\end{aligned}
$$

Note that for the penultimate inequality we used Lemma 3.18 (iv) for $x = 2\sigma p N \geq \varepsilon^2 p n - 3$ and

$$(1+x) \cdot \left(1 - \frac{x}{N}\right)^N \;\leq\; (1+x) \cdot e^{-x} \;\leq\; (1+x) \cdot e^{-p\varepsilon^2 x}.$$

The last inequality follows since $x + e^{-x} \geq 1$ for all $x \in \mathbb{R}$.

If we compute a TSP tour on the sampled customers that costs α times more than optimal, the computed master tour has expected cost at least α times (3.25). Hence, adding up the expected cost of the master tour and the expected connection cost yields at least

$$(1-\varepsilon)^3 \cdot (1-4\varepsilon) \cdot \left(\alpha(\sigma\gamma + e^{-\sigma\gamma}) + 2\gamma + \frac{1}{\sigma} e^{-2\sigma\gamma}\right).$$

□

Proof of Theorem 3.1. Putting together Lemma 3.20 and Lemma 3.21 and considering the limit $\varepsilon \to 0$, the ratio between the expected cost of the master route solution that we get from sampling and the expected cost of an optimum a priori tour is at least

$$\frac{\alpha(\sigma\gamma + e^{-\sigma\gamma}) + 2\gamma + \frac{1}{\sigma} e^{-2\sigma\gamma}}{\gamma + e^{-\gamma}}. \tag{3.26}$$

For any fixed α and γ, this ratio is minimized for the unique positive σ for which

$$\sigma^2\alpha\gamma(e^{2\sigma\gamma} - e^{\sigma\gamma}) \;=\; 1 + 2\sigma\gamma,$$

but apparently there is no closed-form solution. Optimizing this term numerically gives the bounds shown in Fig. 3.4 and Fig. 3.5. Note that due to uniform activation probabilities, the function f in the sampling algorithm is irrelevant except for the value of $f(p)$, which is completely determined by σ. □

3.4 Upper bound on the master route ratio

Our proof of the upper bound on the master route ratio (for normalized A Priori TSP instances) follows a similar line as the proof of Theorem 3.2 in Section 3.2. However, there are some important differences and further complications. In particular, it is not easy to bound the expected cost for connecting the active customers to the master tour in terms of the buckets introduced in Section 3.2.2. This will require another level of aggregation. As in Section 3.2, let β, b_0 be constants that we will choose later.

3.4.1 An optimization problem for the master route ratio

Consider a normalized instance, and let p denote the (uniform) activation probability. Let T^* be a fixed optimum a priori tour, with customers appearing in the order $v_0, v_1, \ldots, v_{n-1}$; here v_0 denotes the depot. Let $v_i := v_0$ for $i < 0$ or $i > n-1$. As in Section 3.2.1, we define for $k \in \mathbb{Z}_{\geq 1}$

$$C_k \;:=\; p^2 \cdot \sum_{j \in \mathbb{Z}} c(v_j, v_{j+k}),$$

which are nonnegative variables satisfying the triangle inequality

$$C_{i+j} \;\leq\; C_i + C_j \tag{3.27}$$

for all $i, j \geq 1$. As seen in Proposition 3.9, the expected cost of T^* is exactly

$$\sum_{i=1}^{\infty} (1-p)^{i-1} \cdot C_i. \tag{3.28}$$

We now design master route solutions. For $k \geq 1$, consider a master route solution in which the master tour contains the depot and, with some offset $h \in \{1, \ldots, k\}$, every k-th customer on T^*, i.e., $\{v_j : j \in \mathbb{Z}, j \equiv h \pmod k\}$. Note that this means that for $h \geq n$, the master tour consists only of the depot (recall that $v_j = v_0$ for $j < 0$ and $j > n-1$). We will now show that, for any fixed k, the expected cost of the best such solution is at most

$$\frac{1}{kp^2}\left(C_k + 2p \cdot \sum_{i=1}^{k-1} \min\left\{C_i,\, C_{k-i}\right\}\right). \tag{3.29}$$

If we choose the offset uniformly at random, the master tour has expected cost $\frac{C_k}{kp^2}$. Indeed, for $k < n$, this directly follows by the definition of C_k, and for $k \geq n$, the master tour has expected cost

$$\mathbb{P}[h < n] \cdot \frac{C_{n-1}}{(n-1)p^2} \;=\; \frac{C_{n-1}}{kp^2} \;=\; \frac{C_k}{kp^2}$$

as claimed.

Now we bound the expected cost of connecting the active customers to the master tour. Again we do this by considering only the following two options for each active customer v: the first sampled customer that we encounter when traversing T^* from v in either direction. Connecting to other sampled customers may be cheaper, but we ignore this again and still obtain an upper bound. Now, for offset h, we obtain an upper bound on the connection cost of

$$2p \cdot \sum_{\substack{j \in \mathbb{Z}: \\ j \equiv h \pmod{k}}} \sum_{i=1}^{k-1} \min\{c(v_j, v_{j+i}), c(v_{j+i}, v_{j+k})\}$$

since the master tour contains precisely the customers v_j with $j \equiv h \pmod{k}$. Choosing h uniformly at random, this results in a bound on the connection cost of

$$\begin{aligned} & \frac{2p}{k} \cdot \sum_{i=1}^{k-1} \sum_{j \in \mathbb{Z}} \min\{c(v_j, v_{j+i}), c(v_{j+i}, v_{j+k})\} \\ \leq\ & \frac{2p}{k} \cdot \sum_{i=1}^{k-1} \min \left\{ \sum_{j \in \mathbb{Z}} c(v_j, v_{j+i}), \sum_{j \in \mathbb{Z}} c(v_j, v_{j+k-i}) \right\} \\ =\ & \frac{2}{pk} \cdot \sum_{i=1}^{k-1} \min\{C_i, C_{k-i}\}. \end{aligned}$$

This concludes the proof that (3.29) is an upper bound on the best "equidistant" master route solution, and hence on the best master route solution per se.

So for every $k \in \mathbb{N}$, the ratio of (3.29) to (3.28) is an upper bound on the master route ratio for that instance. In other words, minimizing (3.28) subject to the constraints that (3.29) is equal to 1 and the C_i are nonnegative and satisfy the triangle inequality (3.27) yields the reciprocal of an upper bound on the master route ratio of all normalized instances with activation probability p. Note that only requiring that (3.29) is at least 1 does not change the minimum. Again, the number n of customers appears neither in (3.28) nor in (3.29). We arrive at the following optimization problem:

$$\min \sum_{i=1}^{\infty} (1-p)^{i-1} \cdot C_i \qquad \text{(Master-Route-Ratio-OP)}$$

$$\begin{aligned} \text{subject to} \quad C_i &\geq 0 && \text{for } i \in \mathbb{N} && (3.30) \\ C_i + C_j &\geq C_{i+j} && \text{for } i, j \in \mathbb{N} && (3.31) \\ C_k + 2p \cdot \sum_{i=1}^{k-1} \min\{C_i,\, C_{k-i}\} &\geq kp^2 && \text{for } k \in \mathbb{N}. && (3.32) \end{aligned}$$

We have proved:

Lemma 3.22. *Let $p > 0$. The reciprocal of the value of* (Master-Route-Ratio-OP) *is an upper bound on the master route ratio for p-normalized instances.* □

3.4.2 Obtaining a single linear program

The optimization problem (Master-Route-Ratio-OP) has infinitely many variables, but one could omit the terms for, say, $i > \frac{100}{p}$ in the infinite sum while still getting a good upper bound. One

could also omit the constraints of type (3.32) for, say, $k > \frac{100}{p}$ while still getting a good upper bound.

However, we would still have an infinite set of LPs (one for each choice of p), and, by Lemma 2.5, we should consider the limit for $p \to 0$.

In the following, exactly as in Section 3.2.2, we require that p is of the form $p = \frac{\beta}{b}$ for some odd integer $b \geq b_0$. Note that $p \to 0$ as $b \to \infty$. In order to obtain a single optimization problem for all such values of p, we again put subsequent C_i's into buckets of size b: We define

$$B_i \;\coloneqq \sum_{j=\max\{1, ib-\frac{b-1}{2}\}}^{ib+\frac{b-1}{2}} C_j \tag{3.33}$$

for $0 \leq i \leq N$ for some integer N that we will choose later. In the following, we show that we can use the constraints in (Master-Route-Ratio-OP) to generate finitely many (slightly relaxed) constraints that only depend on these buckets. Note that for all $i, j \in \mathbb{Z}_{\geq 1}$ with $i + j \leq N$ we have $B_{i+j} \leq B_i + B_j$ by Proposition 3.11.

For each $1 \leq i \leq N$, we sum the constraints of type (3.32) for all k in the i-th bucket (i.e., $k = ib - \frac{b-1}{2}, \ldots, ib + \frac{b-1}{2}$), yielding

$$B_i + 2p \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{ib+k-1} \min\left\{C_j, C_{ib+k-j}\right\} \;\geq \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} (ib+k)p^2 \;=\; i(bp)^2. \tag{3.34}$$

The left-hand side of (3.34) still contains the C_i variables, and we want to replace them by the new B_i variables.

Since it is not possible to express the minima in the left-hand side of (3.34) in terms of the buckets without an additional loss, we will need to be a bit pessimistic here. In order to not lose too much, we form bucket intervals

$$A_j \;\coloneqq \sum_{\ell=\max\{j,0\}}^{\min\{j+a,N\}} B_\ell \tag{3.35}$$

for $j = -a, \ldots, N$, each consisting of $a + 1$ subsequent buckets (except for $j < 0$ or $j > N - a$), where a is some odd integer (think of a large but abp small). Roughly speaking, the purpose of the bucket intervals is the following: When deciding to which sampled neighbor on T^* we connect the active customers, we have to make the same decision for all customers in the same bucket interval. It turns out that this way we will lose only a fraction proportional to $\frac{1}{a}$. See Fig. 3.6 for an illustration.

Then, choosing some offset $h \in \{0, \ldots, a-1\}$, we bound the double sum on the left-hand side of (3.34) as follows, where we use $C_j = 0$ for $j \leq 0$ and $j > Nb + \frac{b-1}{2}$ (i.e., C_j does not occur in

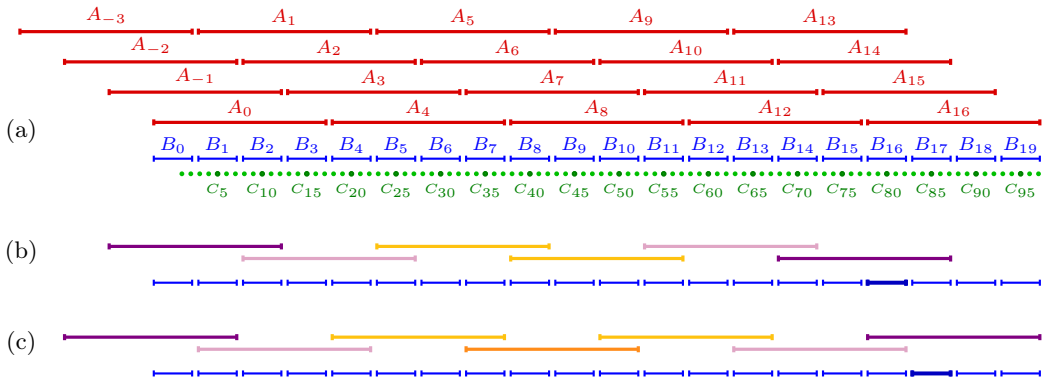


Figure 3.6: (a): The green dots stand for $C_1, C_2, \ldots$, and the centers of the buckets (C_{ib} for $i \geq 1$) are highlighted. Here the bucket size is $b = 5$, and the blue intervals show the buckets $B_0, B_1, B_2, \ldots$ We now form bucket intervals, shown in red on the top (each consisting of $a+1 = 4$ buckets; here $a = 3$; we can think of adding empty buckets on the left). (b): Aggregating the constraints (3.32) for bucket B_{16} (i.e., $k \in \{78, 79, 80, 81, 82\}$) yields (3.34) and then (3.36) for $i = 16$; here we collect the terms on the left-hand side according to the bucket intervals shown. (c): The same for bucket B_{17}.

any bucket), and $B_j = 0$ for $j < 0$ and $j > N$ in intermediate steps to simplify the terms.

$$
\begin{aligned}
&\sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{ib+k-1} \min\left\{C_j,\, C_{ib+k-j}\right\} \\
&\leq \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{ib+k-1+hb} \min\left\{C_{j-hb},\, C_{ib+k-j+hb}\right\} \\
&\leq \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{\left\lceil \frac{h+i+1}{a} \right\rceil} \sum_{\ell=1}^{ab} \min\{C_{(j-1)ab+\ell-hb},\, C_{ib+k-(j-1)ab-\ell+hb}\} \\
&\leq \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{\left\lceil \frac{h+i+1}{a} \right\rceil} \min\left\{\sum_{\ell=1}^{ab} C_{(j-1)ab+\ell-hb},\, \sum_{\ell=1}^{ab} C_{ib+k-(j-1)ab-\ell+hb}\right\} \\
&= \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{\left\lceil \frac{h+i+1}{a} \right\rceil} \min\left\{\sum_{\ell=1}^{ab} C_{(j-1)ab+\ell-hb},\, \sum_{\ell=1}^{ab} C_{ib+k-jab+\ell-1+hb}\right\} \\
&\leq \sum_{k=-\frac{b-1}{2}}^{\frac{b-1}{2}} \sum_{j=1}^{\left\lceil \frac{h+i+1}{a} \right\rceil} \min\left\{\sum_{\ell=0}^{a} B_{(j-1)a+\ell-h},\, \sum_{\ell=0}^{a} B_{i-ja+\ell+h}\right\} \\
&= b \sum_{j=1}^{\left\lceil \frac{h+i+1}{a} \right\rceil} \min\left\{A_{(j-1)a-h},\, A_{i-ja+h}\right\}.
\end{aligned}
$$

To minimize the loss, we choose $h_i = (i \mod a)$ because this implies that different buckets are counted twice for different values of i, i.e., the buckets B_{aj-h_i} $(j \geq 1)$ are counted twice, as Fig. 3.6 (b) and (c) indicate, but note that this consideration is not needed for correctness. In any case, we conclude that for any $1 \leq i \leq N$,

$$B_i + 2bp \sum_{j=1}^{\left\lceil \frac{h_i+i+1}{a} \right\rceil} \min\left\{A_{(j-1)a-h_i},\, A_{i-ja+h_i}\right\} \;\geq\; i(bp)^2 \tag{3.36}$$

is a relaxation of (3.34), and hence of (3.32) summed for $k = ib - \frac{b-1}{2}, \ldots, ib + \frac{b-1}{2}$.

Therefore, we obtain a lower bound on the value of (Master-Route-Ratio-OP) by minimizing $\sum_{i=1}^{\infty}(1-p)^{i-1} \cdot C_i$ subject to (3.35) and (3.36) and $B_{i+j} \geq B_i + B_j$ for $i, j \geq 1$ with $i + j \leq N$, and $B_i \geq 0$ for $i \geq 0$.

The objective function still contains infinitely many variables and depends on p, and we resolve this in the same way as in Section 3.2.2. Again, as in (3.12),

$$\sum_{i=1}^{\infty}(1-p)^{i-1} \cdot C_i \;\geq\; \sum_{i=0}^{\infty}(1-p)^{bi+\frac{b-1}{2}-1} \cdot B_i \;\geq\; \sum_{i=0}^{N}(1-p)^{(i+\frac{1}{2})b} \cdot B_i. \tag{3.37}$$

Moreover, $p = \frac{\beta}{b}$ and $\lim_{b\to\infty}(1-\frac{\beta}{b})^{(i+\frac{1}{2})b} = e^{-(i+\frac{1}{2})\beta}$ for all $i = 0, \ldots, N$. Again, we will use this to replace the objective function by $\sum_{i=0}^{N} e^{-(i+\frac{1}{2})\beta} \cdot B_i$ and still get an upper bound.

Finally, we omit many triangle inequalities and keep only $i \cdot B_1 \geq B_i$ for $i = 2, \ldots, N$. Moreover, we further introduce auxiliary variables for the minima in (3.36). Here is our final linear program:

$$\min \sum_{i=0}^{N} e^{-(i+\frac{1}{2})\beta} \cdot B_i \qquad \text{(Master-Route-Ratio-LP)}$$

$$\begin{aligned}
\text{subject to} \quad i \cdot B_1 \;&\geq\; B_i && \text{for } i = 2, \ldots, N && (3.38)\\
B_i + 2\beta \sum_{j=1}^{\left\lceil \frac{h_i+i+1}{a} \right\rceil} M_{j,i} \;&\geq\; i\beta^2 && \text{for } i = 1, \ldots, N && (3.39)\\
A_{(j-1)a-h_i} \;&\geq\; M_{j,i} && \text{for } i = 1, \ldots, N \text{ and } j = 1, \ldots, \left\lceil \tfrac{h_i+i+1}{a} \right\rceil && (3.40)\\
A_{i-ja+h_i} \;&\geq\; M_{j,i} && \text{for } i = 1, \ldots, N \text{ and } j = 1, \ldots, \left\lceil \tfrac{h_i+i+1}{a} \right\rceil && (3.41)\\
\sum_{\ell=\max\{i,\,0\}}^{\min\{i+a,\,N\}} B_\ell \;&=\; A_i && \text{for } i = -a, \ldots, N && (3.42)\\
B, M \;&\geq\; 0. && && (3.43)
\end{aligned}$$

We conclude:

Lemma 3.23. *Let N, a be integers with a odd and let $\beta > 0$. Let $h_i \in \{0, \ldots, a-1\}$ for $i = 1, \ldots, N$. The reciprocal of the optimum value of* (Master-Route-Ratio-LP) *is an upper bound on the master route ratio for A Priori TSP instances with depot.*

Proof. We proceed analogously to the proof of Lemma 3.14. To this end, we compare the value of (Master-Route-Ratio-LP) to the value of (Master-Route-Ratio-OP). We showed that for any feasible solution C to (Master-Route-Ratio-OP) we obtain a feasible solution (A, B, M) to (Master-Route-Ratio-LP) via (3.33) and (3.35) and $M_{j,i} = \min\{A_{(j-1)a-h_i}, A_{i-ja+h_i}\}$.

Fix $\delta > 0$. Then there exists $b_0 \in \mathbb{N}$ such that for all odd integers $b \geq b_0$

$$\sum_{i=0}^{N} e^{-(i+\frac{1}{2})\beta} \cdot B_i \;\leq\; (1+\delta) \cdot \sum_{i=0}^{N} \left(1-\tfrac{\beta}{b}\right)^{(i+\frac{1}{2})b} \cdot B_i \;\leq\; (1+\delta) \cdot \sum_{i=0}^{\infty} \left(1-\tfrac{\beta}{b}\right)^{i-1} \cdot C_i.$$

Thus we conclude that the value of (Master-Route-Ratio-LP) is at most $(1+\delta)$ times the value of (Master-Route-Ratio-OP) with $p = \frac{\beta}{b}$ for all odd integers $b \geq b_0$. Hence, by Lemma 3.10, $(1+\delta)$ times the reciprocal of the value of (Master-Route-Ratio-LP) is an upper bound on the master route ratio for all $\frac{\beta}{b}$-normalized instances for all odd integers $b \geq b_0$. By Lemma 2.5, the same bound then holds for all instances with depot. Since this bound holds for all $\delta > 0$, it also holds for $\delta = 0$. □

3.4.3 Solving the dual LP

Now we dualize the LP (Master-Route-Ratio-LP). For this, we introduce variables $(x_i)_{i=2,\dots,N}$ for the inequalities of type (3.38), variables $(y_i)_{i=1,\dots,N}$ for the inequalities of type (3.39), variables $v_{i,j}$ and $w_{i,j}$ for the inequalities of type (3.40) and (3.41), respectively, and variables $(z_i)_{i=-a,\dots,N}$ for the inequalities of type (3.42).

$$\max \sum_{i=1}^{N} i\beta^2 \cdot y_i \qquad \text{(Dual-Master-Route-Ratio-LP)}$$

$$\begin{aligned}
\text{subject to} \quad y_i + \sum_{j=i-a}^{i} z_j + \mathbb{1}_{i=1} \sum_{j=2}^{N} j \cdot x_j - \mathbb{1}_{2\leq i\leq N} \cdot x_i \;&\leq\; e^{-(i+\frac{1}{2})\beta} && \text{for } i = 0,\dots,N \\
2\beta \cdot y_i \;&\leq\; v_{j,i} + w_{j,i} && \text{for } i = 1,\dots,N \\
& && \text{and } j = 1,\dots,\left\lceil \tfrac{h_i+i+1}{a} \right\rceil \\
\sum_{i=1}^{N} \sum_{j=1}^{\left\lceil \frac{h_i+i+1}{a} \right\rceil} \left(\mathbb{1}_{k=(j-1)a-h_i} \cdot v_{j,i} + \mathbb{1}_{k=i-ja+h_i} \cdot w_{j,i}\right) \;&=\; z_k && \text{for } k = -a,\dots,N \\
x, y, v, w \;&\geq\; 0.
\end{aligned}$$

Corollary 3.24. *Let N, a be integers with a odd and let $\beta > 0$. Let $h_i \in \{0,\dots,a-1\}$ for $i = 1,\dots,N$. For any feasible solution (x, y, v, w, z) to* (Dual-Master-Route-Ratio-LP)*, $\left(\sum_{i=1}^{N} i\beta^2 \cdot y_i\right)^{-1}$ is an upper bound on the master route ratio restricted to A Priori TSP instances with depot.* □

We have computed a dual solution using Gurobi 10.0.1 with $\beta = \frac{1}{400}$, $a = 199$, $N = 4000$, and $h_i = (i \mod a)$ for $i = 1,\dots,N$, yielding an upper bound of 2.584 and thus proving Theorem 3.5. The dual solution and a Python script that verifies that this is indeed a feasible solution to (Dual-Master-Route-Ratio-LP) can be found at `https://doi.org/10.60507/FK2/JCUIRI`.

3.5 Discussion

We conjecture (but could not prove) that our lower bound examples are really worst-case examples, and that the values of our linear programs converge to these bounds.

Another question is whether the master route ratio is $\frac{1}{1-e^{-1/2}}$ even for low-activity instances, i.e., instances for which the expected number of active customers is small. Currently we only know the upper bound of 3 from [ST08], but know no example with master route ratio larger than $\frac{1}{1-e^{-1/2}}$ (and this value is attained by our example only as the expected number of active customers tends to infinity). The analogous question applies to the sampling algorithm: whether we need to consider the low-activity case separately is an open question.

Finally, we hope that our approach can also help for proving a better bound for related problems where similar random sampling techniques are used (e.g., those referred to in Section 2.2), or for showing that known bounds are best possible.

Chapter 4

Approximating Asymmetric A Priori TSP Beyond the Adaptivity Gap

This is joint work with Manuel Christalla and Vera Traub. Apart from Section 4.6, it is based on [CPT26]. While the reduction we present here in Section 4.6 is deterministic, [CPT26] gives a randomized reduction in this place.

4.1 Introduction

4.1.1 Our contribution

We provide the first nontrivial approximation algorithms for the Asymmetric A Priori TSP. Observe that any cycle on V is an n-approximation. We show that one can obtain an $\mathcal{O}(\sqrt{n})$-approximation, even when comparing to the optimal a posteriori tour. This also implies an upper bound on the adaptivity gap (2.5).

Theorem 4.1. *There is a deterministic polynomial-time algorithm that for any instance (V, c, p) of Asymmetric A Priori TSP, computes a tour T with*

$$\mathbb{E}_{A\sim p}\Big[c\big(T[A]\big)\Big] \;\leq\; O\big(\sqrt{n}\big)\cdot \mathbb{E}_{A\sim p}\Big[\mathrm{TSP(A,c)}\Big],$$

where $n = |V|$ denotes the number of vertices. In particular, the adaptivity gap of the Asymmetric A Priori TSP is $\mathcal{O}(\sqrt{n})$.

Moreover, we prove a polynomial lower bound on the adaptivity gap of Asymmetric A Priori TSP.

Theorem 4.2. *The adaptivity gap of the Asymmetric A Priori TSP is $\Omega(n^{1/4}\cdot \log^{-1} n)$, where n denotes the number of vertices.*

This shows that one cannot achieve subpolynomial approximation factors when comparing solely to the optimal a posteriori tour (even when allowing exponential running time). Our main contribution is an algorithm with a poly-logarithmic approximation factor and quasi-polynomial running time:

Theorem 4.3. *There is a deterministic algorithm with running time $n^{\mathcal{O}(\log n)}$ that given an instance (V, c, p) of Asymmetric A Priori TSP, computes a tour T such that*

$$\mathbb{E}_{A\sim p}\Big[c\big(T[A]\big)\Big] \;\leq\; \mathcal{O}(\log^8 n) \cdot \mathrm{OPT}(V, c, p),$$

where $n = |V|$ denotes the number of vertices.

Together with Theorem 4.2, this shows that we can achieve an approximation factor below the adaptivity gap in quasi-polynomial time. We remark that we did not optimize the exponent of the $\log n$ in the approximation ratio.

In order to prove Theorem 4.3, we provide a series of polynomial-time reductions, leading to a covering problem, and eventually to a problem of finding a path in an acyclic graph minimizing a particular objective function. We emphasize that our algorithm for (approximately) solving the latter problem is the only part that is not polynomial-time, but requires quasi-polynomial running time. Our proofs even imply that the covering problem to which we reduce is equivalent to Asymmetric A Priori TSP up to poly-logarithmic factors in the approximation guarantee and polynomial factors in the running time (see Section 4.9).

Although simple $\mathcal{O}(\log n)$-approximation algorithms for ATSP are known (see e.g. [FGM82; Blä08]), none of them seems to extend to the Asymmetric A Priori TSP. We therefore do not directly generalize any of these algorithms, but develop a different approach.

We introduce a natural generalization of ATSP, where instead of paying the distance of each vertex to its direct successor on the tour, we pay the distance of each vertex to the k vertices succeeding it. Here, k is part of the input and could be large (e.g. $k \approx \sqrt{n}$). We call this problem Hop-ATSP and provide a reduction from Asymmetric A Priori TSP to Hop-ATSP.

In order to find a poly-logarithmic approximation for Hop-ATSP, we use directed low-diameter decompositions [BNW22; BFHL25; Li25], which played an important role in recent breakthroughs in the context of fast graph algorithms, including [BNW22; BCF23]. Directed low-diameter decompositions allow us to assume that our instances of Hop-ATSP have a very particular structure. Exploiting this, we then provide a reduction of Hop-ATSP to a covering problem, which is the main technical challenge in our series of reductions.

In order to solve this covering problem via the Set Cover greedy algorithm (or other well-known Set Cover algorithms), we then need an oracle for finding a "best" set to pick next. This problem turns out to be a problem of finding a path in a directed acyclic graph minimizing a particular objective function. While directly minimizing this objective function in a dynamic programming approach leads to an exponential running time, we show that a suitable approximation of it can be minimized in quasi-polynomial time.

4.1.2 Outline

In Section 4.2 we expand on the techniques we use to prove our main result, Theorem 4.3. We then start with the simple $\mathcal{O}(\sqrt{n})$-approximation algorithm, proving Theorem 4.1 in Section 4.3. In doing so, we provide an upper bound on the adaptivity gap. Next, in Section 4.4, we prove our lower bound on the adaptivity gap of Asymmetric A Priori TSP (Theorem 4.2). In Section 4.5 we explain our reduction to Hop-ATSP. Sections 4.6 and 4.7 contain our reduction of Asymmetric A Priori TSP to a covering problem. Section 4.8 shows how we can solve the covering problem approximately in quasi-polynomial time, and Section 4.9 contains some further discussion.

4.2 Outline of the Proof of Theorem 4.3

In this section we provide an overview of the proof of our main result (Theorem 4.3). As a first step, we reduce to a natural generalization of ATSP which we call Hop-ATSP (Section 4.2.1). Using directed low diameter decompositions, we then show that we may assume that our instances of Hop-ATSP have a very particular structure (Section 4.2.2). Next, we show that for these structured instances, which we call hierarchically ordered instances, the problem of Hop-ATSP can be reduced to a covering problem (Section 4.2.3). In order to solve the covering problem, e.g. via the Set Cover greedy algorithm, we then need an approximation algorithm for finding a path in an acyclic digraph minimizing a particular objective function. In Section 4.2.4 we describe how to achieve this by a dynamic program in quasi-polynomial time.

4.2.1 Reducing to Hop-ATSP

Recall that in Asymmetric TSP, the task is to compute a tour T, i.e. a Hamiltonian cycle on V, that minimizes the cost $c(T)$. By the triangle inequality, every closed walk visiting all vertices in V can be cut short to a Hamiltonian cycle on V without increasing the cost. Thus, we will also allow such walks as solutions for ATSP and we will also call them *tours*.

If a closed walk T visits the vertices $v_1, v_2, \dots, v_{r+1} = v_1$ in this order, then $c(T) \coloneqq \sum_{i=1}^{r} c(v_i, v_{i+1})$. We consider a generalization of ATSP, where we are given a hop-distance $k \in \mathbb{N}$ and the task is to compute a closed walk T visiting all vertices in V while minimizing the k-hop cost

$$c^{(k)}(T) \coloneqq \sum_{\Delta=1}^{k} \sum_{i=1}^{r} c(v_i, v_{i+\Delta}), \tag{4.1}$$

where $v_{r+l} \coloneqq v_l$ for all $l \in \mathbb{N}$. In other words, instead of paying only for the distance of every vertex to its successor on the walk, we pay the distance of every vertex to the k vertices succeeding it on the walk. As we will see in Lemma 4.21, skipping vertices in a closed walk can only decrease its k-hop cost (similarly to how we can take shortcuts using the triangle inequality when considering the usual cost $c(T)$). This justifies also calling closed walks *tours* in the context of Hop-ATSP:

Definition 4.4 (Hop-ATSP)**.** *An instance of Hop-ATSP consists of*

- *a finite vertex set V,*
- *a cost function $c\colon V \times V \to \mathbb{R}_{\geq 0}$ satisfying the triangle inequality, and*
- *a hop-distance $k \in \mathbb{N}$ with $k < |V|$.*

The task in Hop-ATSP is to compute a tour T on V minimizing $c^{(k)}(T)$.

We call an instance $\mathcal{I} = (V, c, k)$ of Hop-ATSP *well-scaled* if the cost function c takes only values in $\{0, 1, \dots, 2n^3\}$ and $\mathrm{OPT}(V, c, k) \geq n^2$, where $n \coloneqq |V|$. In Section 4.5, we provide a reduction of Asymmetric A Priori TSP to well-scaled instances of Hop-ATSP.

Theorem 4.5. *Let $\alpha\colon \mathbb{N} \to \mathbb{R}_{\geq 1}$ and $t\colon \mathbb{N} \to \mathbb{N}$ monotonically increasing. Suppose we have an algorithm that computes for every well-scaled instance $\mathcal{I} = (V, c, k)$ of Hop-ATSP in time $t(n)$ a tour T with $c^{(k)}(T) \leq \alpha(n) \cdot \mathrm{OPT}(\mathcal{I})$, where $n = |V|$.*

Then there is an algorithm that computes for every instance $\mathcal{J} = (V, c, p)$ of the Asymmetric A Priori TSP in time $n \cdot t(\mathrm{poly}(n)) + \mathrm{poly}(n)$ a tour T with

$$\mathbb{E}_{A \sim p}\big[c(T[A])\big] \;\leq\; \mathcal{O}(\alpha(5n^2)) \cdot \mathrm{OPT}(\mathcal{J}),$$

where $n = |V|$ and $\mathrm{poly}(n)$ denotes some polynomial in n.

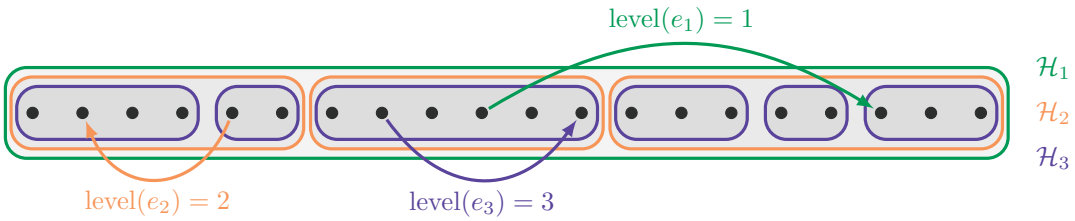


Figure 4.1: An example of a hierarchical partition $\mathcal{H} = (\mathcal{H}_\ell)_{\ell=1}^4$, where $\mathcal{H}_4 = \{\{v\} : v \in V\}$ is not shown. All vertices are drawn from left to right according to the total order $\prec$. The edges e_1 and e_3 are forward edges, while e_2 is a backward edge.

In particular, if $\alpha(n) = \log^q n$ for some constant q, then we achieve an approximation ratio of $\mathcal{O}(\log^q n)$ for the Asymmetric A Priori TSP. We remark that throughout this chapter, we did not make any efforts to optimize constants.

In order to prove Theorem 4.5, we use Proposition 2.3 and assume that all vertices v have the same activation probability $p(v) = \delta$. Then we observe that for this special case of uniform activation probabilities, the objective function of Asymmetric A Priori TSP,

$$\mathbb{E}_{A\sim p}\left[c\left(T[A]\right)\right] = \sum_{\Delta=1}^{r-1}\sum_{i=1}^{r} \delta^2(1-\delta)^{\Delta-1} \cdot c(v_i, v_{i+\Delta})$$

(cf. (2.15) and (3.4)), and the objective function of Hop-ATSP (4.1) scaled by a factor $\frac{1}{k^2}$ differ by at most a constant factor when choosing $k = \lfloor \frac{1}{\delta} \rfloor$. For details we refer to Section 4.5.

4.2.2 Reducing to hierarchically ordered instances

We now focus on approximation algorithms for Hop-ATSP. We show that, at the cost of a poly-logarithmic factor in the approximation ratio, we may assume that the instance has a very particular structure and we will call such structured instances *hierarchically ordered*.

In a hierarchically ordered instance we have a total order $\prec$ on the vertices. Moreover, the instance has some depth $L \in \mathbb{N}$ and we have a partition $\mathcal{H}_\ell$ of V for each level $\ell \in [L]$, where

- $\mathcal{H}_1 = \{V\}$ is the trivial partition and $\mathcal{H}_L = \{\{v\} : v \in V\}$ is the partition of V into singletons,
- every partition $\mathcal{H}_\ell$ (with $\ell \geq 2$) is a refinement of the partition $\mathcal{H}_{\ell-1}$, i.e. for every set $H \in \mathcal{H}_\ell$ there exists a set $H' \in \mathcal{H}_{\ell-1}$ with $H \subseteq H'$, and
- every set $H \in \mathcal{H}_\ell$ for some $\ell \in [L]$ is a set of vertices appearing consecutively in the total order $\prec$.

See Figure 4.1 for an illustration.

For an edge $e = (v, w)$ we define level(e) to be the maximal $\ell \in [L]$ such that both endpoints of e belong to the same element of the partition $\mathcal{H}_\ell$. We call e a *forward edge* if $v \prec w$ and a *backward edge* otherwise. In a hierarchically ordered instance, we have a maximum edge cost D_ℓ for each level $\ell \in [L]$, with the property that $D_\ell \geq 2 \cdot D_{\ell+1}$ for each $\ell \in [L-1]$. The cost function c satisfies

- $c(e) \leq D_{\text{level}(e)}$ if e is a forward edge, and
- $c(e) = D_{\text{level}(e)}$ if e is a backward edge.

We will denote a hierarchically ordered instance as a tuple $(V, L, \mathcal{H}, D, \prec, c, k)$ with the hierarchical partition $\mathcal{H} = \big(\mathcal{H}_\ell\big)_{\ell=1}^{L}$ and the maximum edge costs $D = \big(D_\ell\big)_{\ell=1}^{L}$. We provide the following reduction of general instances of Hop-ATSP to hierarchically ordered instances:

Theorem 4.6. *Given a well-scaled instance $\mathcal{I} = (V, c, k)$ of Hop-ATSP with $n = |V|$ vertices, we can in polynomial time compute a polynomial number of hierarchically ordered instances $\mathcal{J}_1, \dots, \mathcal{J}_m$, where for each $i \in [m]$ we have $\mathcal{J}_i = (V, L, \mathcal{H}^{(i)}, D, \prec_i, \tilde{c}_i, k)$ with $L = \mathcal{O}(\log n)$ and*

$$\tilde{c}_i(v, w) \geq c(v, w) \text{ for all } v, w \in V.$$

Moreover, we require that there exists an index $i^* \in [m]$ such that

$$\mathrm{OPT}(\mathcal{J}_{i^*}) \leq L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot \mathrm{OPT}(\mathcal{I}).$$

With only a polynomial time overhead in the running time, we can compute a solution for each instance $\mathcal{J}_1, \dots, \mathcal{J}_m$ given by Theorem 4.6, and return the best one. Therefore, in order to find a poly-logarithmic approximation for Hop-ATSP, it suffices to consider hierarchically ordered instances with depth $\mathcal{O}(\log n)$.

In order to prove Theorem 4.6, we repeatedly apply directed low-diameter decompositions. See Section 4.6 for details.

4.2.3 Reducing to a covering problem

Next, we explain how we reduce hierarchically ordered instances of Hop-ATSP to a covering problem. To explain the main ideas, we first focus on the special case $L = 2$. In this case, we have only two partitions $\mathcal{H}_1 = \{V\}$ and $\mathcal{H}_2 = \{\{v\} : v \in V\}$. Every forward edge has cost at most D_1 and every backward edge has cost exactly D_1. We provide a reduction to a covering problem, where we have to cover the vertices in V by *monotone paths*.

Definition 4.7 (Monotone path)**.** *A path P is called* monotone *if it contains only forward edges.*

To define our covering problem, we extend the definition of the k-hop costs $c^{(k)}$ from tours to paths. For a path P with vertices $v_1, \dots, v_r$ visited in this order, we define

$$c^{(k)}(P) = \sum_{\Delta=1}^{k} \sum_{i=1}^{r} c(v_i, v_{i+\Delta}),$$

where we define $c(v_i, v_l) \coloneqq 0$ for $l > r$. Then we consider the covering problem of finding a set $\mathcal{P}$ of monotone paths with $V \subseteq \bigcup_{P \in \mathcal{P}} V(P)$ minimizing the weight

$$w(\mathcal{P}) \;\coloneqq\; \sum_{P \in \mathcal{P}} w(P),$$

where $w(P) \coloneqq c^{(k)}(P) + k^2 \cdot D_1$. The definition of the weight $w(P)$ is chosen to ensure the following key properties:

(i) any solution $\mathcal{P}$ to this covering problem can be converted into a tour T with $c^{(k)}(T) \leq w(\mathcal{P})$, and

(ii) any tour T can be converted into a feasible covering $\mathcal{P}$ satisfying $w(\mathcal{P}) \leq \gamma \cdot c^{(k)}(T)$, where $\gamma \geq 1$ is a constant independent of the Hop-ATSP instance.

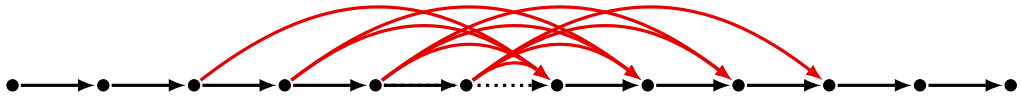


Figure 4.2: If we concatenate the two paths P_1 and P_2 (black, solid) by adding the dotted edge to obtain a path P, the k-hop cost $c^{(k)}(P)$ of the new path P is the sum of $c^{(k)}(P_1) + c^{(k)}(P_2)$ and the cost $c(e)$ of all edges e shown in red, where in this example $k = 4$.

In order to prove (i), we construct a tour T by concatenating the paths from $\mathcal{P}$ in an arbitrary order. Observe that whenever we concatenate two paths P_1 and P_2 to a new path P, we have

$$c^{(k)}(P) \leq c^{(k)}(P_1) + c^{(k)}(P_2) + k^2 \cdot D_1,$$

because $c(v, w) \leq D_1$ for any $v, w \in V$. See Figure 4.2. This implies $w(P) \leq w(P_1) + w(P_2)$. Iteratively applying this observation, we obtain a single path $\hat{P}$ visiting all vertices with $w(\hat{P}) \leq \sum_{P \in \mathcal{P}} w(P)$. If we turn $\hat{P}$ into a tour T by returning to the first vertex of the path $\hat{P}$ once we reached its last vertex, we get $c^{(k)}(T) \leq c^{(k)}(\hat{P}) + k^2 \cdot D_1 = w(\hat{P})$. This shows (i).

The more difficult property to prove is (ii). We start by some simple observations. Using $k < |V|$, one can show $c^{(k)}(T) \geq \Omega(k^2) \cdot D_1$ for any tour T (see Lemma 4.41). Moreover, we may assume that the tour T visits every vertex exactly once (see Lemma 4.21). Then we turn the tour T into a path P by removing an arbitrary edge and observe that $w(P) \leq c^{(k)}(T) + k^2 \cdot D_1 \leq \mathcal{O}(1) \cdot c^{(k)}(T)$.

The key part of the proof of (ii) is to show that we can transform P into a collection of monotone paths covering all vertices without increasing the total weight by more than a constant factor. First, we will ensure that we have at least k vertices between any two backward edges of P. To this end, we partition the vertices of P into intervals of consecutive vertices containing k vertices each (where the last interval may contain fewer vertices). Then we change the order of the vertices in each interval by sorting them according to the total order $\prec$. Using that c satisfies the triangle inequality, one can show that this increases $c^{(k)}(P)$ by at most a constant factor. After this reordering, backward edges can only appear at the boundaries of the sorted intervals and thus we indeed have at least k vertices between any two backward edges of P.

Next, we explain how we get rid of the remaining backward edges. We will maintain a collection $\mathcal{P}$ of paths, where in each path we have at least k vertices between any two backward edges. Now consider a backward edge e contained in a path $P \in \mathcal{P}$. For simplicity, assume k to be even and let $v_1, \ldots, v_{k/2}$ be the $k/2$ vertices that P visits before e, and let $w_1, \ldots, w_{k/2}$ be the $k/2$ vertices that P visits after e in this order.[1] See Figure 4.3. When considering the backward edge e, we will make sure to charge any increase of $w(\mathcal{P})$ to the k-hop cost $c^{(k)}(P^e)$ of the subpath P^e of P with vertices $V^e = \{v_1, \ldots, v_{k/2}, w_1, \ldots, w_{k/2}\}$. This ensures that the weight increase for removing different backward edges is charged against disjoint subpaths of P. Note that because P^e has only k vertices, $c^{(k)}(P^e)$ is the sum of all costs $c(x, y)$ where $x \in V^e$ is visited before $y \in V^e$ by the path P^e.

Observe that $v_1 \prec v_2 \prec \ldots v_{k/2}$ and $w_1 \prec \cdots \prec w_{k/2}$. We distinguish two cases. If $w_{k/2} \prec v_1$, then we split P at the backward edge e into two paths. Because in this case every edge $v_i \prec w_j$ with $i, j \in \left[\frac{k}{2}\right]$ is a backward edge, $c^{(k)}(P^e)$ is at least $\left(\frac{k}{2}\right)^2 \cdot D_1$. Hence, we can indeed charge the weight increase of $k^2 \cdot D_1$ caused by the splitting of P into two paths against $c^{(k)}(P^e)$.

[1] We assume here for simplicity, that P indeed has $\frac{k}{2}$ vertices after the backward edge.

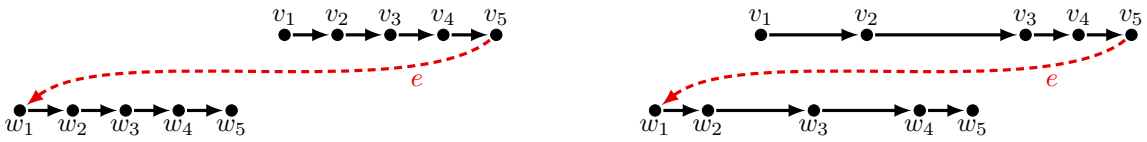


Figure 4.3: Illustration of the path P^e for a backward edge e. Vertices are drawn from left to right according to the total order $\prec$. We distinguish the two cases $w_{k/2} \prec v_1$ (left) and $v_1 \prec w_{k/2}$ (right).

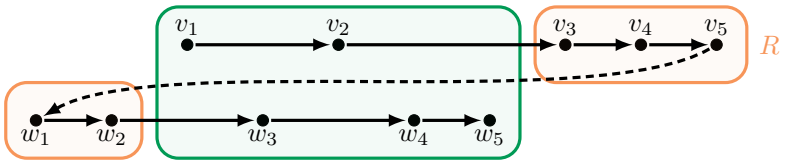


Figure 4.4: Illustration of the case $v_1 \prec w_{k/2}$, where the vertices are drawn from left to right according to the total order $\prec$. We remove the vertices in R (orange) and reorder the remaining vertices of P^e (shown in green) according to $\prec$, i.e., in this example in the order v_1, w_3, v_2, w_4, w_5.

Now consider the remaining case, where we have $v_1 \prec w_{k/2}$, illustrated in Figure 4.4. Then we remove all vertices v_i with $i \in \left[\frac{k}{2}\right]$ and $w_{k/2} \prec v_i$ from the path and put them into a remainder set R, which is initially empty. Similarly, we remove all vertices w_i with $w_i \prec v_1$ from the path and add them to R. Note that v_1 and $w_{k/2}$ were not removed. Now we sort all vertices from V^e that were not removed according to the total order $\prec$. Because these vertices $u \in V^e$ that were not removed satisfy $v_1 \preceq u \preceq w_{k/2}$, this reordering does not introduce any new backward edges.

We will charge the increase of $w(\mathcal{P}) + |R| \cdot 2k \cdot D_1$ against $c^{(k)}(P^e)$ and will later show that we can re-insert every vertex from R into some path while increasing the cost of the path by at most $2k \cdot D_1$. To see that we can indeed charge the increase of $w(\mathcal{P}) + |R| \cdot 2k \cdot D_1$ against $c^{(k)}(P^e)$, we observe that every vertex in R has at least $\frac{k}{2}$ incident edges of cost D_1 that contribute to $c^{(k)}(P^e)$. Moreover, using that c satisfies the triangle inequality, one can show that removing the vertices in R does not increase the k-hop costs $c^{(k)}(P)$ of our path and the reordering of the vertices in $V^e \setminus R$ can increase the cost by at most some constant factor times $c^{(k)}(P^e)$.

It remains to show how to deal with the removed vertices in R. After having considered each backward edge e, all paths in $\mathcal{P}$ are monotone. We insert each vertex from R into an arbitrary path $P \in \mathcal{P}$, maintaining monotonicity. This insertion increases $c^{(k)}(P)$ by at most $2k \cdot D_1$. Doing this for all vertices in R, we obtain a collection $\mathcal{P}'$ of monotone paths covering all vertices with $w(\mathcal{P}') \leq w(\mathcal{P}) + 2k \cdot D_1 \cdot |R|$. This shows (ii).

Recall that so far we only considered the special case $L = 2$. In the general case, we consider a more involved covering problem, taking into account the hierarchical structure of the instance. We will again aim at covering V by monotone paths, but now we will in addition assign a level to every such path we include in our covering.

Definition 4.8 (Path-level pair)**.** *A* path-level pair *is a pair (P, ℓ) where P is a monotone path, $\ell \in [L]$, and $V(P) \subseteq H$ for some $H \in \mathcal{H}_\ell$.*

A feasible solution to our covering problem will now consist of such path-level pairs. In order to be able to turn a solution to our covering problem into a solution to Hop-ATSP without

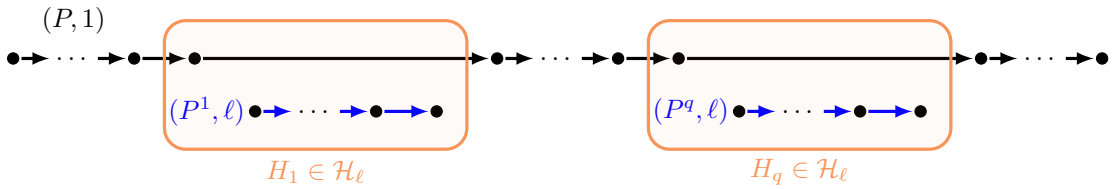


Figure 4.5: Example illustrating why we need condition (b) in Problem 4.9. Suppose that all edges in $\delta(H_i)$ with $i \in [q]$ have cost D_1, and all other forward edges have cost 0. Consider the path-level pairs $(P, 1)$ and $(P^1, \ell), \dots, (P^q, \ell)$ shown in the picture, where each path P^i has k vertices. The path P visits exactly one vertex from each set H_i and it visits k vertices in between any two such vertices. Then $c^{(k)}(P) = 2qk \cdot D_1$ and $c^{(k)}(P^i) = 0$ for $i \in [q]$, and thus $w(P, 1) + \sum_{i=1}^{q} w(P^i, \ell) = k^2 \cdot D_1 + 2qk \cdot D_1 + qk^2 \cdot D_\ell$.
On the other hand, let T be a cycle covering all vertices. For each $i \in [q]$ and each $v \in H_i$, at most k of the $2k$ vertices preceding and succeeding v in T are part of the same H_i. The other k vertices preceding and succeeding v in T incur a cost of at least D_1 each; hence v incurs a cost of at least $k \cdot D_1$. Since we count edges at most twice when summing up over all $v \in H_i$ for all $i \in [q]$, any tour T covering all vertices has k-hop cost $c^{(k)}(T) \geq \frac{1}{2} \cdot q \cdot k^2 \cdot D_1$, which is much larger than $w(P, 1) + \sum_{i=1}^{q} w(P^i, \ell)$ in case D_1 is much larger than D_ℓ and q, k are sufficiently large.

increasing the objective value significantly (analogous to (i) in the case $L = 2$), we define our covering problem as follows.

Problem 4.9: Path Covering

Given a hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP, find a set $\mathcal{P}$ of path-level pairs (P, ℓ) such that

(a) for every vertex $v \in V$ there is some pair $(P, \ell) \in \mathcal{P}$ with $v \in V(P)$, and

(b) for every level $\ell \in \{2, 3, \dots, L\}$ and every set $H \in \mathcal{H}_\ell$, one of the following applies:

  (i) for every vertex $v \in H$ there is a pair $(P, j) \in \mathcal{P}$ with $j < \ell$ and $v \in V(P)$, or
  (ii) there is a pair $(P, j) \in \mathcal{P}$ with $j < \ell$ and $|V(P) \cap H| \geq k$,

while minimizing the weight

$$w(\mathcal{P}) := \sum_{(P,\ell) \in \mathcal{P}} w(P, \ell) = \sum_{(P,\ell) \in \mathcal{P}} \left(c^{(k)}(P) + k^2 \cdot D_\ell \right).$$

We highlight that property (b) is necessary in order to be able to transform $\mathcal{P}$ into a tour T with $c^{(k)}(T) = \mathcal{O}(1) \cdot w(\mathcal{P})$, as the example in Figure 4.5 shows.

In Section 4.7 we show that any α-approximation algorithm for Problem 4.9 leads to an $\mathcal{O}(L^3 \cdot \alpha)$-approximation algorithm for hierarchically ordered instances of Hop-ATSP, with a polynomial overhead in the running time. The main ideas we use to prove this are similar to the special case $L = 2$, but we need a much more careful recursive argument. In particular, we cannot afford to increase the objective value L times by a constant factor γ when transforming a tour into a solution to Problem 4.9 and some care is needed to avoid this. We refer to Section 4.7 for details.

4.2.4 Solving the covering problem via an algorithm for monotone paths

It remains to provide an approximation algorithm for our covering problem. For simplicity, in this outline we again focus on the special case $L = 2$. Recall that in our covering problem, the task is to find a set $\mathcal{P}$ of monotone paths with $V \subseteq \bigcup_{P\in\mathcal{P}} V(P)$ minimizing the weight $w(\mathcal{P}) = \sum_{P\in\mathcal{P}} w(P)$, where $w(P) = c^{(k)}(P) + k^2 \cdot D_1$. We will use the greedy algorithm for Set Cover. It provides an $\mathcal{O}(\alpha \cdot \log n)$-approximation for our covering problem, assuming that we have an α-approximation algorithm for the following problem of finding a monotone path minimizing a particular objective function. Given a hierarchically ordered instance of Hop-ATSP (with $L = 2$) and a set $S \subsetneq V$ of already covered vertices, the task is to find a monotone path P minimizing

$$\frac{w(P)}{|V(P) \setminus S|} \;=\; \frac{c^{(k)}(P) + k^2 \cdot D_1}{|V(P) \setminus S|}, \tag{4.2}$$

which we define to be ∞ for $V(P) \setminus S = \emptyset$.

In order to find such a good path, we will use a dynamic program. A key challenge here is that a naive realization of this approach would lead to a running time of $\Omega(n^k)$. In order to avoid this, we will not directly try to minimize the objective function (4.2), but instead consider some approximation of it. More precisely, for each offset $q \in \{0, 1, \ldots, k\}$ and every monotone path P, we define a weight bound $w^q(P) \geq w(P)$. These weight bounds will have the following key properties:

(i) for every monotone path P and every offset $q \in \{0, 1, \ldots, k\}$, we have $w^q(P) \geq w(P)$,

(ii) for every monotone path P, there exists an offset $q \in \{0, 1, \ldots, k\}$ such that $w^q(P) = \mathcal{O}(\log(k)) \cdot w(P)$, and

(iii) we can find a monotone path P and an offset q minimizing $\frac{w^q(P)}{|V(P)\setminus S|}$ in quasi-polynomial time.

From these properties, it follows immediately that we have a quasi-polynomial-time $\mathcal{O}(\log k)$-approximation algorithm for finding a monotone path minimizing (4.2), and thus a quasi-polynomial-time $\mathcal{O}(\log k \cdot \log n)$-approximation algorithm for our covering problem.

Let us now explain how we define the weight bound $w^q(P)$. See Figure 4.6 for an illustration. For simplicity of the explanation, we will assume $k = 2^\Gamma - 1$ for some $\Gamma \in \mathbb{N}$. We number the vertices of P as $v_0, \ldots, v_r$ and assign a height to every vertex $v_i \in V(P)$. We define height(i) to be the maximum $h \in \{0, 1 \ldots, \Gamma\}$ such that i is divisible by 2^h. For each vertex $v_i \in V(P)$ the distance from the $2^{\text{height}(i+q)} - 1$ vertices preceding v on P and the distance to the $2^{\text{height}(i+q)} - 1$ vertices succeeding v on P will contribute to the weight bound $w^q(P)$. More precisely,

$$w^q(P) \;:=\; k^2 \cdot D_1 + \sum_{i=0}^{r} \sum_{\Delta=1}^{2^{\text{height}(i+q)}-1} 2^{\text{height}(i+q)} \cdot \Big(c(v_{i-\Delta}, v_i) + c(v_i, v_{i+\Delta})\Big)$$

Using that c satisfies the triangle inequality, one can show (i). In order to prove (ii), we show that choosing $q \in \{0, 1, \ldots, k\}$ uniformly at random leads to $\mathbb{E}\big[w^q(P)\big] = \mathcal{O}(\Gamma) \cdot w(P) = \mathcal{O}(\log(k)) \cdot w(P)$. To prove (iii), we use a dynamic program. We highlight that this dynamic program is the only part of our algorithm that does not have a polynomial runtime. For details of how we prove these properties and how we extend these arguments to the general case of arbitrary L, we refer to Section 4.8.

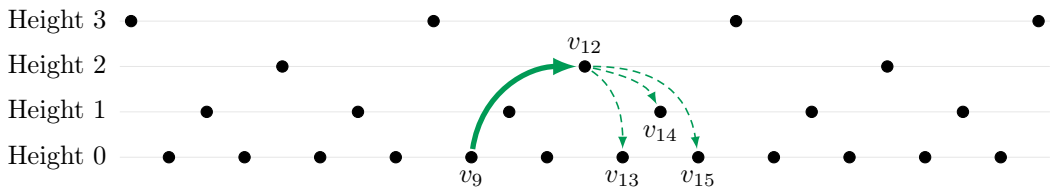


Figure 4.6: An example illustrating the weight bound $w^q(P)$. In this example, we have $q = 0$, $\Gamma = 3$, and $k = 2^3 - 1 = 7$. The vertices $v_0, \ldots, v_{24}$ are drawn from left to right, and each vertex v_i is vertically positioned according to height(i). Suppose that P visits these vertices in this order. To prove that the weight bound is an upper bound on $w(P)$, we use the triangle inequality, bounding the cost $c(v_i, v_j)$ of an edge contributing to $c^{(k)}(P)$ by $c(v_i, v_m) + c(v_m, v_j)$, where we choose $m \in \{i, \ldots, j\}$ maximizing height(m). For example, the edges (v_9, v_j) for $j \in \{12, \ldots, 15\}$ contribute to $c^{(k)}(P)$. Our weight bound $w^0(P)$ bounds the contribution of these edges to $c^{(k)}(P)$ via the triangle inequality: $c(v_9, v_j) \leq c(v_9, v_{12}) + c(v_{12}, v_j)$ for $j \in \{12, \ldots, 15\}$. Note that $c(v_9, v_{12})$ indeed contributes to $w^0(P)$ with a factor of $2^{\text{height}(12)} = 4$.

4.3 A simple $\mathcal{O}(\sqrt{n})$-approximation algorithm

Being able to reduce to instances with a depot (cf. Section 2.3.1) allows us to give a simple polynomial-time approximation algorithm for A Priori ATSP with approximation ratio $\mathcal{O}(\sqrt{n})$ and an upper bound on the adaptivity gap, as claimed in Section 4.1.1:

Theorem 4.1. *There is a deterministic polynomial-time algorithm that for any instance (V, c, p) of Asymmetric A Priori TSP, computes a tour T with*

$$\mathbb{E}_{A\sim p}\Big[c\big(T[A]\big)\Big] \;\leq\; O\big(\sqrt{n}\big) \cdot \mathbb{E}_{A\sim p}\Big[\mathrm{TSP(A, c)}\Big],$$

where $n = |V|$ denotes the number of vertices. In particular, the adaptivity gap of the Asymmetric A Priori TSP is $\mathcal{O}(\sqrt{n})$.

Our approach is to partition the instance into two instances of Asymmetric A Priori TSP with vertex sets V_1 and V_2, which we consider separately. The vertex sets V_1 and V_2 will satisfy $V = V_1 \cup V_2$ with $V_1 \cap V_2 = \{d\}$ (where $d \in V$ is a depot). We will choose a probability threshold $p^* \in [0, 1]$ and define

$$V_1 \coloneqq \{v \in V : p(v) \geq p^*\} \quad \text{and} \quad V_2 \coloneqq \{v \in V : p(v) < p^*\} \cup \{d\}\,.$$

Ultimately, we will choose $p^* \coloneqq \frac{1}{\sqrt{n}}$. Intuitively the instance with vertex set V_1, containing only vertices with "large" activation probabilities, resembles a problem that is close to standard ATSP.

Proposition 4.10. *Let (V, c, p) be an instance of Asymmetric A Priori TSP with a depot $d \in V$ and $p_{\min} \coloneqq \min_{v\in V} p(v)$. Let T be an α-approximate tour for the instance (V, c) of ATSP. Then,*

$$c\,(T) \leq \alpha \cdot \left\lceil \frac{1}{p_{\min}} \right\rceil \cdot \mathbb{E}_{A\sim p}\left[\mathrm{TSP}(A, c)\right].$$

Proof. Let $k \coloneqq \left\lceil \frac{1}{p_{\min}} \right\rceil$. For the sake of this proof, we conduct the following random experiment: Start with $S_1, \ldots, S_k = \{d\}$. Now, for every $v \in V \setminus \{d\}$ choose an index $i \in \{1, \ldots, k\}$ uniformly

and independently at random and add v to S_i. We have

$$\mathbb{P}[v \in S_i] = \frac{1}{k} \leq p_{\min} \leq p(v),$$

for all $v \in V \setminus \{d\}$ and fixed $i \in \{1, \ldots, k\}$. Hence, $\mathbb{E}[\mathrm{TSP}(S_i, c)] \leq \mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)]$, using that the index choices were independently made. Hence,

$$c(T) \leq \alpha \cdot \mathbb{E}\left[\sum_{i=1}^{k} \mathrm{TSP}(S_i, c)\right] \leq \alpha \cdot k \cdot \mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)],$$

since for any realization of $S_1, \ldots, S_k$, their optimal tours can be concatenated into a single tour on V. □

Observe that the length of any tour on k vertices can be at most a factor of k larger than the length of the optimal tour. This leads to an algorithm with an approximation guarantee that depends on the expected number of active vertices, which is good when all vertices (except for the depot) have "small" activation probabilities, as it is the case for the vertices in V_2 .

Proposition 4.11. *Let (V, c, p) be an instance of Asymmetric A Priori TSP with depot $d \in V$. Then any Hamiltonian cycle T on V satisfies*

$$\mathbb{E}_{A\sim p}[c(T[A])] \;\leq\; 6p(V) \cdot \mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)].$$

Proof. By the triangle inequality, we can bound

$$\mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)] \;\geq\; \mathbb{E}_{A\sim p}\left[\frac{\sum_{v\in A} c(v, d) + c(d, v)}{|A|}\right].$$

We can rewrite the right hand side as

$$\mathbb{E}_{A\sim p}\left[\frac{\sum_{v\in A} c(v, d) + c(d, v)}{|A|}\right] \;=\; \sum_{v\in V} p(v) \cdot (c(v, d) + c(d, v)) \cdot \mathbb{E}_{A\sim p}\left[\frac{1}{|A|} \;\middle|\; v \in A\right].$$

By Markov's inequality,

$$\begin{aligned}
\mathbb{E}_{A\sim p}\left[\frac{1}{|A|} \;\middle|\; v \in A\right] &\geq \mathbb{E}_{A\sim p}\left[\frac{1}{|A|+1}\right] \\
&> \mathbb{P}_{A\sim p}\Big[|A| < 2 \cdot \mathbb{E}_{A\sim p}\big[|A|\big]\Big] \cdot \frac{1}{2 \cdot \mathbb{E}_{A\sim p}[|A|] + 1} \\
&\geq \frac{1}{2} \cdot \frac{1}{2 \cdot \mathbb{E}_{A\sim p}[|A|] + 1} \\
&\geq \frac{1}{6p(V)},
\end{aligned}$$

because $\mathbb{E}_{A\sim p}[|A|] = p(V) \geq 1$ as V contains a depot. Therefore, putting everything together yields

$$\mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)] \geq \sum_{v\in V} p(v) \cdot (c(v, d) + c(d, v)) \cdot \frac{1}{6p(V)},$$

which concludes the proof as every Hamiltonian cycle T on V satisfies

$$\mathbb{E}_{A\sim p}[c(T[A])] \leq \sum_{v\in V} p(v) \cdot (c(v, d) + c(d, v))$$

by the triangle inequality. □

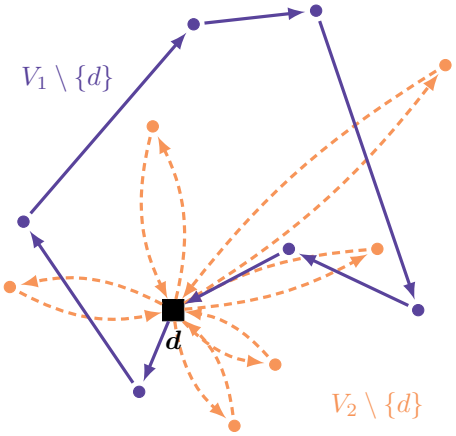


Figure 4.7: Illustration of how we combine Propositions 4.10 and 4.11. We concatenate an α-approximate ATSP tour on V_1 (the vertices with "large" activation probabilities) and any Hamiltonian cycle on V_2 (those with "small" activation probabilities). In our analysis, the expected cost of the latter is upper bounded by the cost of visiting each (active) vertex $v \in V_2$ individually and returning to the depot d between every visit. Note that the depot is important to bound the expected cost of the concatenated tour by the sum of the expected costs of the two individual tours given by Propositions 4.10 and 4.11.

We can now combine Propositions 4.10 and 4.11 into a single algorithm (see also Figure 4.7):

Corollary 4.12. *Let (V, c, p) be an instance of Asymmetric A Priori TSP with a depot $d \in V$. Let $n \coloneqq |V|$ and*

$$V_1 \coloneqq \left\{ v \in V : p(v) \geq \frac{1}{\sqrt{n}} \right\} \quad \text{and} \quad V_2 \coloneqq \left\{ v \in V : p(v) < \frac{1}{\sqrt{n}} \right\} \cup \{d\} .$$

Concatenating an α-approximate tour T_1 for the instance (V_1, c) of ATSP and any Hamiltonian cycle T_2 on V_2 results in an a priori tour T for (V, c, p) with expected cost at most

$$(2\alpha + 12) \cdot \sqrt{n} \cdot \mathbb{E}_{A \sim p} \left[\mathrm{TSP}(A, c) \right] . \tag{4.3}$$

Proof. On the one hand, we can bound $c(T_1)$ by Proposition 4.10 by

$$\begin{aligned} c(T_1) &\leq \alpha \cdot \left\lceil \sqrt{n} \right\rceil \cdot \mathbb{E}_{A \sim p} \left[\mathrm{TSP}(A \cap V_1, c) \right] \\ &\leq \alpha \cdot \left\lceil \sqrt{n} \right\rceil \cdot \mathbb{E}_{A \sim p} \left[\mathrm{TSP}(A, c) \right] \\ &\leq 2\alpha \cdot \sqrt{n} \cdot \mathbb{E}_{A \sim p} \left[\mathrm{TSP}(A, c) \right] . \end{aligned}$$

On the other hand, by Proposition 4.11 and the triangle inequality we have

$$\begin{aligned} \mathbb{E}_{A \sim p} \left[c(T_2[A \cap V_2]) \right] &\leq 6p(V_2) \cdot \mathbb{E}_{A \sim p} \left[\mathrm{TSP}(A \cap V_2, c) \right] \\ &\leq 12\sqrt{n} \cdot \mathbb{E}_{A \sim p} \left[\mathrm{TSP}(A, c) \right] , \end{aligned}$$

because $p(V_2) < 1 + \sum_{v \in V_2} \frac{1}{\sqrt{n}} \leq 2\sqrt{n}$. □

Finally, combining results for constant factor approximations for ATSP ([STV20; TV25]) with Corollary 4.12 and Corollary 2.9 proves Theorem 4.1.

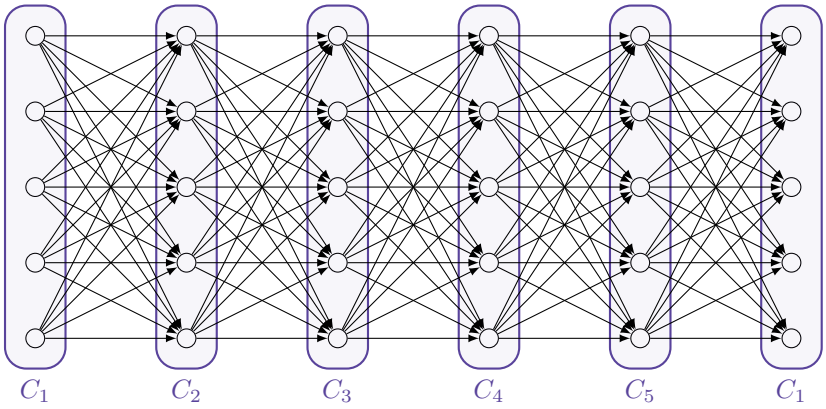


Figure 4.8: The graph G_k for $k = 5$. Each column C_j contains k distinct vertices. Between consecutive columns C_j and C_{j+1}, there is a complete bipartite graph with edges directed towards C_{j+1}. Note that in this figure, the vertices of C_1 are depicted twice for clarity. To form the actual graph G_5, the two representations of each vertex in C_1 should be identified as a single vertex.

4.4 Lower bound on the adaptivity gap

In this section we prove Theorem 4.2, i.e., we prove a polynomial lower bound on the adaptivity gap of the Asymmetric A Priori TSP:

Definition 4.13 (Adaptivity gap)**.** *The* adaptivity gap *of an Asymmetric A Priori TSP instance* $\mathcal{I} = (V, c, p)$ *with* $\mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)] > 0$ *is*

$$\frac{\mathrm{OPT}(\mathcal{I})}{\mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)]}. \tag{4.4}$$

For $n \in \mathbb{N}$ *we define* $\alpha_{\mathrm{adapt}}(n)$ *to be the supremum of this ratio* (4.4) *taken over all instances* (V, c, p) *with* $\mathbb{E}_{A\sim p}[\mathrm{TSP}(A, c)] > 0$ *and* $|V| = n$*.*

We prove that the adaptivity gap $\alpha_{\mathrm{adapt}}(n)$ is at least $\Omega(n^{1/4} \cdot \log^{-1} n)$. To this end, we construct a family of Asymmetric A Priori TSP instances, which we call the *asymmetric grid instances*.

4.4.1 The asymmetric grid instances

The asymmetric grid instances arise as the metric closure of a particular family of directed graphs with unit weight edges, whose construction we present now. See Figure 4.8 for an illustration. We define the graph $G_k = (V_k, E_k)$ for $k \in \mathbb{Z}_{\geq 2}$ as follows: V_k consists of k^2 distinct vertices $\{v_{i,j} : i, j \in \{1, \dots, k\}\}$. As the set of edges we choose

$$E_k \coloneqq \Big\{(v_{i,j},\ v_{i',j'}) \ :\ j + 1 = j' \text{ or } (j = k \text{ and } j' = 1)\Big\}$$

Let $c_k \colon V_k \times V_k \to \mathbb{Z}_{\geq 0}$ denote the shortest-path metric in G_k with unit-weight edges. Then, we obtain an instance (V_k, c_k) of ATSP. We introduce some notation that makes it easier for us to reference certain subsets of V_k. A subset of the form $C_j = \{v_{i,j} : i \in [k]\}$ is called a *column*

for $j \in [j]$. Analogously, subsets $R_i = \{v_{i,j} : j \in [k]\}$ are called *rows*. To obtain instances of Asymmetric A Priori TSP, we define activation probabilities $p_k \colon V_k \to (0,1]$ by $p_k(v) \coloneqq 1/k$ for all $v \in V_k$.

We will prove that for $k \geq 5$, the expected cost of an optimal a priori tour satisfies

$$\mathrm{OPT}\,(V_k, c_k, p_k) \geq \frac{k^{3/2}}{2^9 e^4}, \tag{4.5}$$

and the expected cost of an optimal a posteriori tour satisfies

$$\mathbb{E}_{A\sim p_k}\left[\mathrm{TSP}(A, c_k)\right] \leq 5 \cdot k \log k. \tag{4.6}$$

4.4.2 Lower bound for an optimal a priori tour

We first prove (4.5), i.e., we prove a lower bound for the optimal a priori tour. Let $k \geq 5$ and let T^* be a Hamiltonian cycle on V_k and let us number the vertices as $x_1, \ldots, x_{k^2}$ according to the order in which they are visited by T^*. We write $E^{(k)} \coloneqq V_k \times V_k$ for the edge set of the complete directed graph on V_k, and $\delta^+(X) \coloneqq X \times V_k$ and $\delta^-(X) \coloneqq V_k \times X$ for any $X \subseteq V_k$. We have

$$\mathbb{E}_{A\sim p_k}\left[c_k(T^*[A])\right] = \sum_{e \in E^{(k)}} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right] \cdot c_k(e), \tag{4.7}$$

where, for $e = (x_i, x_j) \in E^{(k)}$,

$$\mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right] = \begin{cases} (1/k)^2 \cdot (1-1/k)^{j-i-1} & \text{if } i < j, \\ (1/k)^2 \cdot (1-1/k)^{k^2+j-i-1} & \text{if } i > j. \end{cases}$$

We analyze the contribution of $\delta^+(C_j)$ to $\mathbb{E}_{A\sim p_k}\left[c_k(T^*[A])\right]$, that is,

$$\sum_{e \in \delta^+(C_j)} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right] \cdot c_k(e)$$

separately for all $j \in [k]$.

Observe that $k \geq 5$ implies $4k < k^2$. Let $j \in [k]$ be fixed and define

$$S(x_i) \;\coloneqq\; \left\{x_\ell : \ell = i+1, \ldots, i+4k\right\}$$

for all $i \in \left\{1, \ldots, k^2\right\}$. In other words, the set $S(x_i)$ contains the $4k$ vertices following x_i in the tour T^*. For $x_\ell \in S(x_i)$, we have

$$\mathbb{P}_{A\sim p_k}\left[(x_i, x_\ell) \in E(T^*[A])\right] \geq (1/k)^2 \cdot (1-1/k)^{4k} \geq (1/k)^2 \cdot \frac{1}{(2e)^4}. \tag{4.8}$$

In our proof, we will focus on lower bounding the contribution of edges (x_i, x_ℓ) with $x_\ell \in S(x_i)$ to (4.7).

Definition 4.14 (Saturated interval)**.** *We say that $I \subseteq V_k$ is an* interval *of T^* if it is the vertex set of a subpath of T^*. Moreover, we call I* saturated *if it satisfies both of the following conditions:*

- *it contains at most $4k$ vertices, and*
- *it contains at least $\sqrt{k}$ vertices from C_j.*

For an interval I of T^*, we denote by $E_j[I] \coloneqq \delta^+(C_j) \cap (I \times I)$ the set of edges in $\delta^+(C_j)$ with both endpoints in I. We prove a lower bound on the contribution of these edges to the overall cost $\mathbb{E}_{A\sim p_k}[c_k(T^*[A])]$ in case the interval I is saturated. We use that each of theses edges has cost k with respect to the cost function c_k (because both endpoints are in the same column), but the edges also have a significant probability of being part of $T^*[A]$ (because both endpoints are in the interval I with at most $4k$ vertices).

Lemma 4.15. *Let I be a saturated interval of T^*. Then*

$$\sum_{e\in E_j[I]} \mathbb{P}_{A\sim p_k}[e \in E(T^*[A])] \cdot c_k(e) \;\geq\; (|I \cap C_j|)^2 \cdot \frac{1}{2^7 e^4 k}.$$

Proof. Let $x_i, x_\ell \in I \cap C_j$ with $x_i \neq x_\ell$. Because $x_i, x_\ell \in I$, we have $x_i \in S(x_\ell)$ or $x_\ell \in S(x_i)$. Without loss of generality, assume that $x_\ell \in S(x_i)$. Since $x_i \neq x_\ell$ and $x_i, x_\ell \in C_j$, we have $c_k(x_i, x_\ell) = k$. There are $\binom{|I\cap C_j|}{2}$ such pairs x_i, x_ℓ. Thus, we have

$$\begin{aligned}
\sum_{e\in E_j[I]} \mathbb{P}_{A\sim p_k}[e \in E(T^*[A])] \cdot c_k(e) &\overset{(4.8)}{\geq} \sum_{e\in E_j[I]} (1/k)^2 \cdot \frac{1}{(2e)^4} \cdot c_k(e) \\
&\geq \binom{|I \cap C_j|}{2} \cdot (1/k)^2 \cdot \frac{1}{(2e)^4} \cdot k \\
&\geq (|I \cap C_j| - 1)^2 \cdot \frac{1}{2^5 e^4 k} \\
&\geq (|I \cap C_j|)^2 \cdot \frac{1}{2^7 e^4 k},
\end{aligned}$$

using that $|I \cap C_j| \geq \sqrt{k} \geq 2$ for the last inequality. □

Lemma 4.16. *For $k \geq 5$,*

$$\mathrm{OPT}\,(V_k, c_k, p_k) \geq \frac{k^{3/2}}{2^9 e^4}.$$

Proof. Let $I_1, \ldots, I_m$ be a family of pairwise disjoint saturated intervals of T^* that maximizes $\sum_{i=1}^m |I_i|$. We say that $v \in V_k$ is *marked* if $v \in \bigcup_{i=1}^m I_i$. Otherwise v is *unmarked*. Observe that $m \leq \sqrt{k}$ since each interval contains at least $\sqrt{k}$ vertices from C_j and $|C_j| = k$. We distinguish the following two cases:

Case 1: Suppose that at least $k/2$ vertices of C_j are marked. In other words, we have $\sum_{i=1}^m |I_i \cap C_j| \geq k/2$. Using the Arithmetic-Quadratic-Mean inequality and the fact that the I_i are pairwise disjoint, we obtain

$$\begin{aligned}
\sum_{e\in\delta^+(C_j)} \mathbb{P}_{A\sim p_k}[e \in E(T^*[A])] \cdot c_k(e) &\geq \sum_{i=1}^m \sum_{e\in E_j[I_i]} \mathbb{P}_{A\sim p_k}[e \in E(T^*[A])] \cdot c_k(e) \\
&\overset{\text{Lem. 4.15}}{\geq} \frac{1}{2^7 e^4 k} \cdot \sum_{i=1}^m (|I_i \cap C_j|)^2 \\
&\geq \frac{1}{2^7 e^4 k} \cdot \frac{\left(\sum_{i=1}^m |I_i \cap C_j|\right)^2}{m} \\
&\geq \frac{1}{2^9 e^4 k} \cdot \frac{k^2}{\sqrt{k}} \\
&= \frac{\sqrt{k}}{2^9 e^4}.
\end{aligned}$$

Case 2: Suppose that at least $k/2$ vertices of C_j are unmarked. We define the *congestion* of a vertex x_ℓ as

$$\operatorname{cong}(x_\ell) \coloneqq |\{x_i \in C_j : x_i \text{ unmarked and } x_\ell \in S(x_i)\}|.$$

We claim that our choice of $I_1, \dots, I_m$ implies $\operatorname{cong}(x_\ell) \leq \sqrt{k}$ for all $\ell \in [k^2]$.

Suppose for the sake of deriving a contradiction that $\operatorname{cong}(x_\ell) > \sqrt{k}$. Without loss of generality, after renumbering we can assume that $\ell = k^2$, i.e, x_ℓ is the last vertex visited by the tour T^*. Let

$$\begin{aligned} i_1 &\coloneqq \min\left\{i \in [k^2] : x_i \in C_j \text{ unmarked and } x_\ell \in S(x_i)\right\} \\ i_2 &\coloneqq \max\left\{i \in [k^2] : x_i \in C_j \text{ unmarked and } x_\ell \in S(x_i)\right\}. \end{aligned}$$

The assumption that $\operatorname{cong}(x_\ell) > \sqrt{k}$ implies that the interval I^* that is given by the subpath $x_{i_1}, x_{i_1+1}, \dots, x_{i_2}$ of T^* is saturated ($|I^*| \leq 4k$ since $x_{i_2} \in S(x_{i_1})$). Every interval I_z for some $z \in [m]$ satisfies $I_z \cap I^* = \emptyset$ or $I_i \subseteq I^*$, because x_{i_1} and x_{i_2} are unmarked. Hence, removing all saturated intervals of $I_1, \dots, I_m$ that intersect I^* and adding the interval I^* instead still yields a family of pairwise disjoint saturated intervals. This contradicts the choice of our family maximizing $\sum_{j=1}^{m} |I_j|$ since I^* contains unmarked vertices. We conclude that, indeed, $\operatorname{cong}(x_\ell) \leq \sqrt{k}$ for all $\ell \in [k^2]$.

We consider the set

$$Z \coloneqq \left\{v \in V_k : c_k(x, v) \geq \sqrt{k} - 1 \text{ for all } x \in C_j\right\}$$

of vertices that are expensive to reach from column C_j. Note that

$$|V_k \setminus Z| \leq \left| \bigcup_{i=0}^{\lfloor \sqrt{k} \rfloor - 1} C_{j+i} \right| \leq \sqrt{k} \cdot k,$$

where $C_z \coloneqq C_{z-k}$ for $z > k$. We consider the edges from an unmarked vertex x_i in column C_j to a vertex in $S(x_i)$:

$$\widetilde{E} \coloneqq \bigcup_{x_i \in C_j} \left\{(x_i, x_\ell) : x_i \text{ unmarked and } x_\ell \in S(x_i)\right\} \subseteq \delta^+(C_j).$$

By definition of $S(x_i)$ and our assumption that at least $k/2$ vertices of C_j are unmarked, we have $|\widetilde{E}| \geq 4k \cdot k/2 = 2k^2$. Because all vertices have congestion at most $\sqrt{k}$, at most $|V_k \setminus Z| \cdot \sqrt{k} \leq k^2$ edges of $\widetilde{E}$ end in a vertex from $V_k \setminus Z$. This shows that $|\widetilde{E} \cap \delta^-(Z)| \geq k^2$.

We conclude

$$\begin{aligned}
\sum_{e\in\delta^+(C_j)} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right]\cdot c_k(e) &\geq \sum_{e\in\widetilde{E}} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right]\cdot c_k(e)\\
&\overset{(4.8)}{\geq} \sum_{e\in\widetilde{E}} (1/k)^2\cdot\frac{1}{(2e)^4}\cdot c_k(e)\\
&\geq \sum_{e\in\widetilde{E}\cap\delta^-(Z)} (1/k)^2\cdot\frac{1}{(2e)^4}\cdot c_k(e)\\
&\geq \sum_{e\in\widetilde{E}\cap\delta^-(Z)} (1/k)^2\cdot\frac{1}{(2e)^4}\cdot(\sqrt{k}-1)\\
&\geq \sum_{e\in\widetilde{E}\cap\delta^-(Z)} (1/k)^2\cdot\frac{1}{2^5e^4}\cdot\sqrt{k}\\
&\geq k^2\cdot(1/k)^2\cdot\frac{1}{2^5e^4}\cdot\sqrt{k}\\
&= \frac{\sqrt{k}}{2^5e^4}.
\end{aligned}$$

We have shown that in both cases

$$\sum_{e\in\delta^+(C_j)} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right]\cdot c_k(e) \geq \frac{\sqrt{k}}{2^9e^4}.$$

This completes our proof because

$$\begin{aligned}
\mathbb{E}_{A\sim p_k}\left[c_k(T^*[A])\right] &= \sum_{e\in E^{(k)}} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right]\cdot c_k(e)\\
&= \sum_{j=1}^{k}\sum_{e\in\delta^+(C_j)} \mathbb{P}_{A\sim p_k}\left[e \in E(T^*[A])\right]\cdot c_k(e)\\
&\geq \frac{k^{3/2}}{2^9e^4}.
\end{aligned}$$

□

The tour shown in Figure 4.9 shows that the result of Lemma 4.16 is asymptotically tight:

Lemma 4.17. *Let $k \in \mathbb{Z}_{\geq 2}$ such that $\sqrt{k} \in \mathbb{Z}$. Then, there exists a tour T on V_k that satisfies*

$$\mathbb{E}_{A\sim p_k}\left[c_k(T[A])\right] \leq \mathcal{O}(k^{3/2}).$$

Proof. The tour T we consider is shown in Figure 4.9. It can be formally described by the tour $x_1, \ldots, x_{k^2}$, where

$$x_i \coloneqq v_{j\cdot\sqrt{k}+l,t} \quad \text{with} \begin{cases} j &= \left\lfloor \frac{i-1}{k\sqrt{k}} \right\rfloor,\\ l &= \left(i-1 \mod \sqrt{k}\right)+1,\\ t &= \left(\left\lfloor \frac{i-1}{\sqrt{k}} \right\rfloor \mod k\right)+1. \end{cases}$$

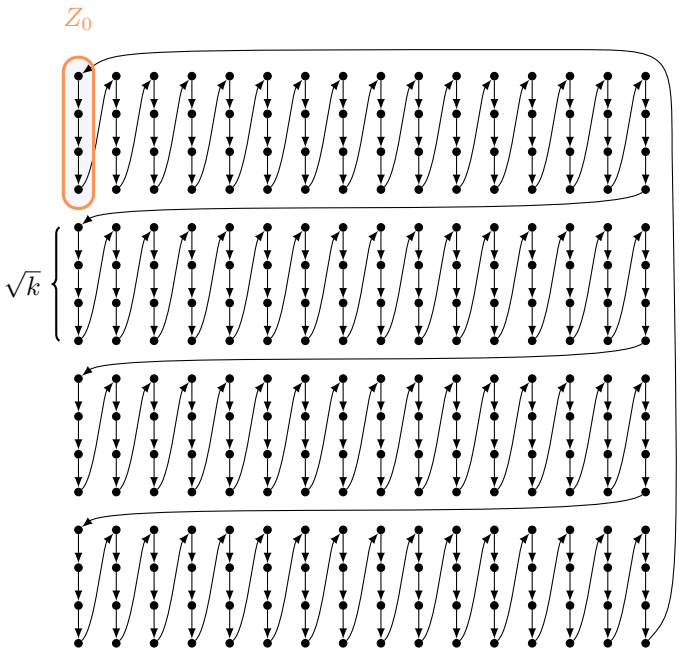


Figure 4.9: An example for $k = 16$. An asymptotically optimal a priori tour T for (V_k, c_k, p_k) is obtained by grouping the rows into blocks of size $\sqrt{k}$ and traversing each block in a column-wise manner.

For a fixed $i \in \left\{1, \ldots, \sqrt{k}\right\}$, we analyze the contribution of $\delta^+(x_i) \coloneqq \{x_i\} \times V_k$ to $\mathbb{E}_{A \sim p_k}\left[c_k(T[A])\right]$, that is,

$$\sum_{e \in \delta^+(x_i)} \mathbb{P}_{A \sim p_k}\left[e \in E(T[A])\right] \cdot c_k(e).$$

By the symmetry of the instance, this is sufficient to obtain a bound on all edges $e \in V_k \times V_k$ because the vertices of V_k can be renumbered accordingly.

We define

$$Z_j \coloneqq \left\{x_l : l = j \cdot \sqrt{k} + 1, \ldots, (j+1) \cdot \sqrt{k}\right\},$$

for $j = 0, \ldots, k \cdot \sqrt{k} - 1$. Then, we can simply bound the contribution of $\delta^+(x_i) \cap \delta^-(Z_0)$ by

$$\begin{aligned}\sum_{e \in \delta^+(x_i) \cap \delta^-(Z_0)} \mathbb{P}_{A \sim p_k}\left[e \in E(T[A])\right] \cdot c_k(e) &\leq \sum_{e \in \delta^+(x_i) \cap \delta^-(Z_0)} (1/k)^2 \cdot k \\ &\leq \sqrt{k} \cdot (1/k) \\ &= \frac{1}{\sqrt{k}}.\end{aligned}$$

For $j \in \left\{1, \ldots, k \cdot \sqrt{k} - 1\right\}$, the cost of an edge in $\delta^+(x_i) \cap \delta^-(Z_j)$ can be upper bounded by j. Furthermore, between x_i and any vertex v of Z_j lie at least $(j-1)\sqrt{k}$ many other vertices on T. Therefore, the activation probability of such an edge (x_i, v) can be upper bounded by

$(1/k)^2 (1-1/k)^{(j-1)\sqrt{k}}$. Hence,

$$\begin{aligned}\sum_{e\in\delta^+(x_i)\cap\delta^-(Z_j)} \mathbb{P}_{A\sim p_k}\left[e\in E(T[A])\right]\cdot c_k(e) &\le \sum_{e\in\delta^+(x_i)\cap\delta^-(Z_j)} \frac{1}{k^2}\cdot\left(1-\frac{1}{k}\right)^{(j-1)\cdot\sqrt{k}}\cdot j\\ &\le \frac{\sqrt{k}}{k^2}\cdot\left(1-\frac{1}{k}\right)^{(j-1)\cdot\sqrt{k}}\cdot j\\ &\le \frac{\sqrt{k}}{k^2}\cdot e^{\frac{-(j-1)}{\sqrt{k}}}\cdot j\\ &\le \frac{e}{k^{3/2}}\cdot\left(e^{-1/\sqrt{k}}\right)^j\cdot j.\end{aligned}$$

This implies an upper bound on the total contribution of $\delta^+(x_i)\cap\delta^-\left(\bigcup_{j=1}^{k^{3/2}-1} Z_j\right)$, since

$$\begin{aligned}\sum_{j=1}^{k^{3/2}-1}\sum_{e\in\delta^+(x_i)\cap\delta^-(Z_j)} \mathbb{P}_{A\sim p_k}\left[e\in E(T[A])\right]\cdot c_k(e) &\le \sum_{j=1}^{k^{3/2}-1}\frac{e}{k^{3/2}}\cdot\left(e^{\frac{-1}{\sqrt{k}}}\right)^j\cdot j\\ &\le \sum_{i=0}^{\infty}\frac{e}{k^{3/2}}\cdot\left(e^{\frac{-1}{\sqrt{k}}}\right)^j\cdot j\\ &= \frac{e}{k^{3/2}}\cdot\frac{e^{\frac{-1}{\sqrt{k}}}}{\left(1-e^{\frac{-1}{\sqrt{k}}}\right)^2}\\ &\le \frac{e}{k^{3/2}}\cdot\frac{1}{\left(\frac{1}{2\sqrt{k}}\right)^2}\\ &= \frac{4e}{\sqrt{k}}.\end{aligned}$$

We conclude that

$$\begin{aligned}\sum_{e\in\delta^+(x_i)} \mathbb{P}_{A\sim p_k}\left[e\in E(T[A])\right]\cdot c_k(e) &= \sum_{j=0}^{k^{3/2}-1}\sum_{e\in\delta^+(x_i)\cap\delta^-(Z_j)} \mathbb{P}_{A\sim p_k}\left[e\in E(T[A])\right]\cdot c_k(e)\\ &\le \frac{1}{\sqrt{k}}+\frac{4e}{\sqrt{k}}.\end{aligned}$$

As argued above, by symmetry, this bound holds for all $i=1,\dots,k^2$. Summing up this bound over all $i=1,\dots,k^2$ yields a total upper bound of $k^2\cdot\frac{1+4e}{\sqrt{k}} = (1+4e)k^{3/2}$ for $\mathbb{E}_{A\sim p_k}\left[c_k(T[A])\right]$. □

4.4.3 Upper bound on the expected cost of an optimal a posteriori tour

We now prove (4.6). We can easily characterize the cost $\mathrm{TSP}(A,c_k)$ of an optimal tour on any set $A\subseteq V_k$:

Lemma 4.18. *For any $A\subseteq V_k$, we have*

$$\mathrm{TSP}(A,c_k) = k\cdot\max\left\{|A\cap C_j| : j=1,\dots,k\right\}.$$

Proof. Because any two vertices in the same column have distance k from each other (in either direction), we have $\mathrm{TSP}(A \cap C_j, c_k) \geq k \cdot |C_j|$ for each $j \in [k]$. Therefore, $\mathrm{TSP}(A, c_k) \geq \mathrm{TSP}(A \cap C_j, c_k) \geq k \cdot |C_j|$, implying that $\mathrm{TSP}(A, c_k)$ is at least as large as claimed.

For the upper bound, suppose that at most t vertices per column are active. After relabeling all vertices, we can assume that without loss of generality all vertices of A are contained in the first t rows. We can visit all vertices in these rows by a tour of length $t \cdot k$ by traversing one row after the other, always visiting first the vertex in C_1, then in C_2, and so on until C_k. □

Using Lemma 4.18, we obtain the following upper bound on the expected cost of an optimal a posteriori tour:

Lemma 4.19. *For $k \geq 3$, we have*

$$\mathbb{E}_{A \sim p_k}\left[\mathrm{TSP}(A, c_k)\right] \leq 5 \cdot k \log k.$$

Proof. Let $X_{i,j}$ be the random indicator variable for the event that $v \in A$ (i.e., $X_{i,j}$ is a Bernoulli random variable with parameter $1/k$). We denote by $\widehat{X}_j \coloneqq \sum_{i=1}^{k} X_{i,j}$ the number of active vertices for each column j. Note that $\mathbb{E}_{A \sim p_k}\left[\widehat{X}_j\right] = 1$. By a Chernoff-bound,

$$\mathbb{P}_{A \sim p_k}\left[\widehat{X}_j \geq 1 + (2 \log k - 1)\right] \leq e^{-\frac{(2 \log k - 1)}{3}} \leq \frac{e}{k^2}.$$

We denote $Y \coloneqq \max_{j=1,\dots,k} \widehat{X}_j$. Then, by a union bound, we obtain $\mathbb{P}_{A \sim p_k}\left[Y \geq 2 \log k\right] \leq e/k$. Thus, we conclude

$$\begin{aligned}
\mathbb{E}_{A \sim p_k}[Y] &= \mathbb{P}_{A \sim p_k}[Y \geq 2 \log k] \cdot \mathbb{E}_{A \sim p_k}[Y \mid Y \geq 2 \log k] \\
&\quad + \mathbb{P}_{A \sim p_k}[Y < 2 \log k] \cdot \mathbb{E}_{A \sim p_k}[Y \mid Y < 2 \log k] \\
&\leq \frac{e}{k} \cdot k + 1 \cdot (2 \log k) \\
&\leq 5 \log k.
\end{aligned}$$

Finally, by Lemma 4.18, we have

$$\mathbb{E}_{A \sim p_k}\left[\mathrm{TSP}(A, c_k)\right] \leq \mathbb{E}_{A \sim p_k}\left[k \cdot Y\right] \leq 5k \log k.$$ □

4.4.4 Lower bound on the adaptivity gap

We can now conclude the claimed lower bound for $\alpha_{\mathrm{adapt}}(n)$:

Theorem 4.2. *The adaptivity gap of the Asymmetric A Priori TSP is $\Omega(n^{1/4} \cdot \log^{-1} n)$, where n denotes the number of vertices.*

Proof. Let $n \in \mathbb{N}, n \geq 25$. For $k' \coloneqq \lfloor \sqrt{n} \rfloor$ and $n' \coloneqq (k')^2$ we have $n \geq n'$. Observe that α_{adapt} is monotonically increasing since we can simply add vertices with activation probability 0, which changes neither the numerator nor the denominator of the ratio in (4.4).[2] Therefore, it suffices to show that $\alpha_{\mathrm{adapt}}(n') \geq c \cdot \frac{n^{1/4}}{\log n}$ for some constant $c \in \mathbb{R}$.

Note that $|V_k| = (k')^2 = n'$ and $k' \geq 5$. Applying Lemma 4.16 and Lemma 4.19 to $(V_{k'}, c_{k'}, p_{k'})$ we obtain

$$\alpha_{\mathrm{adapt}}(n') \geq \frac{\mathrm{OPT}(V_{k'}, c_{k'}, p_{k'})}{\mathbb{E}_{A \sim p_{k'}}\left[\mathrm{TSP}(A, c_{k'})\right]} \geq c \cdot \frac{k' \cdot \sqrt{k'}}{k' \log k'} = c \cdot \frac{\sqrt{k'}}{\log k'}$$

[2] If one does not allow vertices with activation probability zero, it is still possible to add vertices with arbitrarily small activation probability as the adaptivity gap is defined as a supremum.

for $c \coloneqq 1/(5 \cdot 2^9 e^4)$. We have $\sqrt{n} \geq k' \geq \sqrt{n} - 1$. Thus,

$$c \cdot \frac{\sqrt{k'}}{\log k'} \geq c \cdot \frac{\sqrt{\sqrt{n}-1}}{\log \sqrt{n}} \geq c \cdot \frac{\sqrt{\sqrt{n}/2}}{\log \sqrt{n}} \geq \frac{c}{\sqrt{2}} \cdot \frac{n^{1/4}}{\log n}.$$

This concludes the proof. □

4.5 Reducing to Hop-ATSP

The goal of this section is to prove Theorem 4.5, i.e., to reduce Asymmetric A Priori TSP to Hop-ATSP. We start by recalling the definition of Hop-ATSP and explaining the high-level ideas behind our reduction.

4.5.1 Outline of our reduction

For our reduction to Hop-ATSP, we will assume that all vertices have the same activation probability. By Section 2.3.2 (and in particular Proposition 2.3), this assumption is no restriction. Then, when computing the expected cost of an a priori tour T that visits vertices $v_1, v_2, \ldots, v_{r+1} = v_1$ in this order, the contribution of an edge $(v_i, v_{i+\Delta})$ decreases as Δ grows because the probability that at least one more vertex between v_i and $v_{i+\Delta}$ is active increases. Instead of gradually decreasing this contribution, we show that we can also let edges fully contribute to the cost if $\Delta \leq k$ and ignore edges where $\Delta > k$ for some suitable $k \in \mathbb{N}$. To describe this formally, we use the k-hop cost, as introduced in Section 4.2:

Definition 4.20 (k-hop cost)**.** *Given a vertex set V, a cost function $c\colon V \times V \to \mathbb{R}_{\geq 0}$ satisfying the triangle inequality, and a hop distance $k \in \mathbb{N}$ with $k < |V|$, the* k-hop cost *of a closed walk T in V visiting the vertices $v_1, v_2, \ldots, v_r$ in this order, is*

$$c^{(k)}(T) = \sum_{\Delta=1}^{k} \sum_{i=1}^{r} c(v_i, v_{i+\Delta}),$$

where we define $v_i \coloneqq v_{i-r}$ for $i > r$. For a non-closed walk W visiting vertices $v_1, v_2, \ldots, v_r$ in this order, i.e., where $v_1 \neq v_r$, we also define

$$c^{(k)}(W) = \sum_{\Delta=1}^{k} \sum_{i=1}^{r} c(v_i, v_{i+\Delta}),$$

but now define $c(v_i, v_\ell) \coloneqq 0$ for $\ell > r$.

Recall our definition of Hop-ATSP:

Definition 4.4 (Hop-ATSP)**.** *An instance of Hop-ATSP consists of*

- *a finite vertex set V,*
- *a cost function $c\colon V \times V \to \mathbb{R}_{\geq 0}$ satisfying the triangle inequality, and*
- *a hop-distance $k \in \mathbb{N}$ with $k < |V|$.*

The task in Hop-ATSP is to compute a tour T on V minimizing $c^{(k)}(T)$.

Recall that a tour on V is a Hamiltonian cycle on V. However, as we will show next, if a closed walk visits a vertex a second time, we can skip the second visit of this vertex and this does not increase the k-hop cost of the tour. We therefore also call closed walks *tours*, but every Hop-ATSP instance has an optimal solution that is a Hamiltonian cycle on V.

Lemma 4.21. *Let (V, c, k) be an instance of Hop-ATSP. If a tour W' results from a tour W by omitting a (second) visit of a vertex, we have $c^{(k)}(W') \leq c^{(k)}(W)$.*

Similarly, if a walk W' results from a non-closed walk W by omitting one visit of a vertex, we have $c^{(k)}(W') \leq c^{(k)}(W)$.

Proof. Let $w_1, \dots, w_t$ be the vertices visited by W in this order (where the vertices do not need to be distinct). Let $1 \leq i \leq t$ such that W' results from W by omitting the visit w_i, that is W' visits the vertices $w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_t$ in this order.

Suppose the removal of the visit w_i introduces a new k-hop edge $(w_{i-\ell}, w_{i-\ell+k+1})$ for some $\ell \in \{1, \dots, k\}$. Then, the edges $(w_{i-\ell}, w_i)$ and $(w_i, w_{i-\ell+k+1})$ no longer contribute to $c^{(k)}(W')$. Thus, if W is a tour, by the triangle inequality, we have

$$c^{(k)}(W) - c^{(k)}(W') \;=\; \sum_{\ell=1}^{k} \big(c(w_{i-\ell}, w_i) + c(w_i, w_{i-\ell+k+1}) - c(w_{i-\ell}, w_{i-\ell+k+1})\big) \;\geq\; 0,$$

where we define $w_j \coloneqq w_{j-t}$ for $j > t$ and $w_j \coloneqq w_{i+t}$ for $j < 0$. Similarly, if W is a non-closed walk, we have

$$c^{(k)}(W) - c^{(k)}(W') \;=\; \sum_{\ell=\max(1,k+i-t+1)}^{\min(k,i-1)} \big(c(w_{i-\ell}, w_i) + c(w_i, w_{i-\ell+k+1}) - c(w_{i-\ell}, w_{i-\ell+k+1})\big) \;\geq\; 0.$$

□

The main goal of this section is to prove Theorem 4.5, i.e., to reduce Asymmetric A Priori TSP to Hop-ATSP. When assuming that all vertices have the same activation probability $p(v) = \delta$ (which is possible thanks to Proposition 2.3), we show that for $k \coloneqq \lfloor \frac{1}{\delta} \rfloor$, every Hamiltonian cycle T on V has expected cost $\mathbb{E}_{A \sim p}[c(T[A])] = \Theta(1) \cdot \delta^2 \cdot c^{(k)}(T)$ (see Theorem 4.25).

The running time and also the size of the result of the transformation to an instance with uniform activation probabilities as performed in Proposition 2.3 depends on the minimum occurring activation probability in the original instance. If there are vertices of very small activation probability, we would need to choose δ very small, leading to a super-polynomial number of copies of other vertices (and also to a super-polynomial number k). Thus, we handle vertices of very small activation probability separately:

Since we can assume that one vertex is always active, i.e. there is a *depot* d with $p(d) = 1$ (cf. Section 2.3.1), we connect vertices with very low activation probability (less than $\frac{1}{n}$) directly to the depot, thus reducing to instances where all vertices have activation probability at least $\frac{1}{n}$. These instances can then be transformed to instances with uniform activation probability for which we prove a reduction to Hop-ATSP.

Finally, we go one step further and show that it suffices to consider *well-scaled* instances of Hop-ATSP, i.e., we can assume that c takes only integer values, the diameter of the complete graph on V is upper bounded by $2n^3$ and the optimum solution costs at least n^2 (where $n := |V|$). This can be achieved easily by scaling all edge costs appropriately, rounding them down to the next integer, and taking the metric closure to ensure that c still satisfies the triangle inequality (see Section 4.5.3).

4.5.2 Reducing A Priori ATSP with uniform activation to Hop-ATSP

In this subsection we reduce Asymmetric A Priori TSP with uniform activation probabilities to Hop-ATSP. Suppose we have a Hamiltonian cycle T on V and uniform activation probabilities $p(v) = \delta$ for each $v \in V$. We label the vertices of V as $v_1, \dots, v_n$ in the order in which they

are visited by T. We follow our usual convention that we interpret vertices $v_{j+n} \coloneqq v_j$. When sampling a set A according to p, the probability that an edge $e = (v_j, v_{j+\Delta})$ is contained in $T[A]$ is $\delta^2 \cdot (1-\delta)^{\Delta-1}$, because this happens if and only if $v_j, v_{j+\Delta} \in A$ and $\{v_{j+1}, \dots, v_{j+\Delta-1}\} \subseteq V \setminus A$. Hence, we can write the expected cost of T as

$$\mathbb{E}_{A\sim p}\left[c\left(T[A]\right)\right] = \sum_{\Delta=1}^{n-1}\sum_{j=1}^{n} \delta^2(1-\delta)^{\Delta-1} \cdot c(v_j, v_{j+\Delta}), \tag{4.9}$$

We assume for the sake of this explanation that $\delta > \frac{1}{|V|}$ and define $k \coloneqq \lfloor 1/\delta \rfloor < |V|$. In order to reduce Asymmetric A Priori TSP with uniform activation probabilities to Hop-ATSP, we will show that (4.9) is up to constant factors the same as δ^2 times the k-hop cost of T. To this end, we first observe that $(1-\delta)^{\Delta-1} \geq \beta$ for some suitable constant $\beta \in \mathbb{R}_{\geq 0}$ and any $\Delta \in [k]$, implying that (4.9) is lower-bounded by $\delta^2 \cdot \beta \cdot c^{(k)}(T)$,

In order to obtain an upper bound of $\delta^2 \cdot \gamma \cdot c^{(k)}(T)$ for (4.9), where $\gamma \in \mathbb{R}_{\geq 0}$ is another constant, we show that

$$\sum_{\Delta=k+1}^{n-1}\sum_{j=1}^{n}(1-\delta)^{\Delta-1} \cdot c(v_j, v_{j+\Delta}) \leq (\gamma-1) \cdot \sum_{\Delta=1}^{k}\sum_{j=1}^{n} c(v_j, v_{j+\Delta}) = (\gamma-1) \cdot c^{(k)}(T).$$

By the triangle inequality, for an edge $e = (v_j, v_{j+\Delta})$ with $\Delta \in \{k+1, \dots, n-1\}$, we can bound the cost $c(e)$ by the cost of any v_j-$v_{j+\Delta}$ path $c(P^{(e)})$. We will choose $P^{(e)}$ such that each of its edges contributes to the k-hop cost of T. The following lemma will be useful for this:

Lemma 4.22. *Let t and k be integers with $t \geq k \geq 16$. Then there are positive integers $a_1, \dots, a_m$ such that $\sum_{j=1}^{m} a_j = t$ and $2\lceil \frac{k}{16} \rceil \leq a_j \leq \frac{k}{2}$ for all $j \in \{1, \dots, m\}$.*

Proof. Let $m \in \mathbb{Z}$ be the largest integer such that $m \cdot 2\lceil \frac{k}{16} \rceil \leq t$. Define $a_j \coloneqq 2\lceil \frac{k}{16} \rceil$ for $1 \leq j \leq m-1$ and $a_m = t - (m-1) \cdot 2\lceil \frac{k}{16} \rceil$. By maximality of m, we have $a_m < 4\lceil \frac{k}{16} \rceil \leq 4(\frac{k}{16}+1) \leq \frac{k}{2}$ since $k \geq 16$. □

This immediately implies the following lemma:

Lemma 4.23. *Let P be a path on at least k vertices. If $k \geq 16$, then we can subdivide P into vertex disjoint subpaths $P_1, \dots, P_m$ of P such that $\bigcup_{j=1}^{m} V(P_j) = V(P)$ and $2\lceil \frac{k}{16} \rceil \leq |V(P_j)| \leq \frac{k}{2}$ for all $j \in \{1, \dots, m\}$.* □

Recall that $e = (v_j, v_{j+\Delta})$. Applying the previous lemma to the v_j-$v_{j+\Delta}$ subpath of T allows us to choose the path $P^{(e)}$ such that it contains approximately $\frac{\Delta}{k}$ edges. This gives us an upper bound on the number of edges contributing to $c^{(k)}(T)$ that additionally need to pay for the cost of $(1-\delta)^{\Delta-1} \cdot c(v_j, v_{j+\Delta})$. Note that

$$\sum_{j=1}^{n}\sum_{\Delta=k+1}^{n-1} \frac{\Delta}{k} \cdot (1-\delta)^{\Delta-1} = \frac{n}{k} \cdot \mathcal{O}(1/\delta^2) = \mathcal{O}(n \cdot k).$$

Moreover, there are exactly $n \cdot k$ edges that contribute to $c^{(k)}(T)$. By carefully selecting these paths $P^{(v_j, v_{j+\Delta})}$ for any $j \in [n]$ and $\Delta \in \{k+1, \dots, n-1\}$, we will be able to ensure that every edge contributing to $c^{(k)}(T)$ only needs to additionally pay a constant multiple of its own cost.

In fact, we will not only choose a single path $P^{(e)}$ with $e = (v_j, v_{j+\Delta})$, but we choose many such paths $P_1^{(e)}, \dots, P_r^{(e)}$ together with weights $\xi_1^{(e)}, \dots, \xi_r^{(e)}$ such that $\sum_{i=1}^{r} \xi_i^{(e)} = 1$. This allows us to distribute the cost $(1-\delta)^{\Delta-1} \cdot c(v_j, v_{j+\Delta})$ evenly onto all edges contributing to $c^{(k)}(T)$. The following lemma formalizes this:

Lemma 4.24. *Let P be an s-t-path on at least $k+2$ vertices in a complete directed graph G for some $k \in \mathbb{Z}_{\geq 16}$. Enumerate the vertices on P in the order given by P as $s = x_0, x_1, \ldots, x_{\ell+1} = t$. Then there exists a collection of s-t-paths $P_1, \ldots, P_r$ together with weights $\xi_1, \ldots, \xi_r \in [0,1]$ with $\sum_{i=1}^{r} \xi_i = 1$ such that*

$$\begin{cases} \chi_e \leq 2^4/k & \text{if } e = (x_i, x_j) \in (\delta^+(s) \cup \delta^-(t)) \text{ and } j - i \leq k, \\ \chi_e \leq 2^8/k^2 & \text{if } e = (x_i, x_j) \notin \delta^+(s) \cup \delta^-(t) \text{ and } j - i \leq k, \\ \chi_e = 0 & \text{if } e = (x_i, x_j) \text{ with } j - i > k, \end{cases} \tag{4.10}$$

where $\chi \in [0,1]^{E(G)}$ is defined as $\chi \coloneqq \sum_{i=1}^{r} \xi_i \cdot \chi(P_i)$ (writing $\chi(P_i) \in \{0,1\}^{E(G)}$ as the characteristic vector of P_i, i.e. $\chi(P_i)_e = 1$ if and only if $e \in P_i$).

Proof. Apply Lemma 4.23 to the x_1-x_ℓ subpath $P_{[x_1, x_\ell]}$ of P and obtain vertex disjoint subpaths $Q_1, \ldots, Q_m$. We denote $X_j \coloneqq V(Q_j)$ for all $j \in \{1, \ldots, m\}$. Every X_j contains at least $k/16$ and at most $k/2$ vertices.

Next, we define an s-t-flow f in G by

$$f_e = \begin{cases} \frac{1}{|X_1|} & \text{if } e = (s, x) \text{ with } x \in X_1, \\ \frac{1}{|X_j| \cdot |X_{j+1}|} & \text{if } e = (x, y) \text{ with } x \in X_j, y \in X_{j+1}, j \in \{1, \ldots, m-1\}, \\ \frac{1}{|X_m|} & \text{if } e = (x, t) \text{ with } x \in X_m, \\ 0 & \text{else,} \end{cases}$$

for all $e \in E(G)$. One easily sees that f indeed satisfies the flow conservation rule. We observe that its value is $\sum_{x \in X_1} \frac{1}{|X_1|} = 1$. The support of f, i.e., $\operatorname{supp}(f) \coloneqq \{e \in E(G) : f_e > 0\}$, is acyclic by construction. Therefore we can apply flow decomposition to f and $(V, \operatorname{supp}(f))$ yielding paths $P_1, \ldots, P_r$ and $\xi_1, \ldots, \xi_r \in [0,1]$ such that $\sum_{i=1}^{r} \xi_i = 1$ and

$$f = \sum_{i=1}^{r} \xi_i \cdot \chi(P_i)$$

for all $e \in E(G)$. Since each X_j contains at most $k/2$ vertices, $\operatorname{supp}(f)$ only contains edges (x_i, x_j) with $j - i < k$. The remaining properties (4.10) now follow from the definition of f using that all X_j contain at least $k/16$ vertices. □

We are now finally able to prove that the k-hop cost and the expected cost of an a priori tour only differ by some constant factor (and a scaling factor of δ^2):

Theorem 4.25. *Let (V, c, p) be an instance of Asymmetric A Priori TSP, such that $n \geq 2^6$ and $p(v) = \delta$ for all $v \in V$, where $n \coloneqq |V|$ and $\delta \in \left[\frac{2}{n}, \frac{1}{2^5}\right]$. Let T be a Hamiltonian cycle on V. Then, we have*

$$\mathbb{E}_{A \sim p}\left[c\left(T[A]\right)\right] \leq 2^{13} \delta^2 \cdot c^{(k)}(T) \tag{4.11}$$

and

$$\delta^2 \cdot c^{(k)}(T) \leq 4 \cdot \mathbb{E}_{A \sim p}\left[c\left(T[A]\right)\right], \tag{4.12}$$

where $k \coloneqq \left\lfloor \frac{1}{\delta} \right\rfloor$.

Proof. First note that $k \leq 1/\delta \leq n/2 \leq n - 2$ and $k \geq 1/\delta - 1 \geq 1/2\delta \geq 2^4$. Enumerate the vertices of V as $v_1, \ldots, v_n$ in the order that they appear on T. An edge $e = (v_j, v_{j+i})$ (where we

set $v_j = v_{j-n}$ for $j > n$) is part of T after cutting it short to the active vertices A if and only if v_j and v_{j+i} are active, but the vertices in between are not, i.e.

$$\delta^2 \cdot (1-\delta)^{i-1} = \mathbb{P}_{A\sim p}\left[e \in E(T[A])\right]. \tag{4.13}$$

Therefore,

$$\begin{aligned} c^{(k)}(T) &= \sum_{i=1}^{k}\sum_{j=1}^{n} c(v_j, v_{j+i}) \\ &\leq \sum_{i=1}^{k}\sum_{j=1}^{n} c(v_j, v_{j+i}) \cdot \left(1-\frac{1}{k}\right)^i \cdot 4 \\ &\leq 4 \cdot \sum_{i=1}^{n-1}\sum_{j=1}^{n} c(v_j, v_{j+i}) \cdot (1-\delta)^{i-1} \\ &\overset{(4.13)}{\leq} \frac{4}{\delta^2} \cdot \mathbb{E}_{A\sim p}\left[c\left(T[A]\right)\right]. \end{aligned}$$

The first inequality follows from the fact that $k \geq 2$, $i \leq k$ and thus $\left(1-\frac{1}{k}\right)^i \geq \left(1-\frac{1}{k}\right)^k \geq \left(1-\frac{1}{2}\right)^2 = \frac{1}{4}$. This shows (4.12).

In order to show (4.11), we can bound the expected cost after shortcutting inactive vertices by

$$\begin{aligned} \mathbb{E}_{A\sim p}\left[c\left(T[A]\right)\right] &\overset{(4.13)}{=} \sum_{i=1}^{n-1}\sum_{j=1}^{n} c(v_j, v_{j+i}) \cdot \delta^2 \cdot (1-\delta)^{i-1} \\ &\leq \delta^2 \cdot \left(\sum_{i=1}^{k}\sum_{j=1}^{n} c(v_j, v_{j+i}) + \sum_{i=k+1}^{n-1}\sum_{j=1}^{n} c(v_j, v_{j+i}) \cdot (1-\delta)^{i-1}\right). \end{aligned} \tag{4.14}$$

The first part of this sum is just $c^{(k)}(T)$, but we still have to deal with the second part of the sum, i.e. the cost of the edges that skip at least k vertices on the tour (and are thus not counted in $c^{(k)}(T)$). Let $i \in \{k+1, \dots, n-1\}$ and $e = (v_j, v_{j+i})$ be fixed. We denote by $P_e \coloneqq T_{[v_j, v_{j+i}]}$ the subpath from v_j to v_{j+i} on T. Since P_e is a path on at least $k+2$ vertices, we can apply Lemma 4.24 to P_e and obtain paths $Q_1^{(e)}, \dots, Q_{r_e}^{(e)}$ and $\xi_1^{(e)}, \dots, \xi_{r_e}^{(e)}$ with a vector $\chi^{(e)} \coloneqq \sum_{j=1}^{r_e} \xi_j^{(e)} \cdot \chi(Q_j^e)$. By the triangle inequality,

$$c(e) = \sum_{j=1}^{r_e} \xi_j^{(e)} c(e) \leq \sum_{j=1}^{r_e} \xi_j^{(e)} c\left(Q_j^{(e)}\right) = \sum_{j=1}^{r_e} \xi_j^{(e)} c^\intercal \chi\left(Q_j^{(e)}\right) = c^\intercal \chi^{(e)}, \tag{4.15}$$

where we interpret $c \in \mathbb{R}^E_{\geq 0}$ by $c_e = c(e)$ for each $e \in V \times V$.

Now, let $i \in \{k+1, \dots, n-1\}$ and $f = (v_j, v_{j+\Delta})$ for some $\Delta \leq k$ be fixed. We are aiming to find an upper bound on $\sum_{\ell=1}^{n} \chi_f^{(v_\ell, v_{\ell+i})}$. To this end, consider edges $e = (v_\ell, v_{\ell+i})$. There is exactly one such edge e such that e and f end in the same vertex (namely for $\ell = j + \Delta - i$). In this case we have $\chi_f^{(e)} \leq 2^4/k$ by (4.10). By symmetry, the same argument applies in the case that e and f start in the same vertex. Furthermore there are at most $i-2$ edges $e = (v_\ell, v_{\ell+i})$ such that $v_j, v_{j+\Delta} \in V(P_e)$, but $f = (v_j, v_{j+\Delta})$ and e do not share any common vertices. In

these cases, we can bound $\chi_f^{(e)} \leq 2^8/k^2$ by (4.10). For all other choices of e we have $\chi_f^{(e)} = 0$. Hence,

$$\begin{aligned}\sum_{\ell=1}^{n} \chi_f^{(v_\ell, v_{\ell+i})} &\leq 2 \cdot \frac{2^4}{k} + (i-2) \cdot \frac{2^8}{k^2} \\ &\leq 2^6\delta + (i-2) \cdot 2^{10}\delta^2\end{aligned} \tag{4.16}$$

We can use this to bound

$$\begin{aligned}&\frac{1}{2} \cdot \sum_{i=k+1}^{n-1} \sum_{\ell=1}^{n} c(v_\ell, v_{\ell+i}) \cdot (1-\delta)^{i-1} \\ &\leq \sum_{i=k+1}^{n-1} \sum_{\ell=1}^{n} c(v_\ell, v_{\ell+i}) \cdot (1-\delta)^{i} \\ &\overset{(4.15)}{\leq} \sum_{i=k+1}^{n-1} \sum_{\ell=1}^{n} c^\intercal \chi^{(v_\ell, v_{\ell+i})} \cdot (1-\delta)^{i} \\ &= \sum_{\Delta=1}^{k} \sum_{j=1}^{n} c(v_j, v_{j+\Delta}) \cdot \left(\sum_{i=k+1}^{n-1} \sum_{\ell=1}^{n} \chi^{(v_\ell, v_{\ell+i})}_{(v_j, v_{j+\Delta})} (1-\delta)^{i} \right) \\ &\overset{(4.16)}{\leq} \sum_{\Delta=1}^{k} \sum_{j=1}^{n} c(v_j, v_{j+\Delta}) \cdot \left(\sum_{i=k+1}^{n-1} \left(2^6\delta + (i-2) \cdot 2^{10}\delta^2\right) (1-\delta)^{i} \right) \\ &\leq \sum_{\Delta=1}^{k} \sum_{j=1}^{n} c(v_j, v_{j+\Delta}) \cdot 2^{10} \cdot \left(\delta \cdot \sum_{i=0}^{\infty} (1-\delta)^{i} + \delta^2 \cdot \sum_{i=1}^{\infty} i\,(1-\delta)^{i} \right) \\ &= \sum_{\Delta=1}^{k} \sum_{j=1}^{n} c(v_j, v_{j+\Delta}) \cdot 2^{10} \cdot \left(\delta \cdot \frac{1}{1-(1-\delta)} + \delta^2 \cdot \frac{(1-\delta)}{(1-(1-\delta))^2} \right) \\ &\leq \sum_{\Delta=1}^{k} \sum_{j=1}^{n} c(v_j, v_{j+\Delta}) \cdot 2^{11} \\ &= 2^{11} \cdot c^{(k)}(T),\end{aligned}$$

where the penultimate equality follows from well known identities about geometric respectively arithmetico-geometric series. Plugging this into (4.14) yields (4.11). □

4.5.3 Reducing to well-scaled instances of Hop-ATSP

We write $\operatorname{diam}(G, c)$ to denote the diameter, i.e., the maximum length of a shortest path in a graph G with respect to edge costs c. If G is complete and c satisfies the triangle inequality, this is the maximum cost of an edge.

Our next and final goal is to round the edge costs $c(e)$ to integral and polynomially bounded values. We refer to this property as *well-scaled*:

Definition 4.26 (Well-scaled instances)**.** *We say that a Hop-ATSP instance (V, c, k) is* well-scaled *if c takes only integer values, $\operatorname{diam}(G, c) \leq 2n^3$, and $\operatorname{OPT}(V, c, k) \geq n^2$, where G is the complete directed graph on V and $n \coloneqq |V|$.*

Reducing to such instances is rather straightforward and induces only a constant-factor loss (in addition to a scaling factor K):

Proposition 4.27. *Let (V, c, k) be an instance of Hop-ATSP with $\operatorname{diam}(G, c) > 0$ and $k \leq n-1$, where G is the complete directed graph on V and $n \coloneqq |V|$. Then, one can compute a well-scaled instance $(V, \hat{c}, k)$ of Hop-ATSP in $\mathcal{O}(n^3)$, such that*

$$\hat{c}^{(k)}(T) \leq \frac{c^{(k)}(T)}{K} \leq 2\hat{c}^{(k)}(T) \tag{4.17}$$

for any Hamiltonian cycle T on V, where $K \coloneqq \frac{\operatorname{diam}(G,c)}{2n^3}$.

Proof. We set

$$\tilde{c}(e) \coloneqq \left\lfloor \frac{c(e)}{K} \right\rfloor$$

for all $e \in E(G)$. Then, set $\hat{c}$ to be the shortest path metric with respect to $(G, \tilde{c})$. We denote by $P_{x,y}$ the shortest path in G with respect to $\tilde{c}$. Note that

$$\operatorname{diam}(G, \hat{c}) = \operatorname{diam}(G, \tilde{c}) \leq \frac{\operatorname{diam}(G, c)}{K} = 2n^3.$$

Let T be an arbitrary but fixed Hamiltonian cycle on V and enumerate the vertices of V as $v_1, \dots, v_n$ in the order of T. Define $F \coloneqq \{(v_i, v_{\Delta+i}) : \Delta \in [k], i \in [n]\}$ where $v_j = v_{j-n}$ for $j > n$. In particular, $c^{(k)}(T) = c(F)$. Then, we can bound

$$\begin{aligned} K \cdot \hat{c}(F) = \sum_{(x,y)\in F} K \cdot \tilde{c}(P_{x,y}) &\leq \sum_{(x,y)\in F} K \cdot \tilde{c}(x, y) \\ &\leq \sum_{(x,y)\in F} c(x, y) \\ &= c(F). \end{aligned}$$

Moreover, $|F| = nk \leq n^2$ because T is a Hamiltonian cycle. Hence,

$$\begin{aligned} K \cdot \hat{c}(F) = \sum_{(x,y)\in F} \sum_{e\in E(P_{x,y})} K \cdot \tilde{c}(e) &\geq \sum_{(x,y)\in F} \sum_{e\in E(P_{x,y})} (c(e) - K) \\ &\geq \sum_{(x,y)\in F} (c(x, y) - nK) \\ &\geq c(F) - n^3 K \\ &\geq c(F)/2, \end{aligned}$$

because $c(F) = c^{(k)}(T) \geq \operatorname{diam}(G, c) = 2Kn^3$ by the triangle inequality. In particular, we have $\hat{c}(F) \geq \frac{c(F)}{K} - n^3 \geq n^3 \geq n^2$. By Lemma 4.21, this implies that $\operatorname{OPT}(V, \hat{c}, k) \geq n^2$ and $(V, \hat{c}, k)$ is indeed a well-scaled instance. □

Putting everything together, we can now prove Theorem 4.5, which we restate here for convenience:

Theorem 4.5. *Let $\alpha \colon \mathbb{N} \to \mathbb{R}_{\geq 1}$ and $t \colon \mathbb{N} \to \mathbb{N}$ monotonically increasing. Suppose we have an algorithm that computes for every well-scaled instance $\mathcal{I} = (V, c, k)$ of Hop-ATSP in time $t(n)$ a tour T with $c^{(k)}(T) \leq \alpha(n) \cdot \operatorname{OPT}(\mathcal{I})$, where $n = |V|$.*

Then there is an algorithm that computes for every instance $\mathcal{J} = (V, c, p)$ of the Asymmetric A Priori TSP in time $n \cdot t(\text{poly}(n)) + \text{poly}(n)$ a tour T with

$$\mathbb{E}_{A\sim p}\big[c(T[A])\big] \;\leq\; \mathcal{O}(\alpha(5n^2)) \cdot \text{OPT}(\mathcal{J}),$$

where $n = |V|$ and $\text{poly}(n)$ denotes some polynomial in n.

Proof. If we can bound the total activation probability by $\sum_{v\in V} p(v) \leq \frac{1}{2}$, then by Lemma 2.8 any tour on V is a 2-approximation. Otherwise, by Lemma 2.6, there exists a vertex $v \in V$ such that we lose only a factor 5 in the approximation guarantee by changing the activation probability of v to 1. Let $\mathcal{J}_v = (V, c, p_v)$ be the instance that arises from $\mathcal{J}$ by setting $p_v(v) = 1$ and $p_v(w) = p(w)$ for $w \neq v$. We will perform the following steps for each possible choice of v and in the end choose the best. This costs us a factor $n := |V|$ in the running time.

Next, we need to get rid of vertices with activation probability less than $\frac{1}{n}$. Split $\mathcal{J}_v$ into two subinstances: $\mathcal{J}'_v$ with vertex set $V'_v := \{v\} \cup \{w \in V : p_v(w) < \frac{1}{n}\}$ and $\hat{\mathcal{J}}_v$ with vertex set $\hat{V}_v := \{w \in V : p_v(w) \geq \frac{1}{n}\}$. By Proposition 4.11 any tour on V' has expected cost at most $6p_v(V') \cdot \text{OPT}(\mathcal{J}'_v) \leq 6 \cdot (n \cdot \frac{1}{n} + 1) \cdot \text{OPT}(\mathcal{J}'_v) \leq 12 \cdot \text{OPT}(\mathcal{J}_v)$.

If $|\hat{V}_v| \leq 2^5$, we find the best a priori tour in constant time by enumeration. Otherwise, we then transform $\hat{\mathcal{J}}_v$ into an instance $\overline{\mathcal{J}_v} = (\overline{V_v}, \overline{c}, \overline{p})$ with uniform activation probabilities $\overline{p}(w) = \frac{1}{2n}$ for all $w \in V$, as described in Proposition 2.3, using $\varepsilon = \frac{1}{n}$. While doing so, we make sure to replace v (i.e. a vertex with $p(v) = 1$) by at least $4n$ vertices. Together with the bound given in Proposition 2.3, we get $\overline{n} := |\overline{V_v}| \leq 4n + 4|\hat{V}_v| \cdot n \leq 5n^2$.

We define $k := 2n$ and interpret $(\overline{V_v}, \overline{c}, k)$ as an Hop-ATSP instance. Since we made sure that $\overline{n} \geq 4n$, we get $k \leq \frac{\overline{n}}{2}$. Moreover we know that $n \geq |\hat{V}_v| > 2^5$, and thus $\overline{n} \geq 2^6$. Hence for $\delta := \frac{1}{k}$ we get $\frac{2}{\overline{n}} \leq \delta \leq \frac{1}{2^5}$ and therefore we can later apply Theorem 4.25 to $(\overline{V_v}, \overline{c}, k)$ with $\delta = \frac{1}{k}$.

Using Proposition 4.27, we find a cost function $\tilde{c}$ such that $(\overline{V_v}, \tilde{c}, k)$ is well-scaled and $\tilde{c}^{(k)}(T) \leq \frac{\overline{c}^{(k)}(T)}{K} \leq 2\tilde{c}^{(k)}(T)$ for any Hamiltonian cycle T on $\hat{V}_v$ (where $K := \text{diam}(G, \overline{c})/2\overline{n}^3$ and G is the complete directed graph on $\overline{V_v}$).

Now let T^{hop} be an optimum Hop-ATSP-tour in the Hop-ATSP instance $(\overline{V_v}, \tilde{c}, k)$ and T^* be an optimum a priori tour in $\overline{\mathcal{J}_v}$. Suppose we can compute a tour T in time $t(\overline{n})$ with $\tilde{c}^{(k)}(T) \leq \alpha(\overline{n}) \cdot \tilde{c}^{(k)}(T^{\text{hop}})$. Then by Theorem 4.25 we have

$$\begin{aligned}
\mathbb{E}_{A\sim\overline{p}}\left[\overline{c}(T[A])\right] &\leq 2^{13}\delta^2 \cdot \overline{c}^{(k)}(T) \\
&\leq 2^{13}\delta^2 \cdot 2K \cdot \tilde{c}^{(k)}(T) \\
&\leq 2^{13}\delta^2 \cdot 2K \cdot \alpha(\overline{n}) \cdot \tilde{c}^{(k)}(T^{\text{hop}}) \\
&\leq 2^{13}\delta^2 \cdot 2K \cdot \alpha(\overline{n}) \cdot \tilde{c}^{(k)}(T^*) \\
&\leq 2^{13}\delta^2 \cdot 2 \cdot \alpha(\overline{n}) \cdot \overline{c}^{(k)}(T^*) \\
&\leq 2^{13} \cdot 2 \cdot \alpha(\overline{n}) \cdot 4 \cdot \mathbb{E}_{A\sim\overline{p}}\left[\overline{c}(T^*[A])\right].
\end{aligned}$$

Hence we can compute a $(2^{16} \cdot \alpha(\overline{n}))$-approximate a priori tour in $\overline{\mathcal{J}_v}$. By Proposition 2.3 we lose a factor of 4 when transforming this back to a tour in $\hat{\mathcal{J}}_v$, i.e. we get a $(2^{18} \cdot \alpha(\overline{n}))$-approximate a priori tour in $\hat{\mathcal{J}}_v$. Together with the tour of cost at most $12 \cdot \text{OPT}(\mathcal{J}_v)$ for $\mathcal{J}'_v$, we therefore get a tour of cost at most $(2^{18} \cdot \alpha(\overline{n}) + 12) \cdot \text{OPT}(\mathcal{J}_v)$. Additionally respecting the factor 5 that we lost when declaring one vertex as a depot by Lemma 2.6, we end up with a $5 \cdot (2^{18} \cdot \alpha(\overline{n}) + 12)$-approximation for the Asymmetric A Priori TSP. We can bound $5 \cdot (2^{18}) \cdot \alpha(\overline{n}) + 12 \leq 5 \cdot (2^{18} + 12) \cdot \alpha(\overline{n}) \leq 2^{21} \cdot \alpha(\overline{n})$, and we already established $\overline{n} \leq 5n^2$. This proves the statement about the approximation guarantee of the algorithm for Asymmetric A Priori TSP.

For the running time we need $t(\overline{n})$ time to find T, and time that is polynomial in n for the transformation steps where we introduce uniform activation probabilities and a cost-function for a well-scaled Hop-ATSP instance. Since we might need to perform these steps n times to find the best vertex v to declare as a depot, and because $\overline{n} \leq 5n^2$, we end up with the claimed running time.

□

4.6 Reducing to hierarchically ordered instances

In this section we present our reduction of well-scaled instances of Hop-ATSP to hierarchically ordered instances. We first recall the definition of hierarchically ordered instances and introduce some useful notation in this context (Section 4.6.1). Then we describe the known results on directed low-diameter decompositions from [BNW22; BFHL25; Li25] that we will use (Section 4.6.2), and finally we provide our reduction, i.e., the proof of Theorem 4.6 (Section 4.6.3).

4.6.1 Hierarchically ordered instances

First, recall that in a hierarchically ordered instance we have a hierarchical partition $\mathcal{H} = (\mathcal{H}_\ell)_{\ell=1}^L$, consisting of partitions $\mathcal{H}_1, \dots, \mathcal{H}_L$, where each partition $\mathcal{H}_\ell$ is a refinement of the partition $\mathcal{H}_{\ell-1}$.

Definition 4.28 (Hierarchical partition)**.** *A* hierarchical partition *of a finite set V is a family $(\mathcal{H}_\ell)_{\ell=1}^L$ of partitions of V (for some depth $L \in \mathbb{N}$) such that*

(i) $\mathcal{H}_1 = \{V\}$ is the trivial partition into a single set,

(ii) $\mathcal{H}_L = \{\{v\} : v \in V\}$ is the trivial partition into singletons, and

(iii) for all $1 < \ell \leq L$ and $H \in \mathcal{H}_\ell$, there exists a set $H' \in \mathcal{H}_{\ell-1}$ with $H \subseteq H'$.

We say that $A \subseteq V$ is ℓ-confined *if there exists $H \in \mathcal{H}_\ell$ with $A \subseteq H$. We define* $\mathrm{level}(A)$ *as the maximum $\ell \in [L]$ such that A is ℓ-confined. For edges $e = (x, y)$ we simply abbreviate* $\mathrm{level}(e) \coloneqq \mathrm{level}(\{x, y\})$*. For a vertex $v \in V$ and a level $\ell \in [L]$, we denote by $H_\ell(v)$ the unique set $H \in \mathcal{H}_\ell$ such that $v \in H$.*

In a hierarchically ordered instance we also have a total order $\prec$ on the vertex set V.

Definition 4.29 (Compatible partition)**.** *We say that a partition $\{V_1, \dots, V_m\}$ of V is* compatible *with a total order $\prec$ on V if for all $i \in [m]$ the set V_i is a set of vertices appearing consecutively in the order $\prec$. A hierarchical partition $\mathcal{H} = (\mathcal{H}_\ell)_{\ell=1}^L$ is compatible with $\prec$ if the partition $\mathcal{H}_\ell$ is compatible with $\prec$ for all $\ell \in [L]$.*

Then a hierarchically ordered instance of Hop-ATSP is defined as follows.

Definition 4.30 (Hierarchically ordered instance)**.** *A* hierarchically ordered instance *of Hop-ATSP is a tuple $(V, L, \mathcal{H}, D, \prec, c, k)$, where*

- *(V, c, k) is an instance of Hop-ATSP,*
- *$L \in \mathbb{N}$ is the* depth*,*
- *$(\mathcal{H}_\ell)_{\ell=1}^L$ is a hierarchical partition of V,*
- *$D \in \mathbb{Z}^L$ with $D_\ell \geq 2 \cdot D_{\ell+1}$ for each $\ell \in [L-1]$,*
- *$\prec$ is a total order on V such that $\mathcal{H}$ is compatible with $\prec$, and*
- *for every edge $e = (x, y) \in V \times V$, we have $c(e) \leq D_{\mathrm{level}(e)}$ if e is a forward edge, i.e., $x \prec y$, and we have $c(e) = D_{\mathrm{level}(e)}$, if e is a backward edge, i.e., $y \prec x$.*

The goal of this section is to provide a reduction of well-scaled instances of Hop-ATSP to instances of hierarchically ordered instances with depth $L = \mathcal{O}(\log n)$, where $n := |V|$. The reduction will have a poly-logarithmic overhead in the approximation ratio and a polynomial overhead in the running time.

4.6.2 Directed low-diameter decompositions

We will prove Theorem 4.6 by repeatedly applying directed low-diameter decompositions. Directed low-diameter decompositions are defined for general digraphs G with nonnegative cost function on the edges. In this general setting, it is important to distinguish between the weak diameter and the strong diameter of a subset S of vertices of G, where the strong diameter is the diameter of the subgraph $G[S]$ induced by S and the weak diameter is the maximum distance of two vertices from S in the original graph G. (The weak diameter can be smaller than the strong diameter if every shortest path between two vertices in S uses vertices outside of S.) In complete graphs where the edge costs satisfy the triangle inequality, there is no difference between the two diameter notions and the diameter of a graph is the same as the maximum cost of an edge. Because we will only use low-diameter decompositions in this setting, we will not have to worry about the difference between weak and strong diameter in what follows.

Directed low-diameter decompositions allow us to determine a random set F of edges whose removal ensures that every strongly connected component has bounded (weak) diameter, where the probability of an edge e being included in F depends on $c(e)$.

Definition 4.31 (Directed low-diameter decomposition)**.** *Let $G = (V, E)$ be a directed graph and $c\colon E \to \mathbb{Z}_{\geq 0}$. Given positive integers α and D, a* directed low-diameter decomposition with loss α *is a collection of edge sets $\mathcal{F} \subseteq 2^E$ such that*

(i) for all $F \in \mathcal{F}$, every strongly connected component of $(V, E \setminus F)$ has weak diameter at most D,

(ii) for all $e \in E$ we have $\mathbb{P}[e \in F] \leq \alpha \cdot \frac{c(e)}{D}$, where F is drawn uniformly at random from $\mathcal{F}$.

In our reduction we will compute such a directed low-diameter decomposition $\mathcal{F}$ for the complete directed graph (V, E) on the vertex set V with edge cost c, and then sample a set $F \in \mathcal{F}$. We will choose the partition $\mathcal{H}_2$ corresponding to the strongly connected components of $(V, E \setminus F)$ and continue to iteratively refine each set of the current partition, leading eventually to the complete hierarchical partition $\mathcal{H}$. Edges in F correspond to edges that might become backward edges in our hierarchically ordered instance.

Directed low-diameter decompositions have been developed in [BNW22] and improved and simplified versions have been given in [BFHL25; Li25]. The analogous concept of low-diameter decompositions in undirected graphs has already been introduced more than 40 years ago in [Awe85]. In the undirected setting, matching upper and lower bounds of $\Theta(\log n)$ for the achievable loss are known (see e.g. [Bar96; FRT04]). As [BFHL25] remark, the lower bound of $\Omega(\log n)$ for the loss can be transferred to directed low-diameter decompositions. Moreover, they prove:

Theorem 4.32 ([BFHL25])**.** *Let $P(n)$ be a polynomial. Then there is a polynomial $Q_{\mathrm{LDD}}(n)$ such that the following holds: There is a deterministic polynomial-time algorithm that computes for any given G, c and $D \leq P(n)$ as in Definition 4.31 a directed low-diameter decomposition $\mathcal{F}$ with loss at most $\mathcal{O}(\log n \log\log n)$ and $|\mathcal{F}| = Q_{\mathrm{LDD}}(n)$, where $n := |V(G)|$.*

4.6.3 Proof of Theorem 4.6

We now prove Theorem 4.6, which we restate here for convenience.

Theorem 4.6. *Given a well-scaled instance $\mathcal{I} = (V, c, k)$ of Hop-ATSP with $n = |V|$ vertices, we can in polynomial time compute a polynomial number of hierarchically ordered instances $\mathcal{J}_1, \dots, \mathcal{J}_m$, where for each $i \in [m]$ we have $\mathcal{J}_i = (V, L, \mathcal{H}^{(i)}, D, \prec_i, \tilde{c}_i, k)$ with $L = \mathcal{O}(\log n)$ and*

$$\tilde{c}_i(v, w) \geq c(v, w) \text{ for all } v, w \in V.$$

Moreover, we require that there exists an index $i^* \in [m]$ such that

$$\mathrm{OPT}(\mathcal{J}_{i^*}) \leq L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot \mathrm{OPT}(\mathcal{I}).$$

In order to guarantee the existence of the index $i^* \in [m]$ in Theorem 4.6, we give a randomized algorithm that chooses an index $z \in [m]$ uniformly at random and constructs a hierarchically ordered instance $\mathcal{J} = (V, L, \mathcal{H}, D, \tilde{c}, k)$ based on the choice of z. We will then later analyze that

$$\mathbb{E}[\mathrm{OPT}(\mathcal{J})] \leq L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot \mathrm{OPT}(\mathcal{I}),$$

which immediately implies a deterministic algorithm satisfying the properties of Theorem 4.6. The algorithm we use starts with the partition $\mathcal{H}_1 = \{V\}$ and iteratively refines it to partitions $\mathcal{H}_2, \dots, \mathcal{H}_L = \{\{v\} : v \in V\}$. Together with these partitions we will construct a total order $\prec_\ell$ on each partition $\mathcal{H}_\ell$, eventually leading to an order $\prec$ on V.

The algorithm proceeds as follows:

(1) Let G be the complete directed graph on V and set $\mathcal{H}_1 \coloneqq \{V\}$ and $D_1 \coloneqq \mathrm{diam}(G, c)$. Let $\prec_1$ be the trivial total order on $\mathcal{H}_1$.

(2) Define a sequence of integers by $D_\ell \coloneqq \lfloor \frac{D_{\ell-1}}{2} \rfloor$ and let $L \in \mathbb{N}$ be minimal such that $D_L = 0$.

(3) Draw $z \in \{1, \dots, m\}$ uniformly at random, where $m \coloneqq Q_{\mathrm{LDD}}(n)$.

(4) For $\ell = 2, \dots, L-1$, do the following:

- Initialize $\mathcal{H}_\ell = \emptyset$.
- Apply Theorem 4.32 to G with edge costs c and diameter bound D_ℓ to obtain a directed low-diameter decomposition $\mathcal{F}_\ell = \{F_\ell^1, \dots, F_\ell^m\}$.
- For every set $H \in \mathcal{H}_{\ell-1}$, let $G_H \coloneqq G[H] - F_\ell^z$ be the graph that results from the induced subgraph $G[H]$ by deleting the edges of F_ℓ^z with both endpoints in H, and fix a topological order of the strongly connected components of G_H. Then add the vertex sets of the strongly connected components of G_H to $\mathcal{H}_\ell$.
- Define a total order $\prec_\ell$ on $\mathcal{H}_\ell$ by defining $X \prec_\ell Y$ as follows:
  - If X and Y are contained in different parts $\mathcal{H}_{\ell-1}$, say $X \subseteq X' \in \mathcal{H}_{\ell-1}$ and $Y \subseteq Y' \in \mathcal{H}_{\ell-1}$, then $X \prec_\ell Y$ if and only if $X' \prec_{\ell-1} Y'$.
  - If X and Y are contained in the same part H of $\mathcal{H}_{\ell-1}$, then $X \prec_\ell Y$ if and only if and X is before Y in the fixed topological order of the strongly connected components of G_H.

(5) Define $\mathcal{H}_L = \{\{v\} : v \in V\}$ to be the partition of V into singletons and define $\prec$ to be any total order on V such that for any distinct sets $X, Y \in \mathcal{H}_{L-1}$ with $X \prec_{L-1} Y$, we have $x \prec y$ for all $x \in X$ and $y \in Y$.

In this algorithm we defined a total order $\prec_\ell$ on the parts of the partition $\mathcal{H}_\ell$ for $\ell = 1, \dots, L-1$. For our analysis it will be useful to also define an order $\prec_L$ on the singleton partition $\mathcal{H}_L$, which we define as $\{x\} \prec_L \{y\}$ if and only if $x \prec y$.

Having constructed the total order $\prec$ on V, the hierarchical partition $\mathcal{H} = (\mathcal{H}_\ell)_{\ell=1}^{L}$ and the vector $D \in \mathbb{Z}_{\geq 0}^{L}$, we define the cost function $\tilde{c}\colon V \times V \to \mathbb{Z}_{\geq 0}$ by

$$\tilde{c}(e) = \begin{cases} c(e) & \text{if } e \text{ is a forward edge} \\ D_{\mathrm{level}(e)} & \text{if } e \text{ is a backward edge.} \end{cases}$$

A key invariant of our algorithm is that the graph $G[H]$ with edge cost c has (weak) diameter at most D_ℓ for every set $H \in \mathcal{H}_\ell$: For $\ell \in \{1, L\}$ this is clear. If $\ell \in \{2, \dots, L-1\}$, then $H \in \mathcal{H}_\ell$ is the vertex set of a strongly connected component of $G[H'] - F_\ell^z$ for some $H' \in \mathcal{H}_{\ell-1}$. In particular, this is a subset of a strongly connected component of $G - F_\ell^z$ which has diameter at most D_ℓ (by the definition of a directed low-diameter decomposition).

Because G is a complete graph satisfying the triangle inequality, we have $c(e) \leq D_{\mathrm{level}(e)}$ for each edge e of G. In particular, this also implies $c(e) \leq \tilde{c}(e)$ for every edge $e \in V \times V$. We also note that since we consider a well-scaled instance, the maximum cost $c(e)$ of an edge $e \in V \times V$ is at most $2n^3$, implying $L \leq \log_2(\mathrm{diam}(G, c)) + 1 = \mathcal{O}(\log n)$.

In order to prove that $\mathcal{J} = (V, L, \mathcal{H}, D, \prec, \tilde{c}, k)$ is a hierarchically ordered instance, it remains to prove that $\tilde{c}$ satisfies the triangle inequality. All other properties of hierarchically ordered instances follow directly from the construction.

Lemma 4.33. *The function $\tilde{c}\colon V \times V \to \mathbb{Z}_{\geq 0}$ satisfies the triangle inequality.*

Proof. Let $u, v, w \in V$ be three distinct vertices. If $u \prec w$, then

$$\tilde{c}(u, w) = c(u, w) \leq c(u, v) + c(v, w) \leq \tilde{c}(u, v) + \tilde{c}(v, w).$$

Otherwise we have $u \succ w$ and thus $\tilde{c}(u, w) = D_\ell$ for $\ell := \mathrm{level}(u, v) < L$. In this case, $H_{\ell+1}(w) \prec_{\ell+1} H_{\ell+1}(u)$, implying that we have $H_{\ell+1}(v) \prec_{\ell+1} H_{\ell+1}(u)$ or $H_{\ell+1}(w) \prec_{\ell+1} H_{\ell+1}(v)$, implying $\tilde{c}(u, v) \geq D_\ell$ or $\tilde{c}(v, w) \geq D_\ell$. We conclude $\tilde{c}(u, w) = D_\ell \leq \tilde{c}(u, v) + \tilde{c}(v, w)$. □

In order to complete the proof of Theorem 4.6, the only thing left to show is

$$\mathbb{E}[\mathrm{OPT}(\mathcal{J})] \;\leq\; L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot \mathrm{OPT}(\mathcal{I}), \tag{4.18}$$

where $\mathcal{I} = (V, c, k)$ is the given well-scaled instance of Hop-ATSP. The following is the key lemma we use to show this property.

Lemma 4.34. *For every edge $e \in V \times V$, we have*

$$\mathbb{E}\big[\tilde{c}(e)\big] \leq L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot c(e) + 1.$$

Proof. Let B_ℓ be the event that e is a backward edge and $\mathrm{level}(e) = \ell$. Using $D_L = 0$ and $D_{L-1} = 1$, we obtain

$$\mathbb{E}\big[\tilde{c}(e)\big] \;= \mathbb{P}\big[e \text{ forward edge}\big] \cdot c(e) + \sum_{\ell=1}^{L} \mathbb{P}[B_\ell] \cdot D_\ell \;\leq\; c(e) + 1 + \sum_{\ell=1}^{L-2} \mathbb{P}\big[B_\ell\big] \cdot D_\ell.$$

Consider any $\ell \leq L - 2$. In our algorithm we apply Theorem 4.32 to G with edge costs c and diameter bound $D_{\ell+1}$ to obtain a directed low-diameter decomposition $\mathcal{F}_{\ell+1} = \{F_{\ell+1}^1, \dots, F_{\ell+1}^m\}$. Note that the event B_ℓ happens only if $e \in F_{\ell+1}^z$ (using our definition of $\prec_{\ell+1}$ and the fixed topological order on the strongly connected components). Since our algorithm draws $z \in \{1, \dots, m\}$ uniformly at random, Theorem 4.32 implies that

$$\mathbb{P}\big[B_\ell\big] \;\leq\; \mathbb{P}\big[e \in F_{\ell+1}^z\big] \;\leq\; \frac{c(e)}{D_{\ell+1}} \cdot \mathcal{O}(\log n \log\log n).$$

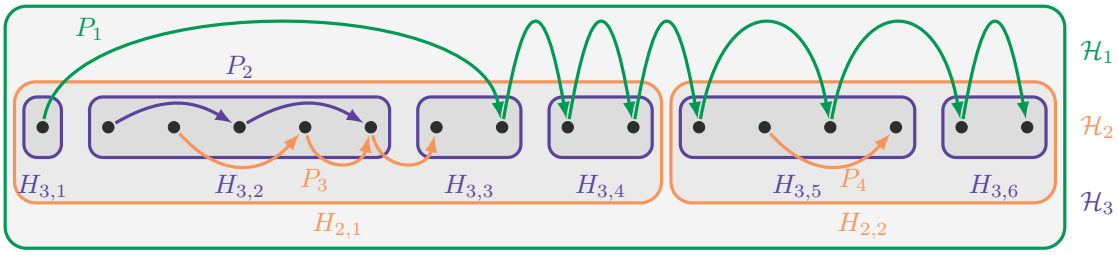


Figure 4.10: A path covering instance with $k = 3$ and $\mathcal{H} = (\mathcal{H}_\ell)_{\ell=1}^4$ where $\mathcal{H}_4$ is not shown; vertices are drawn from left to right according to $\prec$. Then $\mathcal{P} := \{(P_1, 1), (P_2, 3), (P_3, 2), (P_4, 2)\}$ is a feasible solution to the path covering problem: For each $v \in V$ there is $(P, \ell) \in \mathcal{P}$ with $v \in V(P)$. The path-level pair $(P_1, 1)$ takes responsibility for $H_{2,1}$ and $H_{2,2}$; the path-level pair $(P_3, 2)$ takes responsibility for $H_{3,2}$. For every vertex v in the remaining sets of $\mathcal{H}_3$ (i.e., $H_{3,1}$, $H_{3,3}$, $H_{3,4}$, $H_{3,5}$, and $H_{3,6}$), there is a pair $(P, j) \in \mathcal{P}$ with $j < 3$ and $v \in V(P)$.
Note that while $V(P_4) \subseteq H_{3,5}$, i.e., the level of P_4 is 3, we have to work with the path-level pair $(P_4, 2)$ since $H_{3,5}$ otherwise is not covered. Increasing the level in any path-level pair of $\mathcal{P}$ maintains the feasibility of $\mathcal{P}$, but can increase the weight of $\mathcal{P}$.

Since $\frac{D_\ell}{D_{\ell+1}} = \frac{D_\ell}{\lfloor \frac{D_\ell}{2} \rfloor} \leq 4$ for all $\ell \in [L-2]$, we obtain

$$\mathbb{E}\big[\tilde{c}(e)\big] \;\leq\; c(e) + 1 + \sum_{\ell=1}^{L-2} c(e) \cdot \mathcal{O}(\log n \log\log n) \;\leq\; 1 + L \cdot \mathcal{O}(\log n \log\log n) \cdot c(e). \qquad \square$$

Applying Lemma 4.34 to each of the kn edges contributing to the k-hop cost of a fixed optimum solution to the given Hop-ATSP instance $\mathcal{I}$, linearity of expectation implies

$$\mathbb{E}\big[\mathrm{OPT}(\mathcal{J})\big] \;\leq\; L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot \mathrm{OPT}(\mathcal{I}) + kn.$$

Because the instance $\mathcal{I}$ is well-scaled, we have $\mathrm{OPT}(\mathcal{I}) \geq n^2$, which implies (4.18). Hence, explicitly returning the set of instances obtained by trying all possible values $z \in [m]$ completes the proof of Theorem 4.6.

Therefore, in the following, we will focus on hierarchically ordered instances. In this context, we will use the following terminology:

Definition 4.35 (Left/right)**.** *We say that a vertex a is left of a vertex b if $a \prec b$, and a is right of b if $b \prec a$. A vertex a is left/right of a set $B \subseteq V$ if a is left/right of every element of B. If a is left or right of B, we also write $a \prec B$ or $B \prec a$, respectively. Moreover, a vertex set A is left/right of a vertex set B if every element of A is left/right of every element of B. If A is left or right of B, we also write $A \prec B$ or $B \prec A$, respectively.*

4.7 Reducing to a covering problem

In this section we provide a reduction from well-scaled instances of Hop-ATSP to a covering problem (Problem 4.9). We first introduce the covering problem (Section 4.7.1) and state the overall result of the section (Theorem 4.36). Next, we show that every solution to the covering problem can be turned into a Hop-ATSP solution without significant increase in the objective value (Section 4.7.2). The reverse direction is unfortunately not true in general, but we show that it is true for non-degenerate instances (Sections 4.7.4 to 4.7.8) and we can reduce general instances to non-degenerate ones (Section 4.7.3).

4.7.1 Our covering problem

Recall that a path is called monotone if it contains only forward edges. Moreover, a *path-level pair* is a pair (P, ℓ) where P is a monotone path, $\ell \in [L]$, and $V(P) \subseteq H$ for some $H \in \mathcal{H}_\ell$. Then the covering problem to which we reduce is defined as follows:

Problem 4.9: Path Covering

Given a hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP, find a set $\mathcal{P}$ of path-level pairs (P, ℓ) such that

(a) for every vertex $v \in V$ there is some pair $(P, \ell) \in \mathcal{P}$ with $v \in V(P)$, and

(b) for every level $\ell \in \{2, 3, \dots, L\}$ and every set $H \in \mathcal{H}_\ell$, one of the following applies:

(i) for every vertex $v \in H$ there is a pair $(P, j) \in \mathcal{P}$ with $j < \ell$ and $v \in V(P)$, or
(ii) there is a pair $(P, j) \in \mathcal{P}$ with $j < \ell$ and $|V(P) \cap H| \geq k$,

while minimizing the weight

$$w(\mathcal{P}) \coloneqq \sum_{(P,\ell)\in\mathcal{P}} w(P, \ell) = \sum_{(P,\ell)\in\mathcal{P}} \Big(c^{(k)}(P) + k^2 \cdot D_\ell\Big).$$

If a pair (P, j) with $j < \ell$ satisfies (ii) for a set $H \in \mathcal{H}_\ell$, we also say that the pair *takes responsibility* for the set H (see also Fig. 4.10; Fig. 4.5 shows why (b) is needed). We remark that, technically speaking, this is not well-defined in case the same set H appears in multiple partitions $\mathcal{H}_\ell$ for different levels ℓ. To guarantee uniqueness, one could extend this definition by representing elements of $\mathcal{H}_\ell$ explicitly as pairs (H, ℓ). However, to avoid cumbersome notation, we will assume that $H \subseteq V$ implicitly carries its level information (which will always be clear from the context).

The goal of this section is to provide a reduction from Hop-ATSP to Problem 4.9. Let $\gamma > 0$ be a constant such that $L \leq \gamma \cdot \log_2 n$ for every instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ resulting from Theorem 4.6. We say that a hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP has *low depth* if $L \leq \gamma \cdot \log_2 n$. Then the reduction we provide is the following:

Theorem 4.36. *Let $\alpha\colon \mathbb{N} \to \mathbb{R}$ and $t\colon \mathbb{N} \to \mathbb{N}$ be monotonically increasing. Suppose we have an algorithm that computes for every instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Path Covering with low depth, in time $t(n)$ a solution $\mathcal{P}$ with $w(\mathcal{P}) \leq \alpha(n) \cdot \mathrm{OPT}(\mathcal{I})$, where n denotes the number of vertices in $\mathcal{I}$.*

Then there is an algorithm that computes for every well-scaled instance $\mathcal{J} = (V, c, k)$ of Hop-ATSP in time $\mathrm{poly}(t(n) + n)$ a tour T such that $c^{(k)}(T) \leq \mathcal{O}(\log^6 n) \cdot \alpha(n) \cdot \mathrm{OPT}(\mathcal{J})$, where n denotes the number of vertices in $\mathcal{J}$ and $\mathrm{poly}(n)$ denotes some fixed polynomial.

It will be convenient to view Problem 4.9 as an instance of Set Cover. For a hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP, we define the *corresponding Set Cover instance* $(\mathcal{U}, \mathcal{S}, w)$ by defining

- $\mathcal{U} \coloneqq V \times [L]$,
- $\mathcal{S}$ to be the set of all path-level pairs,
- $w(P, \ell) \coloneqq c^{(k)}(P) + k^2 \cdot D_\ell$ for all $(P, \ell) \in S$,

and saying that a path-level pair (P, j) covers an element (v, ℓ) if

- $v \in V(P)$ and $j \leq \ell$, or

- (P, j) takes responsibility for $H_{\ell+1}(v)$, i.e, we have $|V(P) \cap H_{\ell+1}(v)| \geq k$ and $j \leq \ell < L$.

Then our Path Covering problem is indeed equivalent to the corresponding Set Cover problem, as the following lemma shows:

Lemma 4.37. *Let $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ be a hierarchically ordered instance of Hop-ATSP and $(\mathcal{U}, \mathcal{S}, w)$ the corresponding Set Cover instance. Then $\mathcal{P} \subseteq \mathcal{S}$ is a feasible solution to the Path Covering instance $\mathcal{I}$ if and only if every element of $\mathcal{U}$ is covered by some pair $P \in \mathcal{P}$.*

Proof. First consider a set $\mathcal{P} \subseteq \mathcal{S}$ covering each element of $\mathcal{U}$. We show that $\mathcal{P}$ is a feasible solution to the Path Covering problem. To show (a), let $v \in V$ and let (P, j) be a pair covering the element $(v, L) \in \mathcal{U}$. Then by the definition of the Set Cover instance $(\mathcal{U}, \mathcal{S}, w)$, we must have $v \in V(P)$. To show (b), let $\ell \in \{2, \dots, L\}$ and $H \in \mathcal{H}_\ell$. Suppose (i) is not satisfied, i.e., there exists a vertex $v \in H$ such that $j \geq \ell$ for every pair $(P, j) \in \mathcal{P}$ with $v \in V(P)$. Let $(P, j) \in \mathcal{P}$ be the pair covering the element $(v, \ell - 1) \in \mathcal{U}$. Then (P, j) takes responsibility for H.

Now consider a feasible solution $\mathcal{P}$ to the Path Covering problem. We show that $\mathcal{P}$ covers $\mathcal{U}$. To this end, let $(v, \ell) \in \mathcal{U}$. If $\ell = L$, then any pair $(P, j) \in \mathcal{P}$ with $v \in V(P)$ covers (v, ℓ) and such a pair (P, j) exists by (a). Otherwise, since (ii) applies to $H_{\ell+1}(v)$, there must be either a pair $(P, j) \in \mathcal{P}$ with $j \leq \ell$ and $v \in V(P)$, or a pair $(P, j) \in \mathcal{P}$ that takes responsibility for $H_{\ell+1}(v)$. In both cases the pair (P, j) covers the element (v, ℓ) of $\mathcal{U}$. □

4.7.2 Turning covering solutions into tours

In order to prove Theorem 4.36, we will first apply Theorem 4.6 to obtain a collection of hierarchically ordered instances. For such a single hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$, we show that any solution $\mathcal{P}$ to our covering problem can be turned into a tour T of cost $c^{(k)}(T) \leq 2w(\mathcal{P})$. The following lemma states a key argument for this first part of our proof. Condition (b) in our definition of the Path Covering problem plays a crucial role here. (Recall the example from Figure 4.5 to see why this is necessary.)

Lemma 4.38. *Let $W_{\text{main}} = w_1, \dots, w_t$ be a walk on V and $W_{\text{sub}} = w_{s+1}, \dots, w_{s+k}$ a subwalk of W_{main}. Suppose there is a set $H \in \mathcal{H}_\ell$ such that $V(W_{\text{sub}}) \subseteq H$ and let $Q = q_1, \dots, q_m$ be another walk with vertex set $V(Q) \subseteq H$. Define the walk*

$$W'_{\text{main}} \coloneqq w_1, \dots, w_s, q_1, \dots, q_m, w_{s+1}, \dots, w_{s+k}, \dots, w_t$$

that results from W_{main} by inserting Q right before the start of W_{sub}. Then we have

$$c^{(k)}(W'_{\text{main}}) \leq c^{(k)}(W_{\text{main}}) + c^{(k)}(Q) + 2k^2 D_\ell.$$

Proof. For a walk W with vertices $u_1, \dots, u_r$ we write $C(W, u_i) \coloneqq \sum_{\Delta=1}^{k} c(u_i, u_{i+\Delta})$ to denote the total cost of the outgoing edges of v_i that contribute to the k-hop cost of the walk W. (Here, $c(u_i, u_{i+\Delta})$ is defined as zero in case $i + \Delta > r$.) Then $c^{(k)}(W) = \sum_{i=1}^{r} C(W, u_i)$.

To prove the lemma, we therefore need to prove an upper bound on

$$\begin{aligned} &c^{(k)}(W'_{\text{main}}) - \Big(c^{(k)}(W_{\text{main}}) + c^{(k)}(Q)\Big) \\ &= \sum_{i=1}^{t} \Big(C(W'_{\text{main}}, w_i) - C(W_{\text{main}}, w_i)\Big) + \sum_{j=1}^{m} \Big(C(W'_{\text{main}}, q_j) - C(Q, q_j)\Big). \end{aligned}$$

We first observe that $C(W'_{\text{main}}, w_i) - C(W_{\text{main}}, w_i) = 0$ for all $i \in \{1, \dots, s-k\} \cup \{s+1, \dots, t\}$ because for each such vertex w_i, the k vertices following on w_i in the walk did not change when

inserting Q. The same argument shows that $C(W'_{\text{main}}, q_j) - C(Q, q_j) = 0$ for all $j \in \{1, \dots, m-k\}$. To complete the proof, we will show that each of the at most $2k$ remaining summands is at most $k \cdot D_\ell$.

Let $j \in \{m-k+1, \dots, m\}$. To prove that $C(W'_{\text{main}}, q_j) - C(Q, q_j) \leq k \cdot D_\ell$, we observe that the k vertices following on q_j in the walk W'_{main} are all contained in the set $H \in \mathcal{H}_\ell$ that contains q_j. Thus, each of the k edges contributing to $C(W'_{\text{main}}, q_j)$ has cost at most D_ℓ, implying $C(W'_{\text{main}}, q_j) \leq k \cdot D_\ell$. Because $C(Q, q_j) \geq 0$, we can conclude that indeed $C(W'_{\text{main}}, q_j) - C(Q, q_j) \leq k \cdot D_\ell$.

Now, let $i \in \{s-k+1, \dots, s\}$. We will prove $C(W'_{\text{main}}, w_i) - C(W_{\text{main}}, w_i) \leq k \cdot D_\ell$. Let $v_1, \dots, v_k$ be the k vertices following on w_i in the walk W_{main} (i.e., before insertion of Q) and let $v'_1, \dots, v'_k$ be the k vertices following on w_i in the walk W'_{main} (i.e., after insertion of Q). It suffices to show $c(w_i, v'_j) - c(w_i, v_j) \leq D_\ell$ for all $j \in [k]$. First, we observe

$$v'_1 = v_1 = w_{i+1}, \quad v'_2 = v_2 = w_{i+2}, \quad \dots \quad v'_{s-i} = v_{s-i} = w_s$$

and thus $c(w_i, v'_j) - c(w_i, v_j) = 0 \leq D_\ell$ for all $j \in \{1, \dots, s-i\}$.

Next, we observe that the vertices $v_{s-i+1}, \dots, v_k$ are all contained in $\{w_{s+1}, \dots, w_{s+k}\} \subseteq H$ and the vertices $v'_{s-i+1}, \dots, v'_k$ are all contained in $\{q_1, \dots, q_m, w_{s+1}, \dots, w_{s+k}\} \subseteq H$. Thus, for all $j \in \{s-i+1, \dots k\}$, we have $v_j, v'_j \in H \in \mathcal{H}_\ell$, implying

$$c(w_i, v'_j) \leq c(w_i, v_j) + c(v_j, v'_j) \leq c(w_i, v_j) + D_\ell,$$

and therefore $c(w_i, v'_j) - c(w_i, v_j) \leq D_\ell$. □

Using Lemma 4.38, we can now show how we can construct a sufficiently cheap tour from a given covering solution:

Lemma 4.39. *Given a Path Covering solution $\mathcal{P}$, we can construct a tour T on V with $c^{(k)}(T) \leq 2w(\mathcal{P})$ in polynomial time (in $|V|$ and $|\mathcal{P}|$).*

Proof. We may assume that $\mathcal{P}$ is inclusionwise minimal. For every pair $(P, \ell) \in \mathcal{P}$ with $\ell > 1$, there is a unique set $H \in \mathcal{H}_\ell$ containing the vertex set $V(P)$ of the path P. Because $\mathcal{P}$ is inclusionwise minimal, property (i) from Problem 4.9 cannot apply to H. (Otherwise, $\mathcal{P} \setminus \{(P, \ell)\}$ would also be a feasible Path Covering solution.) Therefore, property (ii) must apply, i.e., there exists a pair $(P', j) \in \mathcal{P}$ that takes responsibility for H. We then define $\text{parent}(P, \ell) \coloneqq (P', j)$. There might be several pairs taking responsibility for H, in which case we choose an arbitrary one as the parent of (P, ℓ). This defines a branching B on the node set $\mathcal{P}$ with arcs $\big(\text{parent}(P, \ell), (P, \ell)\big)$ for all pairs $(P, \ell) \in \mathcal{P}$ with $\ell > 1$. Every connected component of this branching is an arborescence with a root $(P, 1) \in \mathcal{P}$.

Throughout our algorithm, we will maintain a branching with node set $\mathcal{W}$, where initially $\mathcal{W} = \mathcal{P}$. We will maintain the following invariants:

(i) Every node is a pair (W, ℓ), where $\ell \in [L]$ and W is a walk completely contained in some set $H \in \mathcal{H}_\ell$.

(ii) A pair $(W, \ell) \in \mathcal{W}$ has an incoming arc if and only if $\ell > 1$.

(iii) For every pair $(Q, \ell) \in \mathcal{W}$ with parent (W, j), we have $j < \ell$ and the walk W visits at least k vertices from H consecutively, where H is the unique set $H \in \mathcal{H}_\ell$ containing the vertices of Q.

(iv) Every vertex $v \in V$ is contained in at least one walk W with $(W, \ell) \in \mathcal{W}$.

We start with the branching B defined above, which satisfies all these invariants. As long as our branching has at least one arc, we consider a pair (Q, ℓ) with maximum level ℓ. Then (Q, ℓ) is a leaf of our branching and it has an incoming arc. Let (W, j) be the parent of (Q, ℓ). By invariant (iii), we can insert Q into W by applying Lemma 4.38, obtaining a walk W'. We delete the leaf (Q, ℓ) and its incoming arc from our branching and replace the node (W, j) by (W', j), maintaining its incident arcs.

We now show that this operation maintains our invariants. For invariants (ii) and (iv) this is obvious from the construction. For invariant (i), let $H_j \in \mathcal{H}_j$ such that all vertices of W are contained in H_j, and let $H_\ell \in \mathcal{H}_\ell$ such that all vertices of Q are contained in H_ℓ. Because W contains vertices from H_ℓ and $j < \ell$, the hierarchical structure of our instance implies $H_\ell \subseteq H_j$. In particular, all vertices of the new walk W' are contained in H_j.

Invariant (iii) could be possibly violated only for outgoing arcs of the new node (W', j). Consider a child $(\tilde{Q}, \tilde{\ell})$ of (W', j). Then our choice of the leaf (Q, ℓ) implies $\tilde{\ell} \leq \ell$. Therefore, the unique set $\tilde{H} \in \mathcal{H}_{\tilde{\ell}}$ containing all vertices from $\tilde{Q}$ is either disjoint from the set $H \in \mathcal{H}_\ell$ containing all vertices of Q, or it is a superset of H. If $\tilde{H}$ is disjoint from H, any k vertices from $\tilde{H}$ that were previously consecutive in W are still consecutive in W'. If $\tilde{H}$ is a superset of H, we observe that W' still contains k consecutive vertices from H, and thus also from $\tilde{H}$.

Lemma 4.38 implies that in every iteration $c^{(k)}(W') \leq c^{(k)}(W) + c^{(k)}(Q) + 2k^2 \cdot D_\ell$, and hence the following quantity is non-increasing during the algorithm:

$$\sum_{(W,\ell)\in\mathcal{W}} \Big(c^{(k)}(W) + 2k^2 \cdot D_\ell \Big)$$

Initially, this value is bounded by $2w(\mathcal{P})$ and hence this will remain the case throughout the algorithm.

After less than $|\mathcal{P}|$ iterations, our branching has no edges anymore. Then the branching has vertices $(W_1, 1), (W_2, 1), \ldots, (W_r, 1)$ for some $r \in \mathbb{N}$, and we have $\sum_{i=1}^{r} \left(c^{(k)}(W_i) + k^2 \cdot D_1 \right) \leq 2 \cdot w(\mathcal{P})$. We concatenate the walks $W_1, \ldots, W_r$ to form a closed walk T. By invariant (iv), this is indeed a tour. Because $c(x, y) \leq D_1$ for all $x, y \in V$, we have

$$c^{(k)}(T) \;\leq\; \sum_{i=1}^{r} \Big(c^{(k)}(W_i) + k^2 \cdot D_1 \Big) \;\leq 2 \cdot w(\mathcal{P}). \qquad \square$$

4.7.3 Degenerate instances

Ideally, we would like to show that every tour T can be transformed into a Covering Solution $\mathcal{P}$ so that the weight $w(\mathcal{P})$ is not much larger than the k-hop cost $c^{(k)}(T)$. Note however, that $\mathcal{P}$ will contain at least one pair (P, ℓ) with level $\ell = 1$ and we therefore have $w(\mathcal{P}) \geq k^2 D_1$. In general the total cost $c^{(k)}(T)$ may be much smaller than $k^2 D_1$. To address this issue, we observe that it can only occur for particular instances, which we call *degenerate*. We will then provide a reduction from general instances of Path Covering to non-degenerate ones.

Definition 4.40 (Degenerate instance)**.** *We say that a hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP is* degenerate *if there exists a set $H \in \mathcal{H}_2$ with $|V \setminus H| \leq \frac{k}{2}$. Otherwise, it is called* non-degenerate*.*

We now show that for non-degenerate instances, we indeed have $c^{(k)}(T) = \Omega(k^2 D_1)$.

Lemma 4.41. *Let $\mathcal{I}$ be a non-degenerate hierarchically ordered instance of Hop-ATSP. Then we have*

$$c^{(k)}(T) \geq 2^{-8} \cdot k^2 D_1$$

for every tour T on V.

Proof. First, we claim that there exists a partition of V into sets A and B such that

- $|A| \geq \frac{k}{4}$ and $|B| \geq \frac{k}{4}$, and
- $c(y, x) = D_1$ for all $x \in A$ and $y \in B$.

To prove this claim, we distinguish two cases. First, consider the case where we have $|H| \geq \frac{k}{4}$ for some $H \in \mathcal{H}_2$. We define

$$V_{\text{left}} \coloneqq \{v \in V : v \text{ is left of } H\}$$
$$V_{\text{right}} \coloneqq \{v \in V : v \text{ is right of } H\}.$$

(Recall the notion of left/right from Definition 4.35.) Because $H \in \mathcal{H}_2$, every edge from V_{right} to H and every edge from H to V_{left} is a backward edge of level 1 and has therefore cost D_1. Because $V = V_{\text{left}} \cup H \cup V_{\text{right}}$ and the instance $\mathcal{I}$ is non-degenerate, we have $|V_{\text{left}}| \geq \frac{k}{4}$ or $|V_{\text{right}}| \geq \frac{k}{4}$. In the former case, we can choose $A \coloneqq V_{\text{left}}$ and $B \coloneqq H \cup V_{\text{right}}$, and in the latter case, we can choose $A \coloneqq V_{\text{left}} \cup H$ and $B \coloneqq V_{\text{right}}$.

Now consider the remaining case, where $|H| < \frac{k}{4}$ for all $H \in \mathcal{H}_2$. We number the elements of $\mathcal{H}_2 = \{H_1, \dots, H_m\}$ so that H_j is left of H_{j+1} for all $j \in [m-1]$, and we define

$$j^* \coloneqq \min \left\{ j \in [m] : \sum_{i=1}^{j} |H_i| \geq \tfrac{k}{2} \right\}.$$

Then we have $\sum_{j=1}^{j^*} |H_j| < \frac{3}{4}k$ because $|H_{j^*}| < \frac{k}{4}$. Therefore, choosing $A \coloneqq \bigcup_{i=1}^{j^*} H_i$ and $B \coloneqq V \setminus A$, we have $|A| \geq \frac{k}{4}$ and $|B| \geq k - \frac{3}{4}k = \frac{k}{4}$. Moreover, every edge starting in B and ending in A is a backward edge of level 1. This completes the proof of our claim.

Now let A, B be a partition of V as in our claim and let T be an arbitrary tour on V. By Lemma 4.21, we can assume that T is a Hamiltonian cycle. Let P be a path obtained from T by removing a single edge. We may assume $k \geq 16$ because otherwise $2^{-8} \cdot k^2 \leq 1$, in which case the statement of the lemma is trivially satisfied (because every tour contains at least one backward edge of level 1). Then we can find vertex disjoint subpaths $P_1, \dots, P_m$ of P with $\bigcup_{j=1}^{m} V(P_j) = V(P)$ and $\frac{k}{8} \leq |V(P_j)| \leq \frac{k}{2}$ for all $j \in [m]$ (cf. Lemma 4.23). Without loss of generality, we can assume that the subpaths are numbered according to their order along the path P. Then, because each subpath P_j contains at most $\frac{k}{2}$ vertices, the cost of every edge (x, y) with $x \in P_j$ and $y \in P_{j+1}$ contributes to the k-hop cost of the tour T. (Here, we define $P_{m+1} \coloneqq P_1$.) In particular, we have

$$c^{(k)}(T) \;\geq\; \sum_{i=1}^{m} \sum_{x \in V(P_i)} \sum_{y \in V(P_{i+1})} c(x, y),$$

where $P_{m+1} \coloneqq P_1$.

Let $j \in [m]$. We say that the subpath P_j is *A-dominated* if P_j contains at least $\frac{k}{16}$ vertices from A. Note that if P_j is not A-dominated, it is B-dominated since we have $|B \cap V(P_j)| \geq \frac{k}{16}$, because $|V(P_j)| \geq \frac{k}{8}$. Of course, P_j can be A-dominated and B-dominated simultaneously.

We now distinguish several cases. If each subpath P_j is A-dominated, then we claim that among the k vertices following on $y \in B$ in the tour T, there are at least $\frac{k}{16}$ vertices from A. Indeed, the vertex y is contained in some subpath P_j and all vertices from P_{j+1} are among the k vertices following on y in the tour T. Because P_{j+1} is A-dominated, at least $\frac{k}{16}$ of its vertices belong to A. This shows that the k outgoing edges of y that contribute to the k-hop cost of T have total cost at least $\frac{k}{16} \cdot D_1$. Summing this over all vertices $y \in B$, we get $c^{(k)}(T) \geq |B| \cdot \frac{k}{16} D_1 \geq 2^{-6} \cdot k^2 D_1$.

If each subpath P_j is B-dominated, we can apply a symmetric argument to see that among the k vertices preceding a vertex $x \in A$ in the tour T, there are at least $\frac{k}{16}$ vertices from B and deduce $c^{(k)}(T) \geq 2^{-6} \cdot k^2 D_1$ also in this case.

It remains to consider the case where at least one subpath is A-dominated and one is not A-dominated. Then, there exists an index $j \in [m]$ such that P_j is not A-dominated and P_{j+1} is A-dominated (where we define $P_{m+1} := P_1$). We obtain

$$c^{(k)}(T) \;\geq\; \sum_{i=1}^{m} \sum_{x \in V(P_i)} \sum_{y \in V(P_{i+1})} c(x,y) \;\geq\; \sum_{\substack{x \in V(P_j),\\ x \in B}} \sum_{\substack{y \in V(P_{j+1}),\\ y \in A}} c(x,y) \geq \left(\tfrac{k}{16}\right)^2 D_1.$$

□

Next, we reduce general instances to non-degenerate ones. In case we have a degenerate instance, we will iteratively remove vertices and shrink our hierarchical partition, until either the resulting instance is non-degenerate, or at most k vertices are remaining.

Lemma 4.42. *Let $\alpha\colon \mathbb{N} \to \mathbb{R}$ and $t\colon \mathbb{N} \to \mathbb{N}$ be monotonically increasing. Suppose we have an algorithm that computes for every non-degenerate hierarchically ordered instance $\mathcal{J}$ of Hop-ATSP with low depth, in time $t(n)$ a tour T with $c^{(k)}(T) \leq \alpha(n) \cdot \mathrm{OPT}(\mathcal{I})$, where n denotes the number of vertices in $\mathcal{I}$.*

Then there is an algorithm that computes for every hierarchically ordered instance $\mathcal{I}$ of Hop-ATSP with low depth, in time $t(n) + \mathcal{O}(nL)$ a tour T with $c^{(k)}(T) \leq (\alpha(n) + 4L) \cdot \mathrm{OPT}(\mathcal{I})$, where n denotes the number of vertices in $\mathcal{I}$.

Proof. We prove the statement by induction on the depth $L \geq 2$ of the hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$. For $L = 2$, we have $\mathcal{H}_1 = \{V\}$ and $\mathcal{H}_2 = \mathcal{H}_L = \{\{v\} : v \in V\}$; if $\mathcal{I}$ is degenerate, there is $H \in \mathcal{H}_2$ with $\frac{k}{2} \geq |V \setminus H| = |V| - 1$. This implies $2|V| - 2 \leq k \leq |V|$, i.e., $|V| \leq 2$. But then there is at most one tour in V. Hence, the statement is trivially satisfied for $L = 2$.

Now, suppose $L > 2$. If we are given a degenerate hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP, let $H^* \in \mathcal{H}_2$ with $|V \setminus H^*| \leq \frac{k}{2}$. We distinguish two cases.

First, consider the case $|H^*| > k$. Then we construct an instance $\mathcal{I}' = (V', L', \mathcal{H}', D', \prec, c, k)$ with $L' = L - 1$ by setting $V' \coloneqq H^*$, and

$$\mathcal{H}'_\ell \coloneqq \left\{H \in \mathcal{H}_{\ell+1} : H \subseteq H^*\right\}$$
$$D'_\ell \coloneqq D_{\ell+1}$$

for all $\ell \in [L']$. By the induction hypothesis applied to this instance $\mathcal{I}'$, we obtain a tour T' on $V' = H^*$ of k-hop cost $c^{(k)}(T') \leq \left((\alpha(|V'|) + 4L'\right) \cdot \mathrm{OPT}(\mathcal{I}')$. Inserting each vertex from $V \setminus H^*$ into T' at an arbitrary position yields a tour T on V with

$$\begin{aligned} c^{(k)}(T) &\leq c^{(k)}(T') + |V \setminus H^*| \cdot 2kD_1 \\ &\leq \left(\alpha(|V'|) + 4L'\right) \cdot \mathrm{OPT}(\mathcal{I}') + |V \setminus H^*| \cdot 2kD_1 \\ &\leq \left(\alpha(n) + 4(L-1)\right) \cdot \mathrm{OPT}(\mathcal{I}) + |V \setminus H^*| \cdot 2kD_1, \end{aligned}$$

where we used $\mathrm{OPT}(\mathcal{I}') \leq \mathrm{OPT}(\mathcal{I})$ by Lemma 4.21. To complete the proof in the first case, it remains to show $|V \setminus H^*| \cdot kD_1 \leq 2 \cdot \mathrm{OPT}(\mathcal{I})$.

Consider a vertex $w \in V \setminus H^*$ and assume that w is left of H^*. (The case with w right of H^* is symmetric.) Because there are at most $\frac{k}{2}$ vertices in $V \setminus H^*$, at least $\frac{k}{2}$ of the incoming edges of w that contribute to the k-hop costs of an optimal tour on V are edges of the form (v, w) with $v \in H^*$. All these edges are backward edges of level 1 and therefore have cost D_1. Because all these edges have w as one endpoint and the other endpoint in H^*, these $\frac{k}{2}$ edges are different ones for each vertex $w \in V \setminus H^*$. This shows $|V \setminus H^*| \cdot kD_1 \leq 2 \cdot \mathrm{OPT}(\mathcal{I})$ and completes the proof in the case $|H^*| > k$.

Now consider the remaining case $|H^*| \leq k$. We choose a set $R \subseteq V \setminus H^*$ such that $|V \setminus R| = k+1$. By Lemma 4.21, the optimal value $\mathrm{OPT}(V \setminus R, c, k)$ of the Hop-ATSP instance $(V \setminus R, c, k)$ is at most $\mathrm{OPT}(\mathcal{I})$ and is attained by a Hamiltonian cycle on $V \setminus R$. Because $|V \setminus R| = k+1$, any such cycle has the same k-hop cost (namely the sum over all costs $c(x, y)$ with $x, y \in V \setminus R$) and hence we can compute an optimal solution T' to the Hop-ATSP instance $(V \setminus R, c, k)$. Inserting each vertex from R into the tour T', we obtain a tour T on V with

$$c^{(k)}(T) \;\leq\; c^{(k)}(T') + |R| \cdot 2kD_1$$

and by the same argument as in the first case, we have $|R| \cdot kD_1 \leq 2 \cdot \mathrm{OPT}(\mathcal{I})$. □

For non-degenerate instances, we will be able to show that a Path Covering solution cannot be much more expensive than the k-hop cost of a tour:

Lemma 4.43. *Let $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ be a non-degenerate instance of Hop-ATSP and T a tour on V. Then there exists a Path Covering solution $\mathcal{P}$ for $\mathcal{I}$ with*

$$w(\mathcal{P}) \;\leq\; \mathcal{O}(L^3) \cdot c^{(k)}(T).$$

Using Lemma 4.43, we can then prove Theorem 4.36, which we restate here for convenience:

Theorem 4.36. *Let $\alpha\colon \mathbb{N} \to \mathbb{R}$ and $t\colon \mathbb{N} \to \mathbb{N}$ be monotonically increasing. Suppose we have an algorithm that computes for every instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Path Covering with low depth, in time $t(n)$ a solution $\mathcal{P}$ with $w(\mathcal{P}) \leq \alpha(n) \cdot \mathrm{OPT}(\mathcal{I})$, where n denotes the number of vertices in $\mathcal{I}$.*

Then there is an algorithm that computes for every well-scaled instance $\mathcal{J} = (V, c, k)$ of Hop-ATSP in time $\mathrm{poly}(t(n) + n)$ a tour T such that $c^{(k)}(T) \leq \mathcal{O}(\log^6 n) \cdot \alpha(n) \cdot \mathrm{OPT}(\mathcal{J})$, where n denotes the number of vertices in $\mathcal{J}$ and $\mathrm{poly}(n)$ denotes some fixed polynomial.

Proof. Given a well-scaled instance $\mathcal{J}$ of Hop-ATSP, we first apply Theorem 4.6 to obtain hierarchically ordered instances $\mathcal{I}_1, \ldots, \mathcal{I}_m$ of Hop-ATSP with low depth (i.e. $L \leq \mathcal{O}(\log n)$) such that there exists an index i^* satisfying $\mathrm{OPT}(\mathcal{I}_{i^*}) \leq L \cdot \mathcal{O}(\log n \cdot \log\log n) \cdot \mathrm{OPT}(\mathcal{J}) \leq \mathcal{O}(\log^3 n) \cdot \mathrm{OPT}(\mathcal{J})$. By trying all polynomially many choices of $i^* \in [m]$ and returning the best solution found, we may assume that our algorithm knows this index.

Hence, by Lemma 4.42, it suffices to show that there is an algorithm that computes for every non-degenerate hierarchically ordered instance $\mathcal{I}$ of Hop-ATSP with low depth and n vertices, in time $t(n) + \mathrm{poly}(n)$ a tour T with $c^{(k)}(T) \leq \mathcal{O}(\log^3 n) \cdot \alpha(n) \cdot \mathrm{OPT}(\mathcal{I})$. Together with the factor $\log^3 n$ that we lost in the reduction from well-scaled instances of Hop-ATSP to hierarchically ordered instances of Hop-ATSP, this then yields the claimed approximation ratio of $\mathcal{O}(\log^6 n) \cdot \alpha(n)$.

Let $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ be a non-degenerate hierarchically ordered instance of Hop-ATSP with n vertices and low depth. By Lemma 4.43, the weight $w(\mathcal{P}^*)$ of a minimum-weight Path

Covering solution for $\mathcal{I}$ satisfies $w(\mathcal{P}^*) \leq \mathcal{O}(L^3) \cdot c^{(k)}(T^*)$, where T^* is a tour on V that minimizes the k-hop cost $c^{(k)}(T^*)$. Therefore, applying the given algorithm for the Path Covering problem to $\mathcal{I}$ yields a Path Covering solution $\mathcal{P}$ with weight

$$w(\mathcal{P}) \;\leq\; \alpha(n) \cdot \mathcal{O}(L^3) \cdot c^{(k)}(T^*)$$

in running time $t(n) + \text{poly}(n)$. Since $\mathcal{P}$ is the output of a polynomial-time algorithm, we may assume that $|\mathcal{P}|$ is polynomially bounded in n. Then by Lemma 4.39, we can turn this Path Covering solution $\mathcal{P}$ into a tour T with

$$c^{(k)}(T) \;\leq\; 2 \cdot w(\mathcal{P}) \;\leq\; \alpha(n) \cdot \mathcal{O}(L^3) \cdot c^{(k)}(T^*) \;=\; \mathcal{O}(\log^3 n) \cdot \alpha(n) \cdot \text{OPT}(\mathcal{I}).$$

□

In the remaining part of this section we prove Lemma 4.43, i.e, we focus on non-degenerate instances and show how to construct a good Path Covering solution from a given tour on V.

4.7.4 Useful observations on hop-costs

Before explaining how our construction works, we make a couple of useful observations about how the k-hop cost of a path changes when we apply certain operations such as changing the order of the vertices in a particular way. Sometimes we will be interested not only in the overall k-hop cost of our path, but also in the contribution of edges of particular levels to the k-hop cost. For a path P visiting the vertices $v_1, \ldots, v_r$ in this order, we define

$$c^{(k)}_{=\ell}(P) := \sum_{i=1}^{r} \sum_{\substack{\Delta \in [k]: \\ \text{level}(\{v_i, v_{i+\Delta}\}) = \ell}} c(v_i, v_{i+\Delta}).$$

to be the contribution of edges with level exactly ℓ, and we define

$$c^{(k)}_{>\ell}(P) := \sum_{j=\ell+1}^{L} c^{(k)}_{=j}(P)$$

to be the contribution of edges with level $> \ell$.

We will often reorder some vertices of a path. In the simplest case, we reorder all vertices of a path according to the total order $\prec$ on V:

Lemma 4.44. *Let P be a path with $|V(P)| \leq k+1$ and let P_{sorted} be the path on the same vertex set that consists only of forward edges. Then, we have $c^{(k)}(P_{\text{sorted}}) \leq c^{(k)}(P)$.*

Proof. We have

$$c^{(k)}(P_{\text{sorted}}) = \sum_{\substack{x,y \in V(P), \\ x \prec y}} c(x, y),$$

because $|V(P_{\text{sorted}})| \leq k+1$. For any $x, y \in V(P)$ with $x \prec y$, either $c(x, y)$ or $c(y, x)$ contributes to $c^{(k)}(P)$ because $|V(P)| \leq k + 1$. Since (x, y) is a forward edge, $c(x, y) \leq c(y, x)$. Hence, we have $c^{(k)}(P_{\text{sorted}}) \leq c^{(k)}(P)$. □

Sometimes we do not want to reorder the whole path, but only interchange two consecutive vertices on the path:

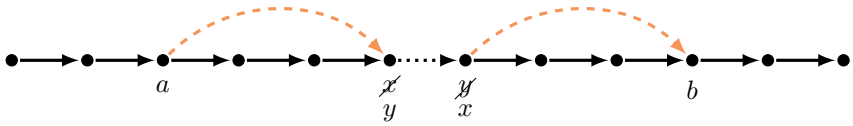


Figure 4.11: Illustration of the interchange of x and y on a path P for $k = 4$. The only edges that contribute to $c^{(k)}(P)$ and are affected by this change are the k-hop edges leaving a and entering b, as well as the edge $(x, y) \in E(P)$ itself.

Lemma 4.45. *Let P be a path and let $e = (x, y)$ be an edge of P. Let P' be the path obtained from P by exchanging the visits of x and y. Then, we have*

$$c^{(k)}(P') \leq c^{(k)}(P) + c(x, y) + c(y, x).$$

Furthermore, if $\text{level}(e) = \ell \in [L]$ *and $x \succ y$, i.e., e is a backward edge, then*

$$c^{(k)}_{>\ell}(P') \leq c^{(k)}_{>\ell}(P) + c(x, y).$$

Proof. Suppose there are vertices $a, b \in V(P)$ such that x is the k-th vertex after a in P, and similarly, b is the k-th vertex after y in P. See Figure 4.11 for an illustration. Then, by the triangle inequality, we have

$$\begin{aligned} c^{(k)}(P') - c^{(k)}(P) &= c(a, y) + c(x, b) + c(y, x) - (c(a, x) + c(y, b) + c(x, y)) \\ &= (c(a, y) - c(a, x)) + (c(x, b) - c(y, b)) + c(y, x) - c(x, y) \\ &\leq 2c(x, y) + c(y, x) - c(x, y) \\ &= c(x, y) + c(y, x). \end{aligned}$$

If no such a or b exists, the corresponding terms can be replaced by zero in the above inequality. This proves the first claim.

Now, consider the case $\text{level}(x, y) = \ell$ and (x, y) is a backward edge. Again, let a and b be as above. The edges (a, y) and (x, b) that newly contribute to the k-hop costs of the path can only contribute to $c^{(k)}_{>\ell}(P')$ if $\text{level}(\{a, y\}) > \ell$ and $\text{level}(\{x, b\}) > \ell$, respectively. In this case, we have $c(a, y) + c(x, b) \leq 2D_{\ell+1} \leq D_\ell = c(x, y)$, where the last inequality uses a property of the numbers D_j from the definition of hierarchically ordered instances (cf. Definition 4.30). This proves the second claim. □

Many arguments in our reduction are based on *replacing* a subpath P_{sub} of a given path P with a different path Q. In most cases we have $V(Q) \subseteq V(P_{\text{sub}})$, so it is clear that the result is indeed a path.

Lemma 4.46. *Let P_{main} be a path and let P_{sub} be a subpath of P_{main}. Let Q be another path with $V(Q) \subseteq V(P_{\text{sub}})$, and denote by P'_{main} the path that results from P_{main} by replacing P_{sub} with Q. Then, we have*

$$c^{(k)}(P'_{\text{main}}) \leq c^{(k)}(P_{\text{main}}) + c^{(k)}(Q) + 2^{11} \cdot c^{(k)}(P_{\text{sub}}).$$

Proof. Without loss of generality, we assume that $V(Q) = V(P_{\text{sub}})$. If $V(Q) \subsetneq V(P_{\text{sub}})$, we simply remove all vertices $v \in V(P_{\text{sub}}) \setminus V(Q)$ from P_{main}, which does not increase the cost by Lemma 4.21.

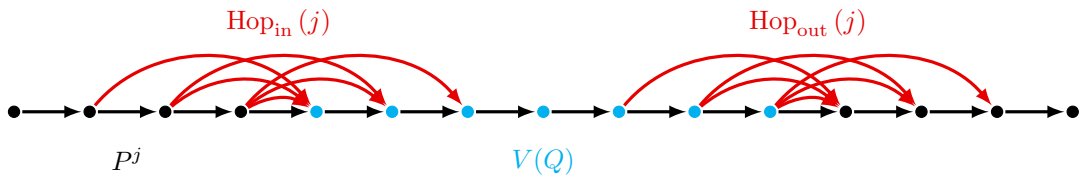


Figure 4.12: Illustration of the sets $\mathrm{Hop}_{\mathrm{in}}(j)$ and $\mathrm{Hop}_{\mathrm{out}}(j)$. The edges contained in these sets are shown in red, where we have $k = 3$ in this example.

We will construct a sequence of paths $P^1, \dots, P^r$ with $P^1 = P_{\mathrm{main}}$ and $P^r = P'_{\mathrm{main}}$ for some $r \in \mathbb{N}$. Each path P^j will arise from P_{main} by changing the order of the vertices in $V(Q) = V(P_{\mathrm{sub}})$. For $j \in [r]$ we define

$$\mathrm{Hop}_{\mathrm{in}}(j) \coloneqq \left\{(v,w) : v \in V(P_{\mathrm{main}}) \setminus V(Q),\ w \in V(Q),\ \text{and } (v,w) \text{ contributes to } c^{(k)}\left(P^j\right)\right\}$$
$$\mathrm{Hop}_{\mathrm{out}}(j) \coloneqq \left\{(v,w) : v \in V(Q),\ w \in V(P_{\mathrm{main}}) \setminus V(Q),\ \text{and } (v,w) \text{ contributes to } c^{(k)}\left(P^j\right)\right\},$$

where an edge (v,w) contributes to $c^{(k)}\left(P^j\right)$ if the path P^j visits v before w and visits fewer than k other vertices in between. See Figure 4.12 for an illustration.

Using this notation, we get

$$\begin{aligned} c^{(k)}(P'_{\mathrm{main}}) - c^{(k)}(P_{\mathrm{main}}) &= c^{(k)}\left(P^r\right) - c^{(k)}\left(P^1\right) \\ &\leq c^{(k)}(Q) + c\left(\mathrm{Hop}_{\mathrm{in}}(r)\right) - c\left(\mathrm{Hop}_{\mathrm{in}}(1)\right) + c\left(\mathrm{Hop}_{\mathrm{out}}(r)\right) - c\left(\mathrm{Hop}_{\mathrm{out}}(1)\right). \end{aligned}$$

Observe that $c\left(\mathrm{Hop}_{\mathrm{in}}(r)\right) - c\left(\mathrm{Hop}_{\mathrm{in}}(1)\right)$ and $c\left(\mathrm{Hop}_{\mathrm{out}}(r)\right) - c\left(\mathrm{Hop}_{\mathrm{out}}(1)\right)$ do not depend on the sequence $P^1, \dots, P^r$, as long as we have $P^1 = P_{\mathrm{main}}$ and $P^r = P'_{\mathrm{main}}$. In particular, we can prove an upper bound on each of these expressions while each time working with a different sequence $P^1, \dots, P^r$. We will show

$$c\left(\mathrm{Hop}_{\mathrm{in}}(r)\right) - c\left(\mathrm{Hop}_{\mathrm{in}}(1)\right) \leq 2^{10} \cdot c^{(k)}(P_{\mathrm{sub}}). \tag{4.19}$$

By symmetry we then also get

$$c\left(\mathrm{Hop}_{\mathrm{out}}(r)\right) - c\left(\mathrm{Hop}_{\mathrm{out}}(1)\right) \leq 2^{10} \cdot c^{(k)}(P_{\mathrm{sub}}), \tag{4.20}$$

implying

$$c^{(k)}(P'_{\mathrm{main}}) - c^{(k)}(P_{\mathrm{main}}) \leq c^{(k)}(Q) + 2^{10} \cdot c^{(k)}(P_{\mathrm{sub}}) + 2^{10} \cdot c^{(k)}(P_{\mathrm{sub}}),$$

as claimed. It remains to prove (4.19) (where (4.20) follows by a symmetric argument).

To show (4.19), we construct a sequence $P^1, \dots, P^r$ with $P^1 = P_{\mathrm{main}}$ and $P^r = P'_{\mathrm{main}}$. During our construction we will maintain a set F of edges, which we initialize as $F \coloneqq \emptyset$. We number the vertices of the path Q as $q_1, \dots, q_m$, in the order in which Q visits them. Starting with $j = 1$ and $P^j = P_{\mathrm{main}}$, we proceed as follows. For $a = 1, \dots, m$ we do the following: While there exists a vertex $q_b \in V(Q)$ with $a < b$ such that the edge (q_b, q_a) is contained in the current path P^j, we interchange the two vertices and obtain P^{j+1}. If $\mathrm{Hop}_{\mathrm{in}}(j) \neq \mathrm{Hop}_{\mathrm{in}}(j+1)$, we add the edge (q_b, q_a) to F. In other words, the sequence $P^{(1)}, \dots, P^{(r)}$ is obtained by applying the Bubblesort algorithm, swapping the vertices that are visited earlier by Q first.

Due to the order in which we interchange vertices, F satisfies the following properties:

(i) $|F| \leq k^2$.

(ii) For every $x \in V$, there are at most k edges in F that are incident to x.

(iii) Every edge $(x, y) \in V \times V$ is added at most once to F.

(iv) If $(x, y) \in F$, then x precedes y on P_{sub}.

Let $(x, y) \in F$ be an edge that was added to F when obtaining P^j from P^{j-1}. Then, we have

$$c\left(\text{Hop}_{\text{in}}(j)\right) - c\left(\text{Hop}_{\text{in}}(j-1)\right) \leq c(x, y),$$

where we used that c satisfies the triangle inequality. In particular, this shows

$$c\left(\text{Hop}_{\text{in}}(r)\right) - c\left(\text{Hop}_{\text{in}}(1)\right) \leq c(F).$$

First, suppose that $k \leq 16$. By property (iv) and the triangle inequality, we can bound $c(x, y) \leq c(P_{\text{sub}}) \leq c^{(k)}(P_{\text{sub}})$ for every $(x, y) \in F$. In particular, property (i) then implies that $c(F) \leq 2^8 \cdot c^{(k)}(P_{\text{sub}})$. Hence, we may assume that $k \geq 16$ in the following.

Let $\widetilde{F} \subseteq F$ be the subset of those edges (x, y) for which P_{sub} visits less than k other vertices between the visits of x and y, i.e., $c(x, y)$ contributes to $c^{(k)}(P_{\text{sub}})$. In particular, we have

$$c(\widetilde{F}) \leq c^{(k)}(P_{\text{sub}}).$$

Next, we upper bound the cost of the edges in $F \setminus \widetilde{F}$. Fix an edge $(x, y) \in F \setminus \widetilde{F}$. By (iv), the vertex x is visited before y by the path P_{sub}. Because $k \geq 16$, we can apply Lemma 4.24 to the x-y subpath of P_{sub} and obtain a vector $\chi^{(x,y)} \in \mathbb{R}_{\geq 0}^{V \times V}$ with

$$\begin{cases} \chi_e^{(x,y)} \leq 2^4/k & \text{if } e \in (\delta^+(x) \cup \delta^-(y)) \text{ and } e \text{ contributes to the } k\text{-hop cost of } P_{\text{sub}}, \\ \chi_e^{(x,y)} \leq 2^8/k^2 & \text{if } e \notin \delta^+(x) \cup \delta^-(y) \text{ and } e \text{ contributes to the } k\text{-hop cost of } P_{\text{sub}}, \\ \chi_e^{(x,y)} = 0 & \text{if } e \text{ does not contribute to the } k\text{-hop cost of } P_{\text{sub}}, \end{cases}$$

The vector $\chi^{(x,y)}$ is a convex combination of incidence vectors of x-y paths and thus, by the triangle inequality, $c(x, y) \leq c^\intercal \chi^{(x,y)}$ (where we view the cost function c as a vector in $\mathbb{R}_{\geq 0}^{V \times V}$).

Now, let $e = (v, w)$ be an edge contributing to the k-hop cost of the x-y subpath of P_{sub}. (In particular, $v, w \in V(P_{\text{sub}})$ by the definition of k-hop cost of non-closed walks (Definition 4.20).) By the guarantees (4.10) of Lemma 4.24, and the properties (i), (ii), and (iii) of F, we have

$$\sum_{(x,y) \in F \setminus \widetilde{F}} \chi_e^{(x,y)} \quad \leq \quad \sum_{\substack{(x,y) \in F \setminus \widetilde{F}, \\ \{x,y\} \cap \{v,w\} \neq \emptyset}} \frac{2^4}{k} + \sum_{\substack{(x,y) \in F \setminus \widetilde{F}, \\ \{x,y\} \cap \{v,w\} = \emptyset}} \frac{2^8}{k^2} \quad \leq \quad 2k \cdot \frac{2^4}{k} + k^2 \cdot \frac{2^8}{k^2} \quad \leq \quad 2^9.$$

For an edge e not contributing to the k-hop cost of P_{sub}, we have $\sum_{(x,y)\in F\setminus\widetilde{F}} \chi_e^{(x,y)} = 0$. This implies

$$\begin{aligned}
c(F \setminus \widetilde{F}) &= \sum_{(x,y)\in F\setminus\widetilde{F}} c(x,y) \\
&\leq \sum_{(x,y)\in F\setminus\widetilde{F}} c^{\intercal}\chi^{(x,y)} \\
&= \sum_{(x,y)\in F\setminus\widetilde{F}} \left(\sum_{e\in V^2} c(e)\cdot \chi_e^{(x,y)} \right) \\
&= \sum_{\substack{e\in V^2 \\ e \text{ contributes} \\ \text{to } c^{(k)}(P_{\text{sub}})}} c(e) \cdot \sum_{(x,y)\in F\setminus\widetilde{F}} \chi_e^{(x,y)} \\
&\leq 2^9 \cdot c^{(k)}(P_{\text{sub}}).
\end{aligned}$$

Hence, we conclude

$$c\left(\text{Hop}_{\text{in}}(r)\right) - c\left(\text{Hop}_{\text{in}}(1)\right) \leq c(F) \leq (2^9+1)\cdot c^{(k)}(P_{\text{sub}}),$$

implying (4.19). □

Sometimes we carefully reorder entire parts of a path P in order to ensure that the resulting path has no backward edges of level ℓ anymore:

Definition 4.47 (ℓ-refinement)**.** *Let P be a path and $\ell \in [L]$. We define a total order $\prec'$ on $V(P)$ as follows: Let $x \neq y \in V(P)$ be two distinct vertices.*

- *If $\text{level}(\{x,y\}) \leq \ell$, then $x \prec' y$ if and only if $x \prec y$.*
- *If $\text{level}(\{x,y\}) > \ell$, then $x \prec' y$ if and only if x appears before y on P.*

Let P' with $V(P') = V(P)$ be the path that visits all vertices in the order given by $\prec'$. We say that P' is the ℓ-refinement *of P.*

See Figure 4.13 for an example. If $\text{level}(P) > \ell$, then P is its own ℓ-refinement. We can prove a strong upper bound on the additional costs that arise by replacing a subpath with its ℓ-refinement:

Lemma 4.48. *Let $\ell \in [L]$. Let P_{main} be a path and let P_{sub} be a subpath of P_{main} with $|V(P_{\text{sub}})| \leq k+1$ and $\text{level}(V(P_{\text{sub}})) \geq \ell$. Let P'_{sub} be the ℓ-refinement of P_{sub} and denote by P'_{main} the path obtained from P_{main} by replacing P_{sub} with P'_{sub}. Then,*

$$c^{(k)}(P'_{\text{sub}}) \leq c^{(k)}(P_{\text{sub}}),$$

and

$$c^{(k)}_{>\ell}(P'_{\text{main}}) \leq c^{(k)}_{>\ell}(P_{\text{main}}) + c^{(k)}_{=\ell}(P_{\text{sub}}).$$

Proof. We start by proving the second inequality. Let $\prec'$ be the total order on $V(P_{\text{sub}})$ as given in Definition 4.47. We modify P_{main} in a similar way as in the proof of Lemma 4.46: In the beginning, set $F := \emptyset$. Then, iteratively interchange the visits of vertices $x \neq y \in V(P_{\text{sub}})$ if $y \prec' x$ and (x,y) is part of the current path, and add (x,y) to F. Again, in other words, we reorder P_{sub} according to $\prec'$ using the Bubblesort algorithm, and the resulting path is P'_{main}.

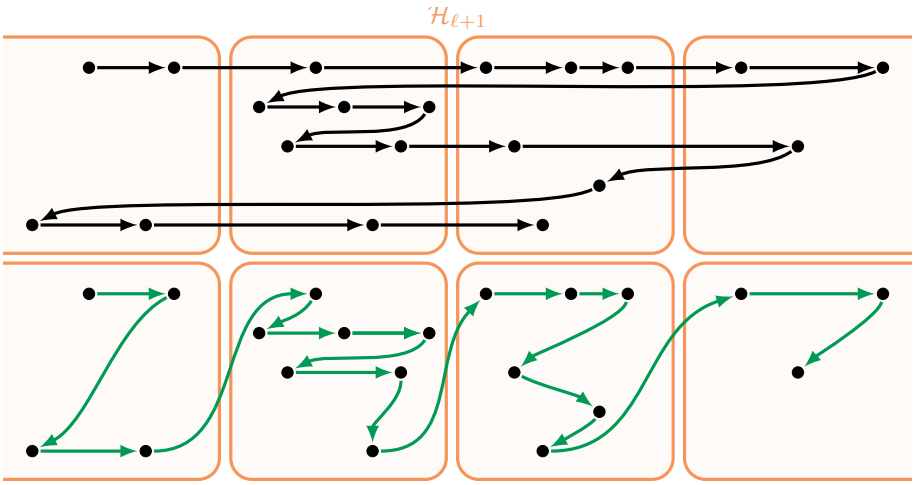


Figure 4.13: An example of a path P (above) and its ℓ-refinement (below). All vertices are drawn from left to right according to the total order $\prec$.

We have $(x, y) \in F$ if and only if $y \prec x$, $\text{level}(\{x, y\}) = \ell$ and x appears before y on P_{sub} because our initial order is given by P_{sub}. Furthermore, a pair of vertices is interchanged at most once. Therefore, by the second part of Lemma 4.45, we have

$$c^{(k)}_{>\ell}(P'_{\text{main}}) \leq c^{(k)}_{>\ell}(P_{\text{main}}) + \sum_{(x,y)\in F} c(x, y).$$

Since $|V(P_{\text{sub}})| \leq k+1$, each edge $(x, y) \in F$ contributes to $c^{(k)}_{=\ell}(P_{\text{sub}})$. Thus, $\sum_{(x,y)\in F} c(x, y) \leq c^{(k)}_{=\ell}(P_{\text{sub}})$. This shows the second inequality.

For the first inequality, simply observe that

$$c^{(k)}(P_{\text{sub}}) = \sum_{\substack{x,y\in V(P_{\text{sub}}) \\ x \text{ precedes } y \\ \text{in } P_{\text{sub}}}} c(x, y)$$

because $|V(P_{\text{sub}})| \leq k+1$; an analogous equality holds for $c^{(k)}(P'_{\text{sub}})$. thus

$$c^{(k)}(P'_{\text{sub}}) = c^{(k)}(P_{\text{sub}}) - \sum_{(x,y)\in F} c(x, y) + \sum_{(x,y)\in F} c(y, x).$$

Since $y \prec x$ if $(x, y) \in F$, we have $c(x, y) \geq c(y, x)$. This proves the first inequality. □

4.7.5 Turning tours into covering solutions

Consider a non-degenerate hierarchically ordered instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Hop-ATSP and a tour T on V. Our goal is to prove that there exists a Path Covering solution $\mathcal{P}$ for the instance $\mathcal{I}$ with $w(\mathcal{P}) \leq \mathcal{O}(L^3) \cdot c^{(k)}(T)$. By Lemma 4.21, we may assume that T is a cycle. We will first remove an arbitrary edge from T to obtain a path P with vertex set V. If the path P

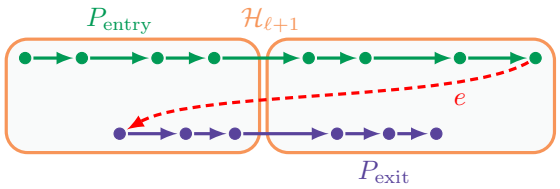


Figure 4.14: Illustration of an ℓ-jump segment. All vertices are drawn from left to right according to the total order $\prec$. The path contains a single backward edge e that connects the two monotone subpaths P_{entry} and P_{exit}.

happens to be monotone, then $\mathcal{P} = \{(P, 1)\}$ would be a Path Covering solution. Because our instance $\mathcal{I}$ is non-degenerate, Lemma 4.41 implies that

$$w(\mathcal{P}) \;=\; c^{(k)}(P) + k^2 D_1 \leq (1 + 2^8) \cdot c^{(k)}(P).$$

Of course P will in general not be monotone, i.e., it will contain some backward edges.

Our strategy will be to first get rid of all backward edges of level 1, then those of level 2, and so on. However, removing the backward edges of a given level can become complicated when there are many such jumps close together. To address this, we first isolate these jumps, which will allow us to treat each jump independently. In order to formalize what we mean by "isolating jumps", we introduce the notion of a *jump segment*.

Definition 4.49 (Jump segment)**.** *An* ℓ-jump segment *is a path* P *with* $|V(P)| \leq k+1$ *and* $\text{level}(V(P)) = \ell$ *that consists of two vertex disjoint monotone paths* P_{entry} *and* P_{exit}*, that are connected by a backward edge* e *(where* P_{entry} *precedes* P_{exit} *on* P*).*

See Figure 4.14 for an illustration. We remark that $\ell \leq \text{level}(e) < L$ because the endpoints of the edge e are contained in $V(P)$ and $\mathcal{H}_L$ contains only singletons. Note however, that $\text{level}(e)$ is not necessarily equal to $\ell = \text{level}(V(P))$, but may be strictly greater.

In a path P with "isolated jumps", we will be able to assign a jump-segment P^e to each backward edge e such that these jump-segments for all backward edges are vertex disjoint subpaths of P. Our strategy will be to change the path only in the local environment of e that is given by the jump-segment P^e when getting rid of the backward edge e. Because the jump segments assigned to different backward edges are vertex disjoint, this will allow us to handle the different jumps independently.

In order to be able to get rid of a backward edge e by only changing P in the local environment given by its assigned jump segment, the jump segment P^e must satisfy certain properties. We will ensure that each jump segment satisfies one of the following two useful properties: If all vertices that the path P visits before the jump segment P^e are left of the vertices of P^e, then intuitively we can handle the part of P before the jump segment independently of P^e. In this case, we will call P^e *entry-safe*.

Definition 4.50 (Entry-safe, exit-safe)**.** *Let* P_{main} *be a path, and let* P_{sub} *be a subpath of* P_{main}*. We say that* P_{sub} *is* entry-safe *within* P_{main} *if all vertices that appear (strictly) before* P_{sub} *on* P_{main} *are left of* $V(P_{\text{sub}})$*.*

Analogously, P_{sub} *is* exit-safe *within* P_{main} *if all vertices that appear (strictly) after* P_{sub} *on* P_{main} *are right of* $V(P_{\text{sub}})$*.*

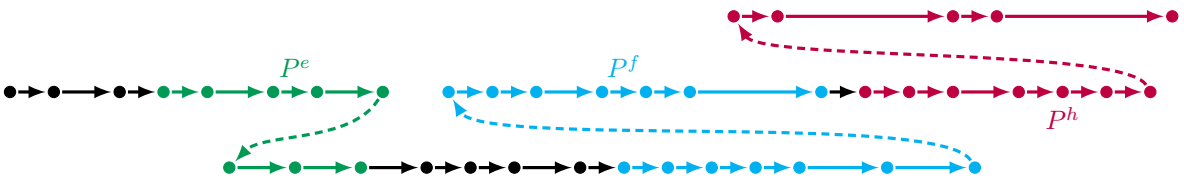


Figure 4.15: Illustration of an ℓ-buffering consisting of three ℓ-jump segments, where backward edges are dashed and vertices are drawn from left to right according to the order $\prec$. The ℓ-jump segment P^e (green) is is entry-safe and exit-safe within the path P. The ℓ-jump segment P^f (blue) is "far away" from other jumps in the sense that P^f_{entry} and P^f_{exit} contain many vertices. More precisely, P^f is entry-buffered and exit-buffered (assuming $7 \geq \frac{k}{16} - (\ell-1) \cdot \frac{k}{32(L-1)}$). The ℓ-jump segment P^h (red) is entry-buffered and exit-safe.

If a jump-segment P^e is not entry-safe, we will ensure that P^e_{entry} contains many vertices, so the jump is "far away" from earlier jumps of P. Analogously, we will ensure that if a jump-segment P^e is not exit-safe, we will ensure that P^e_{exit} contains many vertices, so the jump is "far away" from later jumps of P.

Definition 4.51 (Entry-buffered, exit-buffered)**.** *For $\omega \in \{\text{entry}, \text{exit}\}$, we say that an ℓ-jump segment P is* ω-buffered *if $|V(P_\omega)| \geq \frac{k}{16} - (\ell-1) \cdot \frac{k}{32(L-1)}$.*

We can now formalize the aforementioned concept of "isolated jumps" in a path P. See Figure 4.15 for an illustration.

Definition 4.52 (ℓ-buffering)**.** *An* ℓ-buffering *of a path P is a collection $\mathcal{B}$ of of vertex disjoint subpaths of P such that every backward edge e of P is contained in some path $P^e \in \mathcal{B}$, and each path $Q \in \mathcal{B}$ satisfies the following properties:*

(i) Q is an ℓ'-jump segment with $\ell' \geq \ell$.

(ii) Q is entry-buffered or entry-safe within P.

(iii) Q is exit-buffered or exit-safe within P.

We say that P is ℓ-buffered *if there exists an ℓ-buffering of P.*

To prove that we can turn our path P into a Path Covering solution $\mathcal{P}$ of small weight, we will use a recursive argument. By Lemma 4.37, we want $\mathcal{P}$ to cover all elements of the Set Cover universe $\mathcal{U} = V \times [L]$. To describe our recursive construction, we need the notion of a *main-tour cover*, which will be used to describe what we aim for when recursively applying our argument to a path P that contains only a subset $U \subseteq V$ of the vertices.

Definition 4.53 (Main-tour cover)**.** *Let $U \subseteq V$ be an arbitrary set of vertices. A* main-tour cover *of U is a monotone path P_{main} with $V(P_{\text{main}}) \subseteq U$ together with a set $\mathcal{P} \subseteq \mathcal{S}$ of path-level pairs satisfying the following two conditions:*

(i) $V(P_{\text{main}}) = U$ or $|V(P_{\text{main}})| \geq k$.

(ii) The path-level pairs $\mathcal{P} \cup \big\{\big(P_{\text{main}}, \text{level}(U)\big)\big\}$ cover the elements $U \times \big\{\text{level}(U), \dots, L\big\}$ of the Set Cover universe $\mathcal{U}$.

The key lemma, which we will prove by induction on $L - \ell$, is the following:

Lemma 4.54. *Let P_{old} be an ℓ-buffered path with $\ell \coloneqq \text{level}(V(P_{\text{old}}))$. Then there is a main-tour cover $(P_{\text{new}}, \mathcal{P})$ of $V(P_{\text{old}})$ such that*

$$c^{(k)}(P_{\text{new}}) + w(\mathcal{P}) \;\leq\; \eta^* \cdot (L - \ell + 1) \cdot c^{(k)}(P_{\text{old}}), \tag{4.21}$$

where $\eta^* \coloneqq 3\eta_1 + \big(\eta_2 + 2^{13}\big)$, $\eta_1 \coloneqq 2^{12}L^2$, and $\eta_2 \coloneqq 2^8 L$.

Note that if we apply Lemma 4.54 to a 1-buffered path P_{old} with vertex set V, we obtain a solution $\mathcal{P} \cup \{(P_{\text{new}}, 1)\}$ of our Path Covering problem. In order to apply Lemma 4.54, we need to start with a path that is 1-buffered. Using the tools that we have already established, this is not difficult to obtain:

Lemma 4.55. *Let P be a path. Then, there is a 1-buffered path $\hat{P}$ with $V(P) = V(\hat{P})$ and*

$$c^{(k)}(\hat{P}) \leq 2^{12} \cdot c^{(k)}(P).$$

Proof. If $|V(P)| \leq k$, then we are done by Lemma 4.44. If $k \leq 16$, any jump segment is entry- and exit-buffered. Hence, to obtain P', we simply interchange the endpoints of every second edge that is backward. All remaining backward edges are disjoint and form a 1-buffering (where each jump segment consists of a single backward edge). The cost increase can then be bounded by Lemmas 4.44 and 4.46.

Thus, we may assume $|V(P)| \geq k > 16$. We partition P into vertex disjoint subpaths $P_1, \ldots, P_r$ of P with $\bigcup_{j=1}^{r} V(P_j) = V(P)$ and $2\lceil k/16 \rceil \leq |V(P_j)| \leq k/2$ (cf. Lemma 4.23). For each $j \in [r]$ let P'_j be the path that visits the vertices from $V(P_j)$ according to the order $\prec$. Replace each subpath P_j of P with P'_j and denote by P' the resulting path.

By Lemmas 4.44 and 4.46, we have

$$c^{(k)}(P') \leq c^{(k)}(P) + \sum_{j=1}^{r} 2^{11} \cdot c^{(k)}(P_j) + \sum_{j=1}^{r} c^{(k)}(P'_j) \leq 2^{12} \cdot c^{(k)}(P).$$

It remains to show that P' is 1-buffered. Any backward edge e of P' connects a subpath P'_j and P'_{j+1} (if we number the subpaths appropriately). In that case, we can use $\lceil k/16 \rceil$ vertices from each of P'_j and P'_{j+1} to construct an ℓ-jump segment around e (for some $\ell \in [L]$) that is both entry- and exit-buffered. Doing this for every backward-edge of P' yields a 1-buffering of P'. □

Before proving Lemma 4.54, we first show that its proof will complete our reduction to the Path Covering Problem. In other words, using Lemma 4.54, we are now ready to prove Lemma 4.43, which we restate here for convenience:

Lemma 4.43. *Let $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ be a non-degenerate instance of Hop-ATSP and T a tour on V. Then there exists a Path Covering solution $\mathcal{P}$ for $\mathcal{I}$ with*

$$w(\mathcal{P}) \;\leq\; \mathcal{O}(L^3) \cdot c^{(k)}(T).$$

Proof. Because $\mathcal{I}$ is non-degenerate, Lemma 4.41 implies $k^2 \cdot D_1 \leq 2^8 \cdot c^{(k)}(T)$. Let P be a path resulting from T by removing an arbitrary edge. We apply Lemma 4.55 to P and then apply Lemma 4.54 to the resulting 1-buffered path. We obtain a main-tour cover $(\hat{P}_{\text{main}}, \hat{\mathcal{P}})$ of V such that

$$c^{(k)}(\hat{P}_{\text{main}}) + w(\hat{\mathcal{P}}) \;\leq\; \eta^* \cdot L \cdot 2^{12} \cdot c^{(k)}(P).$$

Then $\mathcal{P} \coloneqq \hat{\mathcal{P}} \cup \left\{\left(\hat{P}_{\text{main}}, 1\right)\right\}$ is a Path Covering solution (by Lemma 4.37) and its weight is

$$\begin{aligned} w(\mathcal{P}) &= w(\hat{\mathcal{P}}) + c^{(k)}(\hat{P}_{\text{main}}) + k^2 \cdot D_1 \\ &\leq 2^{12} \cdot \eta^* \cdot L \cdot c^{(k)}(P) + k^2 \cdot D_1 \\ &\leq \left(2^{12} \cdot \eta^* \cdot L + 2^8\right) \cdot c^{(k)}(T) \\ &= \mathcal{O}(L^3) \cdot c^{(k)}(T), \end{aligned}$$

where we used $\eta^* = \mathcal{O}(L^2)$. □

In the remainder of the section we prove Lemma 4.54.

4.7.6 Splitting jump segments

Recall that we want to consider each jump segment of our given ℓ-buffered path separately and aim at getting rid of the backward edge contained in it. One possibility to get rid of a backward edge is to split the path into two paths. However, this is costly because for every path-level pair (P, ℓ) that we add to our main-tour cover, we need to pay the "fixed cost" $k^2 D_\ell$ in addition to the k-hop costs $c^{(k)}(P)$. Because the total weight of the path-level pairs in our main-tour cover should not be too large compared to the k-hop cost of our initial path, we can only afford paying these fixed costs when our initial path is sufficiently expensive.

We will classify certain ℓ-jump segments P as *splitting* ℓ-jump segments. Such segments will satisfy the property that they contain many j-hop edges ($j \leq k$) that are backward edges of level ℓ. In particular, if they contain at least $\lambda = k^2/(\text{poly}(\log k))$ such j-hop edges, we have $c^{(k)}(P) \geq \lambda \cdot D_\ell$. This will enable us to pay the fixed cost of $k^2 D_\ell$ for a new path-level pair.

In the following, we make this classification more precise. Let P be an ℓ-jump segment and let $H_1, \dots, H_r \in \mathcal{H}_{\ell+1}$ such that

(i) $V(P) \cap H_j \neq \emptyset$ for all $j \in [r]$,

(ii) $V(P) \subseteq \bigcup_{j=1}^{r} H_j$, and

(iii) H_i is left of H_j for all $i, j \in [r]$ with $i < j$.

Note that $r \geq 2$ because $\text{level}(V(P)) = \ell$. If P is entry-buffered, we define

$$j^*_{\text{entry}} \coloneqq \max\left\{ j \in [r] : \sum_{i=j}^{r} |H_i \cap V(P_{\text{entry}})| \geq \frac{k}{64(L-1)} \right\}. \tag{4.22}$$

This is well-defined, because

$$|V(P_{\text{entry}})| \geq \frac{k}{16} - (\ell - 1) \cdot \frac{k}{32(L-1)} \geq \frac{k}{32} \geq \frac{k}{64(L-1)}.$$

Analogously, if P is exit-buffered, we define

$$j^*_{\text{exit}} \coloneqq \min\left\{ j \in [r] : \sum_{i=1}^{j} |H_i \cap V(P_{\text{exit}})| \geq \frac{k}{64(L-1)} \right\}. \tag{4.23}$$

See Figure 4.16 for an illustration. We observe that

$$\sum_{i=j^*_{\text{entry}}+1}^{r} |H_i \cap V(P_{\text{entry}})| < \frac{k}{64(L-1)} \tag{4.24}$$

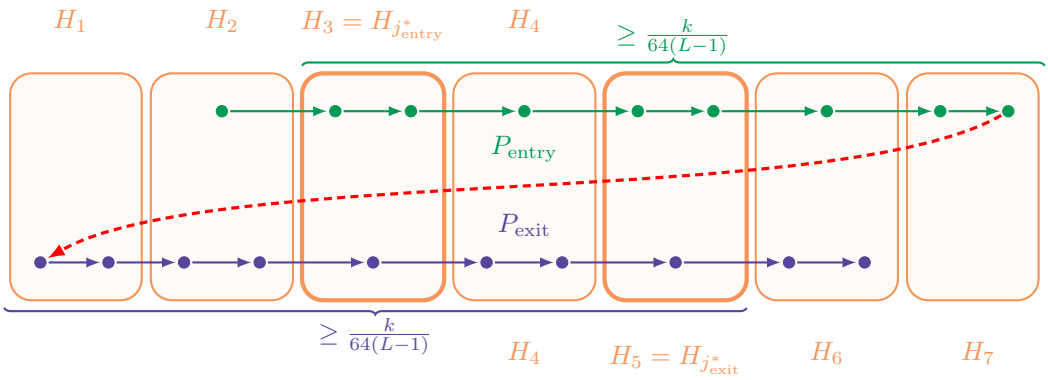


Figure 4.16: Illustration of our choice of j^*_{entry} and j^*_{exit}. All vertices are drawn from left to right according to the total order $\prec$. The ℓ-jump segment P consists of P_{entry} (green), the backward edge e (red), and P_{exit} (purple). In this example, we have a *non-splitting* ℓ-jump segment.

and

$$\sum_{i=1}^{j^*_{\text{exit}}-1} |H_i \cap V(P_{\text{exit}})| < \frac{k}{64(L-1)}, \tag{4.25}$$

because otherwise this would contradict the choice of j^*_{entry} and j^*_{exit} as the maximum and minimum index with their corresponding property, respectively. The sets $H_1, \dots, H_r$ are uniquely determined by the above properties. Hence, the following is well-defined:

Definition 4.56 (Splitting ℓ-jump segment)**.** *Let P be an ℓ-jump segment that is entry- and exit-buffered. Let j^*_{entry} and j^*_{exit} be defined as in* (4.22) *and* (4.23)*. We say that P is a* splitting ℓ-jump segment *if $j^*_{\text{entry}} > j^*_{\text{exit}}$.*

An ℓ-buffering is called ℓ-splitting-free *if it contains no splitting ℓ-jump segment.*

We remark that an ℓ-splitting-free ℓ-buffering may contain a splitting ℓ'-jump segment for some $\ell' > \ell$. From the definition of a splitting ℓ-jump segment, we immediately obtain:

Lemma 4.57. *Let P be a splitting ℓ-jump segment. Then we have*

$$\eta_1 \cdot c^{(k)}(P) \geq k^2 D_\ell,$$

where $\eta_1 := 2^{12} \cdot L^2$.

Proof. We define the $H_1, \dots, H_r \in \mathcal{H}_{\ell+1}$ as above. Any edge $e = (x, y)$ with $x \in \bigcup_{i=j^*_{\text{entry}}}^{r} H_i$ and $y \in \bigcup_{i=1}^{j^*_{\text{exit}}} H_i$ is a backward edge of level ℓ, because $j^*_{\text{entry}} > j^*_{\text{exit}}$. Thus, each such edge e has cost $c(e) = D_\ell$. For each $x \in V(P_{\text{entry}})$ and each $y \in V(P_{\text{exit}})$, the edge (x, y) contributes to $c^{(k)}(P)$, because $|V(P)| \leq k + 1$. Therefore, we conclude

$$\begin{aligned} c^{(k)}(P) &\geq \left|V(P_{\text{entry}}) \cap \left(\bigcup_{i=j^*_{\text{entry}}}^{r} H_i\right)\right| \cdot \left|V(P_{\text{exit}}) \cap \left(\bigcup_{i=1}^{j^*_{\text{exit}}} H_i\right)\right| \cdot D_\ell \\ &\geq \left(\frac{k}{64(L-1)}\right)^2 \cdot D_\ell \\ &\geq \frac{1}{\eta_1} \cdot k^2 D_\ell. \end{aligned}$$

□

4.7.7 Handling non-splitting jump segments

Next, we explain how we handle non-splitting jump segments. Our goal will be to prove the following:

Lemma 4.58. *Let P_{main} be a path with an ℓ-splitting-free ℓ-buffering. Then, there is an $(\ell+1)$-buffered path P'_{main} such that for $R^* := V(P_{\text{main}}) \setminus V(P'_{\text{main}})$ we have*

$$c^{(k)}_{>\ell}(P'_{\text{main}}) \leq c^{(k)}(P_{\text{main}}), \tag{4.26}$$

and

$$c^{(k)}(P'_{\text{main}}) + |R^*| \cdot 2kD_\ell \leq \left(\eta_2 + 2^{12}\right) \cdot c^{(k)}(P_{\text{main}}), \tag{4.27}$$

where $\eta_2 \coloneqq 2^8 L$.

The bound (4.27) will be useful because we will later be able to re-insert the removed vertices, i.e, those from R^*, into some path at cost $2kD_\ell$ per vertex. Observe that the overall cost increases by a factor of $(\eta_2 + 2^{12})$ and we cannot afford this to happen L times, when iteratively applying the argument to each level $\ell \in [L]$. This is where the bound (4.26) will become important. Because we will later recursively apply our argument to subpaths inside sets from $\mathcal{H}_{\ell+1}$, this bound ensures that we only lose an additive term of $(\eta_2 + 2^{12} - 1)$ times the k-hop-cost of our original path in each application of the lemma.

To prove Lemma 4.58, we will remove all backward edges of level ℓ and ensure that the resulting path is ℓ-buffered. We then observe that an ℓ-buffered path with no backward edges of level ℓ is $(\ell+1)$-buffered:

Lemma 4.59. *Let P be an ℓ-buffered path. If P does not contain any backward edge of level ℓ, then P is $(\ell+1)$-buffered.*

Proof. Let $\mathcal{B}$ be an ℓ-buffering of P. If $\mathcal{B}$ contains no ℓ-jump segment (that is, all elements of $\mathcal{B}$ are ℓ'-jump segments for some $\ell' > \ell$), then $\mathcal{B}$ is an $(\ell+1)$-buffering. Otherwise, let P^e be an ℓ-jump segment and let e be the unique backward edge contained in P^e. By our assumption, we have $\text{level}(e) \geq \ell + 1$. Let Q^e be the maximal subpath of P^e that contains e and satisfies $\text{level}(Q^e) \geq \ell + 1$. Then Q^e is an ℓ'-jump segment with $\ell' \geq \ell + 1$. We denote by s and x the start vertices of P^e and Q^e, respectively. See Figure 4.17.

Suppose that $s = x$. If P^e is entry-safe within P, then so is Q^e. If P^e is entry-buffered, then the entry part of Q^e contains at least

$$\frac{k}{16} - (\ell - 1) \cdot \frac{k}{32(L-1)} \;\geq\; \frac{k}{16} - (\ell' - 1) \cdot \frac{k}{32(L-1)}$$

vertices. Hence, Q^e is entry-buffered in this case.

Now, suppose that $s \neq x$. Let s' be the predecessor of x in the path P^e and let $H \in \mathcal{H}_{\ell+1}$ with $V(Q^e) \subseteq H$. Note that (s', x) is a forward edge, since P^e contains only one backward edge and this backward edge e is contained in Q^e. Therefore, because we chose Q^e maximal with the property that $V(Q^e) \subseteq H$, the edge (s', x) is a forward edge of level ℓ. This implies that Q^e is entry-safe within P, because P does not contain any backward edges of level $\leq \ell$. See Figure 4.17 for a visualization. Similarly, one can show that Q^e is exit-buffered or exit-safe within P in any case.

In this way, we can replace every ℓ-jump segment P^e in the ℓ-buffering $\mathcal{B}$ by an ℓ'-jump segment Q^e for some $\ell' > \ell$. □

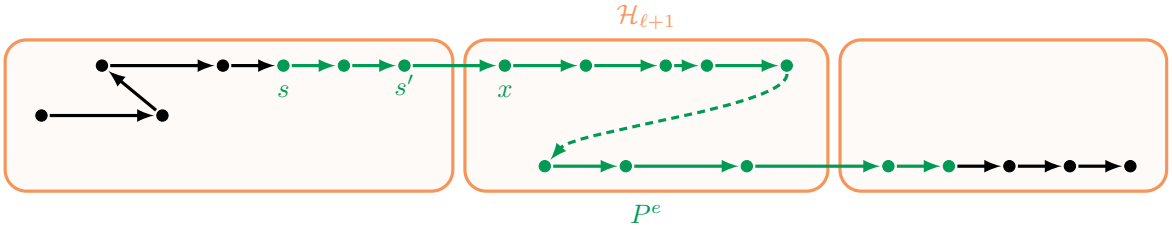


Figure 4.17: Visualization of the proof of Lemma 4.59 in the case $s \neq x$. All vertices are drawn from left to right according to the total order $\prec$. The ℓ-jump segment P_j (green) can be shrunk so that the resulting segment is entry-safe (and in this example also exit-safe).

We now discuss how we can get rid of backward edges of level ℓ. For an entry-safe and exit-safe ℓ-jump segment P containing a backward edge e of level ℓ, we can get rid of e (without introducing new backward edges of level ℓ) by replacing P with its ℓ-refinement. In particular, the path that arises by replacing P with its ℓ-refinement contains fewer backward edges of level ℓ. If P is not entry-safe or not exit-safe, we will need to be more careful. The idea is to first choose a removal set $R \subseteq V(P)$ such that it is safe to replace P with the ℓ-refinement of the path after removing R.

To define the removal set R, we define $H_1, \dots, H_r \in \mathcal{H}_{\ell+1}$ with j^*_{entry} (if P is entry-buffered) and j^*_{exit} (if P is exit-buffered) as previously. If P is entry-buffered, the index j^*_{entry} is defined and we set

$$R_{\text{exit}} \coloneqq \bigcup_{j=1}^{j^*_{\text{entry}}-1} H_j \cap V(P_{\text{exit}}),$$

otherwise, we define $R_{\text{exit}} \coloneqq \emptyset$. Analogously, if P is exit-buffered, we set

$$R_{\text{entry}} \coloneqq \bigcup_{j=j^*_{\text{exit}}+1}^{r} H_j \cap V(P_{\text{entry}}),$$

and otherwise we set $R_{\text{entry}} \coloneqq \emptyset$. We call $R \coloneqq R_{\text{entry}} \cup R_{\text{exit}}$ the *removal set* of the ℓ-jump segment P. See Figure 4.18 for an illustration.

For a path P and $A \subseteq V(P)$, we write $P[A]$ to denote the path obtained by skipping all vertices $v \in V(P) \setminus A$.

To prove Lemma 4.58, we consider a path P_{main} with an ℓ-splitting-free ℓ-buffering $\mathcal{B}$. We denote by $\mathcal{B}_\ell$ the set of jump segments from $\mathcal{B}$ that contain a backward edge of level ℓ.

Our procedure to get rid of all backward edges of level ℓ can then simply be described as follows: In the beginning, we initialize $R^* \coloneqq \emptyset$ as the empty set. For each $P \in \mathcal{B}_\ell$, we replace P with the ℓ-refinement of $P[V(P) \setminus R]$, where R is the removal set of P, and we add the vertices from R to R^*. See also Figure 4.19 for an example. We denote by P'_{main} the path that results from applying this procedure to the path P_{main}.

We now argue that P'_{main}, together with R^*, indeed satisfies the properties claimed in Lemma 4.58. We start by proving that P'_{main} is $(\ell+1)$-buffered. In this proof we exploit the fact that j^*_{entry} was not chosen smaller and j^*_{exit} was not chosen larger than how we chose it in (4.22) and (4.23).

Lemma 4.60. *P'_{main} is $(\ell+1)$-buffered.*

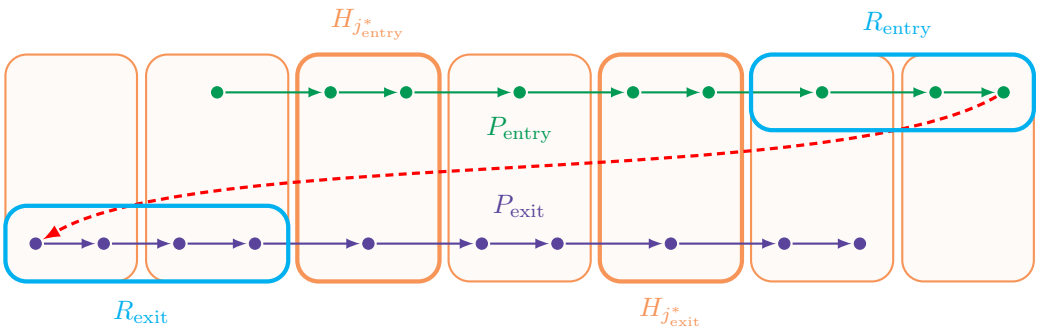


Figure 4.18: Illustration of our choice of R_{exit} and R_{entry}. All vertices are drawn from left to right according to the total order $\prec$. Observe that removing $R \coloneqq R_{\text{exit}} \cup R_{\text{entry}}$ from the depicted non-splitting ℓ-jump segment and replacing the resulting path with its ℓ-refinement changes neither its start vertex nor its end vertex.

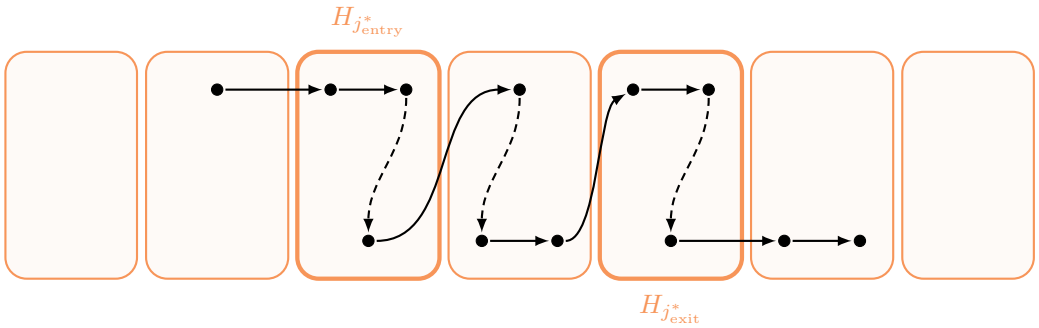


Figure 4.19: The ℓ-refinement Q of the path $P[V(P) \setminus R]$ we obtain for the ℓ- jump segment P from Figure 4.18. All three dashed edges are new backward edges whose level is strictly larger than ℓ.

Proof. We prove the following four claims in this particular order:

(i) Consider an ℓ-jump segment $P \in \mathcal{B}_\ell$ that is replaced with the ℓ-refinement Q of $P[V(P) \setminus R]$ in our procedure. Then we have $V(P) \setminus R \neq \emptyset$, that is, Q contains at least one vertex.

(ii) Let P and Q be as in Claim (i). Then the edge e_1 of P'_{main} that enters the first vertex of Q and the edge e_2 of P'_{main} that leaves the last vertex of Q are forward edges (in case such edges exist).

(iii) P'_{main} contains no backward edge of level $\leq \ell$.

(iv) P'_{main} is ℓ-buffered.

Observe that by Lemma 4.59, the Claims (iii) and (iv) together imply that P'_{main} is $(\ell+1)$-buffered. We prove each claim individually:

(i) Suppose for the sake of deriving a contradiction that $V(P) \setminus R = \emptyset$. Then $R_{\text{entry}} = V(P_{\text{entry}}) \neq \emptyset$ and $R_{\text{exit}} = V(P_{\text{exit}}) \neq \emptyset$. In particular, P is entry-buffered and exit-buffered and thus j^*_{entry} and j^*_{exit} are defined. By the definition of j^*_{entry}, there exists a vertex $v \in V(P_{\text{entry}}) \cap H_{j^*_{\text{entry}}} \neq \emptyset$. Because $P \in \mathcal{B}_\ell$ and $\mathcal{B}$ is ℓ-splitting-free, we have $j^*_{\text{entry}} \leq j^*_{\text{exit}}$, implying $v \notin R_{\text{entry}}$. This contradicts $V(P) \setminus R = \emptyset$.

(ii) To show the second claim, we prove that Q is entry-safe within P'_{main} or the first vertex of P is identical to the first vertex of Q. By a symmetric argument, it then follows that Q is exit-safe within P'_{main} or the last vertex of P is identical to the last vertex of Q. Applying this to all the replacement steps we perform in our procedure proves the second claim because any backward edge of P_{main} must be contained in a jump-segment from $\mathcal{B}$ and these jump segments are vertex disjoint.

First, we observe that if P is entry-safe within P_{main}, this immediately implies that Q is entry-safe within P'_{main}. Now consider the case where P is entry-buffered and we show that in this case the first vertex of the path Q is identical to the first vertex of P. Let $H_1, \ldots, H_r$ be the sets as previously defined for P. If P is entry-buffered, the index j^*_{entry} is defined. Let $j \in [r]$ be such that H_j contains the first vertex v of the path P. Because P_{entry} is monotone, we have $j \leq j^*_{\text{entry}}$ and every vertex $w \in V(P_{\text{entry}})$ is contained in a set H_i with $i \geq j$. We have $j^*_{\text{entry}} \leq j^*_{\text{exit}}$ if j^*_{exit} is defined because $\mathcal{B}$ is ℓ-splitting-free. This implies that $v \notin R_{\text{entry}}$ and therefore $v \in V(Q)$. Moreover, by the definition of R_{exit}, every vertex $w \in V(P_{\text{exit}})$ that is contained in a set H_i with $i < j$ is contained in R. We conclude that every vertex $w \in V(P) \setminus R$ is contained in a set H_i with $i \geq j$, which implies (by the definition of the ℓ-refinement) that indeed v is the first vertex of the path Q.

A symmetric argument shows that Q is exit-safe or the last vertex of P is identical to the last vertex of Q.

(iii) This follows immediately from Claims (i) and (ii) because the ℓ-refinement Q of $P[V(P) \setminus R]$ does not contain any backward edge of level ℓ by the definition of the ℓ-refinement. Hence, all backward edges of level ℓ are removed by our procedure (and by Claims (i) and (ii) no new ones are introduced).

(iv) In order to show that P'_{main} is ℓ-buffered, we extend our procedure described above and explicitly construct an ℓ-buffering of P'_{main} during the process of obtaining P'_{main} from P_{main}. Initially, we set $\mathcal{B}' := \mathcal{B}$. Consider an iteration in which we replace an ℓ-jump segment $P \in \mathcal{B}_\ell$ with the ℓ-refinement Q of $P[V(P) \setminus R]$. As before, let $H_1, \ldots, H_r \in \mathcal{H}_{\ell+1}$ be the sets defined above for P. Because P_{entry} and P_{exit} are monotone, the ℓ-refinement Q of $P[V(P) \setminus R]$ contains at most one backward edge within each set H_i (with $i \in [r]$). See Figure 4.19. For a backward edge e of Q, let H_i be the set containing e and let

$Q^e \coloneqq Q[V(Q) \cap H_i]$. In other words, Q^e is the maximal subpath of Q within the set H_i. Then, we replace P in the current $\mathcal{B}'$ via

$$\mathcal{B}' \;\coloneqq\; \big(\mathcal{B}' \setminus \{P\}\big) \cup \big\{Q^e : e \text{ is a backward edge of } Q\big\}. \tag{4.28}$$

We now show that the resulting $\mathcal{B}'$ at the end of our procedure is an ℓ-buffering of P'_{main}. By construction, $\mathcal{B}'$ is a collection of vertex disjoint subpaths of P'_{main} and every backward edge of P'_{main} is contained in one of these subpaths (using Claim (ii)). Moreover, every element of $\mathcal{B}'$ is an ℓ'-jump segment for some $\ell' \geq \ell$.

We consider a single iteration of our procedure in which we replace P with Q (and modify $\mathcal{B}'$ as in (4.28)). First, note that for any jump segment $P' \in \mathcal{B}' \setminus \{P\}$, the properties of being entry-safe, exit-safe, entry-buffered, and exit-buffered are preserved under this replacement step. Hence, in order to complete the proof of this claim, it suffices to show that each Q^e is entry-buffered or entry-safe within P'_{main}, and exit-buffered or exit-safe within P'_{main}, where e is a backward edge of Q and Q^e is defined as above.

Let Q^e be such a subpath of Q and we show that Q^e is entry-buffered or entry-safe within P'_{main}. Consider the case that Q^e is not an initial part of Q, i.e., the start points of Q^e and Q are not identical. Since Q^e is the maximal subpath of Q that is contained in $H_j \in \mathcal{H}_{\ell+1}$ (for appropriately chosen $j \in [r]$) and Q does not contain any backward edges of level ℓ, the path Q and thus P'_{main} contains a forward edge of level ℓ that enters the first vertex of Q^e. Therefore Claim (iii) implies that, in this case, Q^e is entry-safe within P'_{main}.

Now consider the case that Q^e is an initial part of Q. If P is entry-safe within P_{main}, then Q^e is entry-safe within P'_{main}. So we may assume that P is entry-buffered. Recall from the proof of Claim (ii) that in this case the first vertex of Q is identical to the first vertex of P, and thus the first vertex of P_{entry}.

We have $Q^e = Q[V(Q) \cap H_j]$ for some $j \in [r]$. We claim $j = j^*_{\text{entry}}$. Because Q^e contains a backward edge, it must contain a vertex from P_{exit} and thus we must have $j \geq j^*_{\text{entry}}$ (where we used that the vertices in R_{exit} were removed and are not contained in Q^e). If P is exit-buffered, then $j^*_{\text{entry}} \leq j^*_{\text{exit}}$ because $P \in \mathcal{B}_\ell$ and $\mathcal{B}$ is ℓ-splitting-free. In particular, the vertices from $H_{j^*_{\text{entry}}} \cap V(P_{\text{entry}})$ are not contained in the removal set R. This also applies in case P is not exit-buffered in which case $R_{\text{entry}} = \emptyset$. We conclude that $V(Q) \cap H_{j^*_{\text{entry}}} \neq \emptyset$ (where we used that P contains at least one vertex from $H_{j^*_{\text{entry}}}$ by the definition of j^*_{entry}). Because Q^e contains the first vertex of Q, this implies $j \leq j^*_{\text{entry}}$. Thus, indeed $j = j^*_{\text{entry}}$.

Recall that $R_{\text{entry}} = \emptyset$ (in case P is not exit-buffered) or $j = j^*_{\text{entry}} \leq j^*_{\text{exit}}$ because P is non-splitting. In both cases, $R_{\text{entry}} \cap H_j = \emptyset$. We conclude that the entry part of the jump segment Q^e is $P[V(P_{\text{entry}}) \cap H_j]$. Moreover, the number of vertices in this entry part is

$$\big|V(P_{\text{entry}})\big| - \sum_{i=j^*_{\text{entry}}+1}^{r} |V(P_{\text{entry}} \cap H_i|,$$

which by the definition of j^*_{entry} is at least

$$\big|V(P_{\text{entry}})\big| - \frac{k}{64(L-1)} \;\geq\; \frac{k}{16} - (\ell-1)\cdot\frac{k}{32(L-1)} - \frac{k}{64(L-1)} \;\geq\; \frac{k}{16} - \ell\cdot\frac{k}{32(L-1)},$$

implying that Q^e is entry-buffered (where we used $\text{level}(V(Q^e)) \geq \ell+1$).

An analogous argument implies that Q^e is exit-buffered or exit-safe within P'_{main}. This completes the proof of Claim (iv) and thus of our lemma. □

Because reinserting removed vertices comes at some cost, we need to bound the number of vertices in the removal set R. In the following lemma we use that j^*_{entry} was not chosen larger and j^*_{exit} was not chosen smaller than how we chose it in (4.22) and (4.23).

Lemma 4.61. *Let P be an ℓ-jump segment. Then its removal set R satisfies*

$$|R| \cdot 2kD_\ell \leq \eta_2 \cdot c^{(k)}(P), \tag{4.29}$$

where $\eta_2 \coloneqq 2^8 L$.

Proof. For each vertex $x \in R_{\text{exit}}$ and each vertex $v \in \bigcup_{j=j^*_{\text{entry}}}^{r} \big(V(P_{\text{entry}}) \cap H_j\big)$, the edge (v,x) is a backward edge of level ℓ and this edge contributes to $c^{(k)}(P)$, because $|V(P)| \leq k+1$ and v precedes x in P. Therefore,

$$c^{(k)}(P) \;\geq\; \left(|R_{\text{exit}}| \cdot \sum_{j=j^*_{\text{entry}}}^{r} |V(P_{\text{entry}}) \cap H_j| \right) \cdot D_\ell \;\geq\; \left(|R_{\text{exit}}| \cdot \frac{k}{64(L-1)} \right) \cdot D_\ell,$$

where the second inequality follows from the definition of j^*_{entry}. Analogously, we also obtain $c^{(k)}(P) \geq \big(|R_{\text{entry}}| \cdot \frac{k}{64(L-1)}\big) \cdot D_\ell$, and thus

$$2 \cdot c^{(k)}(P) \;\geq\; \Big(|R_{\text{exit}}| \cdot \tfrac{k}{64(L-1)} + |R_{\text{entry}}| \cdot \tfrac{k}{64(L-1)}\Big) \cdot D_\ell \;=\; |R| \cdot D_\ell \cdot \tfrac{k}{64(L-1)}.$$

□

We are now ready to complete the proof of Lemma 4.58 with the following cost bounds:

Lemma 4.62. *We have*

$$c^{(k)}_{>\ell}(P'_{\text{main}}) \leq c^{(k)}(P_{\text{main}}),$$

and

$$c^{(k)}(P'_{\text{main}}) + |R^*| \cdot 2kD_\ell \leq \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{main}}).$$

Proof. Suppose our procedure constructing P'_{main} from P_{main} as explained above replaces the subpaths $P_1, \ldots, P_m \in \mathcal{B}_\ell$ with $Q_1, \ldots, Q_m$, respectively. We denote by R_j the removal set of each P_j, i.e., $R^* = \bigcup_{j=1}^{m} R_j$.

Note that $P_j[V(P_j) \setminus R_j]$ is a subpath of $P_{\text{main}}[V(P_{\text{main}}) \setminus R^*]$, and P'_{main} can be equivalently obtained from $P_{\text{main}}[V(P_{\text{main}}) \setminus R^*]$ by replacing each $P_j[V(P_j) \setminus R_j]$ with Q_j. Thus, by Lemma 4.48 and Lemma 4.21, we have

$$\begin{aligned}
c^{(k)}_{>\ell}(P'_{\text{main}}) &\leq c^{(k)}_{>\ell}\Big(P_{\text{main}}[V(P_{\text{main}}) \setminus R^*]\Big) + \sum_{j=1}^{m} c^{(k)}_{=\ell}\Big(P_j[V(P_j) \setminus R_j]\Big) \\
&\leq c^{(k)}_{>\ell}\Big(P_{\text{main}}[V(P_{\text{main}}) \setminus R^*]\Big) + c^{(k)}_{=\ell}\Big(P_{\text{main}}[V(P_{\text{main}}) \setminus R^*]\Big) \\
&\leq c^{(k)}\Big(P_{\text{main}}[V(P_{\text{main}}) \setminus R^*]\Big) \\
&\leq c^{(k)}(P_{\text{main}}).
\end{aligned}$$

For $j \in [m]$, Lemmas 4.21 and 4.48 imply that

$$c^{(k)}(Q_j) \leq c^{(k)}\Big(P_j[V(P_j) \setminus R_j]\Big) \leq c^{(k)}(P_j).$$

Combining this with Lemma 4.46 and Lemma 4.61, we obtain

$$\begin{aligned} c^{(k)}(P'_{\text{main}}) + |R^*| \cdot 2kD_\ell &\le c^{(k)}(P_{\text{main}}) + \sum_{j=1}^{m} 2^{11} \cdot c^{(k)}(P_j) + \sum_{j=1}^{m} c^{(k)}(Q_j) + |R^*| \cdot 2kD_\ell \\ &\le c^{(k)}(P_{\text{main}}) + \sum_{j=1}^{m} \Big(\big(2^{11}+1\big) \cdot c^{(k)}(P_j) + |R_j| \cdot 2kD_\ell \Big) \\ &\le c^{(k)}(P_{\text{main}}) + \sum_{j=1}^{m} \big(2^{11}+1+\eta_2\big) \cdot c^{(k)}(P_j) \\ &\le \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{main}}). \end{aligned}$$

□

4.7.8 Proof of Lemma 4.54

Recall that a path-level pair (P, j) covers an element (v, ℓ) if

- $v \in V(P)$ and $j \le \ell$, or
- (P, j) takes responsibility for the set $H_{\ell+1}(v) \in \mathcal{H}_{\ell+1}$ containing v, i.e, we have $|V(P) \cap H_{\ell+1}(v)| \ge k$ and $j \le \ell < L$.

Further, recall that a *main-tour cover* of a set $U \subseteq V$ is a monotone path P_{main} with $V(P_{\text{main}}) \subseteq U$ together with a set $\mathcal{P} \subseteq \mathcal{S}$ of path-level pairs satisfying the following two conditions:

(i) $V(P_{\text{main}}) = U$ or $|V(P_{\text{main}})| \ge k$.

(ii) The path-level pairs $\mathcal{P} \cup \big\{\big(P_{\text{main}}, \text{level}(U)\big)\big\}$ cover the elements $U \times \big\{\text{level}(U), \dots, L\big\}$ of the Set Cover universe $\mathcal{U}$.

We remark that the monotone path with vertex set U together with $\mathcal{P} = \emptyset$ is always a main-tour cover of U.

Finally, we prove the main result of this section, Lemma 4.54. We prove the statement by induction and it will be useful to prove a slightly stronger version:

Lemma 4.63. *Let P_{old} be an ℓ-buffered path with $\ell \coloneqq \text{level}(V(P_{\text{old}}))$. Then there is a main-tour cover $(P_{\text{new}}, \mathcal{P})$ of $V(P_{\text{old}})$ such that*

$$c^{(k)}(P_{\text{new}}) + w(\mathcal{P}) \;\le\; \begin{cases} \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot c^{(k)}(P_{\text{old}}) & \text{if } P_{\text{old}} \text{ has an } \ell\text{-splitting-free } \ell\text{-buffering} \\ (L-\ell+1)\eta^* \cdot c^{(k)}(P_{\text{old}}) & \text{otherwise.} \end{cases}$$

where $\eta^* \coloneqq 3\eta_1 + \big(\eta_2 + 2^{13}\big)$, $\eta_1 \coloneqq 2^{12}L^2$, and $\eta_2 \coloneqq 2^8 L$.

Proof. We prove the statement by induction on $L - \ell$. First, suppose that $\ell = L$. Then, P_{old} consists of a single vertex. In that case, we can choose $P_{\text{new}} \coloneqq P_{\text{old}}$ and $\mathcal{P} \coloneqq \emptyset$. This is a trivial main-tour cover of $V(P_{\text{old}})$ and $c^{(k)}(P_{\text{old}}) = 0$.

For the induction step, suppose that $\ell < L$ and assume as our induction hypothesis that the lemma holds for $\bar{\ell}$-buffered paths $\bar{P}_{\text{old}}$ with $\bar{\ell} \coloneqq \text{level}(V(\bar{P}_{\text{old}}))$ where $\bar{\ell} > \ell$. In order to prove that it also holds for ℓ-buffered paths with level ℓ, we apply another induction on the number of backward edges contained in P_{old}. The base case of this second induction is easy: If P_{old} contains no backward edges, then $P_{\text{new}} \coloneqq P_{\text{old}}$ together with $\mathcal{P} \coloneqq \emptyset$ is a trivial main-tour cover of $V(P_{\text{old}})$ and $c^{(k)}(P_{\text{new}}) + w(\mathcal{P}) = c^{(k)}(P_{\text{old}}) \le ((L-\ell+1)\eta^* - 3\eta_1) \cdot c^{(k)}(P_{\text{old}})$ and P_{old} has an ℓ-splitting-free ℓ-buffering (namely the empty buffering).

For the induction step of our second induction, we distinguish three cases:

Case 1: Assume that $|V(P_{\text{old}})| \leq k$.
Choose P_{new} to be the monotone path with vertex set $V(P_{\text{old}})$ and $\mathcal{P} \coloneqq \emptyset$. This is a trivial main-tour cover of $V(P_{\text{old}})$. By Lemma 4.44 we have $c^{(k)}(P_{\text{new}}) \leq c^{(k)}(P_{\text{old}})$.

Case 2: Assume that P_{old} has no ℓ-splitting-free ℓ-buffering and $|V(P_{\text{old}})| \geq k$.
Let $\mathcal{B}$ be an ℓ-buffering of P_{old}. Then $\mathcal{B}$ is not ℓ-splitting-free. Let $Q_1, \dots, Q_r$ be the paths that result from P_{old} by removing all backward edges contained in some splitting ℓ-jump segment from $\mathcal{B}$. We have $\bigcup_{j=1}^{r} V(Q_j) = V(P_{\text{old}})$ and

$$\sum_{j=1}^{r} c^{(k)}(Q_j) \leq c^{(k)}(P_{\text{old}}). \tag{4.30}$$

Furthermore, $r \geq 2$ and therefore, by Lemma 4.57,

$$3\eta_1 \cdot c^{(k)}(P_{\text{old}}) \geq 3(r-1) \cdot k^2 D_\ell \geq (r+1) \cdot k^2 D_\ell. \tag{4.31}$$

Let $j \in [r]$. There exists an ℓ-splitting-free ℓ-buffering of Q_j (given by those elements of $\mathcal{B}$ from which we did not remove the backward edge and that are subpaths of Q_j). Furthermore, Q_j contains strictly fewer backward edges than P_{old}. Thus, we can apply the induction hypothesis of our second induction and obtain a main-tour cover $(\widetilde{Q}_j, \mathcal{P}_j)$ of $V(Q_j)$.

Choose P_{new} to be the monotone path on k arbitrarily chosen vertices from $V(P_{\text{old}})$. This is possible, because $|V(P_{\text{old}})| \geq k$. Note that $c^{(k)}(P_{\text{new}}) \leq k^2 D_\ell$. For $\mathcal{P}$ we choose

$$\mathcal{P} \coloneqq \bigcup_{j=1}^{r} \Big(\mathcal{P}_j \cup \big\{ (\widetilde{Q}_j, \ell) \big\} \Big).$$

Next, we show that P_{new} together with $\mathcal{P}$ is indeed a main-tour cover of U. Property (i) is satisfied because $|V(P_{\text{new}})| \geq k$. Because $\bigcup_{j=1}^{r} V(\widetilde{Q}_j) = V(P_{\text{old}})$, in order to prove (ii) it suffices to show that for each $j \in [r]$, the elements of $V(Q_j) \times \{\ell, \dots, L\}$ are covered. Since $(\widetilde{Q}_j, \mathcal{P}_j)$ is a main-tour cover of $V(Q_j)$, $\mathcal{P}_j \cup \big\{ \big(\widetilde{Q}_j, \text{level}(V(Q_j))\big) \big\}$ covers each element from $V(Q_j) \times \{\text{level}(V(Q_j)), \dots, L\}$. Now consider (v, ℓ') with $v \in V(Q_j)$ and $\ell \leq \ell' < \text{level}(V(Q_j))$. If $v \in V(\widetilde{Q}_j)$, then (v, ℓ') is covered by the pair $(\widetilde{Q}_j, \ell)$. So we may assume that this is not the case, which in particular implies $V(Q_j) \neq V(\widetilde{Q}_j)$. By property (i) of main-tour coverings, we have $|V(\widetilde{Q}_j)| \geq k$. The hierarchical structure of our instance implies that we have $H_{\ell'+1}(v) \supseteq V(Q_j)$ (where we used $v \in V(Q_j)$ and $\ell' + 1 \leq \text{level}(V(Q_j))$). Thus, we have

$$|H_{\ell'+1}(v) \cap V(\widetilde{Q}_j)| = |V(\widetilde{Q}_j)| \geq k,$$

which implies that the pair $(\widetilde{Q}_j, \ell)$ takes responsibility for the set $H_{\ell'+1}(v)$ and thus covers (v, ℓ'). We conclude that $(P_{\text{new}}, \mathcal{P})$ is a main-tour cover of $V(P_{\text{old}})$.

Now we bound $c^{(k)}(P_{\text{new}}) + w(\mathcal{P})$. Because P_{new} is a path with k vertices contained in a set of level ℓ, we have $c^{(k)}(P_{\text{new}}) \leq k^2 D_\ell$. For each $j \in [r]$ we have

$$w(\widetilde{Q}_j, \ell) \;=\; c^{(k)}\big(\widetilde{Q}_j\big) + k^2 \cdot D_\ell.$$

If $\text{level}(\widetilde{Q}_j)) > \ell$, the induction hypothesis of our first induction implies

$$c^{(k)}\big(\widetilde{Q}_j\big) + w(\mathcal{P}_j) \;\leq\; (L-\ell)\eta^* \cdot c^{(k)}\big(Q_j\big) \;\leq\; \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot c^{(k)}\big(Q_j\big),$$

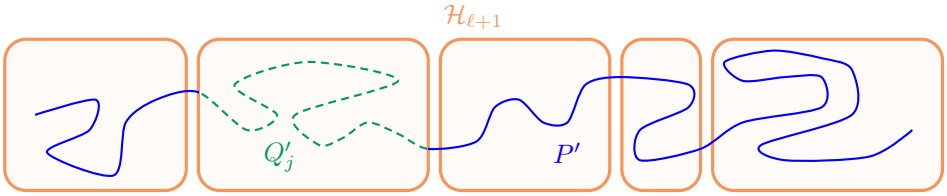


Figure 4.20: Illustration of case 3 in the proof of Lemma 4.63. The path P' visits each $H \in \mathcal{H}_{\ell+1}$ at most once and has no backward edges of level ℓ. Therefore, we can apply the induction hypothesis to each of the Q'_j, which are the maximal $(\ell+1)$-confined subpaths of P'.

where we used $\eta^* \geq 3\eta_1$. If $\text{level}(\widetilde{Q}_j) = \ell$, then Q_j has an ℓ-splitting-free ℓ-buffering as discussed above. Since we obtained $(\widetilde{Q}_j, \mathcal{P}_j)$ by applying the induction hypothesis of our second induction, we hence know that

$$c^{(k)}\big(\widetilde{Q}_j\big) + w(\mathcal{P}_j) \;\leq\; \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot c^{(k)}\big(Q_j\big).$$

We conclude that in both cases

$$\begin{aligned} w\big(\mathcal{P}_j \cup \big\{(\widetilde{Q}_j, \ell)\big\}\big) &= c^{(k)}\big(\widetilde{Q}_j\big) + k^2 \cdot D_\ell + w(\mathcal{P}_j) \\ &\leq k^2 \cdot D_\ell + \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot c^{(k)}\big(Q_j\big). \end{aligned}$$

Summing over all $j \in [r]$ and using $c^{(k)}(P_{\text{new}}) \leq k^2 D_\ell$, we conclude

$$\begin{aligned} c^{(k)}(P_{\text{new}}) + w(\mathcal{P}) &\leq (r+1) \cdot k^2 D_\ell + \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot \sum_{j=1}^{r} c^{(k)}\big(Q_j\big) \\ &\overset{(4.30)}{\leq} (r+1) \cdot k^2 D_\ell + \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot c^{(k)}(P_{\text{old}}) \\ &\overset{(4.31)}{\leq} 3\eta_1 \cdot c^{(k)}(P_{\text{old}}) + \big((L-\ell+1)\eta^* - 3\eta_1\big) \cdot c^{(k)}(P_{\text{old}}) \\ &= (L-\ell+1)\eta^* \cdot c^{(k)}(P_{\text{old}}). \end{aligned}$$

Case 3: Assume that P_{old} has an ℓ-splitting-free ℓ-buffering $\mathcal{B}$.
Let P' and $R \subseteq V(P_{\text{old}})$ be obtained by applying Lemma 4.58 to P_{old}. The path P' is $(\ell+1)$-buffered, hence it contains no backward edges of level ℓ. Therefore, P' enters and leaves each $H \in \mathcal{H}_{\ell+1}$ at most once. Let $Q'_1, \dots, Q'_r$ be the inclusion-wise maximal $(\ell+1)$-confined subpaths of P', numbered so that $V(Q'_j)$ is left of $V(Q'_{j+1})$ for all $j \in [r-1]$. (We call a path $(\ell+1)$-confined if its vertex set is $(\ell+1)$-confined, cf. Definition 4.28.)

In other words, the paths Q'_j are obtained by removing all (forward) edges of P' of level ℓ. See Figure 4.20 for an illustration. Because P' visits each $H \in \mathcal{H}_{\ell+1}$ at most once, we have

$$\sum_{j=1}^{r} c^{(k)}(Q'_j) \;=\; c^{(k)}_{>\ell}(P') \;\overset{(4.26)}{\leq}\; c(P_{\text{old}}), \tag{4.32}$$

where we used (4.26) from Lemma 4.58. Moreover, $\text{level}(Q'_j) \geq \ell+1$ and Q'_j is $\text{level}(Q'_j)$-buffered for all $j \in [r]$ (by Lemma 4.59). Thus, for each $j \in [r]$ we can apply the induction hypothesis of our first induction and obtain a main-tour cover $(\widetilde{Q}_j, \widetilde{\mathcal{P}}_j)$ of $V(Q'_j)$.

Each path $\widetilde{Q}_j$ is monotone and we have $V(\widetilde{Q}_j) \subseteq V(Q'_j)$, where $V(Q'_j)$ is left of $V(Q'_{j+1})$. Thus, concatenating the paths $\widetilde{Q}_j$ into a single path in an order of increasing j yields a monotone path, which we denote by $\widetilde{P}$. In other words, $\widetilde{P}$ results from P' by replacing each Q'_j with $\widetilde{Q}_j$. Thus, by Lemma 4.46, we obtain

$$c^{(k)}(\widetilde{P}) \;\leq\; c^{(k)}(P') + \sum_{j=1}^{r} 2^{11} \cdot c^{(k)}(Q'_j) + \sum_{j=1}^{r} c^{(k)}(\widetilde{Q}_j) \tag{4.33}$$

Let P_{new} be the path that results from $\widetilde{P}$ by inserting each vertex $v \in R$ such that the resulting path remains monotone. When inserting a vertex into a path, there are at most $2k$ new edges that contribute to the k-hop cost of the path. Because all vertices of P_{new} are contained in the set $V(P_{\text{old}})$, which has level ℓ, each such new edge has cost at most D_ℓ. Thus, we obtain

$$\begin{aligned} c^{(k)}(P_{\text{new}}) \;&\leq\; c^{(k)}(\widetilde{P}) + |R| \cdot 2kD_\ell \\ &\overset{(4.33)}{\leq}\; c^{(k)}(P') + |R| \cdot 2kD_\ell + \sum_{j=1}^{r} 2^{11} \cdot c^{(k)}(Q'_j) + \sum_{j=1}^{r} c^{(k)}(\widetilde{Q}_j) \\ &\overset{(4.27)}{\leq}\; \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{old}}) + \sum_{j=1}^{r} 2^{11} \cdot c^{(k)}(Q'_j) + \sum_{j=1}^{r} c^{(k)}(\widetilde{Q}_j), \end{aligned} \tag{4.34}$$

where we used (4.27) from Lemma 4.58.

We define $\mathcal{P} \coloneqq \bigcup_{j=1}^{r} \widetilde{\mathcal{P}}_j$. Since $(\widetilde{Q}_j, \widetilde{\mathcal{P}}_j)$ is a main-tour cover of $V(Q'_j)$ for all $j \in [r]$, we have

$$c^{(k)}(\widetilde{Q}_j) + w(\widetilde{\mathcal{P}}_j) \;\leq\; (L - (\ell+1) + 1)\eta^* \cdot c^{(k)}(Q'_j), \tag{4.35}$$

implying

$$\begin{aligned} c^{(k)}(P_{\text{new}}) + w(\mathcal{P}) \;&\overset{(4.34)}{\leq}\; \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{old}}) + \sum_{j=1}^{r} 2^{11} \cdot c^{(k)}(Q'_j) + \sum_{j=1}^{r} c^{(k)}(\widetilde{Q}_j) + w(\mathcal{P}) \\ &=\; \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{old}}) + \sum_{j=1}^{r} 2^{11} \cdot c^{(k)}(Q'_j) + \sum_{j=1}^{r} \Big(c^{(k)}(\widetilde{Q}_j) + w(\widetilde{\mathcal{P}}_j)\Big) \\ &\overset{(4.35)}{\leq}\; \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{old}}) + \big((L - (\ell+1) + 1)\eta^* + 2^{11}\big) \cdot \sum_{j=1}^{r} c^{(k)}(Q'_j) \\ &\overset{(4.32)}{\leq}\; \big(\eta_2 + 2^{12}\big) \cdot c^{(k)}(P_{\text{old}}) + \big((L-\ell)\eta^* + 2^{11}\big) \cdot c^{(k)}(P_{\text{old}}) \\ &\leq \big((L-\ell)\eta^* + \eta_2 + 2^{13}\big) \cdot c^{(k)}(P_{\text{old}}) \\ &= (L - \ell + 1)\eta^* \cdot c^{(k)}(P_{\text{old}}) - 3\eta_1 \cdot c^{(k)}(P_{\text{old}}). \end{aligned}$$

It remains to argue that $(P_{\text{new}}, \mathcal{P})$ is a main-tour cover of $V(P_{\text{old}})$. If $V(Q'_j) = V(\widetilde{Q}_j)$ for all $j \in [r]$, then $V(P_{\text{old}}) = V(P_{\text{new}})$. Otherwise, we have $|V(P_{\text{new}})| \geq |\bigcup_{j=1}^{r} V(\widetilde{Q}_j)| \geq k$, since each $(\widetilde{Q}_j, \widetilde{\mathcal{P}}_j)$ is a main-tour cover. This shows property (i) of a main-tour cover.

We now prove that $(P_{\text{new}}, \mathcal{P})$ also satisfies (ii). Let $(v, \ell') \in V(P_{\text{old}}) \times \{\ell, \dots, L\}$. If $v \in V(P_{\text{new}})$, then (v, ℓ') is covered by (P_{new}, ℓ). Hence, we may assume that this is not the case. Because $R \subseteq V(P_{\text{new}})$, this implies $v \in V(P')$ and thus $v \in V(Q'_j)$ for some $j \in [r]$.

If $\ell' \geq \text{level}(Q'_j)$, then either a path-level pair from $\widetilde{\mathcal{P}}_j \subseteq \mathcal{P}$ covers (v, ℓ'), or the path-level pair $\big(\widetilde{Q}_j, \text{level}(V(\widetilde{Q}_j))\big)$ covers (v, ℓ') (this follows from property (ii) of $(\widetilde{Q}_j, \widetilde{\mathcal{P}}_j)$ as a main-tour cover of $V(Q'_j)$). In the latter case, the path-level pair (P_{new}, ℓ) covers (v, ℓ') because $\ell \leq \ell'$ and $V(\widetilde{Q}_j) \subseteq V(P_{\text{new}})$.

It remains to consider the case $\ell' < \text{level}(Q'_j)$, i.e., $\ell' + 1 \leq \text{level}(Q'_j)$. In particular $V(Q'_j) \subseteq H_{\ell'+1}(v)$ (by the hierarchical structure of our instance and because $v \in V(Q'_j)$ and $v \in H_{\ell'+1}(v)$). Because $v \notin V(P_{\text{new}}) \supseteq V(\widetilde{Q}_j)$, we have $V(\widetilde{Q}_j) \neq V(Q'_j)$ and thus by property (i) of the main-tour cover $(\widetilde{Q}_j, \mathcal{P}_j)$, we must have $|V(\widetilde{Q}_j)| \geq k$. Therefore,

$$|V(P_{\text{new}}) \cap H_{\ell'+1}(v)| \geq |V(\widetilde{Q}_j) \cap H_{\ell'+1}(v)| \geq |V(\widetilde{Q}_j)| \geq k,$$

so (P_{new}, ℓ) covers (v, ℓ') because $\ell \leq \ell'$. We conclude that $(P_{\text{new}}, \mathcal{P})$ is a main-tour cover of $V(P_{\text{old}})$. □

4.8 Solving the covering problem via an algorithm for monotone paths

In this section we provide a quasi-polynomial-time $\mathcal{O}(\log^2 n)$-approximation for our covering problem. More precisely, we prove the following theorem:

Theorem 4.64. *For every instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ of Path Covering with low depth, we can compute in time $n^{\mathcal{O}(\log n)}$ a solution $\mathcal{P}$ with $w(\mathcal{P}) \leq \mathcal{O}(\log^2 n) \cdot \text{OPT}(\mathcal{I})$, where n denotes the number of vertices in $\mathcal{I}$.*

Combining Theorems 4.5, 4.36 and 4.64 yields our main result, Theorem 4.3. To prove Theorem 4.64, we first discuss what kind of oracle we need in order to apply the greedy Set Cover algorithm (Section 4.8.1). In Sections 4.8.2 to 4.8.5 we then develop such an oracle, using an approach based on dynamic programming.

4.8.1 Applying the greedy Set Cover algorithm

To prove Theorem 4.64, we view Path Covering as a Set Cover problem as in Lemma 4.37, where the task is to cover all elements of the universe $\mathcal{U} = V \times [L]$ by path-level pairs of minimum total weight. Starting with $U = \mathcal{U}$ and $\mathcal{P} = \emptyset$, the greedy algorithm for Set Cover iteratively chooses a path-level pair (P, ℓ) (approximately) minimizing

$$\frac{w(P, \ell)}{|\{(v, j) \in U : (v, j) \text{ covered by } (P, \ell)\}|}, \tag{4.36}$$

adds (P, ℓ) to $\mathcal{P}$, and removes the elements covered by (P, ℓ) from U. Once the set U of uncovered elements becomes empty, the algorithm returns the Path Covering solution $\mathcal{P}$. The well-known analysis of this algorithm (see e.g. [WS11]) shows that if we choose in every iteration of the algorithm a path-level pair (P, ℓ) that minimizes (4.36) up to a factor α, then we obtain an approximation factor of

$$\alpha \cdot \mathcal{O}(\log |\mathcal{U}|) = \alpha \cdot \mathcal{O}(\log n).$$

for our Path Covering problem. Thus, in order to prove Theorem 4.64, it suffices to provide an $\mathcal{O}(\log n)$-approximation algorithm with running time $n^{\mathcal{O}(\log n)}$ for the following problem: Given an instance of Path Covering and a nonempty set $U \subseteq \mathcal{U}$, compute a path-level pair (P, ℓ) minimizing (4.36).

In the remainder of this section we develop such an algorithm. Given a Path Covering instance $\mathcal{I} = (V, L, \mathcal{H}, D, \prec, c, k)$ and a nonempty set $U \subseteq \mathcal{U}$ of uncovered elements, we write

$$S_\ell \coloneqq \Big\{ v \in V : (v, \ell) \notin U \Big\}.$$

to denote the set of vertices for which (v, ℓ) is already covered.

Let (P^*, ℓ^*) denote an optimum solution to our problem, i.e, a path-level pair that minimizes (4.36). We define

- $\gamma^* \coloneqq |\{(v, \ell) \in U : (v, \ell) \text{ covered by } (P^*, \ell^*)\}|$, and
- $H^* \in \mathcal{H}_{\ell^*}$ to be the unique set in the partition $\mathcal{H}_{\ell^*}$ that satisfies $V(P^*) \subseteq H^*$.

Note that such a set H^* exists by the definition of a path-level pair. We also observe there are only polynomially many choices for ℓ^*, H^*, and γ^*. Thus, by enumerating all possibilities, we can assume in the following that we guessed these values correctly.

In the following we abbreviate

$$\begin{aligned} w(P) &\coloneqq w(P, \ell^*), \quad \text{and} \\ \gamma(P) &\coloneqq |\{(v, \ell) \in U : (v, \ell) \text{ covered by } (P, \ell^*)\}| \\ &= \sum_{\ell=\ell^*}^{L} |\{v \in V : (v, \ell) \text{ covered by } (P, \ell^*) \text{ and } v \notin S_\ell\}|, \end{aligned}$$

and we will restrict ourselves to monotone paths P with $V(P) \subseteq H^*$. Thus, our goal is now to find a quasi-polynomial-time $\mathcal{O}(\log n)$-approximation for the problem of finding a monotone path P with $V(P) \subseteq H^*$ that minimizes $\frac{w(P)}{\gamma(P)}$.

4.8.2 Approximating the path weights

As mentioned in Section 4.2.4, we will approximate the weight $w(P)$ of a path by some weight bound $w^q(P)$. In this section we define these weight bounds and prove that they indeed provide a good approximation. We fix an integer $k' \in \{k, \dots, 2k\}$ such that $k' = 2^\Gamma - 1$ for some integer $\Gamma \in \mathbb{Z}$. For each offset $q \in \{0, \dots, k'\}$ and every monotone path P, we will define a weight bound $w^q(P)$. These weight bounds will have the following key properties:

(i) for every monotone path P and every offset $q \in \{0, \dots, k'\}$, we have $w^q(P) \geq w(P)$,

(ii) for every monotone path P, there exists an offset $q \in \{0, \dots, k'\}$ such that $w^q(P) = \mathcal{O}(\log k) \cdot w(P)$, and

(iii) we can find a monotone path P with $V(P) \subseteq H^*$ and an offset $q \in \{0, \dots, k'\}$ minimizing $\frac{w^q(P)}{\gamma(P)}$ in quasi-polynomial time.

To define our weight approximations $w^q(P)$ we assign the vertices of P to different heights, as illustrated in Figure 4.21. More precisely, we define the two functions height$\colon \mathbb{Z} \to \{0, \dots, \Gamma\}$ and width$\colon \mathbb{Z} \to \{0, \dots, k'\}$ by

$$\text{height}(j) \coloneqq \max \left\{ h \in \{0, \dots, \Gamma\} : j \equiv 0 \pmod{2^h} \right\},$$

and

$$\text{width}(j) \coloneqq 2^{\text{height}(j)} - 1.$$

Our weight approximations are then defined as follows:

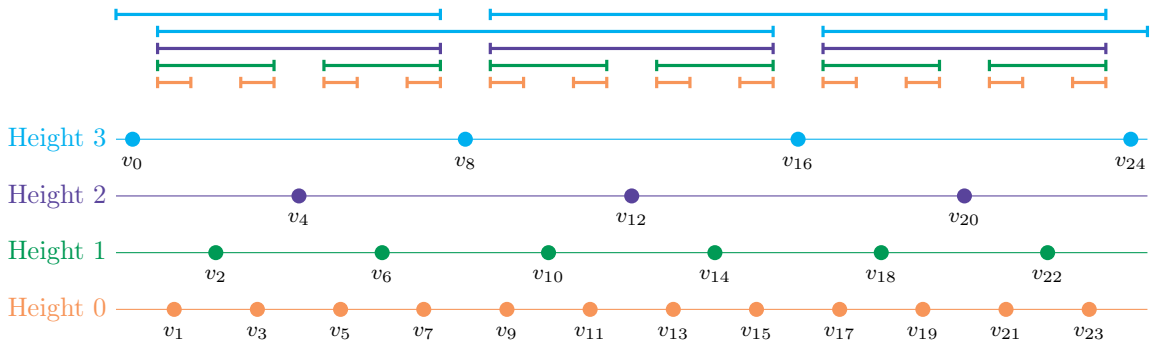


Figure 4.21: An illustration of the functions height and width for $\Gamma = 3$ and $k' = 7$. Each vertex v_i is vertically positioned according to height(i). The intervals shown above indicate for each vertex v_i the set $\{v_j \in V(P) : j = i - \text{width}(i), \dots, i + \text{width}(i)\}$.

Definition 4.65 (Weight-bound)**.** *Let P be a path of length r visiting the vertices $v_0, \dots, v_r$ in this order. For a given $q \in \{0, \dots, k'\}$, we define the* weight-bound *of P as*

$$w^q(P) := k^2 \cdot D_{\ell^*} + \sum_{i=0}^{r} 2^{\text{height}(i+q)} \cdot \sum_{\Delta=1}^{\text{width}(i+q)} \big(c(v_{i-\Delta}, v_i) + c(v_i, v_{i+\Delta})\big),$$

where any term involving a vertex v_z with $z < 0$ or $z > r$ is interpreted as zero.

Now property (i) follows directly from the triangle inequality satisfied by the cost function c:

Lemma 4.66. *Let P be a path of length r visiting the vertices $v_0, \dots, v_r$ in this order. Then, for all $q \in \{0, \dots, k'\}$, we have*

$$w(P) \le w^q(P). \tag{4.37}$$

Proof. Fix an arbitrary $q \in \{0, \dots, k'\}$. First, observe that $k \le k'$ implies $c^{(k)}(P) \le c^{(k')}(P)$. Hence, it suffices to show $w^q(P) \ge k^2 \cdot D_{\ell^*} + c^{(k')}(P)$.

Let $i < j$ such that the edge (v_i, v_j) contributes to $c^{(k')}(P)$, that is, $j - i \le k'$. We choose an index $i^* \in \{i, \dots, j\}$ maximizing height$(i^* + q)$. By the triangle inequality, $c(v_i, v_j) \le c(v_i, v_{i^*}) + c(v_{i^*}, v_j)$. We will show that summing this upper bound over all edges (v_i, v_j) contributing to $c^{(k')}(P)$ is at most $w^q(P) - k^2 \cdot D_{\ell^*}$.

First, we show $i^* - i \le \text{width}(i^* + q)$ and $j - i^* \le \text{width}(i^* + q)$, which implies that the costs $c(v_i, v_{i^*})$ and $c(v_{i^*}, v_j)$ contribute to the weight bound $w^q(P)$. If height$(i^* + q) = \Gamma$, then we have

$$i^* - \text{width}(i^* + q) \le j - k' \le i.$$

Otherwise, we have height$(i^* + q) < \Gamma$. Let $h := \text{height}(i^* + q)$. Then by the definition of height, we have $(i^* + q) \equiv 0 \pmod{2^h}$ and we have $(i^* + q) \not\equiv 0 \pmod{2^{h+1}}$. This implies

$$(i^* + q) - 2^h \equiv 0 \pmod{2^{h+1}}.$$

In particular, this implies height$(i^* + q - 2^h) > h = \text{height}(i^* + q)$, where we used $h < \Gamma$. Therefore, by our choice of the index i^*, we must have $i^* - 2^h < i$. We conclude $i^* - i < 2^h$ and thus $i^* - i \le 2^h - 1 = \text{width}(i^* + q)$.

Hence, in both cases, we have $i^* - i \leq \text{width}(i^* + q)$. The proof of $j - i^* \leq \text{width}(i^* + q)$ is analogous. This shows that the costs $c(v_i, v_{i^*})$ and $c(v_{i^*}, v_j)$ contribute to the weight bound $w^q(P)$. It remains to show that the edges contributing to $w^q(P)$ also contribute with a sufficient multiplicity to account for all edges (v_i, v_j) contributing to $c^{(k')}(P)$.

Finally, for each vertex v_{i^*} and each vertex v_i with $i^* - i \leq \text{width}(i^* + q)$, we have at most $\text{width}(i^* + q) + 1$ indices $j \geq i^*$ such that i^* satisfies $i^* \in \{i, \dots, j\}$ and maximizes $\text{height}(i^* + q)$. (This is the case because we have shown that any such vertex j satisfies $j - i^* \leq \text{width}(i^* + q)$.) Note that the cost $c(v_i, v_{i^*})$ indeed contributes with a factor of $2^{\text{height}(i^*+q)} = \text{width}(i^* + q) + 1$. An analogous argument applies to the cost of the edge (v_{i^*}, v_j). □

To prove property (ii) of our weight bound, we use the following observation:

Lemma 4.67. *For every path P we have $c^{(k')}(P) \leq 4 \cdot c^{(k)}(P)$.*

Proof. Let $v_0, \dots, v_r$ be the vertices visited by P in this order. We upper bound the length of every edge (v_i, v_j) contributing to $c^{(k')}(P)$ by $c(v_i, v_j) \leq c(v_i, v_m) + c(v_m, v_j)$ where $m = \lfloor \frac{i+j}{2} \rfloor$. Observe that $m - i \leq \lfloor \frac{k'}{2} \rfloor \leq k$ and $j - m \leq \lceil \frac{k'}{2} \rceil \leq k$ because $j - i \leq k'$ and $k' \leq 2k$.

For all $i, m \in \{0, \dots, r\}$ there are at most two numbers $j \in \{0, 1, \dots, r\}$ with $m = \lfloor \frac{i+j}{2} \rfloor$. Similarly, for all $m, j \in \{0, \dots, r\}$ there are at most two numbers $i \in \{0, 1, \dots, r\}$ with $m = \lfloor \frac{i+j}{2} \rfloor$. □

Now property (ii) follows from a simple averaging argument:

Lemma 4.68. *Let P be a monotone path of length r visiting the vertices $v_0, \dots, v_r$ in this order. Then, there exists an offset $q \in \{0, \dots, k'\}$ such that*

$$4(\Gamma + 2) \cdot w(P) \geq w^q(P).$$

Proof. We choose $q \in \{0, \dots, k'\}$ uniformly at random and prove $\mathbb{E}\big[w^q(P)\big] \leq 4(\Gamma + 2) \cdot w(P)$. Fix an arbitrary index $j \in \{0, \dots, r\}$. Consider the contribution of v_j to $w^q(P)$, which is

$$2^{\text{height}(j+q)} \sum_{\Delta=1}^{\text{width}(j+q)} \Big(c(v_{j-\Delta}, v_j) + c(v_j, v_{j+\Delta}) \Big), \tag{4.38}$$

where we again follow the convention that edge costs involving undefined vertices (i.e. vertices v_z with $z < 0$ or $z > r$) are zero. Recall that $k' = 2^\Gamma - 1$, and therefore the definition of the function height implies

$$\mathbb{P}\big[\,\text{height}(j + q) = z\big] = \begin{cases} \frac{1}{2^{z+1}} & \text{if } z \in \{0, \dots, \Gamma - 1\}, \\ \frac{1}{2^\Gamma} & \text{if } z = \Gamma. \end{cases}$$

In particular, we obtain

$$\mathbb{E}\Big[2^{\text{height}(j+q)}\Big] \;\leq\; \frac{2^\Gamma}{2^\Gamma} + \sum_{z=0}^{\Gamma-1} \frac{2^z}{2^{z+1}} \;=\; \frac{\Gamma}{2} + 1.$$

Using $\text{width}(j + q) \leq k'$, we can bound the expected value of the contribution (4.38) of v_j to $w^q(P)$ by

$$\mathbb{E}\Big[2^{\text{height}(j+q)} \cdot \sum_{\Delta=1}^{k'} \big(c(v_{j-\Delta}, v_j) + c(v_j, v_{j+\Delta})\big)\Big] \;\leq\; (\tfrac{\Gamma}{2} + 1) \cdot \sum_{\ell=1}^{k'} \big(c(v_{j-\ell}, v_j) + c(v_j, v_{j+\ell})\big).$$

Because $2c^{(k')}(P) = \sum_{j=0}^{r} \sum_{\Delta=1}^{k'} \big(c(v_{j-\Delta}, v_j) + c(v_j, v_{j+\Delta})\big)$, we conclude that

$$\begin{aligned}\mathbb{E}\big[w^q(P)\big] &\leq k^2 D_{\ell^*} + (\Gamma+2) \cdot c^{(k')}(P) \\ &\leq k^2 D_{\ell^*} + 4(\Gamma+2) \cdot c^{(k)}(P) \\ &\leq 4(\Gamma+2) \cdot w(P),\end{aligned}$$

where we used Lemma 4.67 for the second inequality. □

To prove that we can find a monotone path P with $V(P) \subseteq H^*$ and an offset $q \in \{0, \dots, k'\}$ minimizing $\frac{w^q(P)}{\gamma(P)}$ in quasi-polynomial time, we will use a dynamic programming approach and we will no longer assume that the cost function c satisfies the triangle inequality. This allows us to argue that we may assume that the optimal path P^* and an optimal offset q^* satisfy $q^* = 0$ and

$$|V(P^*)| \equiv 1 \pmod{k'+1}. \tag{4.39}$$

These assumptions will be helpful to simplify notation and avoid special cases to consider in our algorithm.

To see that we can indeed assume (4.39) and $q^* = 0$, observe that in case these assumptions are not satisfied, we can add dummy vertices to our instance as follows. We add $2k'$ dummy vertices to H^*, out of which k' are located left of all other vertices in H^* and the other k' vertices are located right of all other vertices of H^*. We extend the hierarchical partition $\mathcal{H}$ to include these dummy vertices. (This can be done in any arbitrary way as long as the dummy vertices belong to H^* and satisfy the above-mentioned left/right-relation to the existing vertices in H^*.) For each dummy vertex, the cost of every incident forward edge is zero. The cost of incident backward edges does not matter, we could set it e.g. to D_ℓ for backward edges on level ℓ. Moreover, for each dummy vertex v, all elements (v, ℓ) of the Set Cover universe are already covered, i.e., none of them is contained in U.

Then including dummy vertices in a monotone path P (while maintaining monotonicity) does not change the number $\gamma(P)$ of newly covered elements of the Set Cover universe. Moreover, including a dummy vertex at the end of the path P does not change the weight bound $w^q(P)$. If a path P' arises from inserting a dummy vertex at the beginning of a path P (with $V(P) \subseteq H^*$), then we have $w^q(P') = w^{q+1}(P)$. Thus, on the instance with the additional dummy vertices, there exists an optimal path P^* and an optimal offset q^* satisfying $q^* = 0$ and (4.39). From an optimal path P^* for the instance with dummy vertices, we can obtain an optimal path P and an optimal offset q by omitting all dummy vertices from P^* and setting q to the number of dummy vertices contained in P^* that are left of the original vertices of H^*.

Thus, to prove that we can find a monotone path P with $V(P) \subseteq H^*$ and an offset $q \in \{0, \dots, k'\}$ minimizing $\frac{w^q(P)}{\gamma(P)}$ in quasi-polynomial time, it suffices to prove the following lemma:

Lemma 4.69. *In time $n^{\mathcal{O}(\log n)}$ we can compute a monotone path P that minimizes $w^0(P)$ among all monotone paths satisfying the following constraints:*

(i) $V(P) \subseteq H^*$,

(ii) $|V(P)| \equiv 1 \pmod{k'+1}$, *and*

(iii) $\gamma(P) = \gamma^*$.

This applies even if the cost function c does not satisfy the triangle inequality.

From Lemma 4.69 we can conclude the main result of this section, Theorem 4.64. As argued in Section 4.8.1, it suffices to obtain an $\mathcal{O}(\log n)$-approximation for the problem of finding a

monotone path P with $V(P) \subseteq H^*$ that minimizes $\frac{w(P)}{\gamma(P)}$ in time $n^{\mathcal{O}(\log n)}$. Let P^* denote an optimal solution for this problem. By Lemma 4.68, there exists an offset q^* such that $\frac{w^{q^*}(P^*)}{\gamma(P^*)} \leq 4(\Gamma+2) \cdot \frac{w(P^*)}{\gamma(P^*)}$. As discussed above, Lemma 4.69 (applied to the instance with dummy vertices) implies that we can find a monotone path P with $V(P) \subseteq H^*$ and an offset q minimizing $\frac{w^q(P)}{\gamma(P)}$. In particular, we have

$$\frac{w^q(P)}{\gamma(P)} \;\leq\; \frac{w^{q^*}(P^*)}{\gamma(P^*)} \;\leq\; 4(\Gamma+2) \cdot \frac{w(P^*)}{\gamma(P^*)},$$

which by Lemma 4.66 implies $\frac{w(P)}{\gamma(P)} \leq 4(\Gamma+2) \cdot \frac{w(P^*)}{\gamma(P^*)} = \mathcal{O}(\log n) \cdot \frac{w(P^*)}{\gamma(P^*)}$, as desired.

We conclude that in order to complete the proof of Theorem 4.64, it remains to prove Lemma 4.69.

4.8.3 Decomposition into blocks

To prove Lemma 4.69, we will first describe how we can decompose every feasible path P into several independent *blocks*. Our algorithm will then precompute possible blocks of an optimal solution, and we will show how to construct an optimal path by combining precomputed blocks in an optimal way.

First, we explain more precisely what we mean by a *block*. Given a monotone path P that satisfies constraint (ii) of Lemma 4.69, we can decompose its vertex set $V(P) = \{x_1, \dots, x_r\} \cup \bigcup_{i=1}^{r-1} A_i$ with

$$x_1 \prec A_1 \prec x_2 \prec A_2 \prec \cdots \prec x_{r-1} \prec A_{r-1} \prec x_r, \tag{4.40}$$

and $|A_i| = k' = 2^\Gamma - 1$ for all $i \in [r-1]$. We refer to the sets A_i as *blocks*. Observe that the vertices $x_1, \dots, x_r$ are exactly those vertices assigned to height Γ in the definition of the weight-bound w^0.

The following lemma shows that if we aim at minimizing $w^0(P)$ and have already fixed the vertices $x_1, \dots, x_r$, the blocks A_i can be chosen independently:

Lemma 4.70. *Let P be a monotone path visiting the vertices $v_0, \dots, v_{r \cdot 2^\Gamma}$ in this order, for some $r \geq 1$. For each $i = 0, \dots, r-1$, consider the subpath Q_i of P visiting $v_{i \cdot 2^\Gamma}, \dots, v_{(i+1) \cdot 2^\Gamma}$ in this order. Then we have*

$$w^0(P) - k^2 \cdot D_{\ell^*} = \sum_{i=0}^{r-1} \Big(w^0(Q_i) - k^2 \cdot D_{\ell^*} \Big). \tag{4.41}$$

Proof. Recall that the contribution of v_j to $w^0(P)$ is

$$2^{\text{height}(j)} \sum_{\Delta=1}^{\text{width}(j)} \Big(c(v_{j-\Delta}, v_j) + c(v_j, v_{j+\Delta}) \Big).$$

We have $\text{height}(j) = \text{height}(j - i \cdot 2^\Gamma)$ for any $i \in \{0, \dots, r-1\}$. Thus, the contribution of any v_j with $j \in \big\{ i \cdot 2^\Gamma : i \in \{0, \dots, r\} \big\}$ is counted equally on both sides of (4.41). Therefore, it suffices to show that

$$\{ j - \text{width}(j), \dots, j + \text{width}(j) \} \subseteq \big\{ i \cdot 2^\Gamma, \dots, (i+1) \cdot 2^\Gamma \big\}$$

for all $i \in \{0, \dots, r-1\}$ and $j \in \big\{ i \cdot 2^\Gamma + 1, \dots, (i+1) \cdot 2^\Gamma + 1 \big\}$.

Fix such indices i and j. For any $\ell \in \big\{ 1, \dots, 2^{\text{height}(j)} - 1 \big\}$ we have $\text{height}(j+\ell) < \text{height}(j) < \Gamma$ by the definition of height. This implies that $\{ j, \dots, j + \text{width}(j) \} \subseteq \big\{ i \cdot 2^\Gamma, \dots, (i+1) \cdot 2^\Gamma \big\}$.

Using an analogous argument, one shows that $\{j, \dots, j - \text{width}(j)\} \subseteq \left\{i \cdot 2^\Gamma, \dots, (i+1) \cdot 2^\Gamma\right\}$. This completes the proof. □

If we do not simply aim at minimizing $w^0(P)$, but also want P to satisfy the covering constraint (iii) from Lemma 4.69, knowing the vertices $x_1, \dots, x_r$ is not sufficient to make the choice of the different blocks independent. We will need additional information, which is captured by the notion of a *covering profile*.

The covering profile realized by a path P should contain enough information to determine $\gamma(P)$. Moreover, if we fix the covering profiles realized by two monotone paths Q_1 and Q_2 and we concatenate them to a new monotone path P, then the covering profile realized by P should be completely determined by the covering profiles for Q_1 and Q_2. This will allow us to optimize the paths Q_i for the different blocks (as in Lemma 4.70) independently once we fixed the vertices x_i and x_{i+1}, as well as the covering profile realized by Q_i.

In order to define covering profiles, we introduce the following notation:

Definition 4.71 (Open and closed intervals)**.** *For $v \preceq w \in V$ we define by*

$$(v, w)_\prec \coloneqq \{x \in V : v \prec x \prec w\}, \quad \text{and} \quad [v, w]_\prec \coloneqq \{x \in V : v \preceq x \preceq w\}$$

the open *and* closed interval *between v and w, respectively.*

Recall that $H_\ell(v)$ denotes the unique set $H \in \mathcal{H}_\ell$ containing the vertex v. Now we define covering profiles as follows:

Definition 4.72 (Covering profile)**.** *A* covering profile Φ *is a tuple $(a_{\min}, a_{\max}, f, g, \gamma)$, where*

(i) $a_{\min} \preceq a_{\max} \in H^*$ *are (not necessarily distinct) vertices,*

(ii) $f\colon \{\ell^*, \dots, L-1\} \times \{\min, \max\} \to \{0, \dots, n\}$ *is a function,*

(iii) $g\colon \{\ell^*, \dots, L-1\} \times \{\min, \max\} \to \{0, \dots, n\}$ *is a function, and*

(iv) $\gamma \in \{0, \dots, nL\}$ *is an integer.*

Given a monotone path P with $V(P) \subseteq H^*$, we say that P realizes *Φ if the following conditions are satisfied:*

(i) $\{a_{\min}, a_{\max}\} \subseteq V(P) \subseteq [a_{\min}, a_{\max}]_\prec$*, that is, $a_{\min}$ and $a_{\max}$ are the leftmost and rightmost vertices of P, respectively,*

(ii) $f(\ell, m) = |V(P) \cap H_{\ell+1}(a_m)|$ *for all $\ell \in \{\ell^*, \dots, L-1\}$ and $m \in \{\min, \max\}$, i.e., $f(\ell, \min)$ and $f(\ell, \max)$ are the number of vertices of P in the leftmost and rightmost $H \in \mathcal{H}_{\ell+1}$ that P intersects, respectively,*

(iii) $g(\ell, m) = \left|\left(H_{\ell+1}(a_m) \cap [a_{\min}, a_{\max}]_\prec\right) \setminus (V(P) \cup S_\ell)\right|$ *for all $\ell \in \{\ell^*, \dots, L-1\}$ and $m \in \{\min, \max\}$, that is, $g(\ell, \min)$ is the number of not yet covered elements (v, ℓ) for which v is in the leftmost $H \in \mathcal{H}_{\ell+1}$ that P intersects, and v is not left of P, but v is not part of P (analogously $g(\ell, \max)$), and*

(iv) we have

$$\gamma = \sum_{\ell=\ell^*}^{L} |V(P) \setminus S_\ell| + \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1}, \\ H \subseteq (a_{\min}, a_{\max})_\prec, \\ |V(P) \cap H| \geq k}} |H \setminus (V(P) \cup S_\ell)|,$$

i.e., the number $\gamma(P)$ of elements newly covered by P equals γ.

We say that Φ is realizable *if there exists a monotone path P with $V(P) \subseteq H^*$ that realizes Φ.*

Observe that there are only quasi-polynomially many covering profiles. Moreover, note that for any monotone path P with $V(P) \subseteq H^*$ there exists a unique covering profile Φ that is realized by P. For any vertex $v \in H^*$, we write $\Phi(v)$ for the unique covering profile of the singleton-path consisting only of v.

We say that a covering profile $\Phi = (a_{\min}, a_{\max}, f, g, \gamma)$ is *left/right* of a vertex v if the interval $[a_{\min}, a_{\max}]_\prec$ is left/right of v. Given another covering profile $\Phi' = (a'_{\min}, a'_{\max}, f', g', \gamma')$, we say that Φ is *left/right* of Φ' if the interval $[a_{\min}, a_{\max}]_\prec$ is left/right of the interval $[a'_{\min}, a_{\max}]'_\prec$.

The following two lemmas state that the covering profile indeed has the desired properties mentioned above. First, the covering profile realized by a path P contains enough information to recover the number of newly covered elements (v, ℓ) by (P, ℓ^*):

Lemma 4.73. *Let Φ be a covering profile. Then there is a number $\gamma(\Phi) \in \{0, \dots, nL\}$ such that $\gamma(\Phi) = \gamma(P)$ for every realization P of Φ.*

Moreover, given Φ, we can compute $\gamma(\Phi)$ in polynomial time.

Second, from two covering profiles Φ and Ψ realized by two monotone paths P_Φ and P_Ψ with $V(P_\phi)$ left of $V(P_\Psi)$, we can determine the covering profile of the monotone path P obtained by concatenating P_Φ and P_Ψ:

Lemma 4.74. *Let Φ and Ψ be two covering profiles such that Φ is left of Ψ. Then there exists a covering profile, denoted by $\Phi \oplus \Psi$, such that for every realization P_Φ of Φ and every realization P_Ψ of Ψ, the monotone path obtained by concatenating P_Φ and P_Ψ realizes $\Phi \oplus \Psi$.*

Moreover, given Φ and Ψ, we can compute $\Phi \oplus \Psi$ in polynomial time.

The proofs of Lemmas 4.73 and 4.74 can be derived in a straightforward way from Definition 4.72 and the definition of what it means for a path-level pair (P, ℓ^*) to cover an element of U. However, the proofs are quite technical and we thus defer them to Section 4.8.5.

Having described the key properties of covering profiles, we can now explain how we use them in our algorithm. We construct a directed auxiliary graph $\widetilde{G}$ whose vertices are pairs (v, Φ), where $v \in H^*$ is a vertex in our original vertex set, and $\Phi = (a_{\min}, a_{\max}, f, g, \gamma)$ is a realizable covering profile with $a_{\max} = v$. The idea is that our desired path P, decomposed into blocks as in (4.40), will correspond to a path in the auxiliary graph $\widetilde{G}$ with vertices (x_i, Φ_i), where Φ_i is the covering profile realized by the x_1-x_i subpath of P. The edge between two vertices (x_i, Φ_i) and (x_{i+1}, Φ_{i+1}) will correspond to the block A_i.

We say that $(v, \Phi) \in V(\widetilde{G})$ is a *designated start vertex* if $\Phi = \Phi(v)$. Moreover, $(v, \Phi) \in V(\widetilde{G})$ is a *designated end vertex* if $\gamma(\Phi) = \gamma^*$ (see Lemma 4.73).

Next, we define the edges of the auxiliary graph $\widetilde{G}$. Recall that each block A_i contains exactly k' vertices. We need the following definition:

Definition 4.75 (Optimal realization)**.** *Let $x \prec y \in H^*$ be two distinct vertices and let Φ be a covering profile that is right of x and left of y. We say that Φ is* (x, y, k')-realizable *if there exists a realization Q of Φ with $|V(Q)| = k'$.*

An optimal (x, y, k')-realization *is a realization Q of Φ with $|V(Q)| = k'$ that minimizes $w^0(Q')$ among all such realizations, where Q' is the monotone path that visits x, then traverses Q, and finally visits y.*

To construct $E(\widetilde{G})$, we need to be able to compute optimal realizations in quasi-polynomial time.

Lemma 4.76. *Let $x \prec y \in H^*$ be two distinct vertices. Then there is an algorithm that computes, for every (x, y, k')-realizable covering profile Φ, an optimal (x, y, k')-realization in time $n^{\mathcal{O}(\log n)}$.*

We will prove Lemma 4.76 in Section 4.8.4 via a dynamic program. We fix an optimal (x, y, k')-realization of Φ for all $x, y \in H^*$ with $x \prec y$ and every (x, y, k')-realizable covering profile Φ. Because there are only $n^{\mathcal{O}(\log n)}$ many covering profiles, Lemma 4.76 implies that we can do this in time $n^{\mathcal{O}(\log n)}$.

Now, in order to construct the edge set of $\widetilde{G}$, we do the following for every pair of vertices $(x, \Phi_x), (y, \Phi_y) \in V(\widetilde{G})$ with $x \prec y$: For each (x, y, k')-realizable covering profile $\widehat{\Phi}$ that is right of x and left of y, we add an edge e from (x, Φ_x) to (y, Φ_y) to $\widetilde{G}$ if

$$\Phi_y = \left(\Phi_x \oplus \widehat{\Phi}\right) \oplus \Phi(y),$$

using the notation of Lemma 4.74. (Recall that $\Phi(y)$ is the covering profile realized by the singleton-path with vertex y.) Additionally, we write $P(e)$ to denote the fixed optimal (x, y, k')-realization of the covering profile $\widehat{\Phi}$. Then we define the weight of the edge e of $\widetilde{G}$ via

$$w(e) \coloneqq w^0(P') - k^2 \cdot D_{\ell^*},$$

where P' is the monotone path that visits x, then traverses $P(e)$, and finally visits y.

Every path in $\widetilde{G}$ naturally corresponds to a path in our original vertex set:

Definition 4.77 (Corresponding path)**.** *Let $\widetilde{Q}$ be a path in $\widetilde{G}$ that visits in this order the vertices $(x_1, \Phi_1), \dots, (x_r, \Phi_r) \in V(\widetilde{G})$. We denote its edges by $e_i \coloneqq ((x_i, \Phi_i), (x_{i+1}, \Phi_{i+1}))$ for $i \in [r-1]$.*

Then we define corresponding path *of $\widetilde{Q}$ as the monotone path that starts in x_1 and, for each $i = 1, \dots, r-1$, traverses $P(e_i)$ and then visits x_{i+1}.*

We will prove that we can find a monotone path P as desired in Lemma 4.69 by finding shortest-paths in the auxiliary graph $\widetilde{G}$ with edge weights w. To this end, we observe the following:

Lemma 4.78. *Let $\widetilde{Q}$ be path in $\widetilde{G}$ that starts in a designated start vertex and ends in a vertex $(y, \Phi) \in V(\widetilde{G})$, and let P be its corresponding path. Then, P realizes Φ and we have $w(\widetilde{Q}) + k^2 \cdot D_{\ell^*} = w^0(P)$.*

Moreover, for any monotone path P' satisfying conditions (i) to (iii) of Lemma 4.69, there exists a path Q' in $\widetilde{G}$ starting in a designated start vertex and ending in a designated end vertex such that $w(Q') + k^2 \cdot D_{\ell^*} \leq w^0(P')$.

Proof. It follows immediately from Lemma 4.74 and an induction on the number of edges of $\widetilde{Q}$ that P realizes Φ. Moreover, Lemma 4.70 implies that $w(\widetilde{Q}) + k^2 \cdot D_{\ell^*} = w^0(P)$.

Let P' be a monotone path that satisfies conditions (i) to (iii) of Lemma 4.69. In particular, we have $|V(P')| = 1 + r \cdot 2^\Gamma$ for some $r \geq 0$. Hence, we can decompose its vertex set into vertices $x_1, \dots, x_r$ and blocks $A_1, \dots, A_{r-1}$, as explained in (4.41). A straightforward induction on r using Lemmas 4.70 and 4.74 shows the existence of a path Q' in $\widetilde{G}$ such that $w(Q') + k^2 \cdot D_{\ell^*} \leq w^0(P')$. □

Lemma 4.78 immediately implies Lemma 4.69:

Proof of Lemma 4.69. We start by constructing the graph $\widetilde{G}$. There are at most $n^{\mathcal{O}(\log n)}$ covering profiles, because $L = \mathcal{O}(\log n)$ as our instance $\mathcal{I}$ has low depth. Hence, by Lemma 4.76, we can construct $\widetilde{G}$ in time $n^{\mathcal{O}(\log n)}$.

For every pair of designated start and end vertices $(x, \Phi_x), (y, \Phi_y) \in V(\widetilde{G})$, we compute a shortest path in $\widetilde{G}$ from (x, Φ_x) to (y, Φ_y) (if there exists one). In particular, $\gamma(\Phi_y) = \gamma^*$. We

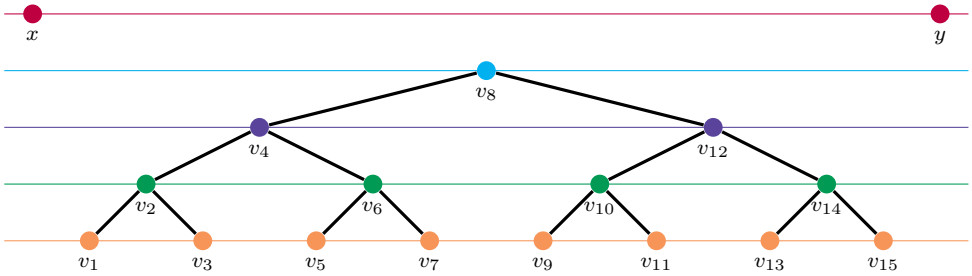


Figure 4.22: Illustration of the full binary tree $\mathcal{B}$ corresponding to an optimal (x, y, k')-realization (for $k' = 15$ and $\Gamma = 4$) that visits the vertices $v_1, \dots, v_{15}$ in this order. The color and vertical position of the vertices indicated in the picture correspond to the height assigned to the vertices in the definition of the weight bound function $w^0(Q')$, where Q' is the monotone path visiting $x, v_1, \dots, v_{15}, y$ in this order (see Definition 4.65).

denote by $\widetilde{Q}$ such a path in $\widetilde{G}$ that minimizes $w(\widetilde{Q})$ among all shortest paths we computed. Let P be its corresponding path. By our construction we have $V(P) \subseteq H^*$ and $|V(P)| \equiv 1 \pmod{k'+1}$. Moreover, Lemmas 4.73 and 4.78 yield that $\gamma(P) = \gamma(\Phi_y) = \gamma^*$.

Finally, Lemma 4.78 implies that

$$w^0(P) = w(\widetilde{Q}) + k^2 \cdot D_{\ell^*} \leq w^0(P')$$

for any monotone path P' satisfying conditions (i) to (iii) of Lemma 4.69. □

4.8.4 Solving a single block

We now show that given two distinct vertices $x, y \in H^*$ with $x \prec y$ and an (x, y, k')-realizable covering profile Φ, we can compute an optimal (x, y, k')-realization of Φ in quasi-polynomial time. That is, we prove Lemma 4.76.

To this end, we fix the vertices $x \prec y \in H^*$ for the remainder of this section. We briefly explain the idea of our dynamic programming approach. Since any (x, y, k')-realization is a monotone path on $k' = 2^\Gamma - 1$ vertices, we can view such a realization as a full binary tree $\mathcal{B}$ of height Γ, as illustrated in Figure 4.22.

Let $h(v)$ denote the height of v in the binary tree $\mathcal{B}$ (where leaves have height 1 and the root has height Γ, thus differing slightly from the notion of the height function as given in Definition 4.65), and let $\pi_v(h+1), \dots, \pi_v(\Gamma)$ denote the vertices on the path from v to the root of $\mathcal{B}$. The cost function we minimize (see Definition 4.65) can then be rewritten as

$$2^\Gamma \cdot \sum_{v \in V(\mathcal{B})} \Big(c(x, v) + c(v, y) \Big) + \sum_{v \in V(\mathcal{B})} \sum_{\ell = h(v)+1}^{\Gamma} 2^{\ell - 1} \cdot \min \Big(c(v, \pi_v(\ell)), c(\pi_v(\ell), v) \Big), \tag{4.42}$$

as we will show in Lemma 4.81.

The summands involving vertices in the subtree $\mathcal{B}_w$ rooted at a vertex w are

$$2^\Gamma \cdot \sum_{v \in V(\mathcal{B}_w)} \Big(c(x, v) + c(v, y) \Big) + \sum_{v \in V(\mathcal{B}_w)} \sum_{\ell = h(v)+1}^{\Gamma} 2^{\ell - 1} \cdot \min \Big(c(v, \pi_v(\ell)), c(\pi_v(\ell), v) \Big),$$

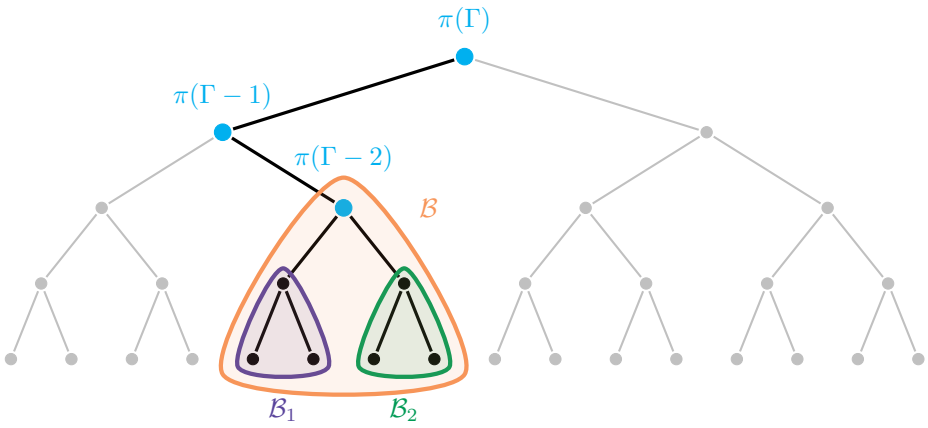


Figure 4.23: Illustration of how two compatible tree solutions $(\mathcal{B}_1, \pi)$ and $(\mathcal{B}_2, \pi)$ of height $h < \Gamma$ are combined. All vertices are drawn from left to right according to the total order $\prec$. The path $\pi(h+1), \dots, \pi(\Gamma)$ from the root of the current subtree to the global root $\pi(\Gamma)$ contains the vertices at which it will later be merged with another subtree.

and we will refer to this as the cost of the subtree rooted at w. We highlight that the cost of any subtree depends on the path between w and the root of the entire tree, but not on any other vertices outside of the subtree. Our dynamic program computes a cheapest subtree of height h for each

- height $h \in [\Gamma]$,
- path $\pi(h+1), \dots, \pi(\Gamma)$ from the root of the subtree to the root of the entire tree,
- covering profile Φ,

such that the monotone path on the vertices of this subtree realizes Φ. We then construct the entire tree bottom-up by merging such subtree solutions of height less than Γ that agree on both their remaining paths to the global root and their covering profiles (see Lemma 4.74).

In the following we make these ideas more precise.

Definition 4.79 (Tree solutions)**.** *Let $h \in [\Gamma]$. A* tree solution of height h *is a pair $(\mathcal{B}, \pi)$, where*

(i) $\mathcal{B}$ is a full binary tree with $V(\mathcal{B}) \subseteq (x, y)_\prec$ and $|V(\mathcal{B})| = 2^h - 1$, such that for every non-leaf $r \in V(\mathcal{B})$, we can label the two full binary subtrees rooted at the children of r as $\mathcal{B}_1$ and $\mathcal{B}_2$ so that $V(\mathcal{B}_1)$ is left of r and $V(\mathcal{B}_2)$ is right of r, and

(ii) $\pi\colon \{h+1, \dots, \Gamma\} \to (x, y)_\prec$ is a function such that $\pi(\ell)$ is left of $V(\mathcal{B})$ or right of $V(\mathcal{B})$ for every $\ell \in \{h+1, \dots, \Gamma\}$.

Given a covering profile Φ, we say that a tree solution $(\mathcal{B}, \pi)$ realizes *Φ if the unique monotone path P whose vertex set is $V(\mathcal{B})$ realizes Φ. Conversely, given a tree solution $(\mathcal{B}, \pi)$, we define its covering profile to be the covering profile of the unique monotone path P whose vertex set is $V(\mathcal{B})$.*

Next, we formalize how we can merge subtree solutions. See Figure 4.23 for an illustration.

Definition 4.80 (Compatible tree solutions and costs)**.** *Given two tree solutions $(\mathcal{B}_1, \pi_1)$ and $(\mathcal{B}_2, \pi_2)$ of height $h \in [\Gamma - 1]$, we say that they are* compatible *if $\pi_1 = \pi_2$ and $V(\mathcal{B}_1)$ is left of*

$\pi(h+1)$ and $V(\mathcal{B}_2)$ is right of $\pi(h+1)$, where $\pi \coloneqq \pi_1 = \pi_2$. We denote by $(\mathcal{B}_1, \pi_1) \oplus (\mathcal{B}_2, \pi_2)$ the tree solution $(\mathcal{B}, \pi\,|_{\{h+2,\dots,\Gamma\}})$ of height $h+1$, where $\mathcal{B}$ is the full binary tree with root $\pi(h)$ and subtrees $\mathcal{B}_1$ and $\mathcal{B}_2$. Note that any tree solution $(\mathcal{B}, \pi)$ of height $h \in \{2, \dots, \Gamma\}$ can be uniquely decomposed via

$$(\mathcal{B}, \pi) = (\mathcal{B}_1, \pi_1) \oplus (\mathcal{B}_2, \pi_2)$$

into two compatible tree solutions $(\mathcal{B}_1, \pi_1)$ and $(\mathcal{B}_2, \pi_2)$ of height $h-1$. Therefore, for $h > 1$, we can recursively define

$$\begin{aligned}\operatorname{cost}(\mathcal{B}, \pi) \coloneqq \operatorname{cost}(\mathcal{B}_1, \pi_1) + \operatorname{cost}(\mathcal{B}_2, \pi_2) + 2^\Gamma \cdot \Big(c(x, r) + c(r, y)\Big) \\ + \sum_{\ell=h+1}^{\Gamma} 2^{\ell-1} \cdot \min\Big(c(r, \pi(\ell)), c(\pi(\ell), r)\Big)\end{aligned}$$

where r is the root of $\mathcal{B}$. For $h = 1$, we replace the term $\operatorname{cost}(\mathcal{B}_1, \pi_1) + \operatorname{cost}(\mathcal{B}_2, \pi_2)$ by zero.

The following lemma formally states that (x, y, k')-realizations of covering profiles can indeed be regarded as tree solutions of height Γ. Moreover, in this case, our cost definition coincides with our objective function in Definition 4.75.

Lemma 4.81. *Let Φ be an (x, y, k')-realizable covering profile. Then for every (x, y, k')-realization P of Φ, there exists a tree solution $(\mathcal{B}, \pi)$ of height Γ with $V(\mathcal{B}) = V(P)$ and $\operatorname{cost}(\mathcal{B}, \pi) = w^0(P') - k^2 \cdot D_{\ell^*}$, where P' denotes the monotone path that starts in x, then traverses P, and ends in y.*

The proof of Lemma 4.81 follows in a straightforward way from the definitions, but it is rather technical. Thus, we defer it to Section 4.8.5 and first show how we use Lemma 4.81 to prove Lemma 4.76:

Proof of Lemma 4.76. We give a dynamic program. Our table entries are indexed by triples (h, Φ, π) where

- $h \in [\Gamma]$ is an integer,
- $\Phi = (a_{\min}, a_{\max}, f, g, \gamma)$ is a covering profile with $a_{\min}, a_{\max} \in (x, y)_\prec$, and
- $\pi\colon \{h+1, \dots, \Gamma\} \to (x, y)_\prec$ is a function.

For each such triple, we compute a tree solution $(\mathcal{B}, \pi)$ of height h that realizes Φ (if one exists), minimizing $\operatorname{cost}(\mathcal{B}, \pi)$ among all such tree solutions. Lemma 4.81 implies that this suffices to find an optimal (x, y, k')-realization of every (x, y, k')-realizable covering profile.

Tree solutions of height 1 are singletons. Thus, finding such an optimal tree solution for all labels (h, Φ, π) with $h = 1$ can be done by enumeration.

Suppose that $h \geq 2$ and assume that for every table index $(h-1, \Psi, \pi')$ we have already computed a tree solution $(\mathcal{B}, \pi)$ of height $h-1$ that realizes Φ and has minimum cost (if such a tree solution exists). Let (h, Φ, π) be an arbitrary but fixed table index and we show how to fill the corresponding entry.

We enumerate all vertices $r \in (x, y)_\prec$, and all covering profiles Ψ_1 and Ψ_2 that are right of x and left of y, such that Ψ_1 is left of r and Ψ_2 is right of r, and

$$\Phi = (\Psi_1 \oplus \Phi(r)) \oplus \Psi_2.$$

Given such r, Ψ_1, and Ψ_2, we define

$$\pi'(\ell) \coloneqq \begin{cases} \pi(\ell) & \text{if } \ell \in \{h+1, \dots, \Gamma\}, \\ r & \text{if } \ell = h. \end{cases}$$

Suppose that there exist tree solutions $(\mathcal{B}_1, \pi')$ and $(\mathcal{B}_2, \pi')$ of height $h-1$ realizing Ψ_1 and Ψ_2, respectively, and we consider such tree solutions of minimum cost. Note that $(\mathcal{B}_1, \pi')$ and $(\mathcal{B}_2, \pi')$ are compatible, and, by Lemma 4.74, $(\mathcal{B}, \pi') \coloneqq (\mathcal{B}_1, \pi') \oplus (\mathcal{B}_2, \pi')$ is a tree solution of height h that realizes Φ.

After enumerating all r, Ψ_1, and Ψ_2, we choose such a tree solution $(\mathcal{B}, \pi) \coloneqq (\mathcal{B}_1, \pi') \oplus (\mathcal{B}_2, \pi')$ of minimum cost for the table entry indexed by (h, Φ, π), if we have constructed at least one such tree solution $(\mathcal{B}, \pi')$.

We argue that this procedure is correct: Suppose that there exists a tree solution $(\mathcal{B}^*, \pi)$ of height h that realizes Φ. Then we can decompose $(\mathcal{B}^*, \pi) = (\mathcal{B}_1^*, \pi_1^*) \oplus (\mathcal{B}_2^*, \pi_2^*)$. Note that $\pi_1^*(h) = \pi_2^*(h) = r^*$, where r^* denotes the root of $\mathcal{B}^*$. Let Ψ_1^* and Ψ_2^* be the covering profiles of $(\mathcal{B}_1^*, \pi_1^*)$ and $(\mathcal{B}_2^*, \pi_2^*)$, respectively. By Lemma 4.74 we have $\Phi = (\Psi_1^* \oplus \Phi(r^*)) \oplus \Psi_2^*$. Hence, r^*, Ψ_1^*, and Ψ_2^* occurred in our enumeration, and we correctly find a realization. Moreover, by the recursive definition of $\mathrm{cost}(\mathcal{B}, \pi)$, the tree solution we find has minimum cost.

To bound the running time, note that there are at most $n^{\mathcal{O}(\log n)}$ covering profiles, since our fixed instance $\mathcal{I}$ of Path Covering has low depth and therefore $L = \mathcal{O}(\log n)$. Since $\Gamma = \mathcal{O}(\log k') = \mathcal{O}(\log n)$, our table contains at most $\Gamma \cdot n^{\mathcal{O}(\log n)} \cdot n^{\Gamma}$ entries (which is still $n^{\mathcal{O}(\log n)}$), and filling each entry takes additional time of at most $n^{\mathcal{O}(\log n)}$ by the procedure we described above. □

4.8.5 Remaining proofs

It remains to prove Lemmas 4.73, 4.74 and 4.81, which we restate here for convenience:

Lemma 4.73. *Let Φ be a covering profile. Then there is a number $\gamma(\Phi) \in \{0, \dots, nL\}$ such that $\gamma(\Phi) = \gamma(P)$ for every realization P of Φ.*

Moreover, given Φ, we can compute $\gamma(\Phi)$ in polynomial time.

Proof. Without loss of generality, assume that $\Phi = (a_{\min}, a_{\max}, f, g, \gamma)$ is realizable and fix an arbitrary realization P of Φ. By the definition of what it means for the path-level pair (P, ℓ^*) to cover an element of U, we have

$$\gamma(P) = \sum_{\ell=\ell^*}^{L} |V(P) \setminus S_\ell| + \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1}, \\ |V(P) \cap H| \geq k}} |H \setminus (V(P) \cup S_\ell)|.$$

Let $\ell \in \{\ell^*, \dots, L-1\}$. Then, because $V(P) \subseteq [a_{\min}, a_{\max}]_\prec$, for every $H \in \mathcal{H}_{\ell+1}$ with $|V(P) \cap H| \geq k$ exactly one of the following two is true:

- $H = H_{\ell+1}(a_{\min})$ or $H = H_{\ell+1}(a_{\max})$.
- $H \subseteq (a_{\min}, a_{\max})_\prec$.

Thus, for

$$I_{\neq} \coloneqq \{\ell \in \{\ell^*, \dots, L-1\} : H_{\ell+1}(a_{\min}) \neq H_{\ell+1}(a_{\max})\}, \text{ and}$$
$$I_{=} \coloneqq \{\ell \in \{\ell^*, \dots, L-1\} : H_{\ell+1}(a_{\min}) = H_{\ell+1}(a_{\max})\},$$

we have

$$\sum_{i=\ell^*}^{L-1} \sum_{\substack{H\in\mathcal{H}_{\ell+1},\\ |V(P)\cap H|\geq k}} |H\setminus(V(P)\cup S_\ell)| = \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H\in\mathcal{H}_{\ell+1},\\ H\subseteq(a_{\min},a_{\max})_\prec,\\ |V(P)\cap H|\geq k}} |H\setminus(V(P)\cup S_\ell)| + \sum_{\ell\in I_{\neq}} \sum_{\substack{m\in\{\min,\max\},\\ |V(P)\cap H_{\ell+1}(a_m)|\geq k}} |H_{\ell+1}(a_m)\setminus(V(P)\cup S_\ell)| + \sum_{\substack{\ell\in I_{=},\\ |V(P)\cap H_{\ell+1}(a_{\min})|\geq k}} |H_{\ell+1}(a_{\min})\setminus(V(P)\cup S_\ell)|.$$

Note that we have

$$H_{\ell+1}(a_m)\setminus V(P) = \Big(\big(H_{\ell+1}(a_m)\cap[a_{\min},a_{\max}]_\prec\big)\setminus V(P)\Big)\,\dot\cup\,\Big(H_{\ell+1}(a_m)\setminus[a_{\min},a_{\max}]_\prec\Big)$$

for every $m\in\{\min,\max\}$ and $\ell\in\{\ell^*,\dots,L-1\}$. Hence, combining this with

$$\begin{aligned} f(\ell,m) &= |V(P)\cap H_{\ell+1}(a_m)|,\\ g(\ell,m) &= |\big(H_{\ell+1}(a_m)\cap[a_{\min},a_{\max}]_\prec\big)\setminus(V(P)\cup S_\ell)|, \quad \text{and}\\ \gamma &= \sum_{\ell=\ell^*}^{L} |V(P)\setminus S_\ell| + \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H\in\mathcal{H}_{\ell+1},\\ H\subseteq(a_{\min},a_{\max})_\prec,\\ |V(P)\cap H|\geq k}} |H\setminus(V(P)\cup S_\ell)|, \end{aligned}$$

we obtain that

$$\begin{aligned} \gamma(P) &= \sum_{\ell=\ell^*}^{L} |V(P)\setminus S_\ell| + \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H\in\mathcal{H}_{\ell+1},\\ |V(P)\cap H|\geq k}} |H\setminus(V(P)\cup S_\ell)|\\ &= \gamma + \sum_{\ell\in I_{\neq}} \sum_{\substack{m\in\{\min,\max\},\\ f(\ell,m)\geq k}} \Big(g(\ell,m) + \big|H_{\ell+1}(a_m)\setminus([a_{\min},a_{\max}]_\prec\cup S_\ell)\big|\Big)\\ &\quad + \sum_{\substack{\ell\in I_{=},\\ f(\ell,\min)\geq k}} \Big(g(\ell,\min) + \big|H_{\ell+1}(a_{\min})\setminus([a_{\min},a_{\max}]_\prec\cup S_\ell)\big|\Big). \end{aligned}$$

Observe that the right-hand side depends only on Φ and not on P. Thus, choosing $\gamma(\Phi)$ as this value completes the proof. □

Lemma 4.74 follows from similar calculations using the definition of covering profiles:

Lemma 4.74. *Let Φ and Ψ be two covering profiles such that Φ is left of Ψ. Then there exists a covering profile, denoted by $\Phi\oplus\Psi$, such that for every realization P_Φ of Φ and every realization P_Ψ of Ψ, the monotone path obtained by concatenating P_Φ and P_Ψ realizes $\Phi\oplus\Psi$.*

Moreover, given Φ and Ψ, we can compute $\Phi\oplus\Psi$ in polynomial time.

Proof. Without loss of generality, assume that both covering profiles $\Phi=\big(a^\Phi_{\min},a^\Phi_{\max},f^\Phi,g^\Phi,\gamma^\Phi\big)$ and $\Psi=\big(a^\Psi_{\min},a^\Psi_{\max},f^\Psi,g^\Psi,\gamma^\Psi\big)$ are realizable and fix two arbitrary realizations P_Φ and P_Ψ,

respectively. We denote by P the concatenation of both paths. In particular, $V(P) = V(P_\Phi) \dot\cup V(P_\Psi)$.

We will construct each component of

$$\Phi \oplus \Psi \coloneqq \left(a^\oplus_{\min}, a^\oplus_{\max}, f^\oplus, g^\oplus, \gamma^\oplus\right)$$

individually.

First, we set $a^\oplus_{\min} \coloneqq a^\Phi_{\min}$ and $a^\oplus_{\max} \coloneqq a^\Psi_{\max}$. We have

$$\left\{a^\oplus_{\min}, a^\oplus_{\max}\right\} \subseteq V(P_\Phi) \cup V(P_\Psi) \subseteq \left[a^\oplus_{\min}, a^\oplus_{\max}\right]_\prec,$$

since $a^\Phi_{\max} \prec a^\Psi_{\min}$ and $a^\Phi_{\min} \in V(P_\Phi) \subseteq \left[a^\Phi_{\min}, a^\Phi_{\max}\right]_\prec$ and $a^\Psi_{\max} \in V(P_\Psi) \subseteq \left[a^\Psi_{\min}, a^\Psi_{\max}\right]_\prec$.

Next, let $\ell \in \{\ell^*, \dots, L-1\}$. We have either $H_{\ell+1}\left(a^\Phi_{\min}\right) = H_{\ell+1}\left(a^\Psi_{\min}\right)$, or $H_{\ell+1}\left(a^\Phi_{\min}\right) \prec H_{\ell+1}\left(a^\Psi_{\min}\right)$. In the former case, since $V(P_\Phi)$ and $V(P_\Psi)$ are disjoint, we have

$$\begin{aligned}\left|H_{\ell+1}\left(a^\oplus_{\min}\right) \cap (V(P_\Phi) \cup V(P_\Psi))\right| &= \left|H_{\ell+1}\left(a^\Phi_{\min}\right) \cap V(P_\Phi)\right| + \left|H_{\ell+1}\left(a^\Psi_{\min}\right) \cap V(P_\Psi)\right| \\ &= f^\Phi(\ell, \min) + f^\Psi(\ell, \min).\end{aligned}$$

In the latter case, we have

$$\begin{aligned}\left|\mathcal{H}_{\ell+1}\left(a^\oplus_{\min}\right) \cap (V(P_\Phi) \cup V(P_\Psi))\right| &= \left|H_{\ell+1}\left(a^\Phi_{\min}\right) \cap V(P_\Phi)\right| + \left|H_{\ell+1}\left(a^\Phi_{\min}\right) \cap V(P_\Psi)\right| \\ &= f^\Phi(\ell, \min).\end{aligned}$$

Hence, we can choose

$$f^\oplus(\ell, \min) \coloneqq \begin{cases} f^\Phi(\ell, \min) + f^\Psi(\ell, \min) & \text{if } H_{\ell+1}\left(a^\Phi_{\min}\right) = H_{\ell+1}\left(a^\Psi_{\min}\right), \\ f^\Phi(\ell, \min) & \text{if } H_{\ell+1}\left(a^\Phi_{\min}\right) \prec H_{\ell+1}\left(a^\Psi_{\min}\right). \end{cases}$$

Observe that the right-hand side depends only on Φ and Ψ, and not on P_Φ or P_Ψ. We can define $f^\oplus(\ell, \max)$ analogously.

We proceed by defining $g^\oplus(\ell, \min)$ for $\ell \in \{\ell^*, \dots, L-1\}$. We follow the same case distinction as before. Again, suppose that $H_{\ell+1}\left(a^\Phi_{\min}\right) = H_{\ell+1}\left(a^\Psi_{\min}\right)$. Then,

$$\begin{aligned}&\left|\left(H_{\ell+1}\left(a^\oplus_{\min}\right) \cap \left[a^\oplus_{\min}, a^\oplus_{\max}\right]_\prec\right) \setminus \left(V(P_\Phi) \cup V(P_\Psi) \cup S_\ell\right)\right| \\ &= \left|\left(H_{\ell+1}\left(a^\Phi_{\min}\right) \cap \left[a^\Phi_{\min}, a^\Phi_{\max}\right]_\prec\right) \setminus \left(V(P_\Phi) \cup S_\ell\right)\right| \\ &\quad + \left|\left(H_{\ell+1}\left(a^\Phi_{\min}\right) \cap \left(a^\Phi_{\max}, a^\Psi_{\min}\right)_\prec\right) \setminus S_\ell\right| \\ &\quad + \left|\left(H_{\ell+1}\left(a^\Psi_{\min}\right) \cap \left[a^\Psi_{\min}, a^\Psi_{\max}\right]_\prec\right) \setminus \left(V(P_\Psi) \cup S_\ell\right)\right| \\ &= g^\Phi(\ell, \min) + g^\Psi(\ell, \min) + \left|\left(H_{\ell+1}\left(a^\Phi_{\min}\right) \cap \left(a^\Phi_{\max}, a^\Psi_{\min}\right)_\prec\right) \setminus S_\ell\right|.\end{aligned}$$

The right-hand side depends only Φ and Ψ, and not on P_Φ or P_Ψ. Thus, in this case, we can define $g^\oplus(\ell, \min)$ to be this value. If we have $H_{\ell+1}\left(a^\Phi_{\min}\right) \prec H_{\ell+1}\left(a^\Psi_{\min}\right)$, then

$$H_{\ell+1}\left(a^\Phi_{\min}\right) \cap \left[a^\Psi_{\min}, a^\Psi_{\max}\right]_\prec = \emptyset,$$

and this case can be treated similarly. Moreover, we can define $g^\oplus(\ell, \max)$ analogously.

Finally, we show how to define $\gamma^{\oplus}$. To this end, consider the index sets

$$I_{\neq} \coloneqq \left\{\ell \in \{\ell^*, \dots, L-1\} : H_{\ell+1}\left(a^{\Phi}_{\max}\right) \neq H_{\ell+1}\left(a^{\Psi}_{\min}\right)\right\}, \quad \text{and}$$
$$I_{=} \coloneqq \left\{\ell \in \{\ell^*, \dots, L-1\} : H_{\ell+1}\left(a^{\Phi}_{\max}\right) = H_{\ell+1}\left(a^{\Psi}_{\min}\right)\right\}.$$

Additionally, we define

$$\widetilde{I}^{\Phi}_{\neq} \coloneqq \left\{\ell \in I_{\neq} : H_{\ell+1}\left(a^{\Phi}_{\max}\right) \subseteq \left(a^{\oplus}_{\min}, a^{\oplus}_{\max}\right)_{\prec}, \left|V(P_{\Phi}) \cap H_{\ell+1}\left(a^{\Phi}_{\max}\right)\right| \geq k\right\},$$
$$\widetilde{I}^{\Psi}_{\neq} \coloneqq \left\{\ell \in I_{\neq} : H_{\ell+1}\left(a^{\Psi}_{\min}\right) \subseteq \left(a^{\oplus}_{\min}, a^{\oplus}_{\max}\right)_{\prec}, \left|V(P_{\Psi}) \cap H_{\ell+1}\left(a^{\Psi}_{\min}\right)\right| \geq k\right\}, \quad \text{and}$$
$$\widetilde{I}_{=} \coloneqq \left\{\ell \in I_{=} : H_{\ell+1}\left(a^{\Phi}_{\max}\right) \subseteq \left(a^{\oplus}_{\min}, a^{\oplus}_{\max}\right)_{\prec}, \left|(V(P_{\Phi}) \cup V(P_{\Psi})) \cap H_{\ell+1}\left(a^{\Phi}_{\max}\right)\right| \geq k\right\}.$$

Note that all of the index sets above can be constructed from Φ and Ψ alone, since, for example,

$$\left|(V(P_{\Phi}) \cup V(P_{\Psi})) \cap H_{\ell+1}\left(a^{\Phi}_{\max}\right)\right| = f^{\Phi}(\ell, \max) + f^{\Psi}(\ell, \min)$$

for all $\ell \in I_{=}$.

For any $\ell \in \{\ell^*, \dots, L-1\}$ and $H \in \mathcal{H}_{\ell+1}$ with $H \subseteq \left(a^{\oplus}_{\min}, a^{\oplus}_{\max}\right)_{\prec}$ and $|(V(P_{\Phi}) \cup V(P_{\Psi})) \cap H| \geq k$ exactly one of the following two is true:

- $H \subseteq \left(a^{\Phi}_{\min}, a^{\Phi}_{\max}\right)_{\prec}$ or $H \subseteq \left(a^{\Psi}_{\min}, a^{\Psi}_{\max}\right)_{\prec}$.
- $H = H_{\ell+1}\left(a^{\Phi}_{\max}\right)$ or $H = H_{\ell+1}\left(a^{\Psi}_{\min}\right)$.

This implies

$$\begin{aligned}
&\sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1},\\ H \subseteq \left(a^{\oplus}_{\min}, a^{\oplus}_{\max}\right)_{\prec},\\ |V(P) \cap H| \geq k}} |H \setminus (V(P) \cup S_{\ell})| \\
&= \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1},\\ H \subseteq \left(a^{\Phi}_{\min}, a^{\Phi}_{\max}\right)_{\prec},\\ |V(P_{\Phi}) \cap H| \geq k}} |H \setminus (V(P_{\Phi}) \cup S_{\ell})| + \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1},\\ H \subseteq \left(a^{\Psi}_{\min}, a^{\Psi}_{\max}\right)_{\prec},\\ |V(P_{\Psi}) \cap H| \geq k}} |H \setminus (V(P_{\Psi}) \cup S_{\ell})| \\
&\quad + \sum_{\ell \in \widetilde{I}^{\Phi}_{\neq}} \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus (V(P_{\Phi}) \cup S_{\ell})\right| + \sum_{\ell \in \widetilde{I}^{\Psi}_{\neq}} \left|H_{\ell+1}\left(a^{\Psi}_{\min}\right) \setminus (V(P_{\Psi}) \cup S_{\ell})\right| \\
&\quad + \sum_{\ell \in \widetilde{I}_{=}} \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus (V(P_{\Phi}) \cup V(P_{\Psi}) \cup S_{\ell})\right|.
\end{aligned}$$

Note that, for all $\ell \in \{\ell^*, \dots, L\}$, we have

$$\begin{aligned}
\left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus (V(P_{\Phi}) \cup S_{\ell})\right| &= \left|\left(H_{\ell+1}\left(a^{\Phi}_{\max}\right) \cap \left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec}\right) \setminus (V(P_{\Phi}) \cup S_{\ell})\right| \\
&\quad + \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(\left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec} \cup S_{\ell}\right)\right| \\
&= g^{\Phi}(\ell, \max) + \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(\left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec} \cup S_{\ell}\right)\right|,
\end{aligned}$$

and similarly,

$$\left|H_{\ell+1}\left(a^{\Psi}_{\min}\right) \setminus (V(P_{\Psi}) \cup S_{\ell})\right| = g^{\Psi}(\ell, \min) + \left|H_{\ell+1}\left(a^{\Psi}_{\min}\right) \setminus \left(\left[a^{\Psi}_{\min}, a^{\Psi}_{\max}\right]_{\prec} \cup S_{\ell}\right)\right|.$$

Moreover, for any $\ell \in I_{=}$, we obtain

$$\begin{aligned}\left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(V(P_\Phi) \cup V(P_\Psi) \cup S_\ell\right)\right| &= \left|\left(H_{\ell+1}\left(a^{\Phi}_{\max}\right) \cap \left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec}\right) \setminus \left(V(P_\Phi) \cup S_\ell\right)\right| \\ &\quad + \left|\left(H_{\ell+1}\left(a^{\Psi}_{\min}\right) \cap \left[a^{\Psi}_{\min}, a^{\Psi}_{\max}\right]_{\prec}\right) \setminus \left(V(P_\Psi) \cup S_\ell\right)\right| \\ &\quad + \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(\left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec} \cup \left[a^{\Psi}_{\min}, a^{\Psi}_{\max}\right]_{\prec} \cup S_\ell\right)\right| \\ &= g^{\Phi}(\ell, \max) + g^{\Psi}(\ell, \min) \\ &\quad + \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(\left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec} \cup \left[a^{\Psi}_{\min}, a^{\Psi}_{\max}\right]_{\prec} \cup S_\ell\right)\right|.\end{aligned}$$

Finally, note that

$$\begin{aligned}\gamma^{\Phi} + \gamma^{\Psi} = \sum_{\ell=\ell^*}^{L} \left|\left(V(P_\Phi) \cup V(P_\Psi)\right) \setminus S_\ell\right| &+ \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1}, \\ H \subseteq \left(a^{\Phi}_{\min}, a^{\Phi}_{\max}\right)_{\prec}, \\ |V(P_\Phi) \cap H| \geq k}} \left|H \setminus \left(V(P_\Phi) \cup S_\ell\right)\right| \\ &+ \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1}, \\ H \subseteq \left(a^{\Psi}_{\min}, a^{\Psi}_{\max}\right)_{\prec}, \\ |V(P_\Psi) \cap H| \geq k}} \left|H \setminus \left(V(P_\Psi) \cup S_\ell\right)\right|\end{aligned}$$

together with the observations above, implies that

$$\begin{aligned}&\sum_{\ell=\ell^*}^{L} |V(P) \setminus S_\ell| + \sum_{\ell=\ell^*}^{L-1} \sum_{\substack{H \in \mathcal{H}_{\ell+1}, \\ H \subseteq \left(a^{\oplus}_{\min}, a^{\oplus}_{\max}\right)_{\prec}, \\ |V(P) \cap H| \geq k}} |H \setminus \left(V(P) \cup S_\ell\right)| \\ &= \gamma^{\Phi} + \gamma^{\Psi} \\ &\quad + \sum_{\ell \in \widetilde{I}^{\Phi}_{\neq}} \left(g^{\Phi}(\ell, \max) + \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(\left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec} \cup S_\ell\right)\right|\right) \\ &\quad + \sum_{\ell \in \widetilde{I}^{\Psi}_{\neq}} \left(g^{\Psi}(\ell, \min) + \left|H_{\ell+1}\left(a^{\Psi}_{\min}\right) \setminus \left(\left[a^{\Psi}_{\min}, a^{\Psi}_{\max}\right]_{\prec} \cup S_\ell\right)\right|\right) \\ &\quad + \sum_{\ell \in \widetilde{I}_{=}} \left(g^{\Phi}(\ell, \max) + g^{\Psi}(\ell, \min)\right) \\ &\quad + \sum_{\ell \in \widetilde{I}_{=}} \left|H_{\ell+1}\left(a^{\Phi}_{\max}\right) \setminus \left(\left[a^{\Phi}_{\min}, a^{\Phi}_{\max}\right]_{\prec} \cup \left[a^{\Psi}_{\min}, a^{\Psi}_{\max}\right]_{\prec} \cup S_\ell\right)\right|.\end{aligned}$$

In particular, the right-hand side depends only on Φ and Ψ. Therefore, choosing $\gamma^{\oplus}$ as this value completes the proof. □

Finally, we prove Lemma 4.81:

Lemma 4.81. *Let Φ be an (x, y, k')-realizable covering profile. Then for every (x, y, k')-realization P of Φ, there exists a tree solution $(\mathcal{B}, \pi)$ of height Γ with $V(\mathcal{B}) = V(P)$ and $\operatorname{cost}(\mathcal{B}, \pi) = w^0(P') - k^2 \cdot D_{\ell^*}$, where P' denotes the monotone path that starts in x, then traverses P, and ends in y.*

Proof. Any (x, y, k')-realization P of Φ satisfies $|V(P)| = k' = 2^\Gamma - 1$. Hence, it is straightforward to construct the unique full binary tree $\mathcal{B}$ with $V(\mathcal{B}) = V(P)$ such that $(\mathcal{B}, \pi)$ is a tree solution of height Γ, where π is the empty function.

It remains to prove that $\mathrm{cost}(\mathcal{B}, \pi) = w^0(P') - k^2 \cdot D_{\ell^*}$, where P' results from P as described above. Given a tree solution $(\mathcal{B}, \pi)$ of height $h > 1$, we recursively define its *internal* cost as

$$\mathrm{cost}_{\mathrm{int}}(\mathcal{B}, \pi) \coloneqq \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_1, \pi_1) + \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_2, \pi_2) + 2^{h-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a, r) + \sum_{a \in V(\mathcal{B}_2)} c(r, a) \right),$$

where r denotes the root of $\mathcal{B}$ and $(\mathcal{B}, \pi) = (\mathcal{B}_1, \pi_1) \oplus (\mathcal{B}_2, \pi_2)$. If $\mathcal{B}$ has height 1, we define $\mathrm{cost}_{\mathrm{int}}(\mathcal{B}, \pi) \coloneqq 0$.

Furthermore, we define its *external* cost as

$$\begin{aligned} \mathrm{cost}_{\mathrm{ext}}(\mathcal{B}, \pi) \coloneqq{}& 2^\Gamma \cdot \sum_{a \in V(\mathcal{B})} \Big(c(x, a) + c(a, y) \Big) \\ &+ \sum_{a \in V(\mathcal{B})} \left(\sum_{\substack{\ell \in \{h+1, \dots, \Gamma\}, \\ \pi(\ell) \prec a}} 2^{\ell-1} \cdot c(\pi(\ell), a) + \sum_{\substack{\ell \in \{h+1, \dots, \Gamma\}, \\ \pi(\ell) \succ a}} 2^{\ell-1} \cdot c(a, \pi(\ell)) \right). \end{aligned}$$

As an intermediate result, we prove that for any tree solution $(\mathcal{B}, \pi)$ we have

$$\mathrm{cost}(\mathcal{B}, \pi) = \mathrm{cost}_{\mathrm{int}}(\mathcal{B}, \pi) + \mathrm{cost}_{\mathrm{ext}}(\mathcal{B}, \pi).$$

To this end, we proceed via induction on the height h of $(\mathcal{B}, \pi)$. Any tree solution of height $h = 1$ consists of a single vertex. Hence, in this case, the claim follows immediately from the definitions of $\mathrm{cost}_{\mathrm{int}}(\mathcal{B}, \pi)$ and $\mathrm{cost}_{\mathrm{ext}}(\mathcal{B}, \pi)$.

Suppose that $h > 1$, and let $(\mathcal{B}_1, \pi_1)$ and $(\mathcal{B}_2, \pi_2)$ be the tree solutions of height $h - 1$ with $(\mathcal{B}, \pi) = (\mathcal{B}_1, \pi_1) \oplus (\mathcal{B}_2, \pi_2)$. Thus, $V(\mathcal{B}_1) \prec \{r\} \prec V(\mathcal{B}_2)$ and $\pi_1(h) = \pi_2(h) = r$, where r denotes the root of $\mathcal{B}$. By the induction hypothesis, we then have

$$\begin{aligned} \mathrm{cost}(\mathcal{B}, \pi) &= \mathrm{cost}(\mathcal{B}_1, \pi_1) + \mathrm{cost}(\mathcal{B}_2, \pi_2) + 2^\Gamma \cdot \big(c(x, r) + c(r, y) \big) \\ &\quad + \sum_{\substack{\ell \in \{h+1, \dots, \Gamma\}, \\ \pi(\ell) \prec r}} 2^{\ell-1} \cdot c(\pi(\ell), r) + \sum_{\substack{\ell \in \{h+1, \dots, \Gamma\}, \\ \pi(\ell) \succ r}} 2^{\ell-1} \cdot c(r, \pi(\ell)) \\ &= \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_1, \pi_1) + \mathrm{cost}_{\mathrm{ext}}(\mathcal{B}_1, \pi_1) + \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_2, \pi_2) + \mathrm{cost}_{\mathrm{ext}}(\mathcal{B}_2, \pi_2) \\ &\quad + 2^\Gamma \cdot \big(c(x, r) + c(r, y) \big) \\ &\quad + \sum_{\substack{\ell \in \{h+1, \dots, \Gamma\}, \\ \pi(\ell) \prec r}} 2^{\ell-1} \cdot c(\pi(\ell), r) + \sum_{\substack{\ell \in \{h+1, \dots, \Gamma\}, \\ \pi(\ell) \succ r}} 2^{\ell-1} \cdot c(r, \pi(\ell)) \end{aligned}$$

$$
\begin{aligned}
&= \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_1,\pi_1) + \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_2,\pi_2) + 2^{\Gamma}\cdot \sum_{a\in V(\mathcal{B})} \big(c(x,a)+c(a,y)\big) \\
&\quad + \sum_{a\in V(\mathcal{B}_1)} \left(\sum_{\substack{\ell\in\{h,\dots,\Gamma\},\\ \pi_1(\ell)\prec a}} 2^{\ell-1}\cdot c(\pi_1(\ell),a) + \sum_{\substack{\ell\in\{h,\dots,\Gamma\},\\ \pi_1(\ell)\succ a}} 2^{\ell-1}\cdot c(a,\pi_1(\ell)) \right) \\
&\quad + \sum_{a\in V(\mathcal{B}_2)} \left(\sum_{\substack{\ell\in\{h,\dots,\Gamma\},\\ \pi_2(\ell)\prec a}} 2^{\ell-1}\cdot c(\pi_2(\ell),a) + \sum_{\substack{\ell\in\{h,\dots,\Gamma\},\\ \pi_2(\ell)\succ a}} 2^{\ell-1}\cdot c(a,\pi_2(\ell)) \right) \\
&\quad + \sum_{\substack{\ell\in\{h+1,\dots,\Gamma\},\\ \pi(\ell)\prec r}} 2^{\ell-1}\cdot c(\pi(\ell),r) + \sum_{\substack{\ell\in\{h+1,\dots,\Gamma\},\\ \pi(\ell)\succ r}} 2^{\ell-1}\cdot c(r,\pi(\ell)) \\
&= \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_1,\pi_1) + \mathrm{cost}_{\mathrm{int}}(\mathcal{B}_2,\pi_2) + 2^{\Gamma}\cdot \sum_{a\in V(\mathcal{B})} \big(c(x,a)+c(a,y)\big) \\
&\quad + 2^{h-1}\cdot \left(\sum_{a\in V(\mathcal{B}_1)} c(a,r) + \sum_{a\in V(\mathcal{B}_2)} c(r,a) \right) \\
&\quad + \sum_{a\in V(\mathcal{B})} \left(\sum_{\substack{\ell\in\{h+1,\dots,\Gamma\},\\ \pi(\ell)\prec a}} 2^{\ell-1}\cdot c(\pi(\ell),a) + \sum_{\substack{\ell\in\{h+1,\dots,\Gamma\},\\ \pi(\ell)\succ a}} 2^{\ell-1}\cdot c(a,\pi(\ell)) \right) \\
&= \mathrm{cost}_{\mathrm{int}}(\mathcal{B},\pi) + \mathrm{cost}_{\mathrm{ext}}(\mathcal{B},\pi).
\end{aligned}
$$

This proves our intermediate result.

Now, let $(\mathcal{B},\pi)$ be a tree solution of height Γ. We denote by P the monotone path on $V(\mathcal{B})$ and by P' the monotone path that additionally visits x and y. Note that in order to complete the proof of our lemma, it suffices to prove

$$
w^0(P') - k^2\cdot D_{\ell^*} = \mathrm{cost}_{\mathrm{int}}(\mathcal{B},\pi) + 2^{\Gamma}\cdot \sum_{a\in V(\mathcal{B})} \Big(c(x,a)+c(a,y)\Big), \tag{4.43}
$$

as this implies

$$
\begin{aligned}
w^0(P') - k^2\cdot D_{\ell^*} &= \mathrm{cost}_{\mathrm{int}}(\mathcal{B},\pi) + 2^{\Gamma}\cdot \sum_{a\in V(\mathcal{B})} \Big(c(x,a)+c(a,y)\Big) \\
&= \mathrm{cost}_{\mathrm{int}}(\mathcal{B},\pi) + \mathrm{cost}_{\mathrm{ext}}(\mathcal{B},\pi) \\
&= \mathrm{cost}(\mathcal{B},\pi).
\end{aligned}
$$

We prove (4.43) by induction on Γ. In the following, we write height_Γ, cost_Γ, and w^0_Γ to make their dependence on Γ explicit.

The case $\Gamma = 1$ is trivial, since the internal cost are zero in this case. Suppose that $\Gamma > 1$. Again, let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the full binary subtrees of height $\Gamma-1$ rooted at the children of the root r of $\mathcal{B}$ with $V(\mathcal{B}_1) \prec \{r\} \prec V(\mathcal{B}_2)$. We denote by P_1 and P_2 the monotone paths whose vertex sets are $V(\mathcal{B}_1)$ and $V(\mathcal{B}_2)$, respectively. We extend these paths to monotone paths P_1' and P_2' by additionally visiting x and y.

We denote by $P'_{[x,r]}$ the x-r subpath of P', and by $P'_{[r,y]}$ the r-y subpath of P'. Then, Lemma 4.70 implies that

$$\begin{aligned} w^0_{\Gamma-1}(P') - k^2 \cdot D_{\ell^*} &= w^0_{\Gamma-1}(P'_{[x,r]}) + w^0_{\Gamma-1}(P'_{[r,y]}) - 2k^2 \cdot D_{\ell^*} \\ &= w^0_{\Gamma-1}(P'_1) + w^0_{\Gamma-1}(P'_2) - 2k^2 \cdot D_{\ell^*} \\ &\quad + 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a,r) + \sum_{a \in V(\mathcal{B}_2)} c(r,a) \right) \\ &\quad - 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a,y) + \sum_{a \in V(\mathcal{B}_2)} c(x,a) \right). \end{aligned}$$

For any $j \in \{1, \dots, 2^\Gamma - 1\}$, we have $\text{height}_{\Gamma-1}(j) = \text{height}_\Gamma(j)$, and for $j' \in \{0, 2^\Gamma\}$, we have $\text{height}_{(\Gamma-1)}(j') + 1 = \text{height}_\Gamma(j')$.

Together with the induction hypothesis, this implies

$$\begin{aligned} &w^0_\Gamma(P') - k^2 \cdot D_{\ell^*} \\ &= w^0_{\Gamma-1}(P') - k^2 \cdot D_{\ell^*} + 2^\Gamma \cdot \Big(c(x,r) + c(r,y) \Big) \\ &\quad + 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(x,a) + \sum_{a \in V(\mathcal{B}_2)} c(a,y) \right) + 2^\Gamma \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a,y) + \sum_{a \in V(\mathcal{B}_2)} c(x,a) \right) \\ &= w^0_{\Gamma-1}(P'_1) + w^0_{\Gamma-1}(P'_2) - 2k^2 \cdot D_{\ell^*} + 2^\Gamma \cdot \Big(c(x,r) + c(r,y) \Big) \\ &\quad + 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(x,a) + \sum_{a \in V(\mathcal{B}_2)} c(a,y) \right) + 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a,y) + \sum_{a \in V(\mathcal{B}_2)} c(x,a) \right) \\ &\quad + 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a,r) + \sum_{a \in V(\mathcal{B}_2)} c(r,a) \right) \\ &\overset{(4.43)}{=} \text{cost}_{\text{int}}(\mathcal{B}_1, \pi_1) + \text{cost}_{\text{int}}(\mathcal{B}_2, \pi_2) + 2^\Gamma \cdot \Big(c(x,r) + c(r,y) \Big) \\ &\quad + 2^\Gamma \cdot \left(\sum_{a \in V(\mathcal{B}_1)} \Big(c(x,a) + c(a,y) \Big) + \sum_{a \in V(\mathcal{B}_2)} \Big(c(x,a) + c(a,y) \Big) \right) \\ &\quad + 2^{\Gamma-1} \cdot \left(\sum_{a \in V(\mathcal{B}_1)} c(a,r) + \sum_{a \in V(\mathcal{B}_2)} c(r,a) \right) \\ &= \text{cost}_{\text{int}}(\mathcal{B}, \pi) + 2^\Gamma \cdot \sum_{a \in V(\mathcal{B})} \Big(c(x,a) + c(a,y) \Big), \end{aligned}$$

which completes the proof. □

4.9 Conclusion

Combining Theorems 4.5, 4.36 and 4.64 directly implies the main result of this chapter, Theorem 4.3. We have thus given a deterministic algorithm for Asymmetric A Priori TSP with

poly-logarithmic approximation factor and quasi-polynomial running time. Interestingly, by Theorem 4.2, the approximation factor is below the adaptivity gap.

A natural open question is whether one can improve the running time of our algorithm and achieve a poly-logarithmic approximation factor in polynomial time. For this, the polynomial-time reductions we provided in this chapter could be an important ingredient. We highlight that our proofs not only imply reductions from Asymmetric A Priori TSP to Hop-ATSP, hierarchically-ordered instances, and finally to the Path Covering problem, but they actually show that these problems are equivalent up to poly-logarithmic factors in the approximation guarantee (and polynomial factors in the running time).

All our reductions rely on showing that a solution for one problem can be turned into a solution of the other problem and vice versa, without changing the cost by more than a poly-logarithmic factor (and possible scaling of the objective). When reducing to a covering problem in Section 4.7, we showed how tours (for non-degenerate instances) can be transformed into covering solutions of not much larger cost. We did not specify running times for this transformation because we only applied this statement to an optimal tour in order to show that the value of an optimal Path Covering solution cannot be much larger than the k-hop cost of an optimal tour. However, we remark that our proof is constructive and gives rise to a polynomial-time algorithm.

Another interesting direction for future research would be to see if approximation factors below the adaptivity gap can also be achieved for other related problems such as the Symmetric A Priori TSP.

Part II

Real-World Applications: Pre-Processing Data for the BonnTour Algorithm

Chapter 5

Introduction to BonnTour

In the second part of this thesis, we will take a closer look at some aspects of tour planning in practice. In cooperation with our industry partner Greenplan (see `www.greenplan.de`), we have implemented a software package called "BonnTour" at the Research Institute for Discrete Mathematics of the University of Bonn. We are continuously developing the algorithm further, constantly improving solution quality or running time or introducing new features. BonnTour is designed for solving real-world vehicle routing problems, and can handle numerous additional constraints, e.g. time windows, working time breaks and heterogeneous vehicle fleets. Our software is successfully used on a daily basis in practice by partners relying on Greenplan.

A lot of the high-level main ideas behind BonnTour are described in [BHMSTTV24] and [Bla24]. Apart from the authors of [BHMSTTV24], Daniel Ebert and our student assistants also contribute significantly to the BonnTour code.

A main feature of BonnTour is the incorporation of time-dependent travel times: In real life instances, travel times may vary a lot during the day due to changes in traffic congestion or time-dependent road restrictions. While many vehicle routing algorithms in theory and in practice work with constant travel times, [BHMSTTV24] show that this simple approach fails as soon as the shipments have time windows in which they need to be delivered. We internally model time-dependent travel times by so-called *arrival time functions*.

This thesis takes a closer look on parts of BonnTour where optimization without full knowledge of all information is needed: In Chapter 6, we will explore a pre-processing step that augments a given road network to a so-called *Contraction Hierarchy* which will later allow for faster *path searches*: Asking for the fastest path from a start address to some target address is a question that naturally arises many times during tour planning. While the classical algorithm to find fastest paths in graphs with non-negative edge weights, Dijkstra's algorithm ([Dij59]), answers this question in almost linear time, its running time is too slow for our purposes, especially when working with time-dependent travel times. Based on ideas from [GSSD08] and [BGSV13], we therefore pre-process the road network to give it a special structure that fulfills the so-called *Contraction Hierarchy property* (cf. Section 6.1). This property enables us to find shortest paths faster in practice.

We show how to make the implementation of Contraction Hierarchies suitable for everyday use by speeding up its computation applying approximation to keep the complexity of the arising arrival time functions under control. Since this pre-processing still takes more time than we are usually willing to spend for solving tour planning instances, we perform it before we even know the actual addresses of our instance. Then, we show how to insert vertices representing the

addresses of interest into the precomputed Contraction Hierarchy without losing the ability to efficiently compute fastest paths.

In Chapter 7, we take a closer look on where exactly we insert addresses into the Contraction Hierarchy in order to model reality as accurately as possible. Since it is not always reasonable to drive as close as possible to an address, we then discuss finding a good parking position before knowing where we come from and where we drive next. This way, we can use this estimated best parking position during BonnTour's core tour optimization routine, and only after having decided for the final tours, we post-optimize the parking positions – now with full knowledge of what happens before and after visiting an address.

Finally, we present experimental results in Chapter 8, displaying running time and memory consumption of our Contraction Hierarchy computations and verifying the advantages of our address mapping routines.

5.1 Model and problem description

As mentioned earlier, our model supports time-dependent travel times that we represent by arrival time functions (see Section 5.4). All graphs that we work with in this part of the thesis are directed graphs, and edges in the graphs are equipped with arrival time functions. We allow heterogeneous vehicle fleets in our tour planning instances, meaning that there can be different vehicles with different characteristics which can lead to varying arrival time functions. In general, we therefore work with multiple graphs (also called *road graphs*) in parallel, one road graph per vehicle type. For different vehicle types, these graphs may agree if the vehicles' properties that are relevant on the road (e.g. length, height, maximum load per axle, etc.) agree. In order to find vehicles that fit together in these characteristics, we perform a vehicle equivalence class analysis in the beginning of BonnTour. All road graphs are based on the same *base graph*, but may be equipped with different arrival time functions. Moreover, we make vehicle-specific changes to the road graph, e.g. when modeling prohibited maneuvers (cf. Fig. 7.1). We also use a pedestrian graph that is derived from the same base graph, too, but edges that correspond to roads unusable for pedestrians are removed. Pedestrian travel times are assumed to be constant throughout the day; the arrival time functions for the pedestrian graph therefore all correspond to a constant speed.

An instance of the tour planning problem considered by BonnTour consists of a set of addresses, a set of shipments, and a set of vehicle types, each assigned one of these road graphs, a number how many vehicles of this type are available, and start and end addresses for the vehicle. Each shipment carries information about its pickup address, its dropping penalty, a time window and a handling time for the pickup action, its delivery address, and a time window and a handling time for the delivery action. BonnTour even allows to set multiple pickup and delivery options for each shipment. Moreover, we can define incompatibilities between vehicle types and shipments. Our time model additionally allows to specify vehicle dependent parking and visit times at each address. Visit and handling times need to take place within the specified time windows whereas parking time may be spent outside of the time window.

Our task is to compute a set of tours using a subset of the available vehicles, delivering the shipments adhering to the time window and vehicle incompatibility constraints. We have to pay the dropping penalty for all shipments that are not delivered. The goal is to find a solution of minimum cost where our cost model allows to define a vehicle dependent fixed cost per tour, cost per time, and cost per distance.

There are further constraints like working time limits, vehicle capacities, or maximum distance constraints in practice. BonnTour actually incorporates them and supports an even more general

cost model, but the simplified model explained here suffices to describe our procedures in the following chapters.

5.2 Outline and contributions

The concept of Contraction Hierarchies was introduced by [GSSD08] and extended to time-dependent travel times by [BGSV13]. A first version of Contraction Hierarchies within BonnTour was implemented by Silas Rathke (see [Rat21]) and further improved by Paula Heinz ([Hei24]). Since then, we introduced a rounding procedure during the construction of the Contraction Hierarchies, and the possibility to insert new vertices while keeping a slightly modified Contraction Hierarchy property that is still sufficient for our purposes. The implementation of a street-side aware variant of this insertion procedure is joint work with our former student assistant Richard Ueltzen (see Section 6.4.3).

With these adjustments, we are now able to compute Contraction Hierarchies for areas as large as Europe (see Chapter 8). The ability to compute Contraction Hierarchies within a reasonable time and with only moderate disk space usage allows our cooperation partner Greenplan to use them in most of their customer projects by default and thereby experience significant speedups. We will examine the improvements that were necessary for enabling the daily use of Contraction Hierarchies in Section 6.3 and Section 6.4.

Chapter 7 focuses on two very closely related features of BonnTour, namely Address Mapping and Shared Parking. As an input to BonnTour, we are (amongst others) given a Contraction Hierarchy and the geo-coordinates of the addresses where shipments have to be picked up or delivered. We then have to map these addresses to vertices in the Contraction Hierarchy – be it already existing vertices or vertices that we add to the Contraction Hierarchy specifically for the purpose of representing a customer address. The first step of Address Mapping consists of finding a vertex whose geographical position is as close as possible to the geo-coordinates of the address in the input. This vertex will then be our *pedestrian position*, i.e., the position where we assume throughout our tour planning procedure that this is the position that the parcel service must visit. See Section 7.1 for the details.

However, the pedestrian position might not always be the best choice for parking the delivery vehicle; it could be not reachable at all (e.g. in a pedestrian zone) or force us to drive large detours (e.g. when it is in a one-way street). Instead it can be reasonable to park the vehicle at most a few hundred meters away and walk the last part. In practice, the choice of the parking position can depend on multiple parameters like the time of day of our visit, the direction where we came from, or where we want to drive next. While BonnTour is able to choose between different address options for the pickup and the delivery of a shipment, each of these addresses needs to be assigned a fixed position during the core algorithm that optimizes the driven tours. We therefore need to find a good *vehicle position* beforehand that will most likely be a good position to park at in the situation where we visit the address eventually. This will require a global view on the reachability of the possible parking spaces around our pedestrian position. The details on how we choose the vehicle position can be found in Section 7.2. After having fixed the tours, BonnTour applies a post-optimization routine finding the best parking spots with respect to the fixed stop order. Our former student assistant Anna Austel contributed to this parking post-optimization code.

In customer cases where the addresses we need to visit are dense it often makes sense to park the vehicle once and then walk to several nearby target positions. In our internal model this corresponds to assigning the same vehicle position to different addresses. Apart from the advantages in practice when walking to the next address is faster than starting the vehicle and

parking again, this even comes with the additional benefit of savings in memory usage during the algorithm because we need to store distance information for fewer point-to-point connections. We therefore encourage the address mapping algorithm to find good parking spots that are feasible for multiple nearby addresses; we call this feature *Shared Parking*. We refer to Section 7.3 for the details.

Finally, Chapter 8 is devoted to computational results of our improvements for Contraction Hierarchy computations as well as our new Address Mapping and Shared Parking procedures on practical instances, and Chapter 9 discusses some ideas for future work.

5.3 Related work

BonnTour solves vehicle routing problems using a heuristic approach described in [BHMSTTV24] and in [Bla24]. As all kinds of vehicle routing problems contain the Traveling Salesperson Problem in some way, they are APX-hard. For certain variants approximation algorithms are known (see e.g. [BBCM04], [CKP12], [NR11], [FMRS22], [BTV23]). However, they are not suitable for practical purposes. Another approach is solving vehicle routing problems via mixed integer programming formulations, as e.g. done in [PSUV20]. This approach can solve instances with several hundred customers optimally, but due to too large running times, these algorithms are again not suitable for practical purposes. Therefore a lot of local search heuristics have been developed, see e.g. [VCGP13], [AS19], [AV21], [QSU21], [Vid22]. Toth and Vigo provide an overview over techniques used for solving vehicle routing problems in [TV14].

In practice, travel times can vary a lot throughout the day, e.g. during rush hour. [MD92] considered time-dependent travel times using piecewise constant travel times. Working with piecewise constant speeds instead leads to piecewise linear travel time functions, which is the most common model nowadays ([IGP03]). Gendreau, Ghiani, and Guerrier gave a survey on vehicle routing with time-dependent travel times in [GGG15]. The arrival time functions between pairs of addresses can directly be given in the input (like in [DMCRG08], [And12], [DRWD13], [PZL21a], and [PZL21b]). Another possibility is to give a road network with arrival time functions for the edges of the graph as input. [Man14], [HZVG17], [Ben17], [BAFQV19], and [GGLP21] work with the latter model. BonnTour is able to process both types of input, pre-computed arrival time functions or a road network. The latter option usually requires an additional pre-processing step that computes the arrival time functions between pairs of addresses from the road network. To this end, some operations on arrival time functions need to be performed, mainly composition and pointwise minima. [Wie86] and [HS86] show that the number of breakpoints of the arrival time functions can grow superlinearly when taking the pointwise minimum. We therefore apply techniques introduced by Imai and Iri ([II86]) to approximate the arrival time functions optimally, i.e., with the minimum number of breakpoints, with a guaranteed error bound in linear time.

In order to be able to quickly answer fastest path queries, we first apply a runtime-intensive pre-processing, building a Contraction Hierarchy. The idea of Contraction Hierarchies was introduced by Geisberger, Sanders, Schultes, and Delling in [GSSD08] for constant edge weights and generalized to time-dependent travel times by Batz, Geisberger, Sanders, and Vetter in [BGSV13]. Geisberger and Sanders complemented this approach with an additional instance-specific pre-processing ([GS10]). Strasser, Wagner, and Zeitz proposed a variant consuming less memory ([SWZ21]). There has previously already been a large body of work on so-called *hierarchical routing*, i.e., an approach for pre-processing a road network to enable faster path search exploiting some kind of hierarchical structure, see e.g. [Gut04], [SS05], [SS07a], [BFSS07]. A general overview over different speed-up techniques for path searches can be found in [SS07b].

A simple idea to accelerate the construction of Contraction Hierarchies is to take advantage of the presence of multiple CPU cores and parallelize the procedure. Vetter made some first observations how a reasonable parallelization can be achieved ([Vet09]). Recently, Wan, Dong, Wang, Zhu, Gu, and San made an effort to obtain a more sophisticated construction method allowing for more speed-ups through parallelization ([WDWZGS25]).

Many implementation details of Contraction Hierarchies in BonnTour are described in the Master's theses of Rathke and Heinz ([Rat21], [Hei24]).

When finding edges in a given road graph with a minimal Euclidean distance to an address for which we want to insert a vertex in the road graph, we use R-trees to answer spatial queries fast: R-trees as introduced by Guttman [Gut84] are a derivative of so-called kd-trees ([Ben75]) and are used in computational geometry. Exploiting the geometrical structure of a given search space, they help answering nearest neighbor queries.

5.4 Preliminaries

5.4.1 Modeling time-dependent travel times

We model the given road network with time-dependent travel time information by a directed graph $G = (V, E)$ where every edge $e \in E$ is equipped with an *arrival time function* f_e. For an edge $e = (v, w) \in E$ and a time $t \in \mathbb{R}$, $f_e(t)$ is the arrival time at w depending on the start time t at v when traversing e. The standard model for describing time-dependent travel times (see e.g. [IGP03], [BHMSTTV24]) is the following:

Definition 5.1 (Arrival time function)**.** *We call a piecewise linear function $f\colon (-\infty, t_{\max}] \to \mathbb{R}$ for some $t_{\max} \in \mathbb{R}$ an* arrival time function *if it has the following properties:*

(i) f is continuous,

(ii) $f(x) \geq f(y)$ if $y < x \leq t_{\max}$,

(iii) $f(x) \geq x$ for all $x \leq t_{\max}$, and

(iv) f is initially constant, i.e. $f(x) = f(t_{\min})$ for some $t_{\min} \leq t_{\max}$ and all $x \leq t_{\min}$.

We will also define $f(x) := \infty$ for $x > t_{\max}$. The domain *of f is $\mathrm{dom}(f) := (-\infty, t_{\max})$. We often write ATF as an abbreviation for arrival time function.*

Note that these conditions all represent reasonable assumptions in the real-world application that we want to model with ATFs: The most controversial condition is probably condition (i) because there are discontinuous events happening in road networks (e.g. time-of-day restricted access to pedestrian zones, barriers at railroad crossings being closed, etc.). But as travel times anyway underlie fluctuations that are impossible to foresee exactly, we interpret the given travel time information as expected arrival times in the road network which should always be continuous. Condition (ii) just says that we cannot arrive earlier by starting later. A simple consideration shows that this is true: For $x < y$, we can always wait until y before traversing the edge, thus arriving at $f(y)$. Condition (iii) corresponds to the travel time, i.e $f(x) - x$, being non-negative. For condition (iv), we can set $t_{\min}$ to the start of the considered time horizon – starting even earlier is not possible and requires us to wait until $t_{\min}$.

An ATF f can be viewed as a list of b points $(t_1, f(t_1)), \ldots, (t_b, f(t_b)) \in \mathbb{R}^2$ with $t_{\min} = t_1 \leq \cdots \leq t_b = t_{\max}$ and affine segments connecting two consecutive points (plus a constant extension to $-\infty$ at the beginning). We call the points $(t_i, f(t_i))$ *breakpoints* for $i = 1, \ldots, b$. The points $(t_1, f(t_1)), \ldots, (t_{b-1}, f(t_{b-1}))$ are the *inner breakpoints*. We can assume that the slope of f changes in each inner breakpoint (otherwise we can remove it from the list of breakpoints).

The minimum number of breakpoints needed to represent f is also called the *complexity* of f and denoted by $|f|$.

Not only travel time on road graph edges can be modeled by ATFs, but all actions that we perform in general during a tour can be assigned an ATF representing the finish time of the action depending on its start time. For instance, a delivery action that takes 1 time unit and has to start between 2 and 4 corresponds to the ATF $f\colon (-\infty, 4] \to \mathbb{R}$ with $f(x) = \max(2, x) + 1$.

5.4.2 Basic operations on arrival time functions

Our model of time-dependent travel times allows us to implement basic operations efficiently. We mention the three operations that are most important for the rest of this thesis here: composition, pointwise minimum, and approximation. Performing these operations is well-known and most of the time straight-forward. Previous works on time-dependent routing problems like [VS20] employ similar algorithms. We still discuss them here for completeness.

Composing arrival time functions

Whenever we have a sequence of actions that all have a corresponding ATF (e.g. traversing several road graph edges consecutively and performing pickup and delivery actions), we can compose all of their ATFs to obtain a function representing the finish time of the whole sequence depending on the start time. As proven in [BHMSTTV24], this composition can be performed in running time almost linear in the sum of the number of breakpoints:

Proposition 5.2 ([BHMSTTV24])**.** *Given arrival time functions $f_1, \dots, f_k$ such that f_i has b_i breakpoints ($i = 1, \dots, k$), we can compute $f_k \circ \cdots \circ f_1$ in $\mathcal{O}((\log k) \sum_{i=1}^{k} b_i)$ time. The resulting function is itself an ATF and has at most $1 + \sum_{i=1}^{k}(b_i - 1)$ breakpoints.*

Composing arrival time functions is particularly important for computing the time needed to travel from some start vertex $s \in V$ to a target vertex $t \in V$: Given an s-t-path P in G, we obtain the arrival time function f_P for P by composing the ATFs of all edges in P. However, different s-t-paths might be optimal throughout the day (depending e.g. on traffic congestion or time-dependent road closures), see also Fig. 5.1. The optimal arrival time between s and t is therefore modeled by the pointwise minimum $f_{s,t}(x) := \min\{f_P(x) : P \text{ is an } s\text{-}t\text{-path}\}$.

Minimum of arrival time functions

The following proposition is proven in [BHMSTTV24]:

Proposition 5.3 ([BHMSTTV24])**.** *Given arrival time functions f_1 and f_2 with b_1 and b_2 breakpoints respectively, we can compute the pointwise minimum $\min\{f_1, f_2\}$ in $\mathcal{O}(b_1 + b_2)$ time. If $\operatorname{dom}(f_1) = \operatorname{dom}(f_2)$ then $\min\{f_1, f_2\}$ is itself an arrival time function with at most $2(b_1+b_2)-3$ breakpoints.*

Note that while the composition of several ATFs did not increase the total number of breakpoints, taking the minimum can introduce new breakpoints. See Fig. 5.2 for an example. [HS86] showed that the pointwise minimum of n line segments in the plane can have at most $\mathcal{O}(n\alpha(n))$ breakpoints, where $\alpha(n)$ is the inverse Ackermann function. [Wie86] gave an example attaining this upper bound. Recursively applying Proposition 5.3 leads to a running time of $\mathcal{O}(m\alpha(m) \log n)$ for taking the pointwise minimum of n functions with in total m breakpoints (see [AS00]). [Sch20] and [BHMSTTV24] show how to reduce this running time to $\mathcal{O}(m \log n)$.

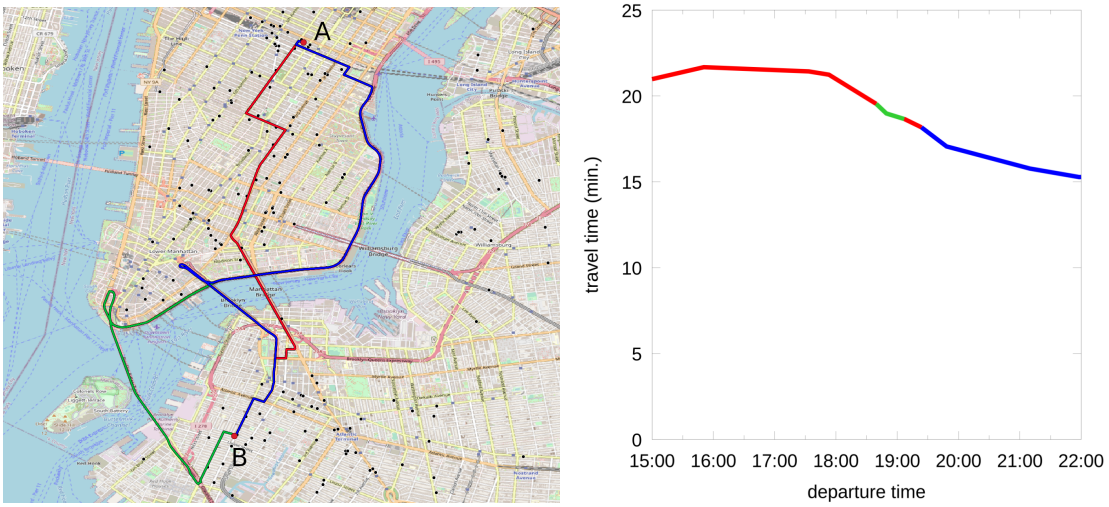


Figure 5.1: There are three different fastest paths from point A in Manhattan to point B in Brooklyn during the time span from 3pm to 10pm: In the afternoon, the red path is optimal. Then there is a short period where the green path is the fastest before red becomes optimal again. In the evening, the blue path is optimal.

The picture is taken from [BHMSTTV24], see also `https://doi.org/10.1016/j.disopt.2024.100848` where it is available under a Creative Commons License (`https://creativecommons.org/licenses/by-nc-nd/4.0/`).

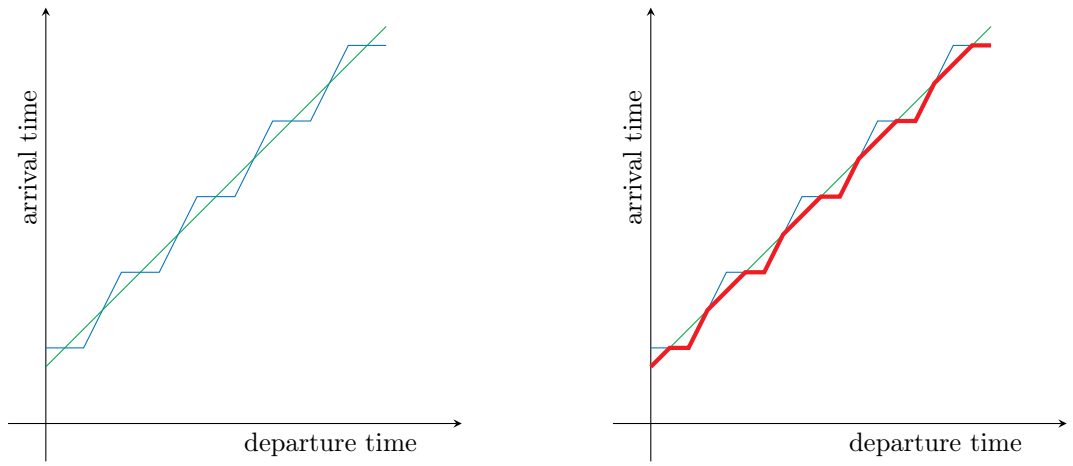


Figure 5.2: The red ATF on the right depicts the minimum of the blue and the green ATF. Note that taking the minimum of the blue and the green ATF introduces new breakpoints not present in the blue or the green ATF.

Approximating arrival time functions

In theory a road network can contain exponentially many paths between s and t that are at some point in time optimal. Even in the simple case where the ATFs associated with the edges are affine (i.e., they only have two breakpoints), the resulting function $f_{s,t}(x) = \min\{f_P(t) : P \text{ is an } s\text{-}t\text{-path}\}$ can have $n^{\Omega(\log n)}$ breakpoints, where $n := |V|$ is the number of vertices in the directed graph G (see [FHS11]). In practice, we usually do not see such extreme cases. However, when composing many ATFs and taking pointwise minima, we still end up with ATFs with many breakpoints. As the running times of many of our algorithms depend on the number of breakpoints of the involved ATFs, it is essential to keep this number small. We do this by approximating ATFs.

Our approach ensures that we never work with an ATF that corresponds to a shorter travel time than what was given in the input: To this end, we guarantee that the original ATF f is a lower bound on the approximated ATF f'. Since we also do not want to be too pessimistic, we define an allowed error $\varepsilon > 0$ and demand $f' \leq f + \varepsilon$ (usually we choose ε as some small percentage of the minimum travel time $\min\{f(x) - x : x \leq t_{\max}\}$). We use the classic polygonal line simplification algorithm due to Imai and Iri [II86] with some adaptations to ensure that the output is always an ATF. The following theorem is due to [BGSV13] and [BHMSTTV24].

Theorem 5.4 ([BGSV13], [BHMSTTV24])**.** *Given an ATF $f\colon (-\infty, t_{\max}] \to \mathbb{R}$ with b breakpoints and $\varepsilon > 0$, we can compute an ATF $f'\colon (-\infty, t_{\max}] \to \mathbb{R}$ with $f(x) \leq f'(x) \leq f(x) + \varepsilon$ for all $x \leq t_{\max}$ such that f' has the fewest possible number of breakpoints in $\mathcal{O}(b)$ time.*

Our approximation routine guarantees to yield an ATF f' with the minimum number of breakpoints while adhering to the lower and upper bound constraints. However, among all such functions with the minimum number of breakpoints, f' may not be the one with the minimum average error (see [BHMSTTV24] for a more detailed explanation of the implementation within BonnTour). In order to improve this, we apply a heuristic post-processing routine that moves individual breakpoints or line segments optimally while maintaining the properties guaranteed by Theorem 5.4.

Chapter 6

Contraction Hierarchies

6.1 Introduction

One of the most basic recurring problems in tour planning is finding fastest paths from a start location to some target location. The underlying mathematical problem is the Shortest Path Problem:

Problem 6.1: Shortest Path Problem

Given a graph $G = (V, E)$, a cost function $c\colon E \to \mathbb{R}$, and two vertices $s, t \in V$, find a path $s, e_1, v_1, e_2, \ldots, e_k, v_k, e_{k+1}, t$ (with $v_i \in V$, $e_j \in E$ for $i = 1, \ldots, k$ and $j = 1, \ldots, k+1$) minimizing

$$\sum_{i=1}^{k+1} c(e_i).$$

In a graph with n vertices, m edges, and non-negative edge cost, Dijkstra's algorithm ([Dij59]) solves this problem in $\mathcal{O}(m + n \log n)$ time when using Fibonacci heaps as introduced by Fredman and Tarjan ([FT87]). Only recently, Duan, Mao, Mao, Shu, and Yin found an algorithm improving this running time for sparse graphs, finding shortest paths from a single source s in time $\mathcal{O}(m \log^{2/3}(n))$ ([DMMSY25]). Since these algorithms are usually not fast enough in practice, heuristics are often used to speed up the search, most notably future costs leading to the so-called A*-search ([HNR68]). Often we are only interested in the length of a shortest path and not in the path itself, but this does not make the problem significantly easier: Dijkstra's algorithm (and similarly, the algorithm of Duan, Mao, Mao, Shu, and Yin,) actually finds the length of the shortest path first and then backtracks to obtain a path attaining this length.

Note that each edge is assigned a constant cost in the definition of Problem 6.1. As discussed before, this is a huge limitation when modeling reality. In real-life instances, travel times may vary a lot during the day due to changes in traffic congestion or time-dependent road restrictions. In order to represent travel times that change during the day more accurately, we use arrival time functions (see Section 5.4).

As we are looking at the travel time needed to traverse edges now, we will no longer talk about shortest paths, but about *fastest* paths instead. However, we can no longer ask for "the" fastest path, but only for the fastest path for a given start time. Moreover, we are interested in an arrival time function mapping each start time to the arrival time when traversing the fastest

path for the given start time. Note that this function is indeed an ATF as it is the pointwise minimum of compositions of edge ATFs.

Problem 6.2: Time-Dependent Fastest Path Problem

Given a graph $G = (V, E)$ where each edge $e \in E$ is equipped with an ATF f_e, and two vertices $s, t \in V$:

(i) for a start time $x \in \mathbb{R}$, find a path $s, e_1, v_1, e_2, \ldots, e_k, v_k, e_{k+1}, t$ (with $v_i \in V$, $e_j \in E$ for $i = 1, \ldots, k$ and $j = 1, \ldots, k+1$) minimizing

$$f_{e_{k+1}}(f_{e_k}(\ldots(f_1(x))\ldots)),$$

or

(ii) find the ATF mapping each start time $x \in \mathbb{R}$ to the arrival time when traversing a fastest path as defined in (i).

As soon as we are interested in the ATF that represents the time-of-day-dependent time needed to go from s to t via a fastest path, the classical algorithm of Dijkstra becomes inefficient: Instead of only propagating single numbers like edge costs, we suddenly need to propagate ATFs which includes composing ATFs and taking minima. Both these operations take a time that is linear in the number of breakpoints and they can result in ATFs with even higher complexity.

These problems motivate us to apply some (possibly runtime intensive) pre-processing to the graph G which will allow us later to find fastest paths efficiently. There are several approaches in the literature how to do this, e.g. via so-called Reach-Based Routing, Highway Hierarchies, Highway-Node Routing, or Transit-Node Routing (see e.g. [Gut04], [SS05], [SS07a], [BFSS07]). A general overview can be found in [SS07b]. Contraction Hierarchies as introduced by [GSSD08] are actually a special case of Highway-Node Routing, and constitute one of the simplest approaches to hierarchical routing.

The idea behind Contraction Hierarchies is motivated from real-world observations about fastest paths: When traveling from some starting position A to a target position B, one often first drives through small roads near A in order to reach some major street. This major street allows us to leave the city of A and may lead to a highway on which we spend most of our trip before returning to a major street leading into the city of B. From there, we might need to take again some small roads to reach our destination B. While certainly not all journeys look exactly like this, we can almost surely neglect small streets in residential areas that are neither near A nor near B, and we can keep in mind that fastest paths tend to first take roads in increasing "importance" before choosing streets with decreasing "importance".

Inspired by this, a Contraction Hierarchy results from a graph $G = (V, E)$ by assigning each vertex a *vertex importance* and inserting so-called shortcut edges, ensuring that for any pair of vertices $s, t \in V$ and any start time $x \in \mathbb{R}$ there is a fastest s-t-path at time x that is an *up-down path*, i.e., a path where the vertex importance first increases and then decreases. It is allowed that the increasing prefix or the decreasing suffix is empty.

Definition 6.3 (Vertex ordering, upward edge, downward edge)**.** *Given a graph $G = (V, E)$, a* vertex ordering *is a total order $\prec$ on V. If $v \prec w$ for $v, w \in V$, we also say that w is more* important *than v.*

Given a graph $G = (V, E)$ with a vertex ordering $\prec$, an edge $e = (v, w) \in E$ is called upward *edge if $v \prec w$, and it is called* downward *edge if $v \succ w$.*

Definition 6.4 (Up-down path)**.** *Given a graph $G = (V, E)$ with a vertex ordering $\prec$, an up-down path is a path $v_1, e_1, v_2, e_2, \dots, v_n, e_n, v_{n+1}$ such that there is an index $i \in \{1, \dots, n+1\}$ such that all edges e_j with $j < i$ are upward edges and all edges e_j with $j \geq i$ are downward edges.*

Contraction Hierarchies were first introduced by [GSSD08] for constant edge cost:

Definition 6.5 (Contraction Hierarchy)**.** *A graph $G = (V, E)$ together with some edge cost $c\colon E \to \mathbb{R}_{\geq 0}$ and a vertex ordering $\prec$ satisfies the* Contraction Hierarchy property *if for any pair of vertices $s, t \in V$ such that t is reachable from s there is an up-down path from s to t that is a shortest s-t-path. We call $(G, c, \prec)$ a Contraction Hierarchy.*

The name stems from the fact that vertices are hierarchically ordered and we construct a Contraction Hierarchy from a given graph by subsequently *contracting* vertices, see Section 6.2.

[BGSV13] extended this notion for time-dependent travel times:

Definition 6.6 (Contraction Hierarchy with time-dependent travel times)**.** *A graph $G = (V, E)$ equipped with ATFs f_e for all $e \in E$ and a vertex ordering $\prec$ satisfies the* Contraction Hierarchy property *if for any pair of vertices $s, t \in V$ and any start time $x \in \mathbb{R}$ such that t is reachable from s at start time x there is an up-down path from s to t that is a fastest s-t-path for the start time x. We call $(G, f, \prec)$ a Contraction Hierarchy with time-dependent travel times.*

The first implementation of Contraction Hierarchies within BonnTour is due to Silas Rathke ([Rat21]).

6.2 Contraction Hierarchies with time-dependent travel times

Given a vertex ordering $\prec$, the following procedure complements a given graph $G = (V, E)$ with ATFs f_e to a Contraction Hierarchy $(G', f, \prec)$ with $G' = (V, E')$ and $E' \supseteq E$ by adding so-called shortcut edges.

We subsequently construct graphs $G_1 := G, G_2, \dots, G_n$ where $n := |V|$. The graph $G' = (V, E')$ with $E' = \bigcup_{i=1}^{n} E(G_i)$ will satisfy the Contraction Hierarchy property. We write $V = \{v_1, \dots, v_n\}$ where the vertices are enumerated in the order of $\prec$, i.e. we have $v_1 \prec \cdots \prec v_n$. We set $V(G_i) := V \setminus \{v_1, \dots, v_{i-1}\}$ and we guarantee $E(G_i[V(G_{i+1})]) \subseteq E(G_{i+1})$, but we can also add some edges in each step. G_{i+1} is constructed from G_i by a procedure that [GSSD08] called "contracting" v_i $(i = 1, \dots, n-1)$, see also Fig. 6.1:

$V(G_{i+1})$ is obtained from $V(G_i)$ by removing v_i. The edge set of G_{i+1} is initialized with $E(G_i[V(G_{i+1})])$. If there is a pair of edges $(e_1, e_2) \in \delta^-_{G_i}(v_i) \times \delta^+_{G_i}(v_i)$ and a start time x for which a, e_1, v_i, e_2, b is faster than any other a-b-path in G_i not traversing v_i (for some $a, b \in V$), we add an edge (a, b) to $E(G_{i+1})$ and equip it with the ATF $f_{e_2} \circ f_{e_1}$. In the case that an edge (a, b) already exists, we *merge* its ATF with $f_{e_2} \circ f_{e_1}$, i.e., we assign it the pointwise minimum of its previous ATF and $f_{e_2} \circ f_{e_1}$. We try to prove that such a pair (e_1, e_2) does not exist by performing Dijkstra searches for each pair of neighbors of v_i. If we fail to prove this, we add a shortcut edge. Note that we sometimes also introduce a shortcut edge that is not necessary: Since an extensive Dijkstra search can take up more running time than we are willing to spend, we allow to abort the search early. In this case, we add a shortcut edge to guarantee the Contraction Hierarchy property for the resulting graph (see Proposition 6.8).

In order to prove that the resulting graph $G' = (V, E')$ with $E' = \bigcup_{i=1}^{n} E(G_i)$ has the Contraction Hierarchy property, we even show a slightly stronger statement. To this end, we introduce the notion of *prefix optimal* paths:

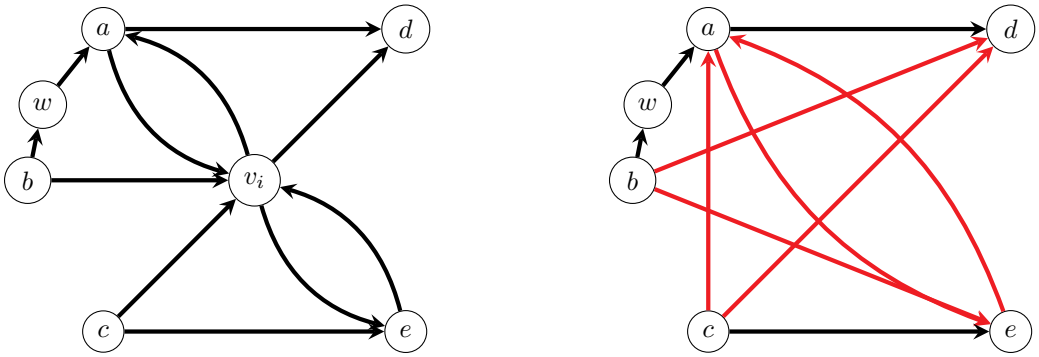


Figure 6.1: The left picture shows the neighbors a, b, c, d, and e of v_i in G_i before contracting v_i. The right picture displays the resulting situation in G_{i+1}: Assume that the ATFs on the edges are chosen such that the fastest paths at some start time in G_i from a to e, from e to a, from b to d, from b to e, from c to a, and from c to d go via v_i. We therefore add the red shortcut edges (a, e), (e, a), (b, d), (b, e), (c, a), and (c, d), respectively. Note that while there is a b-a-path in G_i via v_i, we do not add a shortcut edge (b, a) if the path via w is always faster. Similarly, we do not have to add a shortcut edge from a to d or from c to e. It may happen that the fastest b-d path for some start time is via w and a, but as long as there also is a start time where the fastest b-d-path in G_i is via v_i, we need to add the shortcut edge (b, d).

Definition 6.7 (Prefix optimal paths, [BGSV13])**.** *Given a graph $G = (V, E)$ equipped with ATFs f_e for all $e \in E$, an s-t-path $s := v_0, e_1, v_1, e_2, v_2, \dots, v_{k-1}, e_k, v_k =: t$ (for $s, t, v_i \in V$,$e_i \in E$, $i = 0, \dots k$) is at start time $x \in \mathbb{R}$ prefix optimal if for all $i \in \{0, \dots, k\}$, the path induces an at start time x optimal s-v_i-path $v_0, e_1, \dots, e_i, v_i$ for all $i \in \{0, \dots, k\}$.*

When working with constant travel times, all optimal paths are also prefix optimal. However, this is not necessarily the case with time-dependent travel times: An easy counter-example is an s-t-path P with an early start time where the last edge of P can only be used from some late point in time on. Then there might be many paths with different travel times from s to the vertex that precedes t in P, but after traversing the last edge of P, they all arrive at t at the same time.

Note that if t is reachable from s at start time x, then there also is an at start time x prefix optimal s-t-path: The s-t-path found by a Dijkstra search starting at s at start time x using constant arrival times as vertex labels is prefix optimal.

The following proposition and its proof are adapted from [Rat21] and [BGSV13]:

Proposition 6.8. *The graph $G' = (V, E')$ together with the vertex ordering $\prec$ and the ATFs f_e for each $e \in E'$ constructed as described above is a Contraction Hierarchy with time-dependent travel times.*

Proof. We will even show that for all $s, t \in V$ and any start time $x \in \mathbb{R}$ where there is an s-t-path at start time x, there also is an at start time x prefix optimal s-t-path which is an up-down path. For the sake of obtaining a contradiction, assume this is not the case. Choose $s, t \in V$ and a start time $x \in \mathbb{R}$ such that t is reachable from s at start time x, but there is no at start time x prefix optimal s-t-path that is an up-down path.

We call an internal vertex v in a path P *locally unimportant* if the edge in P that enters v is a down-edge and the edge leaving v in P is an up-edge. Hence any at start time x prefix

optimal s-t-path contains at least one locally unimportant vertex. For a path P, let $v_{\min}(P)$ be the locally unimportant vertex that is minimal w.r.t. $\prec$.

Among all at start time x prefix optimal s-t-paths, let P be the one such that $v := v_{\min}(P)$ is maximal w.r.t. $\prec$. Let $e_{\text{in}} := (u, v)$ and $e_{\text{out}} := (v, w)$ be the edge entering resp. leaving v in P. Define x' to be the time at which we reach u when traversing P starting in s at time x. Consider the step in the construction of G' from G where we contracted v. Choose $i \in \{1, \dots, |V|\}$ such that G_{i+1} results from G_i by contracting v (i.e. $v = v_i$ if $V = \{v_1, \dots, v_n\}$ with $v_1 \prec \cdots \prec v_n$). Since we have $u, w \succ v$, we know that $u, w \in V(G_i)$.

If $u, e_{\text{in}}, v, e_{\text{out}}, w$ is faster than any other u-w-path in G_i not traversing v_i at start time x', we add a shortcut edge from u to w when contracting v. But then replacing the edges e_{in} and e_{out} in P by this shortcut edge (u, w), we obtain a path P' in G' with the same arrival time for start time x as P. Moreover, P' either has no locally unimportant vertex (and thus is an at start time x prefix optimal s-t-path) or $v_{\min}(P') \succ v$ (which contradicts the choice of P).

If $u, e_{\text{in}}, v, e_{\text{out}}, w$ is not faster than any other u-w-path in G_i not traversing v_i at start time x', we take a u-w-path $P_{u,w}$ in G_i that does not traverse v_i and is at start time x' at least as fast as $u, e_{\text{in}}, v, e_{\text{out}}, w$. We replace the edges e_{in} and e_{out} in P by $P_{u,w}$ to obtain P'. But then a prefix of P' is an at start time x faster s-w-path than the prefix of P, contradicting the prefix optimality of P.

This proves that our assumption must be wrong and, in particular, G' is a contraction hierarchy. □

During the construction of G' it is actually not necessary to contract vertices in exactly the given order: When constructing G_{i+1} it is only relevant to contract a vertex $v \in V(G_i)$ such that all neighbors of v in G_i are more important than v. In practice this allows us to contract multiple vertices simultaneously. Hence the construction of the Contraction Hierarchy parallelizes well and multi-threading gives good runtime improvements.

Moreover, we do not even have to know the total order $\prec$ upfront; it suffices to know which vertices are locally least important in each step. [BGSV13] propose a heuristic that computes the vertex ordering while constructing the Contraction Hierarchy: We maintain a certain *importance score* for each $v \in V$ and always contract the vertex with the lowest score. Contracting a vertex can change the score of its neighbors, but the scores of all other vertices remain unaffected. Therefore we always contract a set S of vertices in parallel where S is chosen such that

(i) each $v \in S$ is a vertex with the lowest importance score in $N(v) \cup \{v\}$

(ii) for any $v, w \in S$ there is no v-w-path or w-v-path consisting of at most two edges.

We call a set S of simultaneously contracted vertices a *level* in our Contraction Hierarchy. The vertex ordering is then defined by $v \prec w$ if w was contracted after v; vertices that were contracted at once can be ordered arbitrarily. No matter what vertex ordering $\prec$ we choose, the result will always be a Contraction Hierarchy. But there are certain criteria that a Contraction Hierarchy should ideally meet: On the one hand, we want to minimize the number of inserted shortcuts in order to keep the Contraction Hierarchy small. On the other hand, we want the fastest path searches in the resulting Contraction Hierarchy to be fast. As we only need to look for up-down paths, a path search for a fastest s-t-path consists of the forward search of an up-path starting in s and the backward search of a down-path ending in t. This incentivizes us to minimize the number of levels since it is an upper bound for the depth of the search spaces for these searches. Naturally, these two objectives can conflict with each other, see Fig. 6.2 and [Hei24]. It is therefore important to find a good trade-off.

No non-trivial theoretical guarantees about the resulting number of levels or the number of inserted shortcuts are known for any specific importance score function. [BGSV13] suggest a

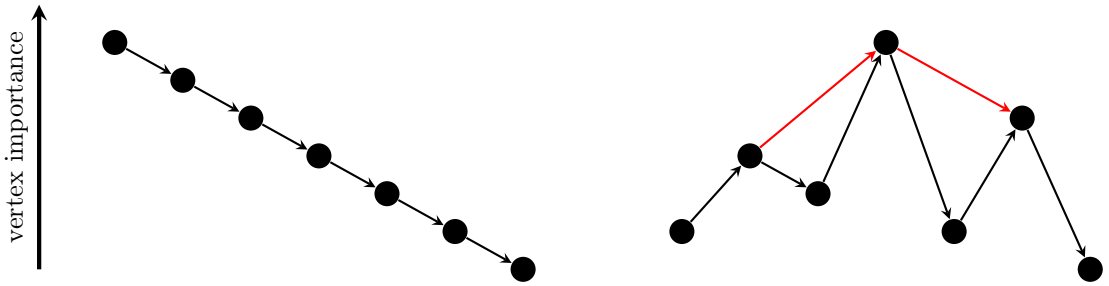


Figure 6.2: There are different possibilities to construct a Contraction Hierarchy for a path on n vertices $v_1, \dots, v_n$: If we choose the vertex importances decreasing (or increasing) along the path (see left picture), no shortcuts have to be added. The depth of a search space however can be $\Theta(n)$. In the right picture, we assign vertex importances such that for all $i \in \{0, \dots, \log n\}$, the vertex importances among $v_{k \cdot 2^i}$ alternate between less important and more important (for $1 \leq k \leq n/2^i$). We have to add $\Theta(n)$ shortcuts (red edges), but obtain search spaces with depth $\mathcal{O}(\log n)$.

heuristic that has proven itself in practice: They define the importance score $c_i(v)$ of a vertex $v \in V(G_i)$ as

$$c_i(v) := 2 \cdot \text{Edges}_i(v) + \text{Depth}_i(v) + \text{Unpack}_i(v) + 2 \cdot \text{Complex}_i(v). \tag{6.1}$$

The different summands are defined as follows:

- Denote the set of shortcut edges that would need to be inserted when contracting v in G_i by $\text{in}_i(v)$. Then $\text{Edges}_i(v) := \frac{|\text{in}_i(v)|}{\max\{1, |\delta_{G_i}(v)|\}}$.
- $\text{Depth}_i(v)$ is defined recursively: In the beginning we set $\text{Depth}_1(v) := 1$ for all $v \in V(G)$. Let u be the vertex contracted when constructing G_{i+1} from G_i. Then $\text{Depth}_{i+1}(v) = \max\{\text{Depth}_i(v), \text{Depth}_i(u) + 1\}$ for neighbors v of u in G_i and $\text{Depth}_{i+1}(v) = \text{Depth}_i(v)$ for all other vertices v.
- For an edge e, let $\pi(e)$ be the length of the longest path represented by e: For an original edge $e \in E(G)$, we have $\pi(e) = 1$; whenever we build a shortcut edge $e' = (u, w)$ from $e_1 = (u, v)$ and $e_2 = (v, w)$, we get $\pi(e') = \pi(e_1) + \pi(e_2)$. If there already existed an edge $e = (u, w)$, we merge it with the new shortcut to an edge e' with $\pi(e') = \max(\pi(e), \pi(e_1) + \pi(e_2))$. We define $\text{Unpack}_i(v) := \frac{\sum_{e \in \text{in}_i(v)} \pi(e)}{\max\{1, \sum_{e \in \delta_{G_i}(v)\pi(e)}\}}$.
- For an edge e, denote its arrival time function by f_e. Recall that $|f_e|$ is the complexity of f_e, i.e., the number of breakpoints of f_e. Then $\text{Complex}_i(v) := \frac{\sum_{e \in \text{in}_i(v)} |f_e|}{\max\{1, \sum_{e \in \delta_{G_i}} |f_e|\}}$.

The summands $\text{Edges}_i(v)$, $\text{Unpack}_i(v)$, and $\text{Complex}_i(v)$ incentivize us to contract a vertex v first for which the newly introduced shortcuts add the least complexity (in form of number of edges, length of the path that the shortcut stands for, or complexity of the involved ATFs) in comparison to the edges that are removed when contracting v. Note that we approximate the ATFs during the construction of the Contraction Hierarchy (see Section 6.3), but using $\text{Complex}_i(v)$ makes sense nevertheless to speed up the Contraction Hierarchy computation, and to reduce the resulting number of levels as well as the number of inserted shortcuts (see the practical results in Chapter 8).

The term $\max_{v \in V} \text{Depth}_n(v)$ is a better upper bound on the depth of an up- or down-search space than the number of levels as discussed above. As $\max_{v \in V} \text{Depth}_i(v)$ only increases

whenever we contract a vertex u that previously attained this maximum, the summand $\text{Depth}_i(v)$ in the importance score $c_i(v)$ helps minimizing the depth of search spaces in the resulting Contraction Hierarchy.

Experiments where we used information about the road segments incident to v as a hint for the importance of v (e.g. letting v be more important if it corresponds to a highway crossing) led to worse results (see [Rat21]).

6.3 Approximating Contraction Hierarchies

In Section 5.4, we have seen that composing ATFs and taking minima leads to ATFs with more breakpoints. As a result, an ATF corresponding to a shortcut edge can be significantly more complex than the ATFs of the original edge. Together with the fact that between a third and a half of the edges of the resulting Contraction Hierarchy are shortcut edges (cf. Chapter 8), we end up with Contraction Hierarchies that occupy a huge amount of disk space. Moreover, shortest path queries in these Contraction Hierarchies take longer if the occurring ATFs have many breakpoints. This stands in the way of our goal to enable fast path searches.

Therefore, in order to be able to employ Contraction Hierarchies in practice, we approximate the ATFs corresponding to shortcut edges (see Section 5.4.2). In connection with the construction of Contraction Hierarchies, there are two main approaches for approximating ATFs: On the one hand, we can completely compute the Contraction Hierarchy, only approximating all ATFs in the end. This approach provides the most precise results because we work with exact ATFs throughout the construction of all shortcut ATFs while also ensuring manageable complexity of the resulting Contraction Hierarchy. [Rat21] already integrated this approximation step in his first implementation of Contraction Hierarchies for BonnTour.

On the other hand, we can directly approximate the ATF with which we equip a newly constructed shortcut edge as soon as this edge is built. This has the disadvantage that whenever we compose this approximated ATF further in later shortcut edges, we propagate the rounding error. In his Master's thesis, Rathke ([Rat21]) discussed several strategies for approximating ATFs already during the construction of the Contraction Hierarchy, but came to the conclusion that the loss in precision is not worth it since building the Contraction Hierarchy usually is not runtime critical. However, not approximating occasionally while building the Contraction Hierarchy is often practically infeasible: Working with increasingly more complex ATFs makes the running time and the memory consumption of the construction process skyrocket. Even though we often only compute a Contraction Hierarchy once in advance and then use it for many BonnTour runs, reading it in from disk each time, for use in practice it is still necessary to be able to create a new Contraction Hierarchy within a few hours. This is needed, for example, when map data in the input changes and we want to incorporate these changes as soon as possible in order to always reflect reality as accurately as possible. Moreover, while [Rat21] only computed Contraction Hierarchies covering the area of a large city or at most a region of a country, we are now able to build Contraction Hierarchies for areas as large as Europe, see Chapter 8.

This section will start with an overview over the general rounding strategy in BonnTour (Section 6.3.1). In Section 6.3.2, we propose a novel approach for approximating Contraction Hierarchies during their construction, and discuss how to control the error that is propagated through many rounding steps.

6.3.1 The general rounding strategy in BonnTour

Since the running time of many operations that we perform during a BonnTour run depends on the number of breakpoints of the involved arrival time functions (see Section 5.4.2), it is

unavoidable to approximate them in order to reduce their complexity. We define a global relative allowed approximation error $\overline{\rho} > 0$ that is given as a parameter in the input which allows BonnTour users to choose the trade-off between running time and precision effort themselves. For simplicity we also provide pre-defined default values; we usually work with a global relative allowed approximation error of $\overline{\rho} = 1\%$.

We process ATFs in multiple steps throughout our algorithm and apply approximation in several places as follows:

Definition 6.9 (Relative ATF approximation error)**.** *Given $\rho \in \mathbb{R}_{>0}$ and an ATF f, an ATF f' is an approximation of f with* relative error ρ *if* $\mathrm{dom}(f') = \mathrm{dom}(f)$, $f(x) \leq f'(x)$ *and* $f'(x) - x \leq (1+\rho) \cdot (f(x) - x)$.

In practice we obtain such an approximation f' from f by applying Theorem 5.4 to compute f' such that $\mathrm{dom}(f') = \mathrm{dom}(f)$ and $f(x) \leq f'(x) \leq f(x) + \varepsilon$ for all $x \in \mathrm{dom}(f)$ where we set $\varepsilon = \rho \cdot \min_{x \in \mathrm{dom}(f)}(f(x) - x)$. This is a tighter approximation error than what would be allowed because we round the whole function f with absolute error $\rho \cdot \min_{x \in \mathrm{dom}(f)}(f(x) - x)$ instead of using a point-wise error of $\rho \cdot (f(x) - x)$.

Let us first describe the rounding procedure that we apply if we do not approximate the ATFs during the construction of the Contraction Hierarchies since the latter is an even more complex issue and is addressed in Section 6.3.2.

(i) After having read in the road map data, including time-dependent travel times, we approximate the resulting ATFs associated with each edge with an allowed relative error of ρ_1.

(ii) After having built the Contraction Hierarchy, for each edge of the resulting graph (i.e., including shortcut and original edges) we approximate the ATF with which the edge is equipped, using a relative error of ρ_2.

(iii) We then construct distance matrices containing point-to-point connection information for each pair of points of interest (i.e., addresses where vehicles start or end, or where shipments have to be picked up or delivered). To this end, we perform shortest path searches and get an ATF for every point-to-point connection. These ATFs are then approximated with an allowed relative error of ρ_3.

We choose ρ_1, ρ_2, and ρ_3 such that $\rho_1, \rho_2, \rho_3 > 0$ and $\rho_1 + \rho_2 + \rho_3 = \overline{\rho}$. Let $f'_{s,t}$ be the approximated ATF in the resulting distance matrix for going from s to t for two points of interest $s, t \in V$, and let $f_{s,t}$ be the exact ATF for going from s to t. Note that $f'_{s,t}$ is not necessarily an approximation of $f_{s,t}$ with relative error $\overline{\rho}$. This is due to two sources of inaccuracy:

Firstly, given an ATF f and some relative approximation errors $a_1, \ldots, a_k > 0$ for some $k \in \mathbb{N}$, approximating f successively k times with relative error $a_1, \ldots, a_k$, respectively, can end up in a total relative error that is larger than $a := \sum_{i=1}^{k} a_i$. We can only bound the resulting total relative error by $\prod_{i=1}^{k}(1 + a_i) - 1$. Expanding this product indeed gives a bound larger than a (if $k > 1$). But by the inequality of the arithmetic and geometric mean, we have

$$\prod_{i=1}^{k}(1 + a_i) \leq \left(\frac{\sum_{i=1}^{k}(1 + a_i)}{k} \right)^k = \left(1 + \frac{a}{k}\right)^k \leq e^a.$$

For $a = 1\%$, the value of e^a is less than 1.01005. In practice, ending up with a relative error of 1.005% instead of 1% is not a problem at all.

Secondly, we compose ATFs in between the different approximation steps. Given two ATFs f and g, let f' and g' be approximations of f and g, respectively, with relative error $\rho > 0$. Then

the relative error of $g' \circ f'$ with respect to $g \circ f$ can be unbounded: Problems arise when g has a large slope because then small changes in the arrival time modeled by f lead to large changes after also traversing g. For example, choose

$$g\colon \mathrm{dom}(g) \to \mathbb{R}, x \mapsto \begin{cases} 0 & x \leq 0 \\ m \cdot x & x > 0 \end{cases}$$

for some $m \in \mathbb{R}_{>0}$ and let $f(x) := \max\{0, x+1\}$. Then $f'(x) := \max\{0, x+1+\rho\}$ is an approximation of f with relative error ρ. We have $g(f(-1)) = 0$, but $g(f'(-1)) = m \cdot \rho$. The relative error of $g \circ f'$ with respect to $g \circ f$ at the point -1 is therefore

$$\frac{g(f'(-1)) - (-1)}{g(f(-1)) - (-1)} = m \cdot \rho + 1$$

which is unbounded since we can choose m arbitrarily large. Thus, if we do not want to take the slope of the involved ATFs into account when giving error bounds, we cannot give any guarantees in terms of Definition 6.9.

Unfortunately, high slopes can appear in real world instances, for example when roads are blocked for a few hours per day. Nevertheless, the expected arrival times computed by BonnTour always look reasonable when comparing it to as-driven data although we use approximation and propagate errors when composing ATFs. One possible explanation is that high slopes do not occur very often. Moreover, arriving later than expected also happens in practice, and is not only a consequence of rounding errors. Hence approximation leading to longer travel times does not only need to be seen as a disadvantage, but also makes the computed tours more robust: If a small approximation error on one edge results in a much higher travel time on the next edge, it may in any case be a bad idea to use the first edge so shortly before travel times increase greatly on the second edge.

In his Master's thesis, Rathke provides a thorough survey over different approximation strategies and the guarantees one can give ([Rat21]), none of them being satisfying enough that he implemented them for approximating the ATFs arising during the construction of Contraction Hierarchies. We therefore introduce a novel approximation strategy in the next subsection. This approximation will have a budget of ρ_4 and will be applied between Step (i) and Step (ii). We will choose the budgets such that $\rho_1 + \rho_2 + \rho_3 + \rho_4 = \overline{\rho}$, the maximum allowed budget given as input parameter.

6.3.2 Controlling the approximation error during the construction of Contraction Hierarchies

Every time that we add a shortcut during the construction of a Contraction Hierarchy, we compose ATFs. When an edge from u to v represents several shortcuts, e.g. $u - w_1 - v$ is faster in the morning and $u - w_2 - v$ is faster in the afternoon, we also compute pointwise minima of ATFs. This leads to ATFs with hundreds of breakpoints during the construction process (see Chapter 8) which slows down building the Contraction Hierarchy significantly. We therefore apply the following approximation strategy: For a given relative error $\rho > 0$, we grant each ATF f an allowed approximation error of $\rho \cdot \min_{x \in \mathrm{dom}(f)}(f(x) - x)$. For each ATF, we store how much of this budget has already been used. When composing two approximated ATFs $\overline{f}$ and $\overline{g}$ (where f and g are the respective un-approximated ATFs) with already used budgets $a \leq \rho \cdot \min_{x \in \mathrm{dom}(f)}(f(x) - x)$ and $b \leq \rho \cdot \min_{x \in \mathrm{dom}(g)}(g(x) - x)$, respectively, we approximate $\overline{g} \circ \overline{f}$ with an allowed error c such that $a + b + c \leq \rho \cdot \min_{x \in \mathrm{dom}(g \circ f)}(g(f(x)) - x)$. Note that there

is such $c \geq 0$ since $\min_{x \in \mathrm{dom}(g \circ f)}(g(f(x)) - x) \geq \min_{x \in \mathrm{dom}(f)}(f(x) - x) + \min_{x \in \mathrm{dom}(g)}(g(x) - x)$. We store $a + b + c$ as already used budget for the resulting approximated function.

This approximation strategy leads to an error guarantee depending on the maximum slope of the appearing ATFs.

Definition 6.10 (Maximum slope)**.** *For an ATF f, denote by $\Delta(f)$ its maximum slope, i.e.*

$$\Delta(f) := \max_{x,y \in \mathrm{dom}(f), x \neq y} \frac{f(x) - f(y)}{x - y}.$$

Note that $\Delta(f)$ is always attained on an affine segment of f.

Remark 6.11. For two ATFs f and g, we have $\Delta(g \circ f) \leq \Delta(f) \cdot \Delta(g)$ since $g \circ f$ consists of affine segments whose slopes are products of slopes of affine segments of f and affine segments of g. Moreover, for an approximation $\overline{f}$ of f, we can assume that $\Delta(\overline{f}) \leq \Delta(f)$: If there is a segment of $\overline{f}$ with higher slope than $\Delta(f)$, we can just tilt this segment, keeping its left endpoint ℓ, such that its slope becomes $\Delta(f)$. As new right endpoint we take the point of intersection of the ray starting in ℓ with slope $\Delta(f)$ with the original function $\overline{f}$. This way we end up with a function whose maximum slope as well as the average error is smaller while keeping the error guarantee and not increasing the number of breakpoints.

Definition 6.12 (Minimum travel time)**.** *For an ATF f, denote its* minimum travel time *by $\delta(f)$, i.e.*

$$\delta(f) := \min_{x \in \mathrm{dom}(f)} (f(x) - x).$$

Theorem 6.13. *Let $f_1, \ldots, f_n$ be ATFs for some $n \in \mathbb{N}$. Define $f := f_n \circ \cdots \circ f_1$. Recursively computing an approximation of f by performing the compositions in any order and approximating after each composition with an error that, together with the already used budgets, stays within the allowed budget, results in an ATF h with*

$$h(x) \leq f(x) + \rho \cdot \delta(f) \cdot \prod_{i=2}^{n} \max\left(\Delta(f_i), 1\right).$$

Proof. We prove this by induction on n. For $n = 1$ the statement holds, no matter with which absolute error $\varepsilon \leq \rho \cdot \delta(f)$ we approximate f_1.

Now let $f_1, \ldots, f_n$ be ATFs and $k \in \{1, \ldots, n-1\}$. Define $g_1 := f_k \circ \cdots \circ f_1$ and $g_2 := f_n \circ \cdots \circ f_{k+1}$. By our induction hypothesis, we have an ATF $\overline{g_1}$ that is an approximation of g_1 with

$$\overline{g_1}(x) \leq g_1(x) + a \cdot \prod_{i=2}^{k} \max\left(\Delta(f_i), 1\right)$$

for some $a \leq \rho \cdot \delta(g_1)$. Analogously, let $\overline{g_2}$ be an ATF that is an approximation of g_2 with

$$\overline{g_2}(x) \leq g_2(x) + b \cdot \prod_{i=k+2}^{n} \max\left(\Delta(f_i), 1\right)$$

for some $b \leq \rho \cdot \delta(g_2)$. We can assume that the slope of $\overline{g_2}$ is at most $\prod_{i=k+1}^{n} \Delta(f_i)$ (see Remark 6.11). Thus

$$\begin{aligned}
\overline{g_2}(\overline{g_1}(x)) &\leq g_2(g_1(x)) + \prod_{i=k+1}^{n} \max(\Delta(f_i), 1) \cdot a \cdot \prod_{i=2}^{k} \max(\Delta(f_i), 1) + b \cdot \prod_{i=k+2}^{n} \max(\Delta(f_i), 1) \\
&\leq g_2(g_1(x)) + (a + b) \cdot \prod_{i=2}^{n} \max(\Delta(f_i), 1).
\end{aligned}$$

Note that $g_2 \circ g_1 = f$. Approximating $\overline{g_2} \circ \overline{g_1}$ with an absolute error c such that $a+b+c \leq \rho \cdot \delta(f)$ therefore results in an ATF h with

$$h(x) \leq f(x) + \rho \cdot \delta(f) \cdot \prod_{i=2}^{n} \max(\Delta(f_i)). \qquad \square$$

Note that the allowed approximation budget always depends on the minimum travel time of the respective un-approximated functions. In practice, we do not have this information anymore because we do not store and compose the ATFs without approximating them in between as this would undermine our effort to eliminate complexity of our functions. We therefore always allow an error budget with respect to the minimum travel time of the composition of the already approximated ATFs. In a setting like the proof of Theorem 6.13, we would approximate $\overline{g_2} \circ \overline{g_1}$ with an absolute error c such that $a + b + c \leq \rho \cdot \delta(\overline{g_2} \circ \overline{g_1})$. Of course, we cannot obtain the guarantee stated in Theorem 6.13 this way. However, as we will see in Chapter 8, this procedure yields sufficiently accurate results in practice.

Since there are more possibilities to decrease the number of breakpoints when approximating with a larger allowed absolute error, we only approximate when there is enough allowed budget left. In our application, we always choose the absolute allowed errors for approximating during the construction of the Contraction Hierarchy as the unused budget rounded down to integer multiples of $\rho \cdot 5$ minutes.

6.4 Customizing the Contraction Hierarchy – adding new vertices

The big advantage of Contraction Hierarchies is that once we have computed one for a given area, we can re-use it every time that we want to run BonnTour for an instance within this area. Even though we do not know in advance for which points of interest we perform fastest path searches, the Contraction Hierarchy enables us to make fast path and ATF queries for an arbitrary pair of vertices. Once we are given the addresses of a specific BonnTour instance we hence need to assign vertices of the Contraction Hierarchy to them. We call this procedure *Address Mapping*. Chapter 7 discusses how to find such a vertex that fits best both in terms of geographic proximity and of reachability in the street network. This section focuses on the implications for our Contraction Hierarchy that follow from Address Mapping:

There are two different approaches on how to find a good vertex for a given address. We compare both of them in practical experiments in Chapter 8. The first possibility is to map an address to a vertex that already exists in the Contraction Hierarchy. This is in so far the easier approach as it allows us to use the Contraction Hierarchy without modification. However, depending on how dense the vertices are spread across the street network covered by the Contraction Hierarchy, we might introduce a lot of inaccuracy here. If we are unlucky, the geographically nearest vertex can be outside the desired address mapping radius, i.e., the maximum distance we allow the mapped point to be away from the original coordinates. That can happen even if there actually is a nearby road, but the edge representing it is too long. In order to prevent this, we can subdivide long edges, introducing some vertices of degree 2, before computing the Contraction Hierarchy. However, we thereby increase the size of the resulting Contraction Hierarchy while we still cannot guarantee maximum accuracy.

The other approach is to choose the edge with the least distance to the given address and insert a new vertex subdividing this edge. While geographical positions in the input are given with longitude and latitude, we convert them internally using the UTM system such that we can

work with Cartesian coordinates. An original edge in the road graph, i.e., an edge corresponding to a road segment and not a Contraction Hierarchy shortcut, is interpreted to represent the straight line connecting the positions of the two vertices incident to the edge. If we want to insert a vertex w subdividing an edge (u, v) for a given address, we add a vertex at the point on (u, v) that is closest to the address. By removing the edge (u, v) and inserting (u, w) and (w, v), we integrate w into our road graph. The edges (u, w) and (w, v) are equipped with ATFs $f_{(u,w)}$ and $f_{(w,v)}$ such that $f_{(w,v)} \circ f_{(u,w)} = f_{(u,v)}$ and the travel times are proportional to the lengths of the respective road segments. This can of course only be done after already having computed the Contraction Hierarchy – simply because we do not know the addresses in advance. Therefore, we have to pay attention to certain details: First of all, the edge that we subdivide should be an original edge corresponding to some road, not a shortcut edge. Furthermore, we need to subdivide all edges that correspond to the chosen road segment. Note that there might be more than one such edge, e.g. because the road can be traversed in both directions or because we cloned the edge to model prohibited maneuvers (see Fig. 7.1). Lastly, the newly added vertex needs to be assigned a vertex importance, and we have to ensure that the Contraction Hierarchy property still holds. We will see that in fact, the strict Contraction Hierarchy property does not hold anymore, but it stays true for all relevant paths.

How to find a good edge that is close to the address of interest and not a shortcut edge is the topic of Section 7.1. Collecting all relevant edges belonging to the same street segment can easily be done with our internal data structures. In this section we therefore concentrate on re-establishing a certain type of Contraction Hierarchy property after having added new vertices.

6.4.1 Subdividing edges

We first observe that there is a reasonable way to equip subdivided edges with ATFs:

Proposition 6.14. *Given $\lambda \in [0, 1]$ and an ATF f with breakpoints $(t_1, f(t_1)), \dots, (t_b, f(t_b))$ such that $t_1 < \dots < t_b$, there is a unique ATF f_1 with $\mathrm{dom}(f_1) = \mathrm{dom}(f)$ such that $f_1(x) - x = \lambda \cdot (f(x) - x)$ for $x \in [t_1, t_b]$ and $f_1(x) = f_1(t_1)$ for $x \le t_1$.*

Furthermore, there is a unique ATF f_2 with $\mathrm{dom}(f_2) = (-\infty, f_1(t_b)]$ such that $f_2 \circ f_1 = f$ and $f_2(x) = f(t_1)$ for $x < f_1(t_1)$. Both f_1 and f_2 have at most as many breakpoints as f.

Proof. The function f_1 is uniquely defined by

$$f_1(x) = \begin{cases} \lambda \cdot f(t_1) + (1 - \lambda) \cdot t_1 & x \le t_1 \\ \lambda \cdot f(x) + (1 - \lambda) \cdot x & t_1 \le x \le t_b \end{cases}.$$

Then f_1 is a piecewise linear function given by the breakpoints

$$(t_1, (1 - \lambda) \cdot t_1 + \lambda \cdot f(t_1)), \dots, (t_b, (1 - \lambda) \cdot t_b + \lambda \cdot f(t_b)).$$

Moreover, f_1 is indeed an ATF since it is continuous, non-decreasing, initially constant, and $f_1(x) \ge (1 - \lambda) \cdot x + \lambda \cdot f(x) \ge \min(x, f(x)) \ge x$ as f is an ATF.

For f_2, note that for two consecutive breakpoints of f at t_i and t_{i+1} (for some index $i \in \{1, \dots, b - 1\}$), the function f between these breakpoints is a linear segment connecting $f(t_i)$ to $f(t_{i+1})$, and f_1 is also linear in this interval, connecting $\lambda \cdot f(t_i) + (1 - \lambda) \cdot t_i$ and $\lambda \cdot f(t_{i+1}) + (1 - \lambda) \cdot t_{i+1}$. In order for $f_2 \circ f_1 = f$ to hold on the interval $[t_i, t_{i+1}]$, f_2 on the interval $[f_1(t_i), f_1(t_{i+1})]$ therefore must be the linear segment connecting $(f_1(t_i), f(t_i))$ to $(f_1(t_{i+1}), f(t_{i+1}))$. Since we demand that $f_2(x) = f_1(t_1)$ for $x < f_1(t_1)$, f_2 has an initial constant segment in the interval $(-\infty, f_1(t_1)]$. Therefore f_2 is uniquely defined by the breakpoints $((1 - \lambda) \cdot t_1 + \lambda \cdot f(t_1), f(t_1)), \dots, ((1 - \lambda) \cdot t_b + \lambda \cdot f(t_b), f(t_b))$.

Let us first check that f_2 is well-defined, i.e., no two breakpoints have the same x-coordinate, but different y-coordinates: If $t_i < t_j$, we have $f(t_i) \leq f(t_j)$. Thus the only possibility where $(1-\lambda)t_i + \lambda f(t_i) = (1-\lambda)t_j + \lambda f(t_j)$ is that $\lambda = 1$ and $f(t_i) = f(t_j)$.

Moreover, the breakpoints actually define an ATF because f_2 is continuous, non-decreasing and initially constant; furthermore, $(1-\lambda) \cdot x + \lambda \cdot f(x) \leq \max(x, f(x)) \leq f(x)$ for $x \in \mathrm{dom}(f)$ and hence $f_2(t) \geq t$ holds if t is the x-coordinate of a breakpoint of f_2 and therefore $f_2(x) \geq x$ holds for all $x \in \mathrm{dom}(f_2)$. □

Definition 6.15 (λ-subdivision ATFs)**.** *Given $\lambda \in [0,1]$ and an ATF f, the pair (f_1, f_2) of ATFs as constructed in Proposition 6.14 is called the* λ-subdivision ATFs *of f.*

Now we can formally define the subdivision procedure:

Definition 6.16 (Subdividing an edge)**.** *Let $G = (V, E)$ be a graph where each edge $e \in E$ is equipped with an ATF f_e. Given a vertex w, an edge $\overline{e} = (u, v) \in E$, and $\lambda \in [0,1]$,* subdividing $\overline{e}$ by w with factor λ *results in a graph $G' = (V', E')$ where*

- $V' := V \cup \{w\}$ *and*
- $E' := E \cup \{(u, w), (w, v)\}$.

We equip (u, w) and (w, v) with $f_{(u,w)}$ and $f_{(w,v)}$, respectively, such that $(f_{(u,w)}, f_{(w,v)})$ is the pair of λ-subdivision ATFs of $f_{\overline{e}}$.

Note that it is not feasible to require $w \notin V$ because we might want to subdivide several edges by the same vertex when we have multiple edges representing the same original road segment. If we were not working with Contraction Hierarchies, keeping the edge $\overline{e}$ would not be necessary as the same travel time information is stored in the new path uwv. However, as we will see later, $\overline{e}$ will serve as a shortcut edge.

As we explain in Section 7.1, we find the most suitable road segment r for a given address a, then collect the sets $\overrightarrow{E_r}, \overleftarrow{E_r} \subseteq E$ that correspond to r in forward resp. backward direction, and compute the fraction λ at which a subdivides r. Then we insert a new vertex w subdividing all edges $e \in \overrightarrow{E_r}$ with factor λ and all edges $e \in \overleftarrow{E_r}$ with factor $(1-\lambda)$.

Let us briefly consider what happens when we insert another vertex w_2 after already having subdivided some edges by a first vertex w_1. Assume we subdivided edges in $\overrightarrow{E_1} \subseteq E$ with factor λ_1 and edges in $\overleftarrow{E_1} \subseteq E$ with factor $(1-\lambda)$ when inserting w_1, resulting in a graph G_1. We might have decided upfront that we would like to subdivide the edge sets $\overrightarrow{E_2} \subseteq E$ and $\overleftarrow{E_2} \subseteq E$ by w_2 with factors λ_2 and $(1-\lambda_2)$, respectively. But now it can happen that $(\overrightarrow{E_1} \cup \overleftarrow{E_1}) \cap (\overrightarrow{E_2} \cup \overleftarrow{E_2}) \neq \emptyset$. In practice this is the case when the addresses for which we want to insert w_1 and w_2 lie on the same road segment. W.l.o.g., we then already have $\overrightarrow{E_1} = \overrightarrow{E_2}$ and $\overleftarrow{E_1} = \overleftarrow{E_2}$. Consider an edge $e \in E$ that has been subdivided by v_1 and that we also want to subdivide by w_2. Since we keep the original edge when subdividing it, e still exists in G_1. But e is now not a reasonable choice anymore for w_2 because w_1 and w_2 should be adjacent in this case. Instead, we subdivide one of the edges resulting from subdividing e by w_1 with factor λ_1, see Fig. 6.3:

Let e', e'' be the edges that we added when inserting w_1 into e, and let λ_2 be the factor with which we planned to subdivide e by w_2. If $\lambda_2 = \lambda_1$, we do not subdivide further and identify w_2 with w_1.

In the case $\lambda_2 < \lambda_1$, we subdivide e' with factor $\frac{\lambda_2}{\lambda_1}$, and if $\lambda_2 > \lambda_1$, we subdivide e'' with factor $\frac{\lambda_2 - \lambda_1}{1-\lambda_1}$. The choice of these factors is indeed reasonable and we get the same resulting ATFs, no matter in which order we subdivide the edge:

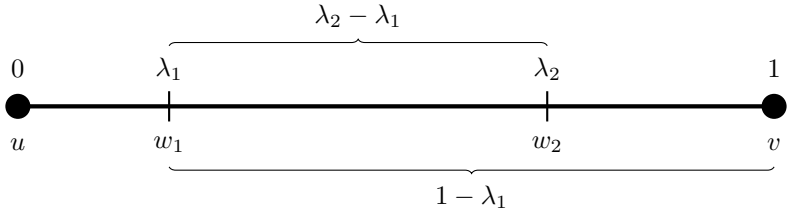


Figure 6.3: We subdivide the edge (u, v) by the vertices w_1 and w_2 with factors λ_1 and λ_2, respectively. Assume for simplicity that the length of (u, v) is 1. Then the distance from w_1 to v is $1 - \lambda_1$ while the distance from w_1 to w_2 is $\lambda_2 - \lambda_1$, implying that w_2 subdivides (w_1, v) with factor $\frac{\lambda_2 - \lambda_1}{1 - \lambda_1}$. The other way around, w_1 subdivides (u, w_2) with factor $\frac{\lambda_1}{\lambda_2}$.

Lemma 6.17. *Let f be an ATF and $\lambda_1, \lambda_2 \in [0, 1]$ with $\lambda_1 < \lambda_2$. Let (g_1, g_2) be the pair of λ_1-subdivision ATFs of f and (h_1, h_2) be the pair of λ_2-subdivision ATFs of f. Define $\lambda_1' := \frac{\lambda_1}{\lambda_2}$ and $\lambda_2' := \frac{\lambda_2 - \lambda_1}{1 - \lambda_1}$. Denote by (f_2, f_3) the pair of λ_2'-subdivision ATFs of g_2 and by (f_1, f_2') the pair of λ_1'-subdivision ATFs of h_1. Then*

(i) $g_1 = f_1$,

(ii) $f_2 = f_2'$, *and*

(iii) $h_2 = f_3$.

Proof. Note that λ_1' and λ_2' are defined in a way such that $\lambda_1' \cdot \lambda_2 = \lambda_1$ and $\lambda_1 + (1 - \lambda_1) \cdot \lambda_2' = \lambda_2$.

Let $(t_1, f(t_1)), \ldots, (t_b, f(t_b))$ be the breakpoints of f for some $b \in \mathbb{N}$ and $t_1 < \cdots < t_b$. Then for $t \geq t_1$, we have $g_1(t) = \lambda_1 \cdot f(t) + (1 - \lambda_1) \cdot t$ and

$$\begin{aligned} f_1(t) &= \lambda_1' \cdot h_1(t) + (1 - \lambda_1') \cdot t = \lambda_1' \cdot (\lambda_2 \cdot f(t) + (1 - \lambda_2) \cdot t) + (1 - \lambda_1') \cdot t \\ &= \lambda_1' \lambda_2 \cdot f(t) + (1 - \lambda_1' \lambda_2) \cdot t. \end{aligned}$$

For $t < t_1$, we have $g_1(t) = g_1(t_1) = f_1(t_1) = f_1(t)$. This settles $g_1 = f_1$.

For $t \geq t_1$, g_2 maps $g_1(t)$ to $f(t)$. Hence

$$\begin{aligned} f_2(g_1(t)) &= \lambda_2' \cdot f(t) + (1 - \lambda_2') \cdot g_1(t) \\ &= \lambda_2' \cdot f(t) + (1 - \lambda_2') \cdot (\lambda_1 \cdot f(t) + (1 - \lambda_1) \cdot t) \\ &= (\lambda_1 + (1 - \lambda_1) \cdot \lambda_2') \cdot f(t) + (1 - \lambda_2') \cdot (1 - \lambda_1) \cdot t \\ &= \lambda_2 \cdot f(t) + (1 - \lambda_2) \cdot t. \end{aligned}$$

The function f_2' maps $f_1(t)$ to $h_1(t) = \lambda_2 \cdot f(t) + (1 - \lambda_2) \cdot t$. Since $g_1(t) = f_1(t)$, this implies $f_2'(g_1(t)) = f_2'(f_1(t)) = \lambda_2 \cdot f(t) + (1 - \lambda_2) \cdot t$. Therefore we have $f_2(t) = f_2'(t)$ for all t that can be written as $t = f_1(x)$ for some $x \in \mathrm{dom}(f_1)$. This is the case for all $t \in [f_1(t_1), f_1(t_b)]$. But for $t < f_1(t_1)$, we have $f_2(t) = f_2(f_1(t_1)) = f_2'(f_1(t_1)) = f_2'(t)$.

It remains to prove $h_2 = f_3$. Once again, let us first consider the case $t \geq t_1$. Then h_2 maps $h_1(t)$ to $f(t)$. Moreover, f_3 maps $f_2(g_1(t))$ to $f(t)$. We already established $f_2 = f_2'$ and $g_1 = f_1$; thus f_3 maps $f_2'(f_1(t))$ to $f(t)$ and we know $f_2' \circ f_1 = h_1$. Hence we have $h_2(t) = f_3(t)$ for all t that can be written as $t = h_1(x)$ for some $x \in \mathrm{dom}(h_1)$. This is the case for all $t \in [h_1(t_1), h_1(t_b)]$. For $t < h_1(t_1)$, we have $h_2(t) = h_2(h_1(t_1)) = f_3(h_1(t_1)) = f_3(t)$., □

Definition 6.18. *In the situation from Lemma 6.17, we also call the ATF $f_2 = f_2'$ the λ_1-λ_2-*intermediate ATF of f.

This procedure of subsequent subdivision can of course be extended to inserting an arbitrary number of new vertices:

Definition 6.19 (Subdivision configuration)**.** *Let $G = (V, E)$ be a graph where each edge $e \in E$ is assigned an ATF f_e. Then a* subdivision configuration *is a quadruple $(W, \lambda, \overrightarrow{E}, \overleftarrow{E})$ consisting of*

- *a set of vertices $W = \{w_1, \dots, w_k\}$ with $W \cap V = \emptyset$ for some $k \in \mathbb{N}$,*
- *$\lambda_i \in [0, 1]$ for $1 \leq i \leq k$, and*
- *edge subsets $\overrightarrow{E_i}, \overleftarrow{E_i} \subseteq E$ for $1 \leq i \leq k$*

such that

- *$\overrightarrow{E_i} \cap \overleftarrow{E_i} = \emptyset$ for each $i \in \{1, \dots, k\}$,*
- *if $(\overrightarrow{E_i} \cup \overleftarrow{E_i}) \cap (\overrightarrow{E_j} \cup \overleftarrow{E_j}) \neq \emptyset$, then $\overrightarrow{E_i} = \overrightarrow{E_j}$ and $\overleftarrow{E_i} = \overleftarrow{E_j}$ for $i, j \in \{1, \dots, k\}$, and*
- *for $i, j \in \{1, \dots, k\}$ with $i \neq j$, but $\overrightarrow{E_i} = \overrightarrow{E_j}$ and $\overleftarrow{E_i} = \overleftarrow{E_j}$, we have $\lambda_i \neq \lambda_j$, and*
- *$f_{e_1} = f_{e_2}$ if $e_1, e_2 \in \overrightarrow{E_i}$ or $e_1, e_2 \in \overleftarrow{E_i}$ for some $i \in \{1, \dots, k\}$.*

Definition 6.20. *Let $G = (V, E)$ be a graph with edge ATFs, and $(W, \lambda, \overrightarrow{E}, \overleftarrow{E})$ be a subdivision configuration with $|W| = k$.*

Then inserting $w_1, \dots, w_k$ in this order into G works as follows: For each w_i ($1 \leq i \leq k$), we maintain a list of pairs where each pair consists of an edge that we want to subdivide by w_i and the factor with which we want to subdivide the edge. Initially, we set the list for w_i to contain all pairs (e, λ_i) for all $e \in \overrightarrow{E_i}$ and all pairs $(e, 1 - \lambda_i)$ for $e \in \overleftarrow{E_i}$. Then we iteratively insert the w_i in their order, starting with w_1 and ending with w_k. This insertion is done by subdividing all edges in the list for w_i by w_i with their respective factors. After inserting w_i, we update the lists for the vertices w_j with $j > i$: If a pair (e, λ') appears in the list of w_j, but e was subdivided by w_i with factor λ into the edges e' and e'' in this order, then we remove (e, λ') from the list of w_j and instead add a new pair:

- *If $\lambda' < \lambda$, we add the pair $(e', \frac{\lambda'}{\lambda})$.*
- *If $\lambda' > \lambda$, we add the pair $(e'', \frac{\lambda' - \lambda}{1 - \lambda})$.*

We denote the resulting graph by $G\langle w_1, \dots, w_k \rangle$. We will also sometimes write $G\langle w_1, \dots, w_k \rangle$ without knowing whether $w_i \notin V(G)$. In the case that there is some $w_i \in V(G)$, we define $G\langle w_1, \dots, w_k \rangle := G\langle w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_k \rangle$, where we set $G\langle\rangle := G$.

6.4.2 Shortest paths with subdivided edges

If we have two edges $e_1 = (u_1, v_1), e_2 = (u_2, v_2) \in \overrightarrow{E_r} \cup \overleftarrow{E_r}$, subdividing the edges by w introduces the new u_1-v_2-path $u_1 w v_2$. Especially if there is a start time x for which this new path is faster than the previous best u_1-v_2-path, this is an undesired side-effect of the insertion of w. Note that this can happen in particular when we clone vertices to model prohibited maneuvers making it all the more undesirable to have a new faster u_1-v_2-path. In order to solve this problem, we take advantage of the up-down path search that we use in Contraction Hierarchies: We define the importance of the new vertex w to be lower than all existing importance levels, i.e., $w \prec v$ for all $v \in V$. Then $u_1 w v_2$ is not an up-down-path and we simply do not add a shortcut edge for this path. Hence a path search restricted to up-down paths will not consider this u_1-v_2-path. The resulting graph therefore does not fulfill the Contraction Hierarchy property, but we are anyway only interested in paths that essentially just use the original Contraction Hierarchy, only visiting newly inserted vertices as part of a prefix or suffix of the path:

Let G' be the graph resulting from inserting a vertex w into a Contraction Hierarchy $(G, f, \prec)$, subdividing some edges. We then want to extend the vertex ordering $\prec$ to w such that the following holds:

- For $u, v \in V(G)$ and a start time x for which a u-v-path exists in G, the fastest u-v-path in G' that is an up-down path is as fast as the fastest u-v-path in G.
- For $u \in V(G)$, the fastest up-down path from u to w in G' is as fast as the fastest path consisting of an in G fastest path from u to a neighbor of w, followed by an edge from this neighbor to w (for any start time x for which such a path exists in G'). An analogous property should hold for w-u-paths.

This can be achieved by setting $w \prec v$ for all $v \in V(G)$: Then up-down paths between two vertices $u, v \in V(G)$ remain unaffected, and up-down paths from u to w look exactly as described in the second bullet point as we can always choose an up-down path for the in G fastest path from u to the neighbor of w.

Next, we want to extend this concept to multiple inserted vertices. This is fairly easy as long as no two vertices subdivide the same original edge. Inserting several vertices into the same original edge works as well, but needs a slightly more thoughtful argument:

Lemma 6.21. *Let $(G, f, \prec)$ be a Contraction Hierarchy with $G = (V, E)$ and let $(W, \lambda, \overrightarrow{E}, \overleftarrow{E})$ be a subdivision configuration with $k := |W|$ such that $\overrightarrow{E_i} = \overrightarrow{E_j}$ and $\overleftarrow{E_i} = \overleftarrow{E_j}$ for all $i, j \in \{1, \dots, k\}$.*

Let $G' := G\langle w_1, \dots, w_k \rangle$ be the graph resulting from successively inserting $w_1, \dots, w_k$ in the edges from $\overrightarrow{E_i}$ with factor λ_i and in the edges from $\overleftarrow{E_i}$ with factor $(1 - \lambda_i)$ as explained in Definition 6.20. Let (f_i, f_i') be the pair of λ_i-subdivision ATFs of f_e for any $e \in \overrightarrow{E_i}$. We extend $\prec$ to $V(G')$ by setting $w_k \prec \cdots \prec w_1 \prec v$ for any $v \in V$. Then:

(i) For any edge $e = (u, v) \in \overrightarrow{E_i}$ and any $w_i \in W$, there is a down path from u to w_i in G' with ATF f_i and an up path from w_i to v with ATF f_i'. Moreover, there is no faster path at any start time from u to w_i in which u is the only non-W vertex. Analogously, there is no faster path at any start time from w_i to v in which v is the only non-W vertex.

(ii) For $w_i, w_j \in W$ with $\lambda_i < \lambda_j$, let f_{ij} be the λ_i-λ_j-intermediate ATF of f_e. Then there is an up path (if $i > j$) respectively down path (if $j > i$) from w_i to w_j in G' with ATF f_{ij}. There is no faster path at any start time from w_i to w_j using only vertices from W.

Proof. Let $\ell := |\overrightarrow{E}|$ and $\ell' := |\overleftarrow{E}|$. Denote by $u_1, \dots, u_\ell$ respectively $u_1', \dots, u_{\ell'}'$ the tail vertices of the arcs in $\overrightarrow{E}$ respectively $\overleftarrow{E}$ and by $v_1, \dots, v_\ell$ respectively $v_1', \dots, v_{\ell'}'$ the head vertices of the arcs in $\overrightarrow{E}$ respectively $\overleftarrow{E}$ such that $\overrightarrow{E} = \{(u_1, v_1), \dots, (u_\ell, v_\ell)\}$ and $\overleftarrow{E} = \{(u_1', v_1'), \dots, (u_{\ell'}', v_{\ell'}')\}$.

Let $i_1, \dots, i_k \in \mathbb{N}$ be a permutation of $1, \dots, k$ such that $\lambda_{i_j} \leq \lambda_{i_{j'}}$ if and only if $j \leq j'$. After having subdivided all edges in $\overrightarrow{E}$ and $\overleftarrow{E}$ by $w_1, \dots, w_k$, there are paths $u_i, w_{i_1}, \dots, w_{i_k}, v_i$ (for $1 \leq i \leq \ell$) and paths $u_i', w_{i_k}, \dots, w_{i_1}, v_i'$ (for $1 \leq i \leq \ell'$).

There are exactly ℓ arcs from w_{i_j} to $w_{i_{j+1}}$ for $1 \leq j < k$, one for each original edge in $\overrightarrow{E}$. They are all assigned the same ATF since all edges in $\overrightarrow{E}$ have the same ATF and they are all subdivided with the same factors λ_{i_j} and $\lambda_{i_{j+1}}$. Similarly, there is exactly one initial edge (u_i, w_{i_1}) and one final edge (w_{i_k}, v_i) for each $i \in \{1, \dots, \ell\}$; all initial edges are assigned the same ATF and all final edges are assigned the same ATF. Analogous statements hold for the edges in the ℓ' paths $u_i', w_{i_k}, \dots, w_{i_1}, v_i'$.

When searching for a fastest path from some u_i to a vertex w_{i_j} (or from w_{i_j} to v_i) in which u_i (or v_i) is the only non-W vertex, it suffices to only consider these edges we just described: The only other edges incident to vertices in W are edges that were later subdivided and only kept

as a shortcut in the Contraction Hierarchy; by Proposition 6.14, their ATF is the same as the composition of the ATFs of the two resulting subdivision edges. Hence $u_i w_{i_1} \dots w_{i_j}$ is a fastest u_i-w_{i_j}-path in which u_i is the only non-W vertex, no matter which edges we choose between any two subsequent vertices in W. Analogously, $w_{i_j} \dots w_{i_k} v_i$ is a fastest w_{i_j}-v_i-path in which v_i is the only non-W vertex. Iteratively applying Lemma 6.17 shows that the ATFs of these paths are f_{i_j} and f'_{i_j}, respectively, where (f_{i_j}, f'_{i_j}) is the pair of λ_{i_j}-subdivision ATFs of f_e for any $e \in \overrightarrow{E}$.

Similarly, for w_{i_j} and $w_{i'_j}$ with $j < j'$, a fastest w_{i_j}-$w_{i'_j}$-path that only uses vertices from W consecutively visits $w_{i_j}, w_{i_{j+1}}, \dots, w_{i_{j'-1}}, w_{i_{j'}}$. The ATF of this path is the λ_{i_j}-$\lambda_{i_{j'}}$-intermediate ATF of f_e (as can again be seen by an iterative application of Lemma 6.17).

It remains to show that we can achieve these ATFs by taking down or up paths, respectively. We prove this by induction on k. For (i), the base case where $k = 1$ holds since we set $w_1 \prec v$ for all $v \in V$. For (ii), the base case where $k = 2$ also holds since $w_i \prec w_j$ if $i > j$ and $w_i \succ w_j$ if $i < j$. Now assume that the lemma is true for a given $k \in \mathbb{N}$. We will see that it also holds if $W = \{w_1, \dots, w_{k+1}\}$. In order to prove (i), take an edge $e = (u, v) \in E$ and $w_i \in W$.

Case 1: If $i \neq k + 1$, we know from the induction hypothesis that there is a down path from u to w_i and an up path from w_i to v in $G\langle w_1, \dots, w_k\rangle$; these paths have the desired ATFs. Moreover, they still exist in G' since we don't remove edges when constructing G' from $G\langle w_1, \dots, w_k\rangle$.

Case 2: Now we consider the case $i = k + 1$. If the tails of the arcs that are subdivided when inserting w_{k+1} into $G\langle w_1, \dots, w_k\rangle$ are $u_1, \dots, u_\ell$, the edge (u, w_{k+1}) is a downward edge and has the desired ATF. Otherwise, let $j \in \{1, \dots, k\}$ such that w_j is the tail of the edges subdivided when inserting w_{k+1} into $G\langle w_1, \dots, w_k\rangle$. By the induction hypothesis, there is a down path from u to w_j. We can extend it by the edge (w_j, w_{k+1}) since this is a downward edge. The resulting down path has the desired ATF. An analogous argument also works for the up path from w_{k+1} to v.

The induction step for (ii) follows analogously. □

Now we can state the following theorem which ensures that we can safely successively subdivide edges, giving us exactly the results of shortest path queries that we expect:

Theorem 6.22. *Let $(G, f, \prec)$ be a Contraction Hierarchy with $G = (V, E)$ and let $(W, \lambda, \overrightarrow{E}, \overleftarrow{E})$ be a subdivision configuration with $k := |W|$.*

Then we can iteratively insert $w_1, \dots, w_k$ in this order into G (see Definition 6.20), resulting in a graph $G' := G\langle w_1, \dots, w_k\rangle$, and extend the vertex ordering $\prec$ to $w_1, \dots, w_k$ such that the following holds: For $s, t \in V(G')$ and a start time $x \in \mathbb{R}$ for which an s-t-path exists in $G\langle s, t\rangle$, the fastest up-down path from s to t in G' is as fast as the fastest s-t-path in $G\langle s, t\rangle$.

Proof. We extend the vertex ordering $\prec$ to $w_1, \dots, w_k$ by setting $w_k \prec \dots \prec w_1 \prec v$ for any $v \in V$. We distinguish the following cases:

Case 1: $s, t \notin W$. Then the claim of the theorem is just the Contraction Hierarchy property of G: Since inserting vertices never removes existing edges, a fastest up-down path from s to t in G' is just a fastest up-down path from s to t in G (as all $w \in W$ are less important than either s or t), and $G\langle s, t\rangle = G$.

Case 2: $s \in W, t \in V$. Let P be a fastest s-t-path in $G\langle s, t\rangle$. Denote by $v \in V$ the vertex succeeding s in P and by P' a fastest up-down v-t-path in G. Since G fulfills the Contraction Hierarchy property, concatenating (s, v) and P' results in a path that is as fast as P. By Lemma 6.21, there is an up s-v-path P'' in G' whose ATF is the same as the ATF of the

edge $(s, v) \in E(G\langle s, t\rangle)$. Thus, concatenating P'' and P' results in an up-down path from s to t in G' that is as fast as P.

For the other direction, let Q be a fastest up-down path from s to t in G'. Let u be the first vertex in Q that is contained in V. Since all vertices on the path from s to u in G' subdivide the same set of original edges of G, u is the head of one of the original edges subdivided by s. Therefore, the edge (s, u) exists in $G\langle s, t\rangle$. By Lemma 6.21, the ATF of (s, u) is the same as the ATF of the prefix of Q from s to u. Denote by Q' the suffix of Q from u to t. Q' also exists in $G\langle s, t\rangle$ since it only visits vertices in G. Concatenating the edge (s, u) and Q' yields a path in $G\langle s, t\rangle$ that has the same ATF as Q.

Case 3: $s \in V, t \in W$. This is analogous to Case 2, just working with the suffix of the path instead of its prefix.

Case 4: $s, t \in W$. If s and t subdivide different original edges, this is still analogous to Case 2, this time handling the prefix of the path as well as its suffix.

Thus, let now s and t subdivide the same set of edges. By Lemma 6.21, there is an up or down path from s to t in G' that only visits vertices in W, and the fastest such path is as fast as the edge (s, t) in $G\langle s, t\rangle$. Moreover, the fastest up-down path from s to t in G' that visits a vertex in V is as fast as the fastest s-t-path in $G\langle s, t\rangle$ that visits a vertex in V (by the same argument as in the case where s and t subdivide different original edges). Putting these observations together yields the statement of the Theorem. □

6.4.3 Street-side aware Address Mapping

The approach for inserting vertices that we have described so far subdivides edges in opposite directions by a single vertex, thus allowing us to leave an address in the same direction where we came from without paying any penalty for making this U-turn. Note that this does not affect u-v-paths where neither u nor v is the newly inserted vertex w because we only consider up-down paths and make w the least important vertex. However, when first searching a u-w-path because we need to visit w, and then searching a w-v-path, we can change directions between arriving at w and leaving for v.

As we will see in Section 7.1, the edge sets $\overrightarrow{E}$ and $\overleftarrow{E}$ that are subdivided by a newly inserted vertex are always chosen as all edges corresponding to the same road segment (which is data given in the road networks in our input). For larger roads with multiple lanes, each lane usually is represented by its own road segment. Smaller streets, e.g. in residential zones, may however be modeled by a single road segment then representing both directions. When transforming the input road network to a road graph, we replace this single road segment by two oppositely directed edges. Then a vertex inserted in both of these oppositely directed edges adds the possibility to change the direction of travel at this point without losing any time for such a maneuver. In a tour planned for a large truck, that is highly unrealistic. We therefore have implemented the possibility to insert vertices street-side aware.

There, we still subdivide edges in $\overrightarrow{E}$ with factor λ and edges in $\overleftarrow{E}$ with factor $1 - \lambda$, but instead of subdividing all edges by the same vertex w, we subdivide edges in $\overrightarrow{E}$ by a vertex w and edges in $\overleftarrow{E}$ by a vertex w'. We then add edges (w, w') and (w', w) equipped with ATFs modeling the time needed for such a U-turn which can be specified by the user in the BonnTour input. The user may also forbid U-turns at addresses completely, in which case we do not add such edges.

In order to guarantee that we can still restrict our path searches to up-down paths (similar to Theorem 6.22), we set $w \prec w' \prec v$ for all $v \in V$. When having added U-turn edges, we not only

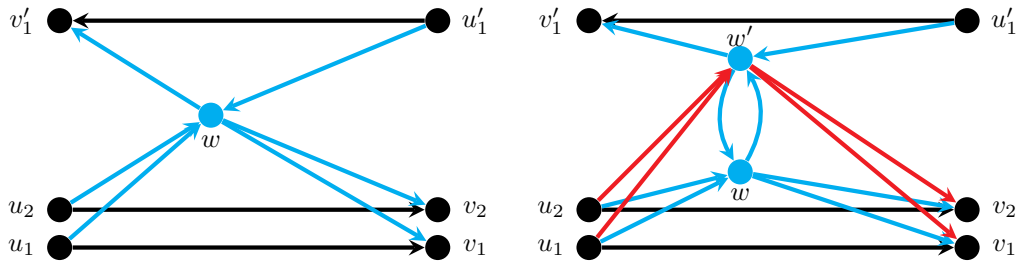


Figure 6.4: Left: inserting a single vertex w into $\overrightarrow{E} = \{(u_1, v_1), (u_2, v_2)\}$ and $\overleftarrow{E} = \{(u'_1, v'_1)\}$; when visiting an address at w, we can make a U-turn for free in w.
Right: inserting w into $\overrightarrow{E} = \{(u_1, v_1), (u_2, v_2)\}$ and w' into $\overleftarrow{E} = \{(u'_1, v'_1)\}$, adding U-turn edges (w, w') and (w', w). Because we set $w \prec w'$, we have to add the red shortcut edges (u_i, w') and (w', v_i) for the paths u_i, w, w' and w', w, v_i, respectively $(i = 1, 2)$.

have to keep the edges in $\overleftarrow{E}$ and $\overrightarrow{E}$ as shortcuts, but have to add more shortcut edges (contrary to our procedure explained in Definition 6.20):

Let G be the graph before inserting w and w', and let $\ell := |\overrightarrow{E}|$ and $\ell' := |\overleftarrow{E}|$. Denote by $u_1, \dots, u_\ell$ respectively $u'_1, \dots, u'_{\ell'}$ the tail vertices of the arcs in $\overrightarrow{E}$ respectively $\overleftarrow{E}$ and by $v_1, \dots, v_\ell$ respectively $v'_1, \dots, v'_{\ell'}$ the head vertices of the arcs in $\overrightarrow{E}$ respectively $\overleftarrow{E}$ such that $\overrightarrow{E} = \{(u_1, v_1), \dots, (u_\ell, v_\ell)\}$ and $\overleftarrow{E} = \{(u'_1, v'_1), \dots, (u'_{\ell'}, v'_{\ell'})\}$.

The vertex w has the lowest importance among all vertices in V; we therefore can easily extend up-down-paths in G to start or end in w. Analogously to Theorem 6.22, one can thus prove that when searching fastest paths starting or ending in w, it suffices to consider up-down-paths.

In order to be able to show the same for w', we add shortcut edges (u_i, w') and (w', v_i) for each $i \in \{1, \dots, \ell\}$, cutting short the paths along u_i, w, w' and w', w, v_i, respectively (see Fig. 6.4).

This way, together with the fact that we do not change the set of up-down-paths from u to v for any $u, v \in V$, we make sure that when restricting our fastest path search to up-down-paths, we find exactly the paths that we want. (The details work analogously to the proof of Theorem 6.22.)

For simplicity however, we will disregard this BonnTour feature in the description of the routines in the following chapters. Also the results shown in Chapter 8 are obtained without using the street-side aware Address Mapping.

Chapter 7

Address Mapping and Shared Parking

When solving tour planning problems, it is crucial to know the distances between addresses that might be visited consecutively. One possibility is to just work with distance matrices that are expected to be given as input: This way, the algorithm can focus on its core competence, namely computing tours. However, this approach is not particularly versatile: Whenever the set of addresses changes, the user needs to come up with a new matrix. Moreover, it is a major limitation to restrict BonnTour to users that know their own distance matrix.

Therefore, BonnTour computes the distance matrix itself. To this end, it reads road network information from a database, and takes the addresses that need to be visited as input. The previous chapter (Chapter 6) explained how this road network data is enhanced to build a Contraction Hierarchy that enables us to compute the distance matrix fast. The topic how to insert new vertices representing the addresses was also discussed, given that we already decided on the set of road graph edges that should be subdivided by the new vertex.

In this chapter, we explore how to find a reasonable choice of road graph edges for a certain address. We do not only consider geographical proximity, but also try to match street names if both the street in the road graph and the address carry street name information. This way, we find a position for the address that is as accurate and close to reality as possible. There are many reasons why this position is not the best choice to park the vehicle, though. For instance, the address may be located in a street with restrictions on motorized traffic (e.g. a pedestrian zone), or in a one-way street that requires us to drive a large detour when we enter it. Another reason can be addresses that are so close by that it is not worth it driving the vehicle from one to the other. We therefore assign two positions to an address, the *pedestrian position* that displays the position of the given address as accurately as possible, but might only be feasible to reach as a pedestrian, and the *vehicle position* that is well reachable by the used vehicle. In our model, the distance matrix that contains the travel time and distance information for connections between two addresses uses the vehicle positions of all addresses. The time needed to walk between the vehicle position and the pedestrian position of an address is then added separately for a given tour (depending on the number of walking trips required to transport all shipments at this address).

We start by presenting how BonnTour finds the pedestrian position for an address in Section 7.1. Section 7.2 deals with finding a good vehicle position with respect to reachability, while Section 7.3 also takes the possible interplay of nearby addresses that can use the same parking position into account.

7.1 Mapping coordinates to the road graph

We define an *address* as follows:

Definition 7.1 (Address)**.** *An* address *a is a pair $a = (\text{coord}(a), \text{streetname}(a))$ where $\text{coord}(a) \in \mathbb{R}^2$ is a position in the plane and $\text{streetname}(a)$ is a string of characters.*

Definition 7.2 (Road Graph)**.** *A* road graph *is a directed graph $G = (V, E)$ with the following additional properties:*

- *Each edge $e \in E$ is assigned an ATF f_e.*
- *Each vertex $v \in V$ is assigned a position $\text{pos}(v) \in \mathbb{R}^2$ in the plane.*
- *Each edge $e \in E$ is assigned a road segment number $\text{rs}(e) \in \mathbb{N}$.*
- *If $\text{rs}(e_1) = \text{rs}(e_2)$ for two edges $e_1 = (v_1, w_1), e_2 = (v_2, w_2) \in E$ with $v_1, v_2, w_1, w_2 \in V$, then $\text{pos}(v_1) = \text{pos}(v_2)$ and $\text{pos}(w_1) = \text{pos}(w_2)$ or $\text{pos}(v_1) = \text{pos}(w_2)$ and $\text{pos}(v_2) = \text{pos}(w_1)$.*
- *If $\text{rs}(e_1) = \text{rs}(e_2)$ for two edges $e_1 = (v_1, w_1), e_2 = (v_2, w_2) \in E$ with $v_1, v_2, w_1, w_2 \in V$ and $\text{pos}(v_1) = \text{pos}(v_2)$ and $\text{pos}(w_1) = \text{pos}(w_2)$, then $f_{e_1} = f_{e_2}$.*
- *Each road segment $r \in \text{rs}(E)$ is assigned a* street name $\text{name}(r)$ *which is a string of characters.*

In BonnTour, we read in a road graph that has been enhanced to a Contraction Hierarchy by defining a vertex ordering and adding shortcut edges. This Contraction Hierarchy is not a road graph anymore because shortcut edges do not correspond to road segments and are thus not assigned a road segment number. However, we do not want to map addresses to shortcut edges anyway. Therefore, it suffices to consider the subgraph of the Contraction Hierarchy that corresponds to the original road graph. In this chapter, we will only work with road graphs for this reason, and not with Contraction Hierarchies.

There can be different vertices with the same position because of how we model prohibited maneuvers (see Fig. 7.1) The road segment number is an identifier for which segment of a road an edge represents; again, different edges can have the same road segment number due to cloning of vertices for modeling prohibited maneuvers. Another reason for two edges having the same road segment number is small roads that can be used in both directions. On larger multi-lane roads, each direction is typically its own road segment in the map data. Smaller roads however are often represented by just one undirected road segment. Since all edges in our road graph are directed, we then introduce two oppositely directed edges that have the same road segment number.

A simple formulation of the goal of Address Mapping could be the following:

Problem 7.3: Address Mapping

Given a road graph $G = (V, E)$ and a set A of addresses, find a *fitting* edge $e \in E$ for each $a \in A$.

As we will see, edges with equal road segment number are indistinguishable in the address mapping procedure. When opting for a certain edge, we therefore implicitly choose all edges with the same road segment number. The technical details follow at the end of this section.

We need to find a measure for how fitting a road segment is for a specified address. The simplest idea is to take the one with the least distance: In our model, we convert all latitude and longitude coordinates given in the input into Cartesian coordinates using the Universal

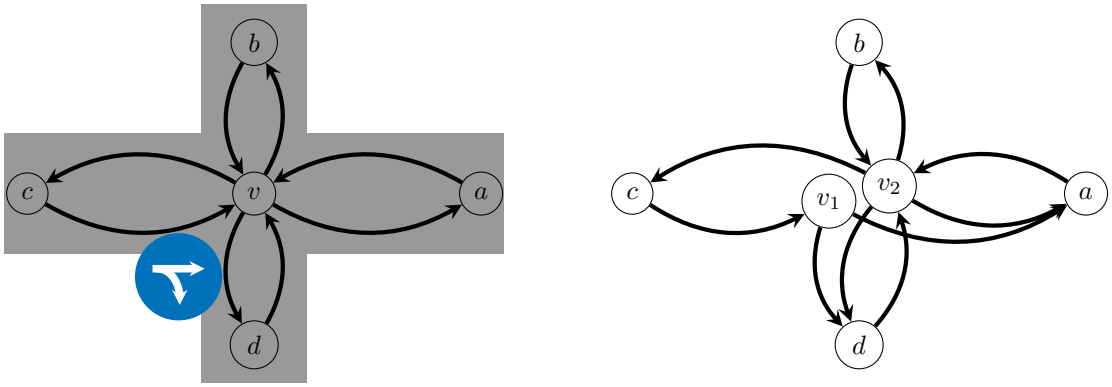


Figure 7.1: The left picture shows a junction in the road network modeled in the road graph by a vertex v. Suppose turning left or making a U-turn when coming from c is forbidden, i.e., we cannot go from c via v directly to b or back to c. Then the right picture shows how we model this restriction in the road graph: We replace the vertex v by two clones v_1 and v_2 with $\text{pos}(v_1) = \text{pos}(v_2) = \text{pos}(v)$. The clone v_1 is used when coming from c, so it has an ingoing edge from c and outgoing edges to a and d, but not to b or c. The clone v_2 has outgoing edges to a, b, c, and d and ingoing edges from a, b, and d, but not from c. This procedure leads to several vertices with the same position (namely v_1 and v_2). Note that (v_1, a), (v_2, a), and (a, v_2) correspond to the same road segment where (v_1, a) and (v_2, a) even go in the same direction.

Transverse Mercator coordinate system (UTM). This allows us to work in the Euclidean plane. An edge $e = (u, v)$ of a road graph represents the straight line connecting $\text{pos}(u)$ and $\text{pos}(v)$. In particular, for a given point $x \in \mathbb{R}^2$, the distance from x to e is given by the Euclidean distance from x to the line segment with endpoints $\text{pos}(u)$ and $\text{pos}(v)$. If roads are curved, we subdivide them upfront into several road segments such that this model does not deviate too much from reality.

Lemma 7.4. *Given three points $a = (a_1, a_2), b = (b_1, b_2), c = (c_1, c_2) \in \mathbb{R}^2$ in the plane, the squared Euclidean distance between a and the line segment s with endpoints b and c can be computed in constant time (when assuming that basic arithmetic operations can be performed in constant time). Moreover, the point $m = (m_1, m_2) \in s$ minimizing $\text{dist}(m, a)$ can be found in constant time, too.*

Proof. Let ℓ be the line going through b and c, and let $\pi(a) = (\pi(a)_1, \pi(a)_2)$ be the orthogonal projection of a onto ℓ. Since $\pi(a) \in \ell$, there is $\lambda \in \mathbb{R}$ such that $\pi(a) = \lambda \cdot b + (1 - \lambda) \cdot c$. Since the line through a and $\pi(a)$ is orthogonal to ℓ,

$$(\pi(a)_1 - a_1) \cdot (b_2 - c_2) + (\pi(a)_2 - a_2) \cdot (b_1 - c_1) = 0$$

must hold. This equation can easily be solved for λ, requiring only multiplication of two factors, addition, and one division. If $\lambda \in [0, 1]$, then $\pi(a) \in s$ and thus $m = \pi(a)$. Otherwise, if $\lambda > 1$ we get $m = b$, and $m = a$ if $\lambda < 0$.

The squared distance between a and the line segment s is just the squared distance between a and m and thus

$$(a_1 - m_1)^2 + (a_2 - m_2)^2.$$ □

Given an address $a = (\text{coord}(a), \text{streetname}(a))$, we can thus find an edge e minimizing the distance from $\text{coord}(a)$ to e in $\mathcal{O}(m)$ time (where $m = |E|$) by iterating over all edges. In practice,

this is often too slow. We therefore make use of so called R-trees introduced by Guttman in 1984 ([Gut84]). An R-tree is a self-balancing search tree for spatial structures like in our case segments that allows for fast queries of its entries that are nearest to a given point. For a fixed $k \in \mathbb{N}$, an R-tree containing m segments finds the k nearest neighbors (among the entries of the R-tree) of a given point in $\mathcal{O}(\log m)$ time. The downside is a longer construction time of $\mathcal{O}(m \log m)$ for the R-tree, but it pays off very quickly to invest in this pre-processing time when mapping several addresses.

In practice, it is not always desirable to choose the edge minimizing the distance to the position of an address. On the one hand, the input data is often not entirely accurate, and the specified geocoordinates may deviate by several meters. On the other hand, even with perfect input, a property may be located between two streets. In this case, it can happen that the entrance to the property is on one street, but the coordinates are closer to the other street. This problem is particularly severe if one has to drive a large detour to get from one street to the other. We have seen this in real-world instances with properties next to a highway where the access to these properties was not directly via the highway.

The given street name information provides a remedy: For an address a and an edge e, we compare streetname(a) and name(rs(e)). A second idea how to define *fitting* in Problem 7.3 is therefore to choose a road segment rs(e) with the least distance to a such that streetname(a) = name(rs(e)). However, it can happen that the street name is spelled differently in the input than in the map material. This may for example be due to simple transcription errors when creating the input or to inconsistent capitalization. To circumvent that problem, we define a *similarity score* for a pair of strings of characters: We specify a certain cost for removing, adding, or replacing characters, respectively, and, using a dynamic program, compute the sequence of changes with the lowest cost for transforming one string into the other. This cost is called *edit distance*. Moreover, we define a maximum edit distance that is still acceptable for two strings. The similarity score of two given strings is then the edit distance if it is at most this maximum, and infinite else. Before computing the similarity score of streetname(a) and name(rs(e)), we *normalize* both strings, converting all letters to lowercase, and applying substitutions for common spelling errors given by regular expressions specified by the user in the input.

Now, given an address a, we collect a set $E_a \subseteq E$ of the k edges that are nearest to a with $k \in \mathbb{N}$ minimum such that $|\text{name}(\text{rs}(E_a))| = 5$. The number 5 is chosen arbitrarily, but has proven to be reasonable in practical experiments. We then choose the edge $e \in E_a$ minimizing the similarity score of name(rs(e)) and streetname(a). Note that if none of the 5 names has an acceptable edit distance to streetname(a) (maybe due to badly generated input), we just opt for the edge that is closest to a.

All edges with the same road segment number are equally good with respect to this described measure. If we decide for an edge $e = (u, v)$, we define $\overrightarrow{E} := \{e' = (u', v') : \text{rs}(e) = \text{rs}(e') \text{ and } \text{pos}(u) = \text{pos}(u')\}$ and $\overleftarrow{E} := \{e' = (u', v') : \text{rs}(e) = \text{rs}(e') \text{ and } \text{pos}(u) = \text{pos}(v')\}$. Moreover, we compute the $\lambda \in [0, 1]$ for which the point m on the segment s corresponding to e attaining the minimum $\min_{x \in s} \text{dist}(x, \text{coord}(a))$ can be written as $m = \lambda \cdot v + (1 - \lambda) \cdot u$. Then $\overrightarrow{E}$, $\overleftarrow{E}$, and λ are the data that is passed on to the routine that inserts new vertices and subdivides edges (see Section 6.4).

We already discussed using R-trees to speed up this part of address mapping. Another approach to save running time is to parallelize the process and use multi-threading. Since the best edge for each address is completely independent of the other addresses, finding the best edges can easily be parallelized. In contrast to that, later inserting new vertices into the selected edges cannot be easily parallelized, especially when multiple addresses get mapped to the same edge. However, this is anyway a simple and fast procedure.

7.2 Global reachability analysis

While mapping the address to the closest street with the most fitting name is certainly necessary for modeling the tour planning instance as accurately as possible, the vertices we obtained in Section 7.1 might not be reachable for a vehicle, but only for pedestrians. We therefore call this chosen vertex the *pedestrian position*. Another possibility is that while the pedestrian position is accessible to vehicles in principle, this is only the case in a limited time window like pedestrian zones where delivery traffic is permitted in the morning.

We obtain information about restrictions on edges from the map data about the corresponding road segments. A first idea to solve the reachability issues is to take these restriction information into account when looking for the best edge to map an address to in the routine explained in Section 7.1. Then we can just exclude edges that have restrictions during the important time window that is relevant for visiting the address, e.g., a delivery time window for a shipment that has to be delivered to this address. However, this approach only provides a local view on the reachability of a road segment. It can happen that while a specific road segment has no relevant restrictions, the surrounding road segments do, and thus we still cannot reach our goal position. Another example where this local perspective can be insufficient is when the address is located at the beginning of a one-way street; entering this one-way street is possible, but can lead to large detours.

Therefore we consider the *global reachability* where we take into account how good a tour is that uses a certain vertex as parking spot, walking from there to the pedestrian position. If we already know the tour that is supposed to visit the address, i.e., we know the direction from which we arrive at the address, the direction where we drive next, and the rough time of day at which we visit the address, the task to find an optimal parking position is pretty straightforward. In fact, we have an implementation of a dynamic program optimizing parking position of fixed tours in BonnTour.

However, we already need a good estimation of where the parking spot will eventually be during the tour optimization routine. To this end, we take the following approach for a given address a: We select a set of parking position candidates in the vicinity of a and choose a vertex v^* in the road graph that is known to be well-reachable. For each parking position candidate p we then compute a round trip starting at v^*, driving to p, walking to the pedestrian position of a and back, and driving back to v^*. Finally, we pick the parking position candidate for which this round trip can be performed taking the least time. The details are explained in the following subsections.

7.2.1 Selecting parking position candidates

The maximum allowed distance between parking and pedestrian position is a parameter called *parking radius* given in the input. It is vehicle specific; so if there are multiple relevant vehicles for an address a, we take the minimum over their parking radii. (A vehicle is *relevant* for an address if there are shipments for this address that are compatible with the vehicle.) Note that BonnTour supports the feature to clone an address, making each clone only compatible with a subset of the shipments, and giving shipments at that address all clones as possible alternative addresses. We then consider each clone to be an independent address. This way, a vehicle with a small parking radius does not restrict the options of a vehicle with a larger radius.

Let $G = (V, E)$ be a graph, let a be an address and $\text{ppos}(a) \in V$ the pedestrian position of a. We then obtain all parking position candidates by performing a Dijkstra search with source $\text{ppos}(a)$, collecting all vertices whose distance from $\text{ppos}(a)$ is at most the parking radius. Note that this Dijkstra search can be done using constant edge weights since we assume that

pedestrians have a constant speed throughout the day, and therefore shortest paths equal fastest paths.

Let us briefly argue that it is reasonable to consider only already existing vertices (of course including the vertices we added when mapping the pedestrian positions) as parking position candidates instead of inserting new vertices by subdividing edges as in Section 6.4.

Proposition 7.5. *Let f be an ATF, $t \in \mathrm{dom}(f)$, and $g\colon [0,1] \to \mathbb{R}_{\geq 0}$ be an affine function. For $\lambda \in [0,1]$ denote by $(f^{(\lambda)}, f_{(\lambda)})$ the pair of λ-subdivision ATFs of f. Define*

$$h\colon [0,1] \to \mathbb{R} \cup \{\infty\}, \lambda \mapsto f_{(\lambda)}(f^{(\lambda)}(t) + g(\lambda)).$$

Then h has a global minimum at 0 or 1.

Proof. We write $\overline{g}(\lambda) := f^{(\lambda)}(t) + g(\lambda)$. Note that $\overline{g}$ is also affine as it is a sum of two affine functions since for a fixed $t \in \mathrm{dom}(f)$, the function $\lambda \mapsto f^{(\lambda)}(t) = (1-\lambda)\cdot t + \lambda \cdot f(t)$ is affine in λ. (For a fixed $x \in \mathrm{dom}(f_\lambda)$, the function $\lambda \mapsto f_{(\lambda)}(x)$ is also affine, but h is not necessarily affine because $f^{(\lambda)}(t) + g(\lambda)$ depends on λ.)

If $h(\lambda) = \infty$ for all $\lambda \in [0,1]$, the claim definitely holds. So assume now h has a global minimum at $\lambda^* \in (0,1)$ with

$$f_{(\lambda^*)}(\overline{g}(\lambda^*)) = h(\lambda^*) < h(0) = f_{(0)}(\overline{g}(0)) = f(\overline{g}(0)) \tag{7.1}$$

and

$$f_{(\lambda^*)}(\overline{g}(\lambda^*)) = h(\lambda^*) < h(1) = f_{(1)}(\overline{g}(1)) = \overline{g}(1). \tag{7.2}$$

Since $h(\lambda^*) \in \mathbb{R}$, we know that $\overline{g}(\lambda^*) \in \mathrm{dom}(f_{(\lambda^*)})$. By the definition of λ-subdivision ATFs, this implies that there is $x \in \mathrm{dom}(f^{(\lambda^*)})$ with $f(x) \geq \overline{g}(\lambda^*)$. Because $\overline{g}(\lambda^*) = f^{(\lambda^*)}(t) + g(\lambda^*) \geq f^{(\lambda^*)}(t)$ and $f^{(\lambda *)}$ is continuous, there has to exist $x \in \mathrm{dom}(f^{(\lambda^*)})$ with $f^{(\lambda^*)}(x) = \overline{g}(\lambda^*)$.

Let $t^* \in \mathbb{R}$ be maximal such that $f^{(\lambda^*)}(t^*) = \overline{g}(\lambda^*)$. Define the function

$$\overline{h}\colon [0,1] \to \mathbb{R}, \lambda \mapsto f^{(\lambda)}(t^*).$$

Note that $\overline{h}$ is again affine in λ. By construction of $\overline{h}$, we have $\overline{h}(\lambda^*) = \overline{g}(\lambda^*)$ and $\overline{h}(0) = t^*$. Moreover, we have

$$f(\overline{h}(0)) = f_{(\lambda^*)}(f^{(\lambda^*)}(t^*)) = f_{(\lambda^*)}(\overline{g}(\lambda^*)) < f(\overline{g}(0))$$

by (7.1). The monotonicity of f now implies $\overline{h}(0) < \overline{g}(0)$. Together with $\overline{h}(\lambda^*) = \overline{g}(\lambda^*)$ and the affinity of $\overline{h}$ and $\overline{g}$, we hence obtain $\overline{h}(1) > \overline{g}(1)$.

On the other hand, by (7.2), we have

$$\overline{h}(1) = f^{(1)}(t^*) = f(t^*) = f_{(\lambda^*)}(f^{(\lambda^*)}(t^*)) = f_{(\lambda^*)}(\overline{g}(\lambda^*)) < \overline{g}(1).$$

This is a contradiction; hence our assumption is wrong and h has indeed a global minimum at 0 or 1. □

Using Proposition 7.5, one can argue that it doesn't make sense to subdivide an edge by a new parking spot: Assume we arrive at time t at the tail of an edge $e = (u, v)$ and subdivide e by a new vertex w with factor λ in order to park at w. A walking path from w to the pedestrian position of the address goes via u or via v. But the time needed to walk from w via u to the address, deliver or pick up shipments there, and walk back to the parking spot (again via u) is an affine function in λ (similarly if we choose the walking path via v). This can easily be seen if

we assume the visit and handling times at the address as well as the pedestrian walking speed to be constant. Hence, we minimize the arrival time at the head of e by choosing $\lambda = 0$ or $\lambda = 1$, i.e., not subdividing e at all, but selecting its tail or head as parking spot.

In general, when there are time windows that we need to comply with for the visit at the address, one can construct situations where introducing new vertices would make sense. However, for our estimate where the parking position will probably be in the final solution, just considering already existing vertices is probably sufficient. This also discretizes our problem and therefore makes it easier to solve.

7.2.2 Computing the reachability score

After having collected all vertices $P \subseteq V$ within the parking radius from the pedestrian position $\text{ppos}(a)$ of the address a, we compute a so called *reachability score* for each of these vertices. We want to measure the global reachability of these vertices (also called parking position candidates from now on) by computing fictional roundtrips and comparing how well the different candidates perform. To this end, we need a well-reachable starting point for these roundtrips. We cannot just take the start address of the vehicle since we have not yet decided on the parking position there. Moreover, the start address of the vehicle does not necessarily lie in the direction from which we come later in the planned tour. Therefore, we take advantage of our Contraction Hierarchy structure and assume that higher importances correlate sufficiently with reachability. Since this starting point has to be near the address, too, because otherwise the roundtrips could be infeasible, we just take the vertex v^* with the highest importance in the Contraction Hierarchy within 5 kilometers of the pedestrian position of a. The choice of 5 kilometers as a radius here is of course arbitrary, but is reasonably large such that we should get rid of local effects like pedestrian zones. We thereby introduce a bias towards the direction in which v^* is in comparison to the address a. But as we will see in Chapter 8, this heuristic still leads to a reasonable estimate for our parking position that we can use during the tour optimization routine. Afterwards, the final tours are polished with our parking position post optimization which finds the best parking space for each address in a tour. In Chapter 9, we discuss possible improvements how to choose the roundtrip starting point for the global reachability analysis.

Let us consider a parking position candidate p. Denote by f_1 the ATF for driving from v^* to p and by f_2 the ATF for driving from p to v^*. We will compute reachability scores for each shipment that is picked up at a or delivered to a and take the average over all shipments in the end. Therefore assume that there is only one relevant shipment at a. Similarly, we assume for now that there is only one relevant vehicle. Of course f_1 and f_2 depend on the vehicle type, but if there are multiple relevant vehicles, we will compute the reachability score separately for each vehicle and again take an average in the end.

Let $t_{\text{parking}} \in \mathbb{R}_{\geq 0}$ be the parking time needed for the vehicle at the address, and let $t_{\text{visit}} \in \mathbb{R}_{\geq 0}$ be the time that we need to spend at the address in our model, i.e., visit time at the address and handling time for the considered shipment. Both these times are given in the input. Denote by t_{walking} the time needed to walk from p to $\text{ppos}(a)$ that we found when performing the Dijkstra search rooted in $\text{ppos}(a)$. Note that we assume walking times to be symmetric.

We are interested in the time which a round trip takes from v^* to $\text{ppos}(a)$ and back to v^*, parking at p and spending parking, visit, and handling time. But we will specify this round trip time depending on the arrival time at $\text{ppos}(a)$ since it is important that we arrive at the address within the time window of the actions that we perform there.

Let t be the time at which we arrive at $\text{ppos}(a)$. This means that we arrived at p at time $t - t_{\text{walking}} - t_{\text{parking}}$. Denote by $f_1^{-1}(t - t_{\text{walking}} - t_{\text{parking}})$ the latest time at which we can leave v^* such that we arrive in p at time $t - t_{\text{walking}} - t_{\text{parking}}$. If there is no such time, then we define

$f_1^{-1}(t - t_{\text{walking}} - t_{\text{parking}}) := -\infty$. We are back at v^* at time $f_2(t + t_{\text{visit}} + t_{\text{walking}})$. Hence we can define the *round trip time* for a parking position candidate p by

$$\text{rt}_p(t) := f_2(t + t_{\text{visit}} + t_{\text{walking}}) - f_1^{-1}(t - t_{\text{walking}} - t_{\text{parking}}).$$

Note that $\text{rt}_p(t)$ implicitly depends on p since f_2, f_1^{-1}, and t_{walking} all depend on p.

Let $I \subseteq \mathbb{R}$ be the time window in which it is feasible to arrive at a. This means, we take the time window for handling the considered shipment at a (i.e., the pickup time window if we pick up the shipment at a, and the delivery time window else), and subtract t_{visit} from the end of the time window since the whole visit action has to take place within the time window. Then the reachability score of p will essentially be the integral

$$\int_I \text{rt}_p(t)\,\mathrm{d}t.$$

However, if $\text{rt}_p(t)$ is infinite for some $t \in I$ (which means that the round trip is infeasible at this arrival time at $\text{ppos}(a)$), this integral immediately becomes infinite. But we need to be able to compare different parking position candidates that all lead to infeasible round trips for subsets of I, of course with a preference for those candidates where the subset of I in which rt_p is infinite is smaller. Therefore we define $t_{\min} := \min_{p \in P, t \in I} \text{rt}_p(t)$ as the minimum time needed for the roundtrip for any parking position candidate $p \in P$. Instead of integrating rt_p, we compute the reachability score of p as

$$\text{reachability}(p) := \int_I \min(\text{rt}_p(t), t_{\min} + c)\,\mathrm{d}t$$

where c is a constant that we internally set to 2 hours. As mentioned above, we made the simplifying assumption that there is exactly one shipment to be handled at the address a. If there are multiple shipments, but they cannot all be handled together, we just compute this integral for each shipment on its own and define the reachability score as the average over all shipments. In the case where it is reasonable to handle all shipments at one address at once, we compute only one round trip with all shipments. The details are explained in Section 7.3.

In the end, we choose the parking position candidate $p \in P$ for which $\text{reachability}(p)$ is minimal as the parking position for a. For our main tour optimization routine, this means that the ATFs and distances reported in the distance matrix denote those between parking positions of addresses, and the time we need to walk from the parking position to the pedestrian position is added to each visit of the address.

Let us briefly discuss the running time in practice and possible speedups for the procedure of finding a good estimate of the parking position. The Dijkstra run at the beginning is pretty fast, as the parking radius is usually reasonably small, and we work with constant edge weights. Computing the round trip time functions is also fast because we can use the Contraction Hierarchy to quickly find f_1 and f_2. Also, we can prune our search for the best parking position when we consider a candidate p where walking from p to $\text{ppos}(a)$ and back together with t_{parking} and t_{visit} already takes longer than the maximum time that a round trip for a previously considered candidate needs. That is especially helpful if we are given large parking radii in the input. For this pruning technique to be efficient, we consider the parking position candidates in the order of increasing walking time to $\text{ppos}(a)$. For further speedups, we parallelize the procedure as parking positions for different addresses are independent of each other.

7.3 Shared Parking

For addresses that are close to each other, it makes sense to find a shared parking spot, at least when it is highly probable that they will be visited consecutively in a tour, i.e., when the shipments that have to be picked up or delivered there are compatible. Shared parking has several advantages: First of all, it is a good model of what delivery drivers do in practice when they visit multiple nearby addresses. It often saves time to just walk between addresses instead of moving the vehicle a few meters. A technical benefit is that the distance matrices with which we deal during the main tour optimization become smaller if several addresses share a parking position. This is because distance matrices store ATFs between parking positions; we can therefore save some memory consumption.

The shared parking routine consists of two parts: In a first step, we partition the set of addresses into sets such that addresses in the same set will end up with a shared parking position. We call these sets *shared parking bundles*. In the second step, we find a good parking position for each shared parking bundle.

For partitioning the addresses into shared parking bundles, we make use of a heuristic that is already implemented in BonnTour for grouping shipments into bundles. This heuristic is otherwise used for bundling shipments upfront in order to reduce complexity for the tour planning algorithm, but it is highly parametrized and therefore also useful in the shared parking context. Moreover, it directly gives an ordering of the addresses in which they should be visited. We will just use this order when computing a round trip through these addresses later.

Since BonnTour cannot model pedestrian subtours yet, we assume star-shaped pedestrian tours for handling all shipments in the same shared parking bundle, i.e., walking back to the vehicle for each shipment. We make sure that all shipments at addresses in the same shared parking bundle have sufficiently compatible time windows and vehicle compatibilities. Additionally, we ensure that the walking distance between any two addresses in a shared parking bundle is at most the parking radius and at most the distance that the pedestrian can walk in the parking time for the vehicle at the addresses. We use this as a heuristic indication if we will be able to find a good shared parking spot. As proven empirically in Chapter 8, this heuristic performs well in practice. Since it is not guaranteed that there actually exists a shared parking spot for all addresses in a shared parking bundle, we use a two-stage approach: We first try to find a shared parking position, but if we fail, we fall back to mapping each address to a parking position on its own as described in Section 7.2.1 and Section 7.2.2.

Note that these shared parking bundles are only used for defining the parking position for each of the elements of a bundle. During the main tour optimization routine of BonnTour, we do not force addresses of the same shared parking bundle to be visited by the same tour, in particular not in the order given by the bundle. Instead, our time model incentivizes us to group these addresses in the same tour: Each time we move the vehicle and park it again, we need to pay parking time. But if we stay in the same parking position, only walking between different addresses, we save the parking time for each but the first address.

For finding a good parking position for a given shared parking bundle B, we proceed in a very similar way as for finding parking positions for single addresses. Analogously to Section 7.2.1, we first find parking position candidates within the parking radius. To this end, we perform multiple Dijkstra searches, rooted in $\text{ppos}(a)$ for each address $a \in B$, and select all vertices that are in the intersection of the results, i.e., that have at most a distance of the parking radius from each address $a \in B$.

Then we evaluate each parking position candidate essentially using the reachability score introduced in Section 7.2.2. However, instead of computing a round trip per shipment, we construct a round trip handling all shipments at all addresses in B. We again choose a well-

reachable vertex v^* by taking the vertex with the highest importance in the Contraction Hierarchy within 5 kilometers of the pedestrian position of the first address in B (using the order of the addresses given by our bundling routine). By f_1, we denote the ATF for driving from v^* to p, and we denote the ATF for driving from p to v^* by f_2. Let $k := |B|$ and $B = \{a_1, \dots, a_k\}$ such that $a_1, \dots, a_k$ are given in this order in the bundle. Let t_{parking} be the time we need for parking the vehicle at the shared parking position. Denote by t_{visit} the sum of all visit and handling times at the addresses in B, and for $a \in B$, denote by t^a_{walking} the time needed to walk from p to $\text{ppos}(a)$.

In order to arrive at a time $t \in \mathbb{R}$ at $\text{ppos}(a_1)$, we need to arrive at p at time $t - t^{a_1}_{\text{walking}} - t_{\text{parking}}$. Denote by $f_1^{-1}(t - t^{a_1}_{\text{walking}} - t_{\text{parking}})$ the latest time at which we can leave v^* such that we arrive in p at time $t - t^{a_1}_{\text{walking}} - t_{\text{parking}}$. Again, we set $f_1^{-1}(t - t^{a_1}_{\text{walking}} - t_{\text{parking}}) := -\infty$ if there is no such departure time. We are back at v^* at time $f_2(t + t_{\text{visit}} + t^{a_1}_{\text{walking}} + 2 \cdot \sum_{i=2}^{k} t^{a_i}_{\text{walking}})$. Hence we can define the round trip time function for a parking position candidate p by

$$\text{rt}_p(t) := f_2\left(t + t_{\text{visit}} + t^{a_1}_{\text{walking}} + 2 \cdot \sum_{i=2}^{k} t^{a_i}_{\text{walking}}\right) - f_1^{-1}\left(t - t^{a_1}_{\text{walking}} - t_{\text{parking}}\right).$$

We will integrate rt_p in order to compute the reachability score. As interval over which we integrate, we first choose the intersection of all time windows of the shipments at the addresses in B. We then subtract t_{visit} as well as the time needed for walking in between the shipment deliveries (or pickups) from the end of the intersection because all actions need to take place within the time window. Of course, it would be feasible to have shipments with slightly shifted time windows in a shared parking bundle, visiting first these with an earlier time window and then those with a later time window. This exactly works out if for example the time needed to walk from the address with the earlier time window to the address with the later time window equals the shift between these time windows. However, these walking times should be negligible in comparison to the time window lengths; especially since we paid attention to compatible time windows when building the shared parking bundles. For getting an estimate of where the parking position will be in the end, taking the intersection of all time windows is therefore reasonable.

The computation of the reachability score then works analogously to the procedure described in Section 7.2.2, including taking the average over several vehicle types and capping the function rt_p in order to be able to compare different parking positions that all are infeasible during a subset of the considered interval.

Chapter 8

Computational Results

We implemented the algorithms described in Chapter 6 and Chapter 7 using C++. They are part of the software package called BonnTour that we maintain in cooperation with our industry partner Greenplan. We carried out extensive experiments to give empirical proof that our suggested techniques lead to high-quality results. For running these tests, we used a 2-way AMD EPYC 7713 Linux server with 128 physical cores in total and 1TB RAM. We ran multiple jobs simultaneously (each using up to 8 threads) with GNU parallel ([Tan25]), never using more than 128 threads in total. Most of the instances that we used were generated by Greenplan. In order to render their confidential data anonymous, we named the instance classes using codes of the form C-x where x is a three-digit number.

8.1 Computing Contraction Hierarchies

Figures 8.1, 8.2, and 8.3 show results of computing Contraction Hierarchies for several road graphs with different parameter settings.

"default": In this setting, we approximate ATFs during the construction of the Contraction Hierarchy, we do not subdivide long edges upfront, and we use the $\text{Complex}_i(v)$ summand when determining the vertex ordering (see Section 6.2).

"100m sub": The difference to "default" is that we subdivide long edges upfront such that all edges represent road segments with a length of at most 100m.

"50m sub": The difference to "default" is that we subdivide long edges upfront such that all edges represent road segments with a length of at most 50m.

"no approx": The difference to "default" is that we do not approximate ATFs during the construction of the Contraction Hierarchy.

"100m no apx": Here we do not approximate ATFs during the construction of the Contraction Hierarchy, and we subdivide long edges upfront such that no edge is longer than 100m. The $\text{Complex}_i(v)$ summand is used for determining the vertex ordering.

"50m no apx": As "100m no apx", but edges are subdivided such that they are not longer than 50m.

"no complex": The difference to "default" is that we do not use the $\text{Complex}_i(v)$ summand for determining the vertex ordering, i.e., the importance score otherwise defined in (6.1) is instead chosen as

$$c_i(v) := 2 \cdot \text{Edges}_i(v) + \text{Depth}_i(v) + \text{Unpack}_i(v).$$

The instances for which we present results in Fig. 8.3 are small enough to also compute Contraction Hierarchies without approximating at all: The **"no apx at all"** Contraction Hierarchies were computed with the original ATFs as read in from our database without approximating them. We neither applied approximation during the construction of the Contraction Hierarchy nor after it was completed. (In contrast to that, the "no approx" Contraction Hierarchies only omit the approximation during the construction, then allowing an even larger approximation budget for the completed Contraction Hierarchy.)

We measured

- the time needed for computing the Contraction Hierarchy (reported in the "Runtime" column),
- the number of vertices of the resulting Contraction Hierarchy (reported in the "#Vertices" column),
- the number of edges that the original road graph had (reported in the "#Original Edges" column),
- the number of shortcut edges that were added during the construction of the Contraction Hierarchy (reported in the "#Shortcut Edges" column),
- the number of levels of the resulting Contraction Hierarchy (reported in the "#Levels" column),
- the average number of segments per ATF for the ATFs assigned to the edges of the resulting Contraction Hierarchy, divided into
  - (i) the ATFs for original edges before the final approximation step (reported in the column "Avg. #ATF segments (i)"),
  - (ii) the ATFs for shortcut edges before the final approximation step (reported in the column "Avg. #ATF segments (ii)"),
  - (iii) the ATFs for original edges after the final approximation step (reported in the column "Avg. #ATF segments (iii)"), and
  - (iv) the ATFs for shortcut edges after the final approximation step (reported in the column "Avg. #ATF segments (iv)"), and
- the disk space needed for storing the resulting Contraction Hierarchy (reported in the "Disk Space" column).

In each instance class, we computed multiple Contraction Hierarchies. The tables in Fig. 8.1, Fig. 8.2, and Fig. 8.3 summarize these computations, i.e., in the columns "Runtime", "#Vertices", "#Original Edges", "#Shortcut Edges", "# Levels", and "Disk Space" report the sums over all Contraction Hierarchies computed in this instance class. The average number of ATF segments per edge is the sum of all ATF segments on the original (respectively shortcut) edges of all Contraction Hierarchies divided by the number of original (respectively shortcut) edges in all Contraction Hierarchies.

C-018	Runtime	#Vertices	#Original Edges	#Shortcut Edges	#Levels	Avg. #ATF Segments (i)	Avg. #ATF Segments (ii)	Avg. #ATF Segments (iii)	Avg. #ATF Segments (iv)	Disk Space
default	2965s	$1.52 \cdot 10^7$	$3.20 \cdot 10^7$	$1.87 \cdot 10^7$	7091	31.7	283.4	10.5	10.1	4.06GB
100m sub	4231s	$2.45 \cdot 10^7$	$4.95 \cdot 10^7$	$3.14 \cdot 10^7$	6989	29.6	261.6	10.0	8.6	5.35GB
50m sub	6014s	$4.05 \cdot 10^7$	$8.02 \cdot 10^7$	$5.48 \cdot 10^7$	7123	32.0	278.7	8.9	7.5	7.10GB
no approx	4155s	$1.52 \cdot 10^7$	$3.19 \cdot 10^7$	$1.87 \cdot 10^7$	7082	31.6	620.4	10.5	9.9	4.03GB
100m no apx	6158s	$2.44 \cdot 10^7$	$4.94 \cdot 10^7$	$3.14 \cdot 10^7$	7056	29.6	583.1	10.0	8.5	5.33GB
50m no apx	9673s	$4.04 \cdot 10^7$	$8.02 \cdot 10^7$	$5.48 \cdot 10^7$	7048	32.0	629.4	8.9	7.5	7.08GB
no complex	3286s	$1.52 \cdot 10^7$	$3.20 \cdot 10^7$	$1.97 \cdot 10^7$	7290	31.8	289.3	10.5	10.2	4.17GB

Figure 8.1: This table shows the results of computing 16 Contraction Hierarchies for the instance class C-018 in different settings, each row reporting the results for one specific setting. The instance class C-018 consists of instances in Germany covering around 5'000 to 40'000 square kilometers and a time horizon between 5 and 24 hours each.

C-030	Runtime	#Vertices	#Original Edges	#Shortcut Edges	#Levels	Avg. #ATF Segments (i)	Avg. #ATF Segments (ii)	Avg. #ATF Segments (iii)	Avg. #ATF Segments (iv)	Disk Space
default	4274s	$2.49 \cdot 10^7$	$4.97 \cdot 10^7$	$3.27 \cdot 10^7$	4545	37.5	324.5	11.7	12.3	7.67GB
100m sub	6167s	$3.81 \cdot 10^7$	$7.32 \cdot 10^7$	$5.07 \cdot 10^7$	4484	36.4	325.6	11.7	11.5	10.13GB
50m sub	8491s	$5.96 \cdot 10^7$	$11.28 \cdot 10^7$	$8.13 \cdot 10^7$	4551	39.1	352.9	10.7	10.6	13.12GB
no approx	6328s	$2.49 \cdot 10^7$	$4.97 \cdot 10^7$	$3.27 \cdot 10^7$	4519	37.5	705.7	11.7	12.1	7.62GB
100m no apx	9236s	$3.81 \cdot 10^7$	$7.32 \cdot 10^7$	$5.07 \cdot 10^7$	4442	36.4	723.2	11.7	11.4	10.09GB
50m no apx	14779s	$5.95 \cdot 10^7$	$11.27 \cdot 10^7$	$8.13 \cdot 10^7$	4463	39.1	812.2	10.7	10.5	13.06GB
no complex	4817s	$2.50 \cdot 10^7$	$4.98 \cdot 10^7$	$3.44 \cdot 10^7$	4715	37.6	330.3	11.7	12.5	7.89GB

Figure 8.2: This table shows the results of computing 7 Contraction Hierarchies for the instance class C-030 in different settings, each row reporting the results for one specific setting. The instance class C-030 consists of instances covering Belgium and the Netherlands (around 85'000 square kilometers) and a time horizon between 5 and 8 hours each.

C-035	Runtime	#Vertices	#Original Edges	#Shortcut Edges	#Levels	Avg. #ATF Segments (i)	Avg. #ATF Segments (ii)	Avg. #ATF Segments (iii)	Avg. #ATF Segments (iv)	Disk Space
default	463s	$7.23 \cdot 10^6$	$14.68 \cdot 10^6$	$8.67 \cdot 10^6$	2824	26.0	221.8	8.5	8.9	1.63GB
100m sub	569s	$9.45 \cdot 10^6$	$18.80 \cdot 10^6$	$11.95 \cdot 10^6$	2868	24.6	209.2	8.3	8.3	1.95GB
50m sub	710s	$14.31 \cdot 10^6$	$28.09 \cdot 10^6$	$19.29 \cdot 10^6$	2762	24.2	201.1	7.4	7.3	2.49GB
no approx	598s	$7.22 \cdot 10^6$	$14.65 \cdot 10^6$	$8.67 \cdot 10^6$	2886	25.9	387.1	8.4	8.8	1.62GB
100m no apx	772s	$9.48 \cdot 10^6$	$18.85 \cdot 10^6$	$11.96 \cdot 10^6$	2846	24.7	364.7	8.4	8.2	1.95GB
50m no apx	1025s	$14.32 \cdot 10^6$	$28.11 \cdot 10^6$	$19.29 \cdot 10^6$	2843	24.2	350.7	7.5	7.3	2.49GB
no complex	515s	$7.25 \cdot 10^6$	$14.71 \cdot 10^6$	$9.11 \cdot 10^6$	2775	26.0	228.6	8.5	9.0	1.68GB
no apx at all	787s	$7.23 \cdot 10^6$	$14.69 \cdot 10^6$	$8.67 \cdot 10^6$	2776	45.0	614.3	45.0	614.3	35.00GB

Figure 8.3: This table shows the results of computing 8 Contraction Hierarchies for the instance class C-035 in different settings, each row reporting the results of one specific setting. The instance class C-035 consists of instances covering an area of roughly 20'000 square kilometers in Switzerland (including the cities of Zurich, Basel, and Bern) and a time horizon of 13 hours.

	Runtime	#Vertices	#Original Edges	#Shortcut Edges	#Levels	Avg. #ATF Segments (i)	Avg. #ATF Segments (ii)	Avg. #ATF Segments (iii)	Avg. #ATF Segments (iv)	Disk Space
Europe	13705s	$9.97 \cdot 10^7$	$21.16 \cdot 10^7$	$13.71 \cdot 10^7$	969	9.4	74.1	3.8	3.9	13.43GB
USA	9848s	$7.14 \cdot 10^7$	$15.02 \cdot 10^7$	$10.77 \cdot 10^7$	1225	7.6	47.3	3.3	3.4	9.16GB

Figure 8.4: This table shows the results of computing a Contraction Hierarchy for Europe and the mainland US, respectively, for a time horizon of 48 hours.

Note that we always approximate all ATFs assigned to the edges of the Contraction Hierarchy after its construction, allowing a larger budget if we did not approximate during the construction of the Contraction Hierarchy (except of course in the "no apx at all" Contraction Hierarchies in Fig. 8.3): The total allowed approximation budget in all reported instances for ATFs in the Contraction Hierarchy is 0.5% of the minimum travel time of the respective ATF, distributed over approximating ATFs directly after having read them in from the database, approximating during the construction of the Contraction Hierarchy and approximating the complete Contraction Hierarchy in a final step. When omitting the approximation during the construction, the budget intended for this step is shifted to the final approximation step. Later, when computing ATF matrices for a given instance, we allow another 0.5% relative approximation budget.

For the used disk space of Contraction Hierarchies, only the number of segments after the final approximation is relevant. However, the time needed for computing the Contraction Hierarchy increases significantly if the ATFs that we work with are more complex. Therefore, it also makes sense to consider the number of ATF segments before the final approximation.

Since we omit the final approximation step in the "no apx at all" Contraction Hierarchies in Fig. 8.3, the number of ATF segments in the final Contraction Hierarchy is the same as the number of ATF segments before the final approximation step in this case.

Although we ran the same set of instances in the different settings, the reported number of vertices and original edges does not coincide, even in the settings that use the same fineness of subdivision. This is because right after having built the Contraction Hierarchy, we delete vertices and edges that are not necessary: Given an area that a Contraction Hierarchy should cover (also called *operation area*), it is not guaranteed that shortest paths between two points inside the operation area never leave it. In order to increase the probability that we find all shortest paths, we therefore compute a Contraction Hierarchy for an extended operation area. Afterwards, we delete vertices outside of the original operation area for which we can prove that they are not needed for shortest paths between vertices inside the operation area. We provide this proof by exploiting the Contraction Hierarchy property and remove e.g. locally least important vertices outside of the operation area. For more details, we refer to [Hei24].

The results confirm the statements we made in Chapters 6 and 7: Using the summand $\text{Complex}_i(v)$ which gives the incentive to contract vertices incident to more complex ATFs later when determining the vertex ordering makes sense: Omitting this summand leads to a higher computation time, more inserted shortcut edges, and in most cases more levels in the Contraction Hierarchy. This is the case even though we approximate ATFs in between, thus distorting the picture of the situation, because the complexity of the ATFs now also depends on whether they have already been approximated. However, the allowed approximation budget is comparatively small, and we still observe that ATFs on shortcut edges have significantly more breakpoints than those on original edges.

Moreover, approximating the ATFs in between is advantageous and reduces the construction time by roughly one third. We save more time if the Contraction Hierarchies are computed for a larger number of vertices. Note that we need slightly more disk space for storing the Contraction Hierarchy if we approximate in between. This is because otherwise, when not approximating ATFs during the construction of a Contraction Hierarchy, we allow a larger budget for the approximation in the end which increases our chances of saving more breakpoints. In Section 8.2, we present the differences in results of BonnTour runs when using the "default" Contraction Hierarchy and the "no approx" Contraction Hierarchy, showing that we usually do not lose any quality by approximating during the construction of the Contraction Hierarchy.

Subdividing edges upfront to ensure maximum edge lengths of 50m or 100m, respectively, drastically increases the number of vertices and therefore the running time. We therefore usually use the "default" setting, adding new vertices for mapping addresses afterwards (see Section 7.1).

The "default" setting also allows us to compute Contraction Hierarchies for areas so large that we fail to construct Contraction Hierarchies within a reasonable time using at most 1TB RAM when subdividing edges upfront or not approximating ATFs during the construction. In Fig. 8.4, we report statistics for Contraction Hierarchies which we built for Europe and the mainland US, respectively (measuring the same metrics as in Figures 8.1, 8.2, and 8.3).

8.2 Comparing approximated Contraction Hierarchies

We ran BonnTour instances of the instance classes C-018, C-030, and C-035, i.e., the classes for which we computed Contraction Hierarchies in Section 8.1. Each instance was solved twice: Once using the "default" Contraction Hierarchy, and once using the "no approx" Contraction Hierarchy. The difference is that for the "default" Contraction Hierarchy, we applied ATF approximation also during its construction. Note that contrary to what the name suggests, also the "no approx" Contraction Hierarchy is approximated: We always approximate all ATFs on road graph edges after having read in the road graph from the database. Moreover, we approximate all ATFs on Contraction Hierarchy edges after having finished the construction, allowing a larger approximation budget if we did not already approximate ATFs during the construction. When running a BonnTour instance, we read in a Contraction Hierarchy and then compute a distance matrix containing ATFs for all pairs of addresses in this instance. The ATFs in the distance matrix are again approximated with a small allowed error. During the tour optimization routine, we frequently access the ATFs in the distance matrix and as usual, working with less complex ATFs speeds up running time. In Figures 8.5, 8.6, and 8.7, we report the results of running BonnTour for the instances of the classes C-018, C-030, and C-035, respectively. To render the instances anonymous, we just numbered them consecutively (e.g. C-018-01 to C-018-30), always running an "a" version (with Contraction Hierarchy "no approx") as well as a "b" version (with Contraction Hierarchy "default"). We report the number of shipments in each instance (column "#Shipm.") and compare the resulting solutions:

- In the column "#Tours", we report the number of tours that BonnTour computed.
- In the column "#Drops", we report the number of shipments not delivered by the respective solution.
- In the column "Cost", we report the resulting cost of the solution, respecting our extensive cost model (including dropping penalty cost for the shipments that we do not deliver).
- In the column "Dist.", we report the distance driven (cumulative over all tours in the solution).
- In the column "Drv. time", we report the driving time that is spent (again cumulative over all tours in the solution).

Moreover, we compare the ATFs in the distance matrices. For each instance, we consider the distance matrix A resulting from the "no approx" Contraction Hierarchy and the distance matrix B resulting from the "default" Contraction Hierarchy. For each entry i, j of A resp. B (where $1 \leq i, j \leq n$ and n is the number of addresses in the instance), we compute the maximum relative difference of the ATF $f^B_{i,j}$ in B with respect to the ATF $f^A_{i,j}$ in A as

$$\max_{x \in \mathrm{dom}(f^A_{i,j}) \cap \mathrm{dom}(f^B_{i,j})} \frac{f^B_{i,j}(x) - f^A_{i,j}(x)}{f^A_{i,j}(x) - x},$$

i.e., we measure the difference relative to the driving time needed according to A. In the column "glob. max. ATF diff.", we report the maximum of these maximum relative differences over all

entries of A and B. In the column "avg. max. ATF diff.", we report the average of these maximum relative differences, taken over all non-diagonal entries that existed in both A and B. Note that when it is not possible to drive from address i to address j within the working time, the entry i, j is empty. There were pairs of addresses for which an entry existed in A, but not in B, but for those, the domain of the ATF in A was at most a few seconds long.

In Fig. 8.5, Fig. 8.6, and Fig. 8.7, we observe that while the average maximum relative ATF difference is mostly less than $\frac{1}{4}$ percent, we cannot give a bound on the global maximum relative ATF difference. In the C-030 instances (Fig. 8.6), there is an instance (C-030-23) where this global maximum difference is almost 79%. However, the solution quality in this case is not affected; total cost, distance, and driving time even decrease slightly in the version with approximation during the construction of the Contraction Hierarchy.

In the C-018 instances (Fig. 8.5), we see even more extreme global maximum relative ATF differences of more than 120% in the instances C-018-17 and C-018-18. In the C-018-17 instance, the cost slightly increases by a little over 1%; the cost for the C-018-18 instance however decreases by almost 3%. When the large errors appear only on connections that are not relevant for the final tours, they of course do not affect the solution.

The instance C-018-28 even gets significantly better in the version where we approximate Contraction Hierarchies during their construction. The cost of this solution is dominated by the penalties for dropping shipments; each shipment in the instance has a dropping penalty of 100000€. There is only little slack in this instance; already slight changes in the distances can therefore lead to more or fewer dropped shipments. Note that it can of course also happen that ATFs in the "default" Contraction Hierarchy have faster travel times than the corresponding ATF in the "no approx" Contraction Hierarchy because both Contraction Hierarchies are approximated in a final step after the construction. When not having approximated during the construction, we even allow a larger error in the final step.

Overall, when summing up all instances of one class, but omitting the instance C-018-28, the total cost of the C-018 instances increase by less than 0.4% when using the "default" Contraction Hierarchy instead of the "no approx" Contraction Hierarchy. The total cost of the C-030 instances increases by slightly more than 0.1% while the total cost in the C-035 instance even decreases by almost 0.1%. This justifies saving running time when constructing Contraction Hierarchies by approximating in between. But one has to keep in mind that it is impossible to rule out large errors and therefore, when one observes poor results, one should always consider the possibility that they are due to excessive approximation errors.

The instance from class C-035 is small enough such that it is feasible to compute Contraction Hierarchies without approximating at all, see Fig. 8.3. We computed ATF matrices from these Contraction Hierarchies, omitting the approximation step that is usually applied to ATF matrices. When using these exact ATF matrices as a baseline in our comparison, the maximum relative difference of ATFs in the "default" setting (i.e., using the "default" Contraction Hierarchy and approximating the resulting ATF matrix) is 6.61%. The average of the maximum relative differences, taken over all non-diagonal entries in the ATF matrix, is 0.66%. That is within the range we would usually expect with an allowed approximation budget of 1% of the minimum travel time. We also computed tours using these exact ATF matrices, resulting in tours that are even slightly worse (the cost in comparison to the "default" run increases by 0.1%), which can always happen since our algorithm is not guaranteed to find the best solution. Since the ATFs with which BonnTour has to work are significantly more complex in the exact computation, the running time increased from roughly 18 minutes in the "default" setting to almost 8 hours. Together with the fact that the disk space which is needed to store the Contraction Hierarchy is 35GB for the exact ATFs, and only 1.63GB for the "default" Contraction Hierarchy (see Fig. 8.3), this demonstrates nicely that approximation is essential for using BonnTour in practice.

Instance	#Shipm.	#Tours	#Drops	Cost (in €)	Dist. (in km)	Drv. time (hh:mm:ss)	glob. max. ATF diff.	avg. max. ATF diff.
C-018-01 a	126	12	0	1091.04	1189.76	30:24:29		
C-018-01 b	126	12	0	1086.37	1207.92	30:58:58	3.63%	0.21%
C-018-02 a	83	10	0	939.72	1160.44	24:49:53		
C-018-02 b	83	10	0	939.27	1157.53	24:52:08	6.49%	0.21%
C-018-03 a	49	9	1	100633.12	586.72	16:20:08		
C-018-03 b	49	9	1	100628.96	572.82	16:13:38	1.85%	0.10%
C-018-04 a	123	15	0	989.33	1264.28	32:04:09		
C-018-04 b	123	15	0	989.75	1264.17	32:05:32	5.29%	0.14%
C-018-05 a	62	8	0	532.36	691.98	15:34:17		
C-018-05 b	62	8	0	532.20	700.85	15:34:56	0.50%	0.04%
C-018-06 a	2413	162	2	222721.43	30761.09	716:06:41		
C-018-06 b	2413	159	2	222698.82	31176.01	726:42:40	16.57%	0.23%
C-018-07 a	496	31	3	7775.82	7084.30	135:36:05		
C-018-07 b	496	31	3	7773.16	7143.46	137:27:53	5.66%	0.24%
C-018-08 a	554	44	0	6340.72	10174.84	196:26:42		
C-018-08 b	554	46	0	6267.20	9959.14	194:07:44	8.94%	0.16%
C-018-09 a	591	59	0	12795.68	19813.55	304:37:08		
C-018-09 b	591	59	0	12557.44	19636.13	303:08:00	2.12%	0.06%
C-018-10 a	560	56	0	11181.51	18452.48	284:57:46		
C-018-10 b	560	59	0	11847.49	18671.42	287:14:42	2.12%	0.06%
C-018-11 a	1219	90	0	22490.51	30504.87	484:23:38		
C-018-11 b	1219	89	0	22262.58	30635.59	484:39:02	10.36%	0.26%
C-018-12 a	1266	91	0	23291.18	31508.14	495:51:57		
C-018-12 b	1266	93	0	23579.52	31145.80	493:25:27	13.11%	0.27%
C-018-13 a	376	28	0	4974.15	7825.59	121:46:00		
C-018-13 b	376	29	0	4967.04	7769.71	121:33:06	0.87%	0.09%
C-018-14 a	1036	72	0	10576.34	15611.11	313:52:30		
C-018-14 b	1036	73	0	10769.26	16034.24	319:34:51	12.40%	0.17%
C-018-15 a	1035	73	0	10697.72	15929.63	316:00:52		
C-018-15 b	1035	75	0	10726.21	15793.44	316:10:08	10.32%	0.18%
C-018-16 a	239	10	0	2283.11	3520.73	62:51:32		
C-018-16 b	239	10	0	2307.93	3572.24	63:39:27	5.91%	0.14%
C-018-17 a	222	9	0	1551.40	2496.15	46:33:20		
C-018-17 b	222	9	0	1569.36	2547.39	46:36:11	120.15%	1.78%
C-018-18 a	223	10	0	1653.85	2640.87	48:52:22		
C-018-18 b	223	9	0	1606.53	2591.07	48:02:20	120.15%	1.84%
C-018-19 a	202	11	0	1685.69	2339.28	40:20:07		
C-018-19 b	202	11	0	1672.23	2317.65	39:38:02	2.56%	0.17%
C-018-20 a	383	11	0	1674.57	2176.93	51:41:55		
C-018-20 b	383	11	0	1699.98	2238.48	52:21:03	6.71%	0.12%
C-018-21 a	324	10	0	1457.31	2113.39	52:19:55		
C-018-21 b	324	9	0	1433.50	2047.38	51:10:42	1.29%	0.10%
C-018-22 a	372	31	2	204603.13	6726.76	131:52:07		
C-018-22 b	372	30	2	204469.55	6500.26	127:17:44	7.83%	0.11%
C-018-23 a	339	38	1	104713.56	7080.16	123:14:30		
C-018-23 b	339	36	1	104694.42	6989.93	120:57:10	3.05%	0.20%
C-018-24 a	509	25	0	4422.45	6077.28	119:14:19		
C-018-24 b	509	26	0	4450.88	6125.54	118:43:55	9.09%	0.17%
C-018-25 a	214	8	0	2070.21	2563.11	45:30:02		
C-018-25 b	214	8	0	2105.27	2646.83	46:29:13	1.21%	0.10%
C-018-26 a	357	61	0	4261.34	5794.64	104:13:10		
C-018-26 b	357	61	0	4264.23	5778.11	104:47:29	0.76%	0.07%
C-018-27 a	787	55	0	10713.76	15100.63	261:05:54		
C-018-27 b	787	56	0	10944.00	15101.50	261:40:35	30.95%	0.16%
C-018-28 a	984	99	36	3627264.61	27871.53	466:34:20		
C-018-28 b	984	99	33	3427303.32	27818.50	457:50:40	8.94%	0.15%
C-018-29 a	1291	100	1	126629.59	32276.73	508:32:00		
C-018-29 b	1291	98	1	126228.30	32185.93	512:45:26	13.11%	0.26%
C-018-30 a	202	23	0	3430.12	4809.62	71:53:19		
C-018-30 b	202	24	0	3428.71	4817.15	71:18:14	0.61%	0.09%

Figure 8.5: Results of running the C-018 instances using the “no approx” Contraction Hierarchy (a) and the “default” Contraction Hierarchy (b)

Instance	#Shipm.	#Tours	#Drops	Cost (in €)	Dist. (in km)	Drv. time (hh:mm:ss)	glob. max. ATF diff.	avg. max. ATF diff
C-030-01 a	282	13	0	8953.56	1136.53	35:25:02		
C-030-01 b	282	13	0	8980.30	1171.94	35:41:53	5.81%	0.10%
C-030-02 a	261	12	0	8254.86	1081.93	33:02:41		
C-030-02 b	261	11	0	8046.49	1147.76	33:26:55	2.69%	0.15%
C-030-03 a	298	13	0	9151.50	1291.07	38:02:35		
C-030-03 b	298	13	0	9147.50	1236.11	37:54:33	0.77%	0.11%
C-030-04 a	376	17	0	12218.85	1924.82	52:55:23		
C-030-04 b	376	17	0	12253.13	1881.46	53:37:44	2.06%	0.13%
C-030-05 a	412	20	0	14077.44	1966.55	58:21:34		
C-030-05 b	412	20	0	14121.33	2018.22	59:37:48	1.32%	0.08%
C-030-06 a	311	14	0	10303.80	1636.73	45:16:41		
C-030-06 b	311	14	0	10235.63	1647.85	44:36:34	1.04%	0.13%
C-030-07 a	381	16	0	11308.85	1506.02	44:38:32		
C-030-07 b	381	16	0	11335.60	1530.85	44:19:57	0.68%	0.08%
C-030-08 a	252	11	0	7643.48	974.75	29:05:42		
C-030-08 b	252	11	0	7626.50	967.49	28:32:27	0.79%	0.10%
C-030-09 a	164	8	0	5616.21	836.25	23:53:16		
C-030-09 b	164	8	0	5634.29	858.73	24:20:21	0.84%	0.06%
C-030-10 a	386	18	0	12712.11	1752.20	49:46:24		
C-030-10 b	386	18	0	12708.36	1713.71	49:29:57	1.05%	0.10%
C-030-11 a	365	16	0	11134.92	1451.44	43:06:25		
C-030-11 b	365	16	0	11131.62	1424.86	43:16:19	9.13%	0.13%
C-030-12 a	304	14	0	9894.91	1373.55	42:24:10		
C-030-12 b	304	14	0	9850.24	1304.44	40:41:45	0.90%	0.12%
C-030-13 a	301	13	0	9082.97	1142.23	34:53:25		
C-030-13 b	301	13	0	9057.98	1123.93	34:34:16	8.86%	0.15%
C-030-14 a	560	26	0	18973.46	2969.20	89:37:47		
C-030-14 b	560	26	0	19224.13	3131.24	91:24:24	22.97%	0.12%
C-030-15 a	450	22	0	15798.30	2589.46	74:41:16		
C-030-15 b	450	22	0	15868.07	2547.76	74:36:22	8.90%	0.15%
C-030-16 a	447	23	0	66434.32	3003.28	85:05:57		
C-030-16 b	447	23	0	66367.04	2921.33	83:21:47	12.32%	0.10%
C-030-17 a	398	21	0	15118.70	2563.68	73:02:06		
C-030-17 b	398	21	0	15225.45	2708.32	75:05:55	9.97%	0.14%
C-030-18 a	266	13	0	59464.15	1941.49	51:48:21		
C-030-18 b	266	13	0	59413.78	1931.37	51:43:36	3.74%	0.07%
C-030-19 a	345	16	0	11611.14	1761.11	51:15:11		
C-030-19 b	345	16	0	11389.02	1737.47	50:08:27	0.94%	0.12%
C-030-20 a	285	14	0	9984.98	1482.12	44:02:26		
C-030-20 b	285	14	0	9980.51	1534.94	44:48:02	0.69%	0.06%
C-030-21 a	437	22	0	17072.50	3239.28	87:50:51		
C-030-21 b	437	23	0	17232.46	3252.64	88:39:28	15.00%	0.12%
C-030-22 a	478	23	0	16741.03	2789.05	79:52:33		
C-030-22 b	478	23	0	16710.46	2706.95	78:18:35	3.66%	0.15%
C-030-23 a	527	26	0	68743.29	3433.20	96:45:19		
C-030-23 b	527	26	0	68677.74	3365.82	94:37:45	78.97%	0.14%
C-030-24 a	466	22	1	115913.02	2462.84	72:19:04		
C-030-24 b	466	23	1	116087.80	2289.83	69:52:17	8.17%	0.18%
C-030-25 a	471	23	0	16852.80	2771.86	81:03:33		
C-030-25 b	471	23	0	16822.00	2854.50	81:49:44	2.40%	0.11%

Figure 8.6: Results of running the C-030 instances using the "no approx" Contraction Hierarchy (a) and the "default" Contraction Hierarchy (b)

Instance	#Shipm.	#Tours	#Drops	Cost (in CHF)	Dist. (in km)	Drv. time (hh:mm:ss)	glob. max ATF diff.	avg. max. ATF diff.
C-035-1 a	167	19	12	595615.98	5450.06	83:42:43		
C-035-1 b	167	18	13	590254.80	5452.79	85:00:00	0.67%	0.16%

Figure 8.7: Results of running the C-035 instance using the "no approx" Contraction Hierarchy (a) and the "default" Contraction Hierarchy (b)

8.3 Address Mapping and Shared Parking

We compare the accuracy of both address mapping modes described in Section 7.1, i.e., mapping to already existing vertices versus inserting new vertices. For mapping to already existing vertices, we used the Contraction Hierarchies computed with the settings "50m sub" and "100m sub" while we used the "default" Contraction Hierarchies for inserting new vertices. In Fig. 8.8 and Fig. 8.9, we report the results: We ran instances of the instance classes for which we computed Contraction Hierarchies in Section 8.1, comparing the average distance between the position of an address given in the input (also called *original position*) and the pedestrian position when mapping to edges. We report the computation times needed for mapping the addresses, the average Euclidean distance between the original position and the pedestrian position, and the average Euclidean distance between the pedestrian position and the parking position when

(i) inserting new vertices into the nearest edge (row "new vertex"),

(ii) mapping to the nearest vertex, having subdivided edges upfront such that they have a length of at most 50m (row "50m sub"), and

(iii) mapping to the nearest vertex, having subdivided edges upfront such that they have a length of at most 100m (row "100m sub").

Note that the averages are taken over all addresses across all instances from the respective instance class. In particular, an address that appears in multiple instances of the same class contributes multiple times to that average.

We observe that the best accuracy, i.e., the shortest average distance between original position and pedestrian position, is indeed achieved when inserting new vertices into edges. When subdividing edges at least every 50m and then mapping to vertices, the average distance still seems acceptable; when subdividing only at least every 100m, we already see an increase in the average distance of around $\frac{1}{2}$ compared to inserting new vertices. Note that when we subdivide edges at least every 50m, the resulting average edge length is usually less than 50m: For instance, an edge of length 60m is subdivided into two edges of length 30m each.

Since computing the Contraction Hierarchies takes longer when subdividing edges upfront (see Section 8.1), mapping to newly inserted vertices is clearly the favorable option.

In Fig. 8.10, we compare results of running the same instances once with the shared parking feature switched off and once using the shared parking feature. The instances we used are parcel delivery instances in a single city and dense enough such that using shared parking spots makes sense. We defined the parking radius of the delivery van to be 100 meters and the parking time to be 2 minutes. In these instances, our shared parking routine (when switched on) aggregated on average slightly less than 2.5 addresses in a shared parking bundle. We compare the number of tours, the total cost of the solution, the total driving distance (not including walking distance), the total work time, the total driving time (including parking time, but not walking time), the running time of BonnTour, and the memory usage of the BonnTour runs.

The memory usage of BonnTour runs is usually dominated by the size of the ATF distance matrices. As these matrices contain the ATFs between the parking positions of two addresses,

C-018	Average Distance orig ↔ pedestrian	Average Distance pedestrian ↔ parking	Runtime map pedestrian	Runtime map parking
new vertex	19.3m	4.2m	39s	120s
50m sub	23.2m	5.3m	120s	198s
100m sub	27.0m	4.7m	74s	148s

Figure 8.8: We compare the results of Address Mapping on the 30 instances from the C-018 instance class where we have mapped 9037 addresses in total. We report the average distance between the original position given in the input and the mapped pedestrian position, the average distance between the pedestrian position and the parking position, and the computation time needed to map all pedestrian positions and all parking positions. The vehicles in the C-018 instances all have a parking radius of 200m. The reported running times are the sum over all 30 instances. We compare mapping to newly inserted vertices ("new vertex") and mapping to already existing vertices after having subdivided edges upfront such that their maximum length is 50m ("50m sub") or 100m ("100m sub"), respectively.

C-030	Average Distance orig ↔ pedestrian	Runtime map pedestrian
new vertex	16.7m	145s
50m sub	20.7m	191s
100m sub	25.9m	109s

C-035	Average Distance orig ↔ pedestrian	Runtime map pedestrian
new vertex	8.0m	2.2s
50m sub	13.2m	1.7s
100m sub	17.0m	1.0s

Figure 8.9: We compare the results of Address Mapping on the 25 instances from the C-030 instance class where we have mapped 9298 addresses in total, and on the instance from the C-035 instance class where we have mapped 69 addresses. We report the average distance between the original position given in the input and the mapped pedestrian position and the computation time needed to map all pedestrian positions. The instances of the class C-030 and C-035 model the transportation of goods that are too heavy to be carried by a pedestrian. Therefore, the parking radius of the vehicles is 0, and the parking position is always exactly at the vehicle position. The reported running times for the instance class C-030 are the sum over all 25 instances. We compare mapping to newly inserted vertices ("new vertex") and mapping to already existing vertices after having subdivided edges upfront such that their maximum length is 50m ("50m sub") or 100m ("100m sub"), respectively.

their size decreases significantly when mapping several addresses to the same parking spot. The running time of BonnTour then also benefits from a reduced memory usage. Moreover, total cost, driven distance, working time, and driving time all improve (i.e., decrease) when using the shared parking feature. Hence these results clearly demonstrate that the shared parking feature leads to a significant improvement in results on instances where addresses are sufficiently close in comparison to the parking time.

To showcase the utility of the global reachability analysis explained in Section 7.2, we constructed an instance delivering medicine to all pharmacies in the city center of Bonn. This instance has a lot of interesting features, like a pedestrian zone only usable for vehicles in the morning, one-way streets and forbidden turns. Fig. 8.11 shows the resulting tour before running the parking post optimization routine. Thus, the tour still uses the parking positions determined by the global reachability analysis. In Fig. 8.12, we see the same tour after parking spots being post-optimized.

Instance	#Shipm.	#Tours	#Drops	Cost (in €)	Distance (in km)	Work time (hhhh:mm:ss)	Drv. time (hhh:mm:ss)	Park time (hhh:mm:ss)	Walk time (hhh:mm:ss)	Runtime (in s)	Memory usage
C-020-1 a	4561	54	76	95813.18	1978.89	400:40:25	207:09:36	142:10:00	11:17:26	5188	19.28GB
C-020-1 b	4561	54	76	95036.05	1918.98	363:47:55	124:36:33	64:56:00	56:58:20	3753	7.58GB
C-020-2 a	6414	75	133	160344.50	2640.88	550:40:39	279:18:32	193:42:00	15:47:38	8441	35.15GB
C-020-2 b	6414	74	133	158908.44	2509.44	492:38:47	156:24:01	79:42:00	80:40:30	6102	10.99GB
C-020-3 a	5965	70	113	138547.66	2491.78	514:30:40	261:23:03	181:30:00	15:02:49	7109	31.48GB
C-020-3 b	5965	69	113	137203.73	2364.23	461:02:38	148:44:29	76:32:00	74:13:33	4949	10.50GB
C-020-4 a	5537	65	107	130744.74	2324.86	478:43:58	243:40:54	168:58:00	14:02:56	7167	26.70GB
C-020-4 b	5537	65	107	129740.46	2221.59	431:26:55	139:57:07	72:16:00	70:29:08	4968	9.22GB
C-020-5 a	5799	68	121	145797.33	2397.70	499:33:47	253:51:44	176:58:00	14:35:10	5535	27.86GB
C-020-5 b	5799	68	121	144696.06	2297.84	447:19:16	141:55:14	73:08:00	74:17:12	4108	9.37GB
C-020-6 a	3302	39	63	77200.17	1443.56	283:53:06	147:46:41	103:44:00	07:23:40	2043	8.77GB
C-020-6 b	3302	39	63	76678.56	1380.96	259:22:59	90:58:09	50:46:00	39:42:10	1597	4.37GB
Sum a	31578	371	613	748447.58	13277.67	2728:02:35	1393:10:30	967:02:00	78:09:39	35483	149.25GB
Sum b	31578	369	613	742263.30	12693.04	2455:38:30	802:35:33	417:20:00	396:20:53	25477	52.02GB

Figure 8.10: Results of running the C-020 instances using the shared parking feature (b) and running with the shared parking feature switched off (a). The C-020 instances are parcel delivery instances in a single city. The delivery van has a parking radius of 100 meters and a parking time of 2 minutes. There are no time windows in these instances, so the integral for computing the reachability score is always taken over the vehicles' work time horizons.

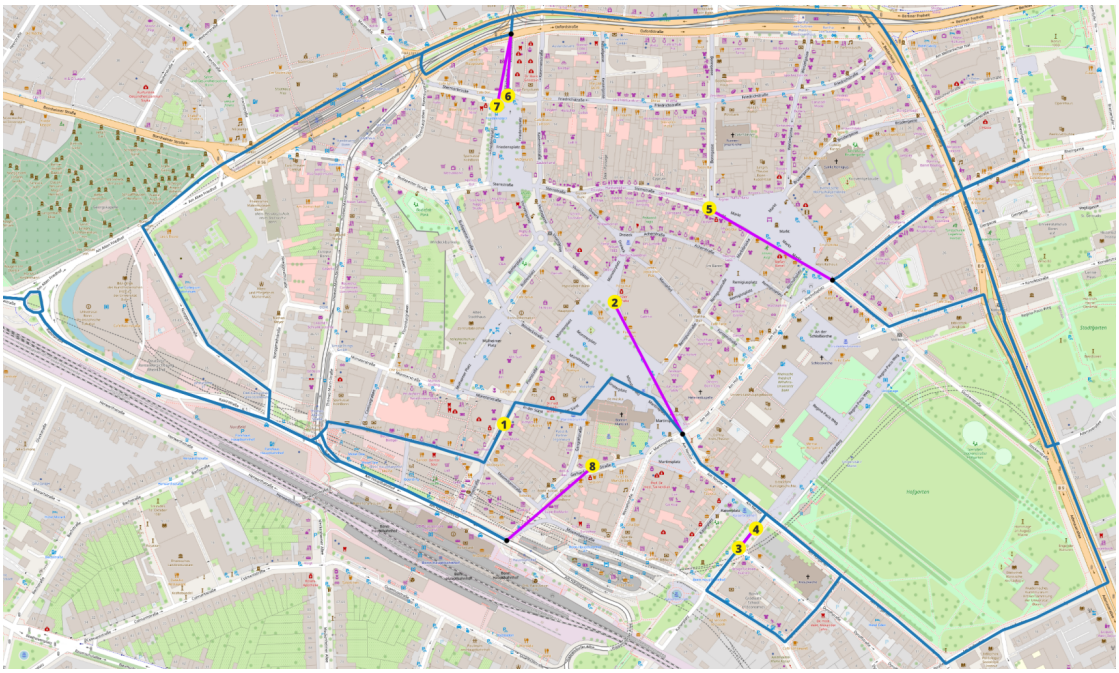


Figure 8.11: A tour computed by BonnTour, delivering medicine from a depot in the west of Bonn (outside the picture) to eight pharmacies (numbered in yellow circles) in the city center of Bonn. The tour driven by the delivery van is marked in blue while the connections between pedestrian position and parking position of the addresses are marked in purple. These connections have to be made on foot. The pharmacies are numbered in the order in which we visit them. Most of the city center of Bonn is a pedestrian zone through which delivery vehicles are only allowed to drive from 6am to 12pm. The shipment delivered to pharmacy 1 has a delivery time window in the morning while all other shipments have to be delivered in the afternoon. We therefore drive through the pedestrian zone to deliver the first shipment, but then park outside the pedestrian zone to deliver the next shipment (at pharmacy 2) – even though we have already been significantly nearer to pharmacy 2. The tour then continues to deliver the medicine, using shared parking spots for pharmacies 3 and 4 and for pharmacies 6 and 7, respectively. The extra loop driven for pharmacies 3 and 4 is due to one-way streets. In the east of the city center of Bonn, we can observe another peculiarity: Coming from the south, we need to turn left to visit pharmacy 5. But turning left is forbidden at this crossroad which we also model inside our road graph (see Fig. 7.1). We therefore turn right and then make a U-turn at the next crossroad.

The map data for the background is taken from OpenStreetMap (`https://www.openstreetmap.org/copyright`).

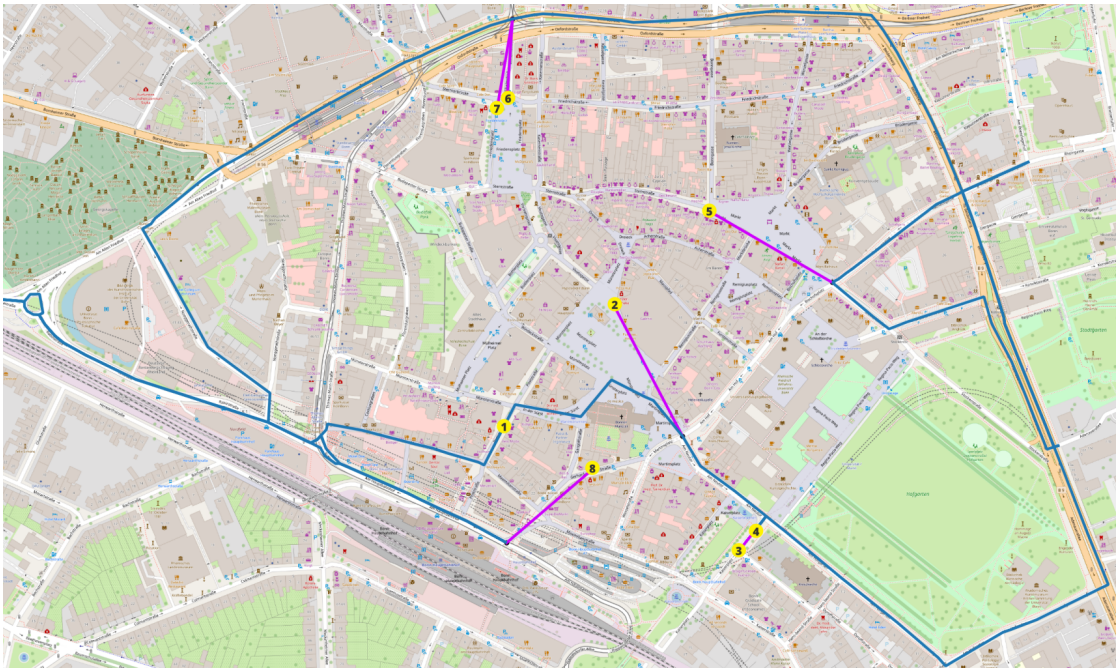


Figure 8.12: The same tour as in Fig. 8.11, but after running our parking post optimization routine on it. The pharmacies are still visited in the same order as before, but we optimize the parking positions with respect to that order. This changes the tour in two places: For the pharmacies 3 and 4, we no longer drive the loop. We did not choose this parking position in the global reachability analysis because it only makes sense when coming from the west. When not driving through the pedestrian zone, coming from the west however is only possible with large detours. The other change is for the pharmacies 6 and 7, where we do not park at the side of the street closer to the pharmacies, but decide to cross the street by foot. This makes sense because there are only limited possibilities to make U-turns on this street. In total, the driven distance is reduced from 12.23km for the tour shown in Fig. 8.11 to 11.58km here (including the way from the depot and back not shown in the picture). The total working time is reduced by 2 minutes: while the driving time decreases by roughly 3 minutes, the walking time increases by approximately 1 minute.

The map data for the background is taken from OpenStreetMap (`https://www.openstreetmap.org/copyright`).

Chapter 9

Future work

There are still many ideas how to improve on the algorithms and heuristics presented in the second part of this thesis. One of them is linked to how ATFs are approximated throughout our algorithm. As explained in Section 6.3, we approximate ATFs on shortcut edges using a variant of the Imai-Iri algorithm with an allowed approximation budget. However, due to the post-processing routine we apply (see Section 5.4.2), it can happen that only part of the budget is used. Therefore, it might be worthwhile to measure the actual resulting approximation error, and store this as already used budget. This would enable us to use more approximation budget later, increasing the chances to get rid of more breakpoints. Since ATFs are approximated in multiple places, not only when constructing Contraction Hierarchies, this approach could even be more broadly used, yielding overall simpler ATFs and thus decreasing the running time and memory consumption of BonnTour runs. Hence, it seems probable that investing the effort to internally rework the handling of approximation budgets would pay off.

Another idea affects Address Mapping: We have discussed several times that the best choice of a parking position for an address depends on the vehicle. When collecting parking position candidates, we therefore only search within the minimum of the parking radii of all relevant vehicles (see Section 7.2.1). This might unnecessarily restrict our options for vehicles with a larger parking radius. Afterwards, when computing reachability scores for parking position candidates, we take the average over all relevant vehicles (see Section 7.2.2). In cases where the vehicle types differ a lot in relevant features, this compromise often is not the best choice for any of the vehicles. We already brought up the idea of cloning addresses, making each clone compatible to a set of similar enough vehicles, and then using BonnTour's multiple addresses feature to specify several pickup or delivery options for each shipment at the cloned address. Considering each clone as an independent address then allows us to find parking positions only for similar vehicles. However, having multiple address options per shipment always comes with a runtime overhead (see [Bla24]). Changing our internal model to allow for different vehicle-dependent parking positions per address therefore is probably worth the effort.

We have also briefly discussed the choice of the starting vertex for the roundtrip tours computed in the global reachability analysis (cf. Section 7.2.2). In order to eliminate the bias for the direction in which this starting point lies, one could compute multiple roundtrip tours for each address, starting in different directions around the address, and take the average reachability score. Given an address a for which we want to find a good parking spot, a more sophisticated approach would be to pre-compute addresses that are likely to be visited directly before and directly after a. These addresses can be determined e.g. by beeline proximity, matching time windows, and matching vehicle compatibilities. Then we can compute a trip visiting a, start-

ing in another address and ending in a third address which is probably also more realistic than the roundtrips where we leave a in the same direction from which we came from. Since it is probably hard to decide for one single address that is most likely to be visited directly before a, and another single address that succeeds a on our tour, we can again take multiple choices and compute the reachability score as the average over all these possibilities. To even go one step further, this approach could be combined with the previous idea, allowing multiple parking positions per address, e.g. one per relevant direction.

Finding good predecessors and successors of a in a tour is of course one of the main tasks of BonnTour and therefore hard to estimate correctly upfront. Therefore it remains to be seen whether such a significantly more complex approach brings sufficient improvements compared to the current method which – together with the parking post optimization routine – already now yields high-quality results.

Bibliography

[AKB17] Mohamed Abdellahi Amar, Walid Khaznaji and Monia Bellalouna. 'An exact resolution for the probabilistic traveling salesman problem under the a priori strategy'. In: *Procedia Computer Science* 108 (2017), pp. 1414–1423 (cit. on p. 9).

[AV21] Luca Accorsi and Daniele Vigo. 'A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems'. In: *Transportation Science* 55.4 (2021), pp. 832–856 (cit. on p. 140).

[AGLW17] Marek Adamczyk, Fabrizio Grandoni, Stefano Leonardi and Michal Wlodarczyk. 'When the optimum is also blind: a new perspective on universal optimization'. In: *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Vol. 80. 2017, 35:1–35:15 (cit. on p. 9).

[AS00] Pankaj K. Agarwal and Micha Sharir. 'Davenport–Schinzel sequences and their geometric applications'. In: *Handbook of Computational Geometry*. 2000, pp. 1–47 (cit. on p. 142).

[And12] Miguel Andres Figliozzi. 'The time dependent vehicle routing problem with time windows: benchmark problems, an efficient solution algorithm, and solution characteristics'. In: *Transportation Research Part E: Logistics and Transportation Review* 48.3 (2012), pp. 616–636 (cit. on p. 140).

[AS19] Florian Arnold and Kenneth Sörensen. 'Knowledge-guided local search for the vehicle routing problem'. In: *Computers & Operations Research* 105 (2019), pp. 32–46 (cit. on p. 140).

[AGMGS17] Arash Asadpour, Michel X. Goemans, Aleksander Mądry, Shayan Oveis Gharan and Amin Saberi. 'An O(log n/log log n)-approximation algorithm for the asymmetric traveling salesman problem'. In: *Operations Research* 65.4 (2017), pp. 1043–1061 (cit. on p. 5).

[Awe85] Baruch Awerbuch. 'Complexity of network synchronization'. In: *Journal of the ACM* 32.4 (1985), pp. 804–823 (cit. on p. 82).

[BBCM04] Nikhil Bansal, Avrim Blum, Shuchi Chawla and Adam Meyerson. 'Approximation algorithms for deadline-TSP and vehicle routing with time-windows'. In: *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*. 2004, pp. 166–174 (cit. on p. 140).

[Bar96] Y. Bartal. 'Probabilistic approximation of metric spaces and its algorithmic applications'. In: *1996 IEEE 37th Annual Symposium on Foundations of Computer Science (FOCS)*. 1996, pp. 184–193 (cit. on p. 82).

[BFSS07] Holger Bast, Stefan Funke, Peter Sanders and Dominik Schultes. 'Fast routing in road networks with transit nodes'. In: *Science* 316.5824 (2007), pp. 566–566 (cit. on pp. 140, 146).

[BGSV13] G. Veit Batz, Robert Geisberger, Peter Sanders and Christian Vetter. 'Minimum time-dependent travel times with contraction hierarchies'. In: *ACM J. Exp. Algorithmics* 18 (2013) (cit. on pp. 137, 139, 140, 144, 147–149).

[Ben17] Hamza Ben Ticha. 'Vehicle routing problems with road-network information'. PhD thesis. Université Clermont Auvergne, 2017 (cit. on p. 140).

[BAFQV19] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, Alain Quilliot and Tom Van Woensel. 'A branch-and-price algorithm for the vehicle routing problem with time windows on a road network'. In: *Networks* 73.4 (2019), pp. 401–417 (cit. on p. 140).

[Ben75] Jon Louis Bentley. 'Multidimensional binary search trees used for associative searching'. In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517 (cit. on p. 141).

[BNW22] Aaron Bernstein, Danupon Nanongkai and Christian Wulff-Nilsen. 'Negative-weight single-source shortest paths in near-linear time'. In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 600–611 (cit. on pp. 2, 54, 81, 82).

[Ber88] Dimitris Bertsimas. 'Probabilistic combinatorial optimization problems'. PhD thesis. Massachusetts Institute of Technology, 1988 (cit. on p. 9).

[BJO90] Dimitris J. Bertsimas, Patrick Jaillet and Amedeo R. Odoni. 'A priori optimization'. In: *Operations Research* 38.6 (1990), pp. 1019–1033 (cit. on p. 9).

[BCK11] Anand Bhalgat, Deeparnab Chakrabarty and Sanjeev Khanna. 'Optimal lower bounds for universal and differentially private Steiner trees and TSPs'. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. 2011, pp. 75–86 (cit. on p. 9).

[Blä08] Markus Bläser. 'A new approximation algorithm for the asymmetric TSP with triangle inequality'. In: *ACM Trans. Algorithms* (2008) (cit. on pp. 5, 54).

[Bla24] Jannis Blauth. 'Vehicle routing in theory and practice'. PhD thesis. Rheinische Friedrich-Wilhelms-Universität Bonn, 2024 (cit. on pp. 2, 137, 140, 191).

[BHMSTTV24] Jannis Blauth, Stephan Held, Dirk Müller, Niklas Schlomberg, Vera Traub, Thorben Tröbst and Jens Vygen. 'Vehicle routing with time-dependent travel times: theory, practice, and benchmarks'. In: *Discrete Optimization* 53 (2024), p. 100848 (cit. on pp. 2, 137, 140–144).

[BNPV25] Jannis Blauth, Meike Neuwohner, Luise Puhlmann and Jens Vygen. 'Improved guarantees for the a priori TSP'. In: *Mathematics of Operations Research* 50.4 (2025), pp. 2909–2940 (cit. on pp. 1, 5, 7, 27, 29–31).

[BTV23] Jannis Blauth, Vera Traub and Jens Vygen. 'Improving the approximation ratio for capacitated vehicle routing'. In: *Mathematical Programming* 197.2 (Feb. 2023), pp. 451–497 (cit. on p. 140).

[BFB03] Neill E. Bowler, Thomas M. A. Fink and Robin C. Ball. 'Characterization of the probabilistic traveling salesman problem'. In: *Phys. Rev. E* 68.3 (2003), p. 036703 (cit. on p. 9).

[BCF23] Karl Bringmann, Alejandro Cassis and Nick Fischer. 'Negative-weight single-source shortest paths in near-linear time: now faster!' In: *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. 2023, pp. 515–538 (cit. on p. 54).

[BFHL25] Karl Bringmann, Nick Fischer, Bernhard Haeupler and Rustam Latypov. 'Near-optimal directed low-diameter decompositions'. In: *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*. Vol. 334. 2025, 35:1–35:18 (cit. on pp. 2, 54, 81, 82).

[CKP12] Chandra Chekuri, Nitish Korula and Martin Pál. 'Improved algorithms for orienteering and related problems'. In: *ACM Trans. Algorithms* 8.3 (2012) (cit. on p. 140).

[CPT26] Manuel Christalla, Luise Puhlmann and Vera Traub. 'Approximating asymmetric a priori TSP beyond the adaptivity gap'. In: *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2026, pp. 5374–5439. Full version on arXiV. arXiv: 2510.17595.

[Chr76] Nicos Christofides. *Worst-case analysis of a new heuristic for the travelling salesman problem.* Technical report, Carnegie-Mellon University, [reprinted in Operations Research Forum3 (2022), Article20]. 1976 (cit. on p. 5).

[DRWD13] Said Dabia, Stefan Ropke, Tom van Woensel and Ton De Kok. 'Branch and price for the time-dependent vehicle routing problem with time windows'. In: *Transportation Science* 47.3 (2013), pp. 380–396 (cit. on p. 140).

[Dij59] Edsger W Dijkstra. 'A note on two problems in connexion with graphs'. In: *Numerische Mathematik* 1 (1959), pp. 269–271 (cit. on pp. 137, 145).

[DMCRG08] Alberto V. Donati, Roberto Montemanni, Norman Casagrande, Andrea E. Rizzoli and Luca M. Gambardella. 'Time dependent vehicle routing problem with a multi ant colony system'. In: *European Journal of Operational Research* 185.3 (2008), pp. 1174–1191 (cit. on p. 140).

[DMMSY25] Ran Duan, Jiayi Mao, Xiao Mao, Xinkai Shu and Longhui Yin. 'Breaking the sorting barrier for directed single-source shortest paths'. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC)*. 2025, pp. 36–44 (cit. on p. 145).

[EIJS18] Martijn van Ee, Leo van Iersel, Teun Janssen and René Sitters. 'A priori TSP in the scenario model'. In: *Discrete Applied Mathematics* 250 (2018), pp. 331–341 (cit. on p. 9).

[ES18] Martijn van Ee and René Sitters. 'The a priori traveling repairman problem'. In: *Algorithmica* 80.10 (2018), pp. 2818–2833 (cit. on pp. 6, 9, 28).

[EGRS10] Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoß and Guido Schäfer. 'Connected facility location via random facility sampling and core detouring'. In: *Journal of Computer and System Sciences* 76.8 (2010), pp. 709–726 (cit. on pp. 5, 9, 16, 28).

[FRT04] Jittat Fakcharoenphol, Satish Rao and Kunal Talwar. 'A tight bound on approximating arbitrary metrics by tree metrics'. In: *Journal of Computer and System Sciences* 69.3 (2004), pp. 485–497 (cit. on p. 82).

[FS07] Uriel Feige and Mohit Singh. 'Improved approximation ratios for traveling salesperson tours and paths in directed graphs'. In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. 2007, pp. 104–118 (cit. on p. 5).

[FS20] Finn Fernstrøm and Teresa Anna Steiner. 'A constant approximation algorithm for the uniform a priori capacitated vehicle routing problem with unit demands'. In: *Information Processing Letters* 159-160 (2020), p. 105960 (cit. on p. 9).

[FMSSSSZ14] Esteban Feuerstein, Alberto Marchetti-Spaccamela, Frans Schalekamp, René Sitters, Suzanne van der Ster, Leen Stougie and Anke van Zuylen. 'Scheduling over scenarios on two machines'. In: *Computing and Combinatorics*. 2014, pp. 559–571 (cit. on p. 9).

[FHS11] Luca Foschini, John Hershberger and Subhash Suri. 'On the complexity of time-dependent shortest paths'. In: *Proceedings of the 2011 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2011, pp. 327–341.

[FT87] Michael L. Fredman and Robert Endre Tarjan. 'Fibonacci heaps and their uses in improved network optimization algorithms'. In: *Journal of the ACM* 34.3 (July 1987), pp. 596–615 (cit. on p. 145).

[FGM82] A. M. Frieze, G. Galbiati and F. Maffioli. 'On the worst-case performance of some algorithms for the asymmetric traveling salesman problem'. In: *Networks* 12.1 (1982), pp. 23–39 (cit. on pp. 5, 54).

[FMRS22] Zachary Friggstad, Ramin Mousavi, Mirmahdi Rahgoshay and Mohammad R. Salavatipour. 'Improved approximations for capacitated vehicle routing with unsplittable client demands'. In: *Integer Programming and Combinatorial Optimization*. 2022, pp. 251–261 (cit. on p. 140).

[GMP23] Arun Ganesh, Bruce M. Maggs and Debmalya Panigrahi. 'Robust algorithms for TSP and Steiner tree'. In: *ACM Trans. Algorithms* 19.2 (2023) (cit. on p. 9).

[GGLS08] Naveen Garg, Anupam Gupta, Stefano Leonardi and Piotr Sankowski. 'Stochastic analyses for online combinatorial optimization problems'. In: *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2008, pp. 942–951 (cit. on pp. 1, 5, 6, 9).

[GS10] Robert Geisberger and Peter Sanders. 'Engineering time-dependent many-to-many shortest paths computation'. In: *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*. Vol. 14. 2010, pp. 74–87 (cit. on p. 140).

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes and Daniel Delling. 'Contraction hierarchies: faster and simpler hierarchical routing in road networks'. In: *Experimental Algorithms*. 2008, pp. 319–333 (cit. on pp. 2, 137, 139, 140, 146, 147).

[GGG15] Michel Gendreau, Gianpaolo Ghiani and Emanuela Guerriero. 'Time-dependent routing problems: a review'. In: *Computers & Operations Research* 64 (2015), pp. 189–197 (cit. on p. 140).

[GGLP21] Maha Gmira, Michel Gendreau, Andrea Lodi and Jean-Yves Potvin. 'Tabu search for the time-dependent vehicle routing problem with time windows on a road network'. In: *European Journal of Operational Research* 288.1 (2021), pp. 129–140 (cit. on p. 140).

[GK98] Michel Goemans and Jon Kleinberg. 'An improved approximation ratio for the minimum latency problem'. In: *Mathematical Programming* 82.1 (1998), pp. 111–124 (cit. on p. 9).

[GKSS10] Igor Gorodezky, Robert D. Kleinberg, David B. Shmoys and Gwen Spencer. 'Improved lower bounds for the universal and a priori TSP'. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. 2010, pp. 178–191 (cit. on p. 9).

[GGLMSS13] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski and Mohit Singh. 'Set covering with our eyes closed'. In: *SIAM Journal on Computing* 42.3 (2013), pp. 808–830 (cit. on p. 9).

[GHR06] Anupam Gupta, MohammadTaghi Hajiaghayi and Harald Räcke. 'Oblivious network design'. In: *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*. 2006, pp. 970–979 (cit. on p. 9).

[GKL24] Anupam Gupta, Gregory Kehne and Roie Levin. 'Set covering with our eyes wide shut'. In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2024, pp. 4530–4553.

[GKPR07] Anupam Gupta, Amit Kumar, Martin Pál and Tim Roughgarden. 'Approximation via cost sharing: simpler and better approximation algorithms for network design'. In: *Journal of the ACM* 54.3 (2007) (cit. on pp. 5, 9).

[GKR03] Anupam Gupta, Amit Kumar and Tim Roughgarden. 'Simpler and better approximation algorithms for network design'. In: *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*. 2003, pp. 365–372 (cit. on pp. 5, 9).

[GPRS04] Anupam Gupta, Martin Pál, R. Ravi and Amitabh Sinha. 'Boosted sampling: approximation algorithms for stochastic optimization'. In: *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*. 2004, pp. 417–426 (cit. on pp. 5, 9).

[Gut04] Ron Gutman. 'Reach-based routing: a new approach to shortest path algorithms optimized for road networks.' In: *Proc. 6th International Workshop on Algorithm Engineering and Experiments*. 2004, pp. 100–111 (cit. on pp. 140, 146).

[Gut84] Antonin Guttman. 'R-trees: a dynamic index structure for spatial searching'. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. 1984, pp. 47–57 (cit. on pp. 141, 168).

[HKLR05] MohammadTaghi Hajiaghayi, Jeong Han Kim, Tom Leighton and Harald Räcke. 'Oblivious routing in directed graphs with random demands'. In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*. 2005, pp. 193–201 (cit. on p. 9).

[HKL06] MohammadTaghi Hajiaghayi, Robert D Kleinberg and Tom Leighton. 'Improved lower and upper bounds for universal TSP in planar metrics'. In: *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*. 2006, pp. 649–658 (cit. on p. 9).

[HNR68] Peter E. Hart, Nils J. Nilsson and Bertram Raphael. 'A formal basis for the heuristic determination of minimum cost paths'. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 145).

[HS86] Sergiu Hart and Micha Sharir. 'Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes'. In: *Combinatorica* 6.2 (1986), pp. 151–177 (cit. on pp. 140, 142).

[Hei24] Paula Heinz. 'Contraction hierarchies for vehicle routing'. Master's thesis. Research Institute for Discrete Mathematics, University of Bonn, 2024 (cit. on pp. 139, 141, 149, 179).

[HZVG17] Yixiao Huang, Lei Zhao, Tom Van Woensel and Jean-Philippe Gross. 'Time-dependent vehicle routing problem with path flexibility'. In: *Transportation Research Part B: Methodological* 95 (2017), pp. 169–195 (cit. on p. 140).

[IGP03] Soumia Ichoua, Michel Gendreau and Jean-Yves Potvin. 'Vehicle dispatching with time-dependent travel times'. In: *European Journal of Operational Research* 144.2 (2003), pp. 379–396 (cit. on pp. 140, 141).

[II86] Hiroshi Imai and Masao Iri. 'An optimal algorithm for approximating a piecewise linear function'. In: *Information Processing Letters* 9.3 (1986), pp. 159–162 (cit. on pp. 140, 144).

[Jai85] Patrick Jaillet. 'Probabilistic traveling salesman problems'. PhD thesis. Massachusetts Institute of Technology, 1985 (cit. on p. 9).

[Jai88] Patrick Jaillet. 'A priori solution of a traveling salesman problem in which a random subset of the customers are visited'. In: *Operations Research* 36.6 (1988), pp. 929–936 (cit. on p. 9).

[JMMSV03] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi and Vijay V. Vazirani. 'Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP'. In: *Journal of the ACM* 50.6 (2003), pp. 795–824 (cit. on p. 9).

[JLNRS05] Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman and Ravi Sundaram. 'Universal approximations for TSP, Steiner tree, and set cover'. In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*. 2005, pp. 386–395 (cit. on p. 9).

[KLSS05] Haim Kaplan, Moshe Lewenstein, Nira Shafrir and Maxim Sviridenko. 'Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs'. In: *Journal of the ACM* 52.4 (2005), pp. 602–626 (cit. on p. 5).

[KKO23] Anna R. Karlin, Nathan Klein and Shayan Oveis Gharan. 'A deterministic better-than-3/2 approximation algorithm for metric TSP'. In: *Integer Programming and Combinatorial Optimization*. 2023, pp. 261–274 (cit. on pp. 5, 6, 31).

[KKO24] Anna R. Karlin, Nathan Klein and Shayan Oveis Gharan. 'A (slightly) improved approximation algorithm for metric TSP'. In: *Operations Research* 72.6 (2024), pp. 2543–2594 (cit. on p. 6).

[KV18] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2018 (cit. on p. 43).

[Li25] Jason Li. *Simpler and faster directed low-diameter decompositions*. 2025. arXiv: 2505.10244 (cit. on pp. 2, 54, 81, 82).

[MD92] Chryssi Malandraki and Mark S. Daskin. 'Time dependent vehicle routing problems: formulations, properties and heuristic algorithms'. In: *Transportation Science* 26.3 (1992), pp. 185–200 (cit. on p. 140).

[Man14] Simona Mancini. 'Time dependent travel speed vehicle routing and scheduling on a real road network: the case of Torino'. In: *Transportation Research Procedia* 3 (2014), pp. 433–441 (cit. on p. 140).

[NR11] Viswanath Nagarajan and R. Ravi. 'The directed orienteering problem'. In: *Algorithmica* 60.4 (Aug. 2011), pp. 1017–1030 (cit. on p. 140).

[NGN20] Fatemeh Navidi, Inge Li Gørtz and Viswanath Nagarajan. 'Approximation algorithms for the a priori traveling repairman'. In: *Operations Research Letters* 48.5 (2020), pp. 599–606 (cit. on p. 9).

[PZL21a] Binbin Pan, Zhenzhen Zhang and Andrew Lim. 'A hybrid algorithm for time-dependent vehicle routing problem with time windows'. In: *Computers & Operations Research* 128 (2021), p. 105193 (cit. on p. 140).

[PZL21b] Binbin Pan, Zhenzhen Zhang and Andrew Lim. 'Multi-trip time-dependent vehicle routing problem with time windows'. In: *European Journal of Operational Research* 291.1 (2021), pp. 218–231 (cit. on p. 140).

[PY93] Christos H. Papadimitriou and Mihalis Yannakakis. 'The traveling salesman problem with distances one and two'. In: *Mathematics of Operations Research* 18.1 (1993), pp. 1–11 (cit. on p. 6).

[PSUV20] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa and François Vanderbeck. 'A generic exact solver for vehicle routing and related problems'. In: *Mathematical Programming* 183.1 (Sept. 2020), pp. 483–523 (cit. on p. 140).

[Puh21] Luise Puhlmann. 'Vehicle routing with incomplete information'. Master's thesis. Research Institute for Discrete Mathematics, University of Bonn, 2021 (cit. on pp. 2, 7, 29–31).

[QSU21] Eduardo Queiroga, Ruslan Sadykov and Eduardo Uchoa. 'A POPMUSIC matheuristic for the capacitated vehicle routing problem'. In: *Computers & Operations Research* 136 (2021), p. 105475 (cit. on p. 140).

[Rat21] Silas Rathke. 'Time-dependent shortest paths with preprocessing'. Master's thesis. Research Institute for Discrete Mathematics, University of Bonn, 2021 (cit. on pp. 2, 139, 141, 147, 148, 151, 153).

[RSL77] Daniel J. Rosenkrantz, Richard E. Stearns and Philip M. Lewis II. 'An analysis of several heuristics for the traveling salesman problem'. In: *SIAM Journal on Computing* 6.3 (1977), pp. 563–581 (cit. on p. 5).

[SS05] Peter Sanders and Dominik Schultes. 'Highway hierarchies hasten exact shortest path queries'. In: *Algorithms – ESA 2005*. 2005, pp. 568–579 (cit. on pp. 140, 146).

[SS07a] Peter Sanders and Dominik Schultes. 'Dynamic highway-node routing'. In: *Experimental Algorithms*. 2007, pp. 66–79 (cit. on pp. 140, 146).

[SS07b] Peter Sanders and Dominik Schultes. 'Engineering fast route planning algorithms'. In: *Experimental Algorithms*. 2007, pp. 23–36 (cit. on pp. 140, 146).

[SS08] Frans Schalekamp and David B. Shmoys. 'Algorithms for the universal and a priori TSP'. In: *Operations Research Letters* 36.1 (2008), pp. 1–3 (cit. on p. 9).

[Sch20] Niklas Schlomberg. 'Scheduling in vehicle routing'. Master's thesis. Research Institute for Discrete Mathematics, University of Bonn, 2020 (cit. on p. 142).

[Ser78] A.I. Serdjukov. *Some extremal bypasses in graphs.* Upravlyaemye Sistemy 17, 76–79. [in Russian]. 1978 (cit. on p. 5).

[ST08] David Shmoys and Kunal Talwar. 'A constant approximation algorithm for the a priori traveling salesman problem'. In: *Integer Programming and Combinatorial Optimization*. 2008, pp. 331–343 (cit. on pp. 1, 5–8, 10, 16, 17, 27–29, 51).

[SWZ21] Ben Strasser, Dorothea Wagner and Tim Zeitz. 'Space-efficient, fast and exact routing in time-dependent road networks'. In: *Algorithms* 14.3 (2021) (cit. on p. 140).

[STV20] Ola Svensson, Jakub Tarnawski and László A. Végh. 'A constant-factor approximation algorithm for the asymmetric traveling salesman problem'. In: *Journal of the ACM* 67.6 (2020) (cit. on pp. 5, 64).

[Tan25] Ole Tange. *GNU Parallel 20251122 ('Mamdani')*. Nov. 2025. URL: `https://doi.org/10.5281/zenodo.17692695` (cit. on p. 175).

[THP14] Alejandro Toriello, William B. Haskell and Michael Poremba. 'A dynamic traveling salesman problem with stochastic arc costs'. In: *Operations Research* 62.5 (2014), pp. 1107–1125 (cit. on p. 9).

[TV14] Paolo Toth and Daniele Vigo. *Vehicle Routing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014 (cit. on p. 140).

[TV22] Vera Traub and Jens Vygen. 'An improved approximation algorithm for the asymmetric traveling salesman problem'. In: *SIAM Journal on Computing* 51.1 (2022), pp. 139–173 (cit. on p. 5).

[TV25] Vera Traub and Jens Vygen. *Approximation Algorithms for Traveling Salesman Problems*. Cambridge University Press, 2025 (cit. on pp. 5, 64).

[Vet09] Christian Vetter. *Parallel time-dependent contraction hierarchies*. Student Research Project. Universität Karlsruhe, 2009 (cit. on p. 141).

[Vid22] Thibaut Vidal. 'Hybrid genetic search for the CVRP: open-source implementation and SWAP* neighborhood'. In: *Computers & Operations Research* 140 (2022), p. 105643 (cit. on p. 140).

[VCGP13] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau and Christian Prins. 'A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows'. In: *Computers & Operations Research* 40.1 (2013), pp. 475–489 (cit. on p. 140).

[VS20] Thomas R. Visser and Remy Spliet. 'Efficient move evaluations for time-dependent vehicle routing problems'. In: *Transportation Science* 54.4 (2020), pp. 1091–1112 (cit. on p. 142).

[WDWZGS25] Zijin Wan, Xiaojun Dong, Letong Wang, Enzuo Zhu, Yan Gu and Yihan Sun. 'Parallel contraction hierarchies can be efficient and scalable'. In: *Proceedings of the 39th ACM International Conference on Supercomputing*. 2025, pp. 670–688 (cit. on p. 141).

[Wie86] Ady Wiernik. 'Planar realizations of nonlinear Davenport-Schinzel sequences by segments'. In: *27th Annual Symposium on Foundations of Computer Science (SFCS)*. 1986, pp. 97–106 (cit. on pp. 140, 142).

[WS11] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011 (cit. on p. 114).

[Wol80] Laurence A. Wolsey. 'Heuristic analysis, linear programming and branch and bound'. In: *Mathematical Programming Study* 13 (1980), pp. 121–134 (cit. on p. 31).

[Zuy11] Anke van Zuylen. 'Deterministic sampling algorithms for network design'. In: *Algorithmica* 60.1 (2011), pp. 110–151 (cit. on pp. 6, 7, 10, 28–31).