Routing under Uncertainty
Approximation and Complexity

Martijn van Ee

Reading committee:

prof.dr. N. Bansal
prof.dr. N. Megow
prof.dr. G. Schäfer (chair)
prof.dr. M. Uetz
dr. T. Vredeveld

The author was supported by the Netherlands Organisation for Scientific Research (NWO), grant number 612.001.215.

ISBN 978 90 3610 496 8

The book is printed by Drukkerij Haveka which also formatted the cover. The illustration on the cover is made by Remco Wetzels (www.remcowetzels.nl).

This dissertation is number 23 in the ABRI series.

VRIJE UNIVERSITEIT

Routing under Uncertainty
Approximation and Complexity

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de School of Business and Economics
op woensdag 6 december 2017 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Martijn van Ee

geboren te Zeevang

promotor: prof.dr. L. Stougie
copromotor: dr.ir. R.A. Sitters

Contents

CHAPTER 1

Introduction

This thesis covers part of the research I performed during my appointment as Ph.D. candidate. All the chapters in this thesis concern *routing problems* that deal with some sort of *uncertainty*. For these problems, my co-authors and myself investigated the *complexity* and *approximability*. This chapter gives an introduction to these optimization problems and to the field of *combinatorial optimization* in general.

1.1 Routing problems

In this thesis, we mainly consider routing problems. In the most basic setting, we are given a set of clients together with their pairwise distances. The goal in routing problems is to construct a tour connecting the clients at minimum cost. This cost can differ for each problem as is demonstrated by the following two examples of routing problems, which will play a central role in this thesis.

In the *traveling salesman problem* (TSP), the goal is to construct a tour visiting all clients of minimum total length. Here, the length is the sum of the distances between every pair of consecutive clients in the tour. The TSP is one of the most well-known problems in combinatorial optimization and it clearly has many practical applications.

In the *traveling repairman problem* (TRP), the goal is to find a tour connecting all clients having the minimum total latency. Here, the latency of a client is equal to the length of the path between a designated start location and the client. The total latency is the sum of all latencies. Although this problem looks very similar to the TSP, it needs different approaches and handling. A further comparison of the two problems shows that the TSP could be considered as server-based, since it minimizes the time that a server visiting the clients is busy (assuming it moves at unit speed), whereas TRP could be considered as client-based, since it minimizes the average arrival time at a client.

Formally, an instance of the TSP and the TRP is defined as a set of n points with distances c_{ij} for $i, j = 1, \ldots, n$, where c_{ij} denotes the distance between point i and j. A tour can be characterized as a permutation τ of the points. Any cyclic permutation of τ represents the same tour. The length of tour $\tau = (\tau(1), \ldots, \tau(n))$ is equal to

$$\sum_{i=1}^{n-1} c_{\tau(i),\tau(i+1)} + c_{\tau(n),\tau(1)}.$$

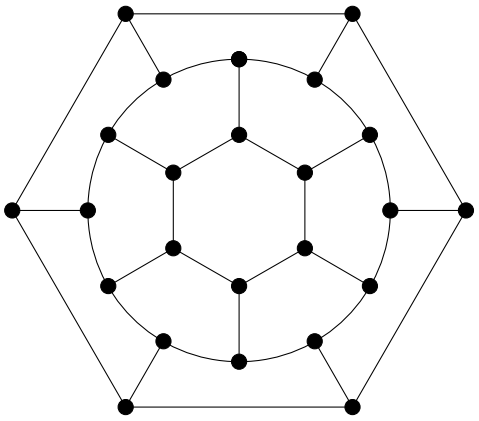


Fig. 1.1: Example of a graph.

In TRP, we have a specific point, called the root r, from which we need to start. Now, the total latency of tour $\tau = (\tau(1), \ldots, \tau(n))$ with $\tau(1) = 1 = r$ is

$$c_{\tau(1),\tau(2)} + (c_{\tau(1),\tau(2)} + c_{\tau(2),\tau(3)}) + \ldots + (c_{\tau(1),\tau(2)} + \ldots + c_{\tau(n-1),\tau(n)}) = \sum_{i=1}^{n-1} (n-i) c_{\tau(i),\tau(i+1)}.$$

Usually, we assume that distances obey the triangle inequality, i.e., $c_{ij} \leq c_{ik} + c_{kj}$ for all i, j, k. If so, we say that the points lie in a metric space.

Sometimes the metric space is induced by a graph. A graph $G = (V, E)$ is a set of vertices V and a set of edges E between vertices. An example of a graph is presented in Figure 1.1. If the metric space is induced by a graph, it means that the distances correspond to shortest paths, in terms of number of edges, between vertices in the graph. Note that such an instance satisfies the triangle inequality by its definition. Thinking this way, another way to represent the points and the distances for the general TSP is through a complete weighted graph, i.e., there is an edge between each pair of vertices, with a weight function that assigns a positive integer number to each edge, i.e., $c : E \to \mathbb{Z}_+$. Edge weights can be interpreted as length. Therefore, when considering routing problems, we may also speak of edge lengths instead. In this thesis, we will talk about vertices or points instead of clients.

A Hamiltonian cycle in a graph is a tour that uses only edges to go from one vertex to another, and visits each vertex exactly once. Now, suppose our metric is induced by the graph in Figure 1.1. If a Hamiltonian cycle exists, it is the optimal solution for the TSP. Moreover, any tour that is not a Hamiltonian cycle is not optimal in this case. It is left as an exercise to the reader to determine whether the graph in Figure 1.1 contains a Hamiltonian cycle. (The answer can be found on page 107.)

1.2 Routing under uncertainty

The traveling salesman problem is an example of a deterministic problem. In deterministic problems, it is assumed that every aspect of the input is certain. On the other hand, we have stochastic problems. In these problems, we incorporate uncertainties in the input of a problem. For example, one could think of TSP where the distances are uncertain [78]. This could be motivated practically by congestion in traffic. The

problems addressed in this thesis lie in the domains of *a priori routing* [19] and *graph search* [74]. In a priori routing, it is uncertain which subset of the vertices needs to be visited. In graph search, we have to visit one vertex but is not known beforehand which one it is. We will now define these domains more formally.

1.2.1 A priori routing

Suppose you are faced with a specific routing problem every day, but each day the set of vertices to be visited is different. The natural approach would be to optimize your route every day. However, this may be impractical or even impossible in certain cases (like in an application described in Chapter 3). In a priori routing, we choose to make just one tour on all the vertices that could possibly appear in one of the instances. Let us call this tour the first-stage tour, denoted by τ. At the start of a day, we see which vertices need to be visited, called the active set which is denoted by A. Then, we shortcut the first-stage tour to the active set (Figure 1.2). Let us call this tour the second-stage tour, denoted by τ_A, which follows directly from the first-stage tour and the active set. We also get a probability distribution over the active sets, i.e., a function $p : 2^V \to [0,1] \cap \mathbb{Q}$ with $\sum_{A \subseteq 2^V} p(A) = 1$, where 2^V denotes the set of all subsets of V (the power set). Our goal is now to find a first-stage tour that minimizes the expected cost (with respect to p) of the second-stage tour.

We now define a priori TSP and TRP. In *a priori TSP*, the goal is to find a first-stage tour minimizing the expected length of the second-stage tour. In *a priori TRP*, the goal is to find a first-stage tour minimizing the expected total latency of the second-stage tour. The problems we consider differ further in the way the probability distribution is specified. In the literature, basically three models are used.

In the black-box model [94], we do not know the probability distribution, but we can ask an oracle to generate a sample. In this thesis, we assume we have some more knowledge by using either the scenario or the independent decision model. In the scenario model [44], we are given an explicit list of active sets, $\{S_1, \ldots, S_m\}$, called scenarios, and a probability distribution over the scenarios. This model will be used in Chapter 3, 4 and 5. The third model used in the literature is the independent decision model [95]. Here, each vertex has its own probability of being active, i.e., being part of the active set, independent of the other vertices. This model will be used in Chapter 2 and 5.

It is important to note that the a priori problems are generalizations of the standard problems. In the scenario model, the standard problem can be modeled as the a priori problem with one scenario that contains all vertices. In the independent decision model, we set all probabilities equal to 1 to model the standard problem.

In Chapter 2, we consider a priori TRP in the independent decision model. The a priori TSP in the scenario model is handled in Chapter 3. A related problem, called the master tour problem (in the scenario model), is treated in Chapter 4.

1.2.2 Graph search

The most common setting for searching in graphs is the following. We are given an edge-weighted graph and we imagine a target is hidden at one of the vertices. We are given a probability distribution p over the vertices and the target is at vertex i

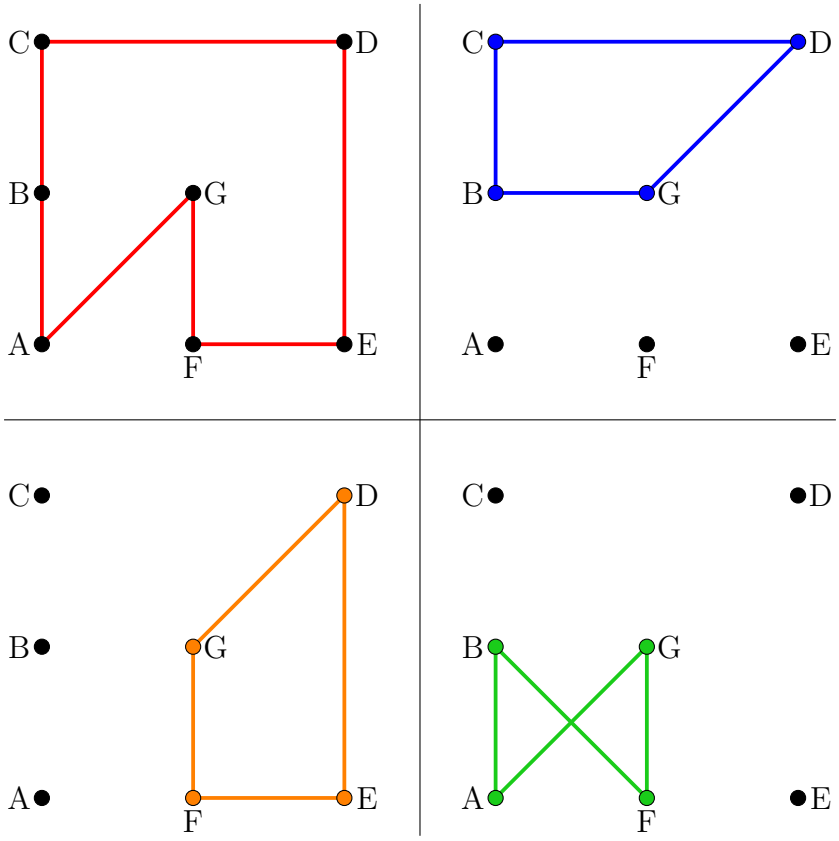


Fig. 1.2: Illustration of shortcutting. Top-left: tour τ. Top-right: tour τ_X, where $X = \{B, C, D, G\}$. Bottom-left: tour τ_Y, where $Y = \{D, E, F, G\}$. Bottom-right: tour τ_Z, where $Z = \{A, B, G, F\}$.

with probability p_i, where $\sum_i p_i = 1$. Our goal is to find the target, starting from a designated root vertex, as fast as possible, i.e., we want to construct a walk in the graph minimizing the expected length until finding the target. This problem is called the *graph search problem*. In Figure 1.3, an example of an instance of the graph search problem and two walks are illustrated.

Note that if $p_i = \frac{1}{n}$ for all $i \in V$, where $n = |V|$, the walk minimizing the expected length until finding the target is the same as the walk minimizing the total latency. Hence, TRP reduces to the graph search problem. The graph search problem is generally known as the weighted TRP, since the probabilities p_i can be modeled as weights on the vertices. This problem turns out be efficiently solvable on a star or a line graph.

In Chapter 6, we generalize the graph search problem on the line graph to more general distributions. More specifically, we also allow continuous distributions on the real line. This chapter is named after the *lost cow* problem, since one can think about this problem as if a cow is searching for a gate in a fence, where the cow only has probabilistic information about the location of the gate. In Chapter 7, we consider the complexity of TRP on graphs that look like stars. Therefore, we define the graph parameter *starwidth*. This parameter measures to what extend the graph looks like a star graph in a similar way as treewidth [23] measures to what extend the graph looks like a tree.

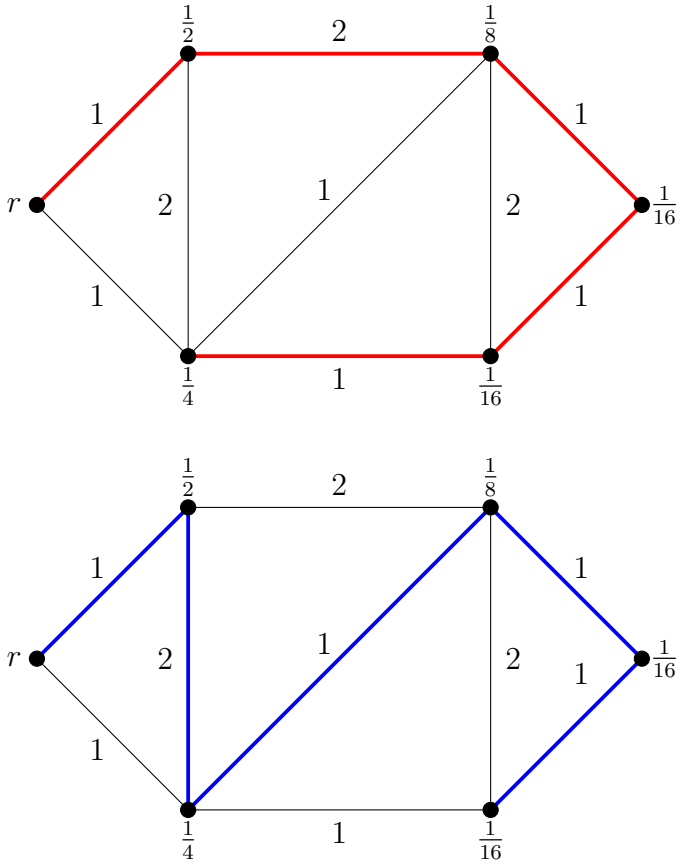


Fig. 1.3: Illustration of two walks. Top: red path with expected length $\frac{1}{2} \cdot 1 + \frac{1}{8} \cdot 3 + \frac{1}{16} \cdot 4 + \frac{1}{16} \cdot 5 + \frac{1}{4} \cdot 6 = \frac{47}{16}$. Bottom: blue path with expected length $\frac{39}{16}$.

In Chapter 5, we consider the *Canadian traveler problem* (CTP). Later, we will explain why this problem is a graph search problem. In the CTP, we are given an edge-weighted graph, where one vertex is the start vertex and one is the terminal vertex, denoted by s and t respectively. We have to construct a short walk from s to t. However, edges may not be present and we only discover this when we are at one of its incident vertices. More precisely, we are given a probability distribution over the edge sets, i.e., a function $p : 2^E \to [0, 1] \cap \mathbb{Q}$ with $\sum_{A \subseteq 2^E} p(A) = 1$, where with probability $p(A)$ only the edges in A are present. Our goal is now to design a policy that minimizes the expected length of the walk. Here, a policy may use the observed realizations as input to decide where to go next. A policy differs from a walk since a policy can condition its actions based on observed realizations. We consider the CTP in both the scenario and the independent decision model. In Chapter 5, we restrict our solution space by only considering non-adaptive solutions, i.e., policies that do not condition on past observations. However, this is only well-defined in very specific cases. For example, when the graph without t and its incident edges is a tree, a non-adaptive solution is just an ordering of the vertices. A closer look at the non-adaptive CTP shows that it can be seen as a generalization of the graph search problem to the case with multiple targets and failing edges.

1.3 Combinatorial optimization

The routing problems in this thesis are examples of combinatorial optimization problems. Here, we give an introduction in combinatorial optimization by discussing three famous combinatorial optimization problems and their applications in practice. These problems are called *bin packing*, the *assignment problem* and *minimum cut*. Let us first formally define an optimization problem.

Definition 1.1. *An optimization problem* Π *is a three-tuple* $(\mathcal{I}, \mathcal{S}, f)$*, where*

1. $\mathcal{I}$ *is the set of problem instances,*
2. *every instance* $I \in \mathcal{I}$ *has a set of solutions* $\mathcal{S}(I)$*,*
3. *every solution* $x \in \mathcal{S}(I)$ *has a value* $f(x, I) \in \mathbb{R}$*.*

The objective is to determine for a given instance I *if there exists a solution, and if so, either to find a solution that has the smallest value among all solutions for* I*, in which case we call* Π *a minimization problem, or to find a solution that has the largest value among all solutions for* I*, in which case we call* Π *a maximization problem.*

We consider optimization problems in which the solution set is finite for every instance. This type is called a combinatorial optimization problem, in this thesis often abbreviated to problem. An example of a combinatorial optimization problem is the TSP, as introduced in Section 1.1. Here, the problem instance is defined by an edge-weighted graph, the solutions are tours on all vertices and the value of a solution is the tour length. Three other problems are bin packing, the assignment problem and minimum cut.

Bin packing In transportation, one of the major challenges is to efficiently pack our vehicles. In the bin packing problem, we are given a set of items. Each item has a weight and we want to transport the items. The items have to be packed into bins, think of this as containers. Each bin can carry a maximum amount of weight, called the capacity of the bin, which is equal for all bins. The goal is to pack the items into a minimum number of bins. The problem is an oversimplified version of reality, but is already challenging when considered on a large scale.

Using the terminology of Definition 1.1, an instance is a list of item weights and a capacity. A solution is a packing and the value of the solution is the number of bins used. An example of an instance of bin packing is the instance where bins have capacity 20 and items have weights 11, 11, 7, 7, 6, 6, 6, 3 and 3. As an exercise, try to pack these items into a minimum number of bins such that no bin exceeds the capacity of 20. (The answer can be found on page 108.)

Assignment problem In the assignment problem, we have a set of people and an equally sized set of tasks to perform. We have to assign tasks to people in the best way. Suppose we are facing the problem of selecting swimmers for a 4×50 meters relay medley. There are four swimmers and we only need to assign disciplines to swimmers. For each swimmer, we know his personal record for each discipline as shown in Table 1.1. Let us assume that the swimmers are able to swim exactly this time at the race.

We now have to decide which swimmer competes on which discipline. Our goal is to minimize the total time (and to win the race). The assignment problem and its generalizations obviously have applications in other domains, e.g., managers assigning jobs to working staff. In general, an instance of the assignment problem consists of a square matrix with values, a solution is a pairing of rows and columns and the value of the solution is the sum of the chosen values. As an exercise, one could try to assign one swimmer to each discipline such that the total time is minimized. (The answer can be found on page 109.)

	A	B	C	D
Backstroke	39	33	38	35
Breaststroke	34	41	34	33
Butterfly	29	30	33	31
Freestyle	28	27	30	29

Tab. 1.1: Swimmers and their personal records.

Minimum cut The minimum cut problem finds its origin in military applications. In this problem, we are facing an edge-weighted graph in which our enemy wants to travel from location s to location t. An example of a minimum cut-instance is depicted in Figure 1.4, where the numbers along the edges represent the importance of the edge. If we look at it as a rail network, this number can be interpreted as the number of parallel tracks between the two locations. Our goal is to disrupt the network by deleting edges, i.e., bombing tracks. We want to do this as efficient as possible, i.e., we delete the set of edges with minimum total weight. Define $\delta(X)$ as the set of edges with one endpoint in X and one not in X, also called the cut separating X from $V \setminus X$. Feasible solutions to the minimum cut problem are cuts with $s \in X$ and $t \in V \setminus X$. The cut with minimum total weight is called the minimum cut. This problem was studied by [63], who considered the train network of the Warschau Pact that connected Moskow with the capitals of satellite countries. Following Definition 1.1 of combinatorial optimization problems, an instance is an edge-weighted graph with special vertices s and t, a solution is a cut and the value of the solution is the sum of the edge weights in the cut. Although this problem turns out to be efficiently solvable, it is not trivial to find the minimum cut. As a final exercise, try to find the minimum cut of the network in Figure 1.4. (The answer can be found on page 110.)

1.4 Complexity and approximation

For the problems considered in this thesis, we will study their complexity and approximability. In this section, we will formalize both these terms.

1.4.1 Complexity

Recall the definition of a combinatorial optimization problem (Definition 1.1). Given a problem, we would like to tackle it using an algorithm. This is a step-by-step procedure that for each instance I outputs a solution $x \in \mathcal{S}(I)$. We call an algorithm *exact* if

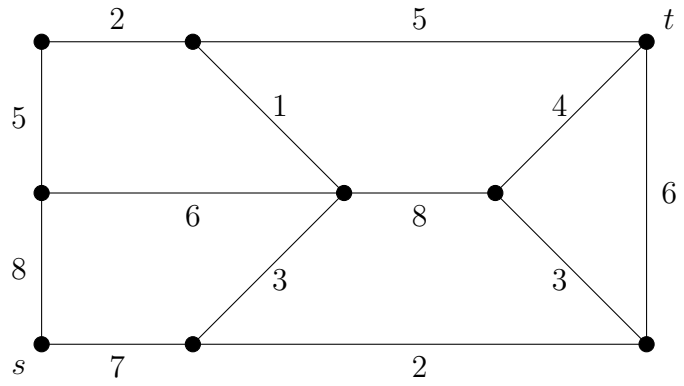


Fig. 1.4: An edge-weighted graph with special vertices s and t. In the minimum cut problem, we have to delete edges such that there is no path from s to t. Such a set of edges of minimum weight is called a minimum cut.

it always outputs the optimal solution, the solution with either the smallest or the largest value (depending on the problem) among all solutions. An example of an exact algorithm for TSP is complete enumeration. Here, you try all possible tours and return one with the smallest tour length. However, for instances with a lot of vertices, this might be a very time-consuming approach. Therefore, we define the notion of an efficient algorithm.

We say an algorithm is *efficient* when the number of computational steps it needs to output a solution, i.e., the running time of the algorithm, is bounded by a polynomial function of the size of the instance. The size of an instance, denoted by $|I|$, is the number of bits needed to represent the instance. For example, if we have an instance of TSP with n vertices and each distance is at most C, then $|I| \leq n^2 \log_2 C$.

For counting computational steps and quantifying the input size, we use big O notation [57]. For two real-valued functions g and h, we say that $g = O(h)$ if there is a positive constant ζ and a real number x_0 such that $g(x) \leq \zeta h(x)$ for all $x \geq x_0$. Similarly, we say that a function $g = \Omega(h)$ if there is a positive constant ζ and a real number x_0 such that $g(x) \geq \zeta h(x)$ for all $x \geq x_0$. We write $g = \Theta(h)$ if $g = O(h)$ and $g = \Omega(h)$. Now, we can say that the size of the aforementioned TSP-instance is $O(n^2 \log C)$. Moreover, the running time of the complete enumeration algorithm is $\Omega((n-1)!)$, since there are exactly $(n-1)!/2$ different tours on n points. This is clearly not a polynomial function of the input size. Of course, we want to find an efficient exact algorithm for TSP. To decide whether or not this is possible, we need computational complexity theory.

In computational complexity theory, we classify the hardness of decision problems. These are problems for which the answer is either yes or no. We say that an algorithm for a decision problem is exact if it always outputs the correct answer. Note that for each optimization problem, we can define a corresponding decision problem. For example, given a TSP-instance and a number B, is there a tour with length at most B? If we have an efficient exact algorithm for the optimization problem, we also have one for the decision problem. Under some mild restriction on the objective function f, this also works the other way around. By trying multiple values for B and using binary search, we can use the algorithm for the decision problem to solve the optimization

problem.

The class P (short for polynomial-time) contains all decision problems that have an efficient exact algorithm. Usually, we say that these problems are solvable. Examples of problems in this class are the assignment problem and the minimum cut problem. So far, no one has been able to find an efficient exact algorithm for TSP. However, this does not show that there is none. In order to provide evidence for the non-existence of such an algorithm, the theory of NP-completeness has been developed [51].

Formally, the class NP (short for non-deterministic polynomial-time) contains all decision problems that have a polynomially bounded certificate for a yes-instance that can be checked in polynomial time. To make this more concrete, we show that the decision version of TSP is contained in NP. A certificate for a yes-instance is a tour on all the vertices with length at most B. Since we need only $O(n \log n)$ bits to represent this solution, we have a polynomially bounded certificate. We can check whether this is indeed a solution by simply taking the sum of the edge weights, which will take $O(n \log C)$ time, where C is again the largest distance in the instance. Hence, we have shown that TSP is in NP. It is easy to see that a lot of natural problems are contained in NP. Also oberve that P is contained in NP. It is still open whether or not P=NP. It is widely believed that P≠NP.

Before giving the definition of NP-completeness, we have to define a polynomial reduction. We say that a problem Π polynomially reduces to problem Π' if there exists a polynomial-computable function that maps any instance I of Π to an instance I' of Π' such that I is a yes-instance for Π if and only if I' is a yes-instance for Π'. Hence, if Π reduces to Π' then any efficient exact algorithm for Π' implies an efficient exact algorithm for Π.

Definition 1.2. *Decision problem* Π *is NP-hard if all problems in NP polynomially reduce to* Π*. If* Π *is NP-hard and contained in NP, we say that* Π *is NP-complete.*

If we are able to give an efficient exact algorithm for an NP-complete problem, we can use this to solve all problems in NP in polynomial time. Hence, this would imply that P=NP. Since the latter is unlikely, we may say that the NP-complete problems are the hardest among the problems in NP. To show that a decision problem Π is NP-complete, it is sufficient to show that a known NP-complete problem polynomially reduces to Π and that it is contained in NP. In 1971, it was shown by Cook [29] that the satisfiability problem is NP-complete. Using the approach above, many problems have been proven to be NP-complete, e.g., Karp [70] showed NP-completeness of 21 classical problems. As a consequence of the work of Karp, the TSP and the bin packing problem are NP-hard problems (their decision versions are NP-complete).

There are many optimization problems with numbers in their input. For example, in the TSP there are weights on the edges. We call a decision problem strongly NP-complete if it remains NP-complete when the numbers are polynomially bounded in the remaining input size. The TSP is an example of a strongly NP-hard problem, since it is already NP-hard if all the distances are either 1 or 2 [70]. If a problem is NP-hard, but we were not able to prove strong NP-hardness, we say that a problem is weakly NP-hard. Often we omit the word 'weakly'. We say that a problem is pseudopolynomial solvable if there is an exact algorithm with a running time that is polynomial in the input size and the largest number. Note that, by definition, a

strongly NP-hard problem cannot have a pseudopolynomial exact algorithm, unless P=NP.

Finally, we would like to mention that there are more complexity classes than just P and NP. In this thesis, classes like Δ_2^p and PSPACE will be used. Completeness in these classes is defined similarly to NP-completeness. In the final chapter, we will also use the class FPT, which further classifies our ability to give 'good' algorithms for NP-hard problems. All these extra classes will be defined where they appear.

1.4.2 Approximation

In the previous section we saw that for some problems, like the TSP, it is not possible to find an efficient exact algorithm. However, we might be able to design an efficient algorithm that always comes provably close to the optimal solution. This is formalized for minimization problems by the following definition.

Definition 1.3. *An algorithm* ALG *is an α-approximation for a minimization problem Π if for any instance I with a feasible solution, the value returned by the algorithm,* ALG*(I), is at most α times the optimal value,* OPT*(I), for instance I, i.e.,* ALG*$(I) \leq \alpha$*OPT*(I) for all $I \in \mathcal{I}$. Moreover,* ALG *has to run in polynomial time.*

The value α is called the approximation ratio, approximation guarantee or performance guarantee. Because this definition is concerned with minimization problems, we have $\alpha \geq 1$. The algorithm is exact if and only if $\alpha = 1$. An example of an approximation algorithm is the $\frac{3}{2}$-approximation for metric TSP by Christofides [27].

There are problems which allow approximation algorithms that output solutions with a value that is arbitrarily close to the optimal solution. Such an algorithm is called a polynomial-time approximation scheme (PTAS). Formally, a PTAS is a family of algorithms $\{A_\epsilon\}_{\epsilon>0}$ for problem Π such that A_ϵ is an efficient $(1+\epsilon)$-approximation. For TSP, there is a PTAS when the metric space is the Euclidean plane [3]. Note that in the definition of a PTAS, the running time is not measured in terms of ϵ, that means, it is for example allowed to have a running time that is exponential in $\frac{1}{\epsilon}$. If the running time depends polynomially on $\frac{1}{\epsilon}$, we say the family of algorithms is an FPTAS.

Similar to proving the hardness of finding optimal solutions, we can also prove hardness of finding approximate solutions. This is done by using so called approximation preserving reductions. For example, it has been shown that metric TSP cannot have an approximation algorithm with guarantee less than $\frac{123}{122}$, unless P=NP [71]. In Chapter 3, we will use such reductions. Sometimes, we can prove stronger inapproximability results under other assumptions than P$\neq$NP. In this thesis, we will also use the unique games conjecture [73]. We will elaborate some more on this conjecture in Chapter 3.

1.5 Outline

We end this introduction by giving an overview of the results in this thesis. Each chapter concerns problems in routing under uncertainty. As explained before, this can either be a problem in the a priori setting or a graph search problem. For these problems we consider the computational complexity and the approximability.

In Chapter 2 we study the a priori traveling repairman problem in the independent decision model. Using a series of reductions to other routing problems, we were able

to give a constant-factor approximation for the uniform case, i.e., when $p_i = p$ for all $i \in V$. We were the first to study this problem and therefore the first to obtain an approximation algorithm.

Chapter 3 concerns the complexity and approximability of the a priori traveling salesman problem in the scenario model. We show that a natural lower bound on the optimal value, the master tour lower bound, will not help us to beat the $O(\log n)$-approximation guarantee that was known from prior work. We show that the problem is even NP-hard when the scenarios are small, i.e., when $|S_j| \leq 4$ for all j. We also link the problem to the so-called permutation constraint satisfaction problems, which leads to the fact that there is no algorithm with guarantee less than $\frac{41}{30}$, unless the unique games conjecture fails. When scenarios are big, i.e., $|S_j| \geq n - \xi$ for some constant ξ, we show that the optimal solution on all vertices is a constant-factor approximation. We also show how to obtain a 9-approximation for nested scenarios, i.e., when $S_1 \subseteq \ldots \subseteq S_m$. Finally, we show that if we are able to find an approximation algorithm for a similar a priori minimum spanning tree problem, we also obtain an approximation algorithm with the same guarantee (up to a factor 2) for a priori TSP. The a priori minimum spanning tree problem turns out to be NP-hard.

In Chapter 4, we consider the complexity of the *master tour problem* and other related problems. The problem is closely related to the a priori traveling salesman problem in the scenario model, discussed in Chapter 3. In the master tour problem, we are given a graph and a set of scenarios. The question is whether there exists a tour such that the shortcut solution on each scenario has the same value as the optimal solution on just that scenario. We show that this problem is Δ_2^p-complete.

The Canadian traveler problem is investigated in Chapter 5. We study both the independent decision and the scenario model. For the independent decision model, we show NP-hardness for series-parallel graphs, solving an open question posed in the literature. Moreover, we investigate what it means to be non-adaptive and show results on the implications of being non-adaptive through analyzing the adaptivity gap. For the scenario model, we show NP-hardness on disjoint-path graphs and cactus graphs. Although this chapter is named after the Canadian traveler problem, it mainly concerns the *multi-target graph search problem*, which can be reduced easily from a special case of CTP. For this problem, in the independent decision model, we show NP-hardness on trees and give a 14.4-approximation for general metrics. In the scenario model, the problem is already NP-hard on stars.

Chapter 6 deals with the *lost cow problem*. This problem can be seen as a non-discrete generalization of the graph search problem on the line graph to the real line. For symmetric probability distributions, we characterize for which of them it is optimal not to turn, i.e., to walk to the end of the line on one side and then to the other end. We generalize this result to more than two directions and also consider non-symmetric distribution functions.

In Chapter 7, we introduce the graph parameter *starwidth*. It naturally measures to what extend a graph looks like a star graph in the sense that it is the star-equivalent of treewidth [23]. Our main goal in introducing this parameter was to extend the polynomial solvability of TRP on stars to graphs that have small starwidth. Unfortunately, TRP turns out to be NP-hard on trees of starwidth 4. In this chapter, we consider characterizations of graphs with small starwidth, the complexity of computing the starwidth of a given graph and the relation between other graph parameters.

Finally, we show three problems that are NP-hard on trees and remain NP-hard on trees with small starwidth.

To summarize this introductory chapter, the connection between the different chapters is illustrated in Figure 1.5. Here, one can see that Chapter 2 combines the two domains by considering a graph search problem in the a priori setting. The first three chapter of this thesis discuss a priori routing, whereas the last three concern graph search problems.

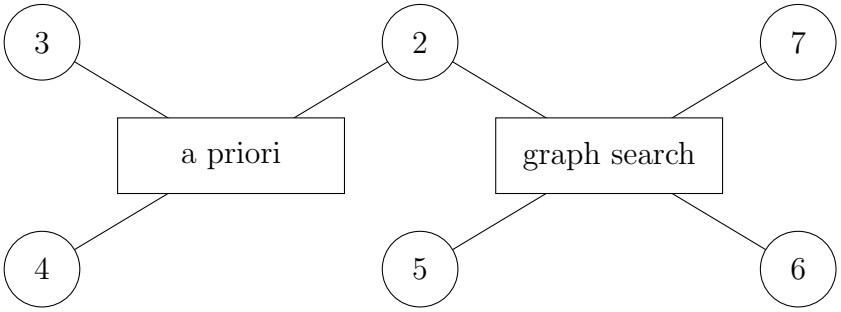


Fig. 1.5: Connection between chapters.

CHAPTER 2

The a priori traveling repairman problem

The results in this chapter were published in [41] and [39].

2.1 Introduction

In the last few decades, a lot of research has been done in stochastic combinatorial optimization. This field is concerned with classical combinatorial optimization problems, like the shortest path problem and the minimum Steiner tree problem, but with additional uncertainty in the instance. For example, there are situations where the problem instance changes on a daily basis. Instead of reoptimizing every instance, because it might be impossible or undesirable, one can alternatively choose to pick one solution that will be good on average. This is the setting of a priori optimization. In this chapter, we consider the *a priori traveling repairman problem* (a priori TRP). In our setting, the uncertainty is in the subset of vertices to be visited, modeled as a probability distribution over subsets of vertices.

Thus, in a priori routing, we are given a complete weighted graph $G = (V, E)$ and a probability distribution on subsets of V. Depending on the model, this distribution is either given or unknown. It is assumed that the instances are metric. In the first stage, a tour τ on V has to be constructed. In the second stage, an active set $A \subseteq V$ is revealed, which is the set of vertices to be visited. The second-stage tour τ_A is obtained by shortcutting the first-stage tour over the active set. For each active set, the first-stage tour has a second-stage objective value. The goal is to find a first-stage tour that minimizes the expected cost of the second stage tour. When it is clear form the context, we may refer to this expected second-stage cost simply as the expected cost of the solution.

As discussed in Chapter 1, several models for the probability distribution over the active sets are used. Here, we consider the independent decision model, where each vertex has its own probability of being active, independent of the other vertices. The special case where all probabilities are equal, i.e., $p_i = p$ for all i, is called the uniform model. Recall that in the black-box model, we can only sample from the distribution.

In the *a priori traveling salesman problem* (a priori TSP), the goal is to minimize the expected length of the tour. The problem was introduced in the PhD-theses of Jaillet [68] and Bertsimas [19]. An approximation algorithm was achieved by Schalekamp and Shmoys [94], who showed that there is a $O(\log n)$-approximation algorithm in the black-box model. Later, Gorodezky et al. [58] showed that no deterministic al-

gorithm can beat this bound. Constant-factor approximations were achieved for the first time by Shmoys and Talwar [95], who showed that there exists a randomized 4-approximation and a deterministic 8-approximation in the independent decision model. The deterministic approximation guarantee was later improved to 6.5 by Van Zuylen [105]. The authors of [53] independently obtained a constant-factor approximation for a priori TSP in the independent decision model.

The standard (not a priori) *traveling repairman problem* (TRP) is also known as the *minimum latency problem*. Here, we are given a complete graph $G = (V, E)$, a metric weight function c over the edges and a root vertex r. We want to find a tour τ starting at the root which minimizes the total latency. Here, the latency of a vertex i is defined as the length of the path from r to i along τ. The problem is known to be NP-hard in general [93] and it is even NP-hard on weighted trees [96]. The best known approximation guarantee is 3.59 for general metrics [25] and there is a polynomial time approximation scheme for the Euclidean plane and weighted trees [97]. The a priori traveling repairman problem is defined similarly to the a priori traveling salesman problem. The goal is to find a first-stage tour which minimizes the expected second-stage total latency. Here, the second-stage total latency for active set A is obtained by shortcutting the first-stage tour over A and summing up the latencies in the second-stage tour.

In this chapter, we establish a constant-factor approximation for the a priori traveling repairman problem in the uniform model. In the next section, we will discuss some papers that play a central role in this chapter. In Section 2.3, the basic ideas for our algorithm will be discussed. After that, it will be shown how the *a priori k-TSP* can be used to obtain a constant-factor approximation for a priori TRP on trees. In Section 2.6, we will discuss how to get a constant-factor approximation for the a priori TRP on general metrics. In order to get there, we investigate the *tour single-sink rent-or-buy problem* and its prize-collecting version. The a priori k-TSP and the tour single-sink rent-or-buy problem will be defined in their corresponding sections. Finally, we end with some remarks on open problems.

2.2 Background knowledge

Let us first discuss some papers that play a central role in this chapter. We are going to adapt the results of these papers to our setting, whereas results from other papers (like [99]) are used as black box. Therefore, we have dedicated a separate section for these two papers. First, we discuss the 4-approximation for a priori TSP in the independent decision model by Shmoys and Talwar [95]. Then, we consider the algorithms by Blum et al. [21] and Goemans and Kleinberg [54] for the traveling repairman problem.

The algorithm by Shmoys and Talwar fits in the framework of 'sample and augment'-algorithms [60]. It starts with sampling a set of vertices using the given probabilities. Now, the algorithm builds a tour on this sampled set, for example by using a double-tree algorithm (as was done in [95]). The other, non-sampled, vertices are connected to the tour by adding two edges from each vertex to the closest vertex in the sampled set. It is easy to show that the randomized 4-approximation can be improved to a factor $\alpha + 2$ by replacing the double-tree subroutine in the algorithm of Shmoys and Talwar by an α-approximation algorithm for TSP. Hence, using Christofides' algorithm [27] gives a randomized 3.5-approximation. We can summarize the analysis of

Shmoys and Talwar by Equation (2.1). Here, ALG and OPT denote the value of the solution of the algorithm and the optimal solution respectively. The tour constructed on sampled set X is denoted by $\textsc{Tour}(X)$ and $D_i(X)$ denotes the shortest edge from i to a vertex in $X \setminus \{i\}$. We get

$$\textsc{Alg} \leq \textsc{Tour}(X) + 2\sum_{i \notin X} D_i(X) \leq (\alpha + 2)\textsc{Opt}. \tag{2.1}$$

Unfortunately, we have not been able to extend this approach to a priori TRP in the independent decision model. In this chapter, we therefore follow the literature on TRP. An important result, used again in Chapter 5, is the algorithm by Blum et al. [21]. It uses the intuition that a good solution for TRP should visit vertices close to the root early. The algorithm uses a subroutine in which one has to find a tour of a certain length containing as many vertices as possible. For now, suppose there is an exact algorithm for this problem. Let $L_j = 2^j$ for $j = 0, 1, \ldots$. Now, the algorithm constructs a tour of length at most L_j containing as many vertices as possible for all j until all vertices are visited. Then, the algorithm concatenates these tours, i.e., it visits these tours in increasing order of j and shortcut already visited vertices.

One can show easily that this approach will result in a constant-factor approximation. Suppose that the ℓth vertex in the optimal tour is visited in the interval $(2^j, 2^{j+1}]$. Then, we know that there exists a tour containing ℓ vertices of length at most 2^{j+2}, and hence our subroutine will find a tour of length at most 2^{j+2} with at least ℓ vertices. We then observe that the ℓth vertex of the algorithm's tour is visited latest at time $2^0 + 2^1 + \ldots + 2^{j+2} < 2^{j+3}$, which is at most a factor 8 larger than the latency of the optimal tour at its ℓth vertex. Summing up over all vertices, we obtain an 8-approximation.

Goemans and Kleinberg [54] showed that by using randomization one can decrease this factor for TRP to 3.59. This is done by picking a random starting length, i.e.,, the length of the first tour, and a random direction for each tour. Finally, note that we do not have an exact algorithm for the subroutine. However, if there is a β-approximation for k-TSP, the problem of finding a minimum length tour containing k vertices, we can approximately solve the subproblem. This will result in an approximation guarantee of 3.59β. As said before, we will use this approach to tackle the a priori traveling repairman problem.

2.3 Preliminaries

The a priori traveling repairman problem in the independent decision model is formally defined as follows. We are given an edge-weighted graph $G = (V, E)$ with weight function c. The graph has a designated start vertex called the root $r = 1$. Each vertex has a probability of being active, i.e., vertex i is active with probability p_i for $i = 2, \ldots, n$. The goal is to construct a tour on V, starting at the root, minimizing the expected total latency of the second-stage tour. The root is active with probability 1, but since it has latency zero in any tour we may abuse notation by giving the root a probability $p_1 \neq 1$.

In this chapter, it is assumed that the edge weights are non-negative integers satisfying the triangle inequality. Moreover, we assume that the smallest edge weight is

equal to 1. In the following, we denote an active set of vertices by A. When the set is drawn from a probability distribution, we denote the expectation with respect to this distribution as $\mathbb{E}_A[\cdot]$.

The a priori TRP is NP-hard since the standard TRP is a special case of it (as observed in Section 1.2.1). However, it is not obvious that its decision version is also in NP. In the decision version of the a priori traveling repairman problem in the independent decision model, we are an instance of the optimization version of a priori TRP and a number B. The question is whether there exists a tour, starting at the root, that has an expected total latency of at most B. The next theorem shows that this decision version is contained in NP. Since we already know that a priori TRP is NP-hard, we can say that the decision version is NP-complete.

Theorem 2.1. *The decision version of a priori TRP in the independent decision model is in NP.*

Proof. We will show that for a given solution τ we can compute its value in polynomial time. Assume w.l.o.g. that $\tau = (1, 2, \ldots, n)$, where $r = 1$. The contribution of edge (i, j), with $i < j$, to the objective value is equal to

$$c_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1 - p_k) \sum_{k=0}^{n} \mathbb{P}(Succ(j) = k)(k + 1), \tag{2.2}$$

where $\mathbb{P}(Succ(j) = k)$ is the probability that exactly k vertices after j on τ are active. If we can compute these probabilities in polynomial time, then we can compute (2.2) in polynomial time for every edge and sum over all edges. To compute the previously mentioned probability for a given j, we define sets $S_t = [t] \setminus [j]$ for $j \leq t \leq n$, where $[n] = \{1, \ldots, n\}$. Let $\mathbb{P}(S_t, k)$ be the probability that there are exactly k active vertices in set S_t. In the end, we want to know $\mathbb{P}(S_n, k) = \mathbb{P}(Succ(j) = k)$ for all $0 \leq k \leq |S_n|$. Initially, we have the following probabilities.

$$\mathbb{P}(S_j, 0) = 1, \mathbb{P}(S_j, 1) = 0,$$
$$\mathbb{P}(S_t, -1) = 0, t = j, \ldots, n.$$

We can now find all probabilities using the following recursion for $t = j + 1, \ldots, n$ and for $k = 0, \ldots, |S_t|$.

$$\mathbb{P}(S_t, k) = p_t \mathbb{P}(S_{t-1}, k - 1) + (1 - p_t)\mathbb{P}(S_{t-1}, k).$$

Note that the procedure above runs in $O(n^2)$ time. □

The decision version of a priori TRP is also in NP in the scenario model. Since the input contains an explicit list of the scenarios, the second-stage latencies can simply be computed for each scenario.

There are some intriguing difficulties with a priori TRP. Finding an approximation algorithm for this problem turns out to be much harder than for a priori TSP. It is easy to adjust the proof in [58] to show a $\Omega(\log n)$ lower bound on the approximation guarantee in the black-box model. Getting positive results is even non-trivial if all vertices are on a line graph. In the deterministic setting, TRP on the line graph can be solved using dynamic programming [1]. This algorithm is discussed in detail in

Chapter 6. This result relies on the fact that vertices will always be visited when the tour comes across them. In the a priori setting, this is not true. Consider the example from the scenario model shown in Figure 2.1.

Example 2.2. *There is a point at v_1 at distance 1 from the root which is always active. Further, there are 100 points at v_2 at distance 10 from the root which are simultaneously active with probability 0.01, and there are 10 points at v_3 at distance 2 on the other side of the root which are simultaneously active with probability 0.1. These two events at v_2 and v_3 are independent. Note that this gives four possible scenarios. It is easy to compute that the optimal a priori tour is (r, v_2, v_3, v_1), meaning that we pass by the point at v_1 twice before visiting it. The intuition behind this is that we do not want to visit v_1 before v_3, but we do want to visit v_2 before v_3.*

Fig. 2.1: Instance of a priori TRP in the scenario model. The optimal tour passes the point at v_1 twice before visiting it.

Hence, skipping may be optimal in the scenario model. However, we conjecture that in the independent model skipping is never optimal. If this is true, then dynamic programming may be used to solve this problem.

For general metric spaces, the independent decision model is non-trivial. The intuitive approach of using the probabilities as weights, i.e., $w_i = p_i$, and solving the weighted version of TRP turns out to give arbitrarily bad solutions, as shown in Example 2.3.

Example 2.3. *Consider a star graph with $k+1$ leafs. Assign a weight of ℓ to the edge going to leaf $k+1$, and assign weight 1 to the remaining edges. Then, add $\ell-1$ vertices and additional edges such that leaf $k+1$ and the added vertices form a clique containing ℓ vertices. Assign an edge weight equal to zero to all these added edges. Each vertex is active with probability $1/\ell$.*

Now, if we take the probabilities as weights, we see that every solution (visiting all vertices of leaf $k+1$ at the same time) for the created weighted TSP instance has the same value. However, in the a priori setting, it is optimal to visit leaf $k+1$ last. Moreover, by choosing $k = \ell\sqrt{\ell}$, we can show that the ratio between the solution starting with leaf $k+1$ and the optimal solution is $\Omega(\sqrt{\ell})$.

The example shows that simply replacing probabilities by weights does not work. However, for vertex-weighted star graphs the problem is still easy to solve. Here, the weight of a vertex represents the number of clients at that vertex. Each of these clients has its own probability of being in the active set. It can be shown by an interchange argument that the vertices have to be visited in non-increasing order of $\mathbb{E}[N_i]/\mathbb{E}[L_i]$. Here, $\mathbb{E}[N_i]$ is the expected number of clients at vertex i and $\mathbb{E}[L_i]$ is the expected length to vertex i, i.e., the length of the edge times the probability that at least one of the clients at the endpoint has to be visited. Even for slightly more general graphs, such as spiders of depth two, the complexity is still open.

2.4 Algorithm

Before presenting our algorithm, we will rewrite the objective function and state a basic lemma that we will need in the analysis. For a given tour and active set A, we denote ℓ_i^A as the latency of vertex $i \in A$ in the tour shortcutted over A. If vertex i is not in A, then we define $\ell_i^A = 0$. If C_i is the expected latency of vertex i given that i is active, the law of total probability gives that our objective becomes minimizing

$$\mathbb{E}_A[\sum_{i \in V} \ell_i^A] = \sum_{i \in V} p_i \mathbb{E}_A[\ell_i^A | i \text{ is active}] = \sum_{i \in V} p_i C_i. \tag{2.3}$$

Recall that using c_{ri} is the cheapest way to travel from the root r to vertex i. Note that C_i is the expected latency of vertex i, *given* that it is active. Hence, we obtain the following lemma.

Lemma 2.4. *For any tour and vertex i, we have $C_i \geq c_{ri}$.*

Our algorithm is based on algorithms for the deterministic TRP [21, 25, 54]. However, the a priori setting makes the problem a lot harder to solve. As explained above, even the problem on the line is non-trivial in the a priori setting and is not known to be solvable in polynomial time. Our algorithm makes use of an (α, β)-TSP-approximator in the a priori setting, which is similar to the one introduced in [21]. Suppose we have an instance of a priori TSP and a number L. The goal is to find a tour of expected length at most L which minimizes the number of unvisited vertices. An (α, β)-TSP-approximator in the a priori setting will find a tour of expected length at most βL with a number of unvisited vertices at most α times the optimal number of unvisited vertices. More formally, it is defined as follows.

Definition 2.5. *An (α, β)-TSP-approximator in the a priori setting will find, for any given L, a tour that visits at least $(1 - \alpha\epsilon)n$ vertices and is of expected length at most βL if there exists a tour that visits at least $(1 - \epsilon)n$ vertices and is of expected length at most L.*

The algorithm works as follows. Let $L_0 = 2$ (twice the minimum edge length) and $\gamma > 1$ be a parameter to be determined later, and define $L_j = L_0 \gamma^j$ for $j = 0, 1, \ldots$. Now for each length L_j, we obtain a tour $T(L_j)$ by applying the (α, β)-TSP-approximator in the a priori setting. These tours will then be concatenated, i.e., we first traverse tour $T(L_0)$, then we traverse tour $T(L_1)$ and so on until all vertices are visited, where we shortcut already visited vertices. We output the resulting tour.

Theorem 2.6. *Given an (α, β)-TSP-approximator in the a priori setting, our algorithm with $\gamma = 2$ is a $(8\lceil\alpha\rceil\beta + 1)$-approximation for the a priori traveling repairman problem in the uniform model, i.e., $p_i = p$ for all $i \in V$.*

Proof. Assume that α is an integer, otherwise use its ceiling as upper bound. Relabel the vertices such that the tour is $(1, 2, \ldots, n)$. Partition the vertices of the algorithm's tour in blocks of at most α vertices. We define the block B_x to be the subset containing the vertices $n - \alpha(x+1) + 1, n - \alpha(x+1) + 2, \ldots, n - \alpha x$ for $x = 0, 1, \ldots, \lceil \frac{n}{\alpha} \rceil - 1$. Let C^*_{n-x} denote the expected latency of the $(n-x)$th vertex on the optimal a priori TRP-tour, given that it is active, for $x = 0, 1, \ldots, n-1$. Now, suppose that the $(n-x)$th

vertex visited by the optimal tour has a conditional expected latency between L_{j-1} until L_j, i.e., $L_{j-1} \leq C^*_{n-x} < L_j$. Then, we know that there exists a tour visiting at least $n - x$ vertices with expected length at most $2C^*_{n-x} \leq 2L_j = L_{j+1}$, so the (α, β)-TSP-approximator (with respect to L_{j+1}) finds a tour visiting at least $n - \alpha x$ vertices of expected length at most βL_{j+1}. This implies that each vertex $v \in B_x$ is visited in $T_0 \cup \ldots \cup T(L_{j+1})$. We can bound the conditional expected latency, denoted as C_v^{Alg}, in the following way. Let v be visited for the first time in $T(L_{j+1})$. Now, construct a new tour by removing vertex v from tour $T(L_{j+1})$ and visit it after the vertices of $T(L_{j+1})$. Denote the expected latency of v in the new tour by C'_v and note that we have $C_v^{\text{Alg}} \leq C'_v$. Finally note that the expected latency of v in the new tour is bounded by $\beta(L_0 + \ldots + L_{j+1}) + c_{rv}$. If we sum over all vertices in B_x, we get

$$\begin{aligned}
\sum_{v \in B_x} C_v^{\text{Alg}} &\leq \alpha(\beta(L_0 + L_1 + \ldots + L_{j+1})) + \sum_{v \in B_x} c_{rv} \\
&\leq 2\alpha\beta L_{j+1} + \sum_{v \in B_x} c_{rv} \\
&= 8\alpha\beta L_{j-1} + \sum_{v \in B_x} c_{rv} \\
&\leq 8\alpha\beta C^*_{n-x} + \sum_{v \in B_x} c_{rv}.
\end{aligned}$$

If we multiply by p and sum over all blocks, we can bound the objective (2.3) as follows

$$\begin{aligned}
\sum_{x=0}^{\lceil \frac{n}{\alpha} \rceil - 1} \sum_{v \in B_x} pC_v^{\text{Alg}} &\leq 8\alpha\beta \sum_{x=0}^{\lceil \frac{n}{\alpha} \rceil - 1} pC^*_{n-x} + \sum_{v \in V} pc_{rv} \\
&\leq 8\alpha\beta \sum_{v \in V} pC^*_v + \sum_{v \in V} pc_{rv} \\
&\leq (8\alpha\beta + 1)\textsc{Opt}.
\end{aligned}$$

□

Note that uniformity is essential in the last step, since we are comparing different tours vertex by vertex. This approximation guarantee might be improved by choosing another value of γ, but it turns out that $\gamma = 2$ is optimal for our analysis. We can improve the approximation factor by randomizing the starting length. Set $L_0 = 2\gamma^U$ and $L_j = L_0\gamma^j$, where U is a random variable uniformly distributed on $[0, 1]$, and optimize over γ.

Theorem 2.7. *Given an (α, β)-TSP-approximator in the a priori setting, our algorithm with $L_0 = 2\gamma^U$ and $\gamma = e$ is a $(2e\lceil\alpha\rceil\beta + 1)$-approximation for the a priori traveling repairman problem in the uniform model, where U is a random variable uniformly distributed on $[0, 1]$.*

Proof. The proof is similar to that of Theorem 2.6. Assume w.l.o.g. that the algorithms a priori tour is $(1, 2, \ldots, n)$. Partition the vertices of the resulting tour in blocks of size at most α. Let $x \in \{0, 1, \ldots, n-1\}$ and assume that for some ℓ the (conditional)

expected latency of the $(n-x)$th vertex of the optimal tour can be written as a polynomial function of degree ℓ of γ, i.e., $C^*_{n-x} = \rho\gamma^\ell$, where $1 \le \rho < \gamma$. We distinguish two cases: $\rho < \gamma^U$ and $\rho \ge \gamma^U$.

If $\rho < \gamma^U$, then there exists a path from the root with expected length at most $\gamma^U\gamma^\ell$ visiting at least $n-x$ vertices. This means that $T(L_\ell)$ contains at least $n-\alpha x$ vertices and is of length at most $2\beta\gamma^U\gamma^\ell$. So, for $v \in B_x$, we have $C_v^{\mathrm{Alg}} \le \beta\sum_{j=0}^{\ell} L_0\gamma^j + c_{rv} \le \beta L_0\gamma^\ell(\frac{\gamma}{\gamma-1}) + c_{rv}$.

Now for the other case, $\rho \ge \gamma^U$, there must be a path from the root with expected length at most $\gamma^U\gamma^{\ell+1}$. This means that $T(L_{\ell+1})$ contains at least $n-\alpha x$ vertices and is of length at most $2\beta\gamma^U\gamma^{\ell+1}$. So, for $v \in B_x$, we have $C_v^{\mathrm{Alg}} \le \beta\sum_{j=1}^{\ell+1} L_0\gamma^j + c_{rv} \le \beta L_0\gamma^{\ell+1}(\frac{\gamma}{\gamma-1}) + c_{rv}$. In the first case, we have $\log_\gamma \rho \le U \le 1$ and we have $0 \le U \le \log_\gamma \rho$ in the second case. Taking expectations over U gives

$$\begin{aligned}
C_v^{\mathrm{Alg}} \le& \int_{\log_\gamma \rho}^{1} \left(\beta L_0\gamma^\ell\left(\frac{\gamma}{\gamma-1}\right) + c_{rv}\right) dU + \int_0^{\log_\gamma \rho} \left(\beta L_0\gamma^{\ell+1}\left(\frac{\gamma}{\gamma-1}\right) + c_{rv}\right) dU \\
=& 2\beta\gamma^\ell\left(\frac{\gamma}{\gamma-1}\right)\int_{\log_\gamma \rho}^{1} \gamma^U dU + 2\beta\gamma^{\ell+1}\left(\frac{\gamma}{\gamma-1}\right)\int_0^{\log_\gamma \rho} \gamma^U dU + c_{rv} \\
=& 2\beta\gamma^\ell\left(\frac{\gamma}{\gamma-1}\right)\left(\frac{\gamma-\rho}{\ln\gamma}\right) + 2\beta\gamma^{\ell+1}\left(\frac{\gamma}{\gamma-1}\right)\left(\frac{\rho-1}{\ln\gamma}\right) + c_{rv} \\
=& \frac{2\gamma\beta}{\ln\gamma} C^*_{n-x} + c_{rv}.
\end{aligned}$$

If we multiply by p and sum over all vertices in block B_x and then over all blocks, we get a bound of $\frac{2\gamma}{\ln\gamma}\alpha\beta + 1$. Optimizing over γ gives $\gamma = e$ and a bound of $2e\alpha\beta + 1$. □

The algorithm can be derandomized by trying multiple values for U. This will give an approximation guarantee that is arbitrarily close to $2e\alpha\beta + 1$ by using techniques from [54]. Note that if $\alpha = 1$, the approximator corresponds to a β-approximation for *a priori k-TSP*, the problem of finding a tour on k vertices of minimum expected length. This yields the following corollary.

Corollary 2.8. *If there is a β-approximation for the a priori k-TSP, then there is a $(2e\beta + 1)$-approximation for the a priori traveling repairman problem in the uniform model.*

2.5 Tree metrics

To obtain an approximation guarantee for the a priori TRP on trees, we use Corollary 2.8. Note that finding a k-tour in a tree is similar to finding a k-tree in a tree. So, in this case we can solve the a priori k-MST problem, in which we have to find a tree spanning k vertices such that the expected weight of the tree is minimized. Here, shortcutting the tree is done by removing inactive vertices provided that the tree on the active vertices remains connected.

Theorem 2.9. *The a priori k-TSP in the uniform model on tree metrics can be solved to optimality in polynomial time.*

Proof. First, we turn the tree into a binary tree with the original vertices at the leaves by adding vertices with probability zero and edges with weight zero. Next, we use dynamic programming to solve the a priori k-MST problem. Define the function $f(i, y)$ to be the minimum expected weight of a subtree rooted at i containing y leaves. For all leaves i, we have $f(i, 0) = f(i, 1) = 0$. For a certain state (i, y), the best tree follows from a combination of z vertices from the left subtree and $y - z$ vertices from the right subtree. For a given combination, the expected weight is equal to the sum of the expected weight of each of the two subtrees plus, for each subtree, the weight of the edge connecting i with the subtree times the probability that at least one of the vertices in the subtree is active. We denote $\hat{i}$ and $\tilde{i}$ for the left and right child of i respectively. To enhance readability, we slightly change our notation and write $c(i, j)$ for the weight of the edge between i and j. Now, we get the following recursive formula:

$$\begin{aligned} f(i, y) = \min_{0 \le z \le y} \{ & f(\hat{i}, z) + (1 - (1 - p)^z) c(i, \hat{i}) \\ & + f(\tilde{i}, y - z) + (1 - (1 - p)^{y-z}) c(i, \tilde{i}) \}. \end{aligned}$$

The optimal tree containing k vertices is the solution corresponding to $f(r, k)$, where r is the root of the tree. Note that the dynamic program needs $O(nk^2)$ time, so a priori k-MST (and hence k-TSP) on trees can be solved in polynomial time. □

Corollary 2.10. *There is a* $2e + 1 \approx 6.44$*-approximation for the a priori traveling repairman problem in the uniform model on trees.*

It is not clear how to generalize this result to the non-uniform case. The difficulty is that we have to store the minimum weight tree for each possible probability that at least one vertex in the subtree is active. Since the probability that at least one vertex in the subtree is active can take exponentially many different values, the DP will not run in polynomial time. On the other hand, it is easy to extend the DP above to the case where it is almost uniform in the sense that there is a constant number of different probabilities p_i.

2.6 General metrics

For general metrics, we show how to obtain an (α, β)-TSP-approximator for some constants α and β. It turns out that finding such an approximator boils down to finding a approximation algorithms for certain variations of the tour single-sink rent-or-buy problem (tour-SRoB).

In the *single-sink rent-or-buy problem* (SRoB) [99], we are given a graph $G = (V, E)$ with a metric weight function c_e on the edges. There is a client j at each vertex in V with demand d_j. We have to open a facility at some of the vertices and connect the clients to the facilities. We denote c_{ij} as the length of the shortest path between i and j in G. Note that $c_{ij} = c_e$ if $e = (i, j)$. Connecting facility i with client j costs $d_j c_{ij}$ and buying edge e costs $M c_e$, where $M \ge 1$. We need to buy edges such that the open facilities are joined by a Steiner tree. The goal is to minimize the sum of connection cost and Steiner cost. In the *tour-SRoB*, G is a complete graph. Here, edges have to be bought such that the open facilities are joined by a tour.

The next two variants are used to get the desired approximation results for a priori TRP. In the *prize-collecting tour-SRoB*, it is not needed to connect every client, but if

client j is not connected, then we have to pay penalty π_j. The goal is to minimize the sum of connection cost, tour cost and penalty cost.

In the *k-client tour-SRoB*, it also not needed to connect every client. One has to connect at least k vertices at minimum total cost. Approximating the latter problem is done by using the following definition.

Definition 2.11. *An (α, β)-tour-SRoB-approximator will find, for any given L, a tour-SRoB-solution containing at least $(1 - \alpha\epsilon)n$ vertices of cost at most βL if there exists a tour-SRoB-solution containing at least $(1 - \epsilon)n$ vertices of cost at most L.*

In this section, we will follow the structure depicted below. We start with showing that there is a 5-approximation for tour-SRoB. We then use this result to show that there is 5.52-approximation for the prize-collecting tour-SRoB. In section 2.6.2, we first show that if we have an (α, β)-tour-SRoB-approximator, we get an $(\alpha, 3\beta)$-TSP-approximator. Finally, we show that the 5.52-approximation for prize-collecting tour-SRoB can be used to obtain an $(11.04, 11.04)$-tour-SRoB-approximation which together with the former statement results in an $(11.04, 33.12)$-TSP-approximator. Hence, by Theorem 2.7 this results in a $O(1)$-approximation for a priori TRP in the uniform model on general metrics.

tour-SRoB	5	(Theorem 2.12)
↓	↓	
prize-collecting tour-SRoB	5.52	(Theorem 2.15)
↓	↓	
(α, β)-tour-SRoB-approximator	(11.04,11.04)	(Lemma 2.18)
↓	↓	
(α, β)-TSP-approximator	(11.04,33.12)	(Corollary 2.17)
↓	↓	
a priori TRP	$O(1)$	(Theorem 2.19)

2.6.1 Prize-collecting tour-SRoB

The prize-collecting tour-SRoB has, to the best of our knowledge, not been considered explicitly in the literature. We can obtain a randomized 3-approximation for tour-SRoB by adjusting the analysis for tour-connected facility location (a generalization of tour-SRoB) by Eisenbrand et al. [42]. This can be derandomized by adapting the analysis of Van Zuylen [105] to obtain a deterministic 3-approximation. However, it is not clear how to extend these results to prize-collecting tour-SRoB. Therefore, we will use the primal-dual algorithm for SRoB by Swamy and Kumar [99] instead.

Tour-SRoB

First, consider SRoB. We assume without loss of generality that a facility is opened at root vertex r. In the ILP-formulation below, we define x_{ij} to be 1 if i is on the tree and j is connected to i. We define z_e to be 1 if we use edge e in the tree. Without loss of generality, we assume that we have unit demand. The reader is referred to [99] for

further details. Recall that $\delta(X)$ denotes the cut separating X from $V \setminus X$ (page 7).

$$
\begin{aligned}
\text{(P)}\ \min\quad & \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} + M\sum_{e\in E} c_e z_e & & \\
\text{s.t.}\quad & \sum_{i\in V} x_{ij} \geq 1 & & \forall j \in V \\
& \sum_{i\in S} x_{ij} \leq \sum_{e\in\delta(S)} z_e & & \forall S \subseteq V\setminus\{r\}, j \in V \\
& x_{ij}, z_e \in \{0,1\} & & \forall i,j \in V, e \in E.
\end{aligned}
$$

Relaxing the integrality constraints and taking the dual gives the following problem.

$$
\begin{aligned}
\text{(D)}\ \max\quad & \sum_{j\in V} \mu_j & & \\
\text{s.t.}\quad & \mu_j \leq c_{ij} + \sum_{S\subseteq V: i\in S, r\notin S} \theta_{S,j} & & \forall i \in V\setminus\{r\}, j \in V \\
& \mu_j \leq c_{ij} & & \forall j \in V \\
& \sum_{j\in V}\sum_{S\subseteq V: e\in\delta(S), r\notin S} \theta_{S,j} \leq M c_e & & \forall e \in E \\
& \mu_j, \theta_{S,j} \geq 0 & & \forall j \in V, S \subseteq V\setminus\{r\}.
\end{aligned}
$$

In any solution for the tour-SRoB, each client j is connected to some facility i on the tour (possibly $i = j$). In that case, any cut separating i from r must contain at least two edges. Hence, an LP-relaxation for tour-SRoB is obtained by relaxing the integrality constraints in (P) and by putting a factor 2 in front of x_{ij} in the second constraint. We obtain the following LP-relaxation and its dual.

$$
\begin{aligned}
\text{(P}'\text{)}\ \min\quad & \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} + M\sum_{e\in E} c_e z_e & & \\
\text{s.t.}\quad & \sum_{i\in V} x_{ij} \geq 1 & & \forall j \in V \\
& 2\sum_{i\in S} x_{ij} \leq \sum_{e\in\delta(S)} z_e & & \forall S \subseteq V\setminus\{r\}, j \in V \\
& x_{ij}, z_e \geq 0 & & \forall i,j \in V, e \in E.
\end{aligned}
$$

$$
\begin{aligned}
\text{(D}'\text{)}\ \max\quad & \sum_{j\in V} \mu_j & & \\
\text{s.t.}\quad & \mu_j \leq c_{ij} + 2\sum_{S\subseteq V: i\in S, r\notin S} \theta_{S,j} & & \forall i \in V\setminus\{r\}, j \in V \\
& \mu_j \leq c_{ij} & & \forall j \in V \\
& \sum_{j\in V}\sum_{S\subseteq V: e\in\delta(S), r\notin S} \theta_{S,j} \leq M c_e & & \forall e \in E \\
& \mu_j, \theta_{S,j} \geq 0 & & \forall j \in V, S \subseteq V\setminus\{r\}.
\end{aligned}
$$

We can now use the primal-dual algorithm for SRoB to obtain an approximation algorithm for tour-SRoB. Given an instance of tour-SRoB, we divide all edge weights by 2, i.e., $c'_e = c_e/2$ and $c'_{ij} = c_{ij}/2$. To keep the remaining restrictions of the dual and the Steiner costs the same, we also set $M' = 2M$. Secondly, we use the primal-dual algorithm of Swamy and Kumar [99] on the new instance to obtain a solution for SRoB. Finally, we double the tree and shortcut the resulting Eulerian tour. This algorithm and its analysis are similar to the work of Goemans and Williamson [55], who showed how to obtain a 2-approximation for the prize-collecting TSP using a 2-approximation for the prize-collecting Steiner tree problem. Further, this ratio is worse than the ratio that can be obtained from [42]. However, that result is based on a sampling approach which we do not know how to extend to the prize-collecting version of the problem.

Theorem 2.12. *The approach above gives a 5-approximation for the tour-SRoB. Moreover, the value is at most 5 times the optimal value of the LP-relaxation* (P').

Proof. The primal-dual algorithm of Swamy and Kumar gives two feasible solutions, namely (μ^1, θ^1) and (μ^2, θ^2). Then, $(2\mu^1, \theta^1)$ and $(2\mu^2, \theta^2)$ are feasible solutions for (D$'$). By duality, we have

$$2\sum_{j\in V} \mu_j^1 \leq \textsc{Opt} \text{ and } 2\sum_{j\in V} \mu_j^2 \leq \textsc{Opt}, \tag{2.4}$$

where Opt is the optimal value for tour-SRoB. Given the solution of SRoB with connection costs C' and Steiner cost S, the cost of the solution for tour-SRoB produced by the algorithm is at most $C + 2S = 2(C' + S)$. By Swamy and Kumar, we get $C' + S \leq 3\sum_j \mu_j^1 + 2\sum_j \mu_j^2$ (proof of Theorem 3.6 in [99]). Combining these two equations and using (2.4), we get that the solution of our algorithm has cost at most

$$C + 2S = 2(C' + S) \leq 2(3\sum_{j\in V} \mu_j^1 + 2\sum_{j\in V} \mu_j^2) \leq 3\textsc{Opt} + 2\textsc{Opt} \leq 5\textsc{Opt}.$$

Note that the solution of our algorithm contains a tour on the open facilities and it is therefore a feasible solution for tour-SRoB. □

The prize-collecting version.

In this version, it is not needed to connect all clients. However, a penalty π_j is incurred if client j is not connected. For the LP-relaxation of the prize-collecting tour-SRoB problem, we add the variable s_j, which is set to 1 if client j is not connected. In an integral solution, the first constraint corresponds to a client being either connected with an open facility or not connected at all.

$$\begin{aligned}
(\mathrm{P}'') \ \min \quad & \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} + M\sum_{e\in E} c_e z_e + \sum_{j\in V} \pi_j s_j & \\
\text{s.t.} \quad & s_j + \sum_{i\in V} x_{ij} \geq 1 & \forall j \in V \\
& 2\sum_{i\in S} x_{ij} \leq \sum_{e\in\delta(S)} z_e & \forall S \subseteq V \setminus \{r\}, j \in V \\
& x_{ij}, z_e, s_j \geq 0 & \forall i, j \in V, e \in E.
\end{aligned}$$

Using the ellipsoid method, the LP-relaxation can be solved in polynomial time. Note that, like for the other two LP-formulations, the separation problem can be solved by using a min-cut algorithm. The algorithm for the prize-collecting version works as follows (see [104], Section 4.4). Let (x^*, z^*, s^*) be an optimal solution for (P″). If $s_j^* \geq \delta$, then we set $\hat{s_j} = 1$, else we set $\hat{s_j} = 0$, where $0 \leq \delta \leq 1$ is determined later, and let $T = \{j : \hat{s}_j = 0\}$. The vertices in $V \setminus T$ will not be visited. Next, we obtain a solution of tour-SRoB on T by applying the algorithm from Theorem 2.12. This results in a feasible solution for prize-collecting tour-SRoB on V. Partition the optimal LP-value in the connection plus tour cost C_{LP} and penalty cost Π_{LP}.

Lemma 2.13. *The algorithm above finds a solution for the prize-collecting tour-SRoB such that the resulting tour and connection cost is bounded by* $5/(1-\delta)C_{LP}$ *and the resulting penalty cost is bounded by* $(1/\delta)\Pi_{LP}$.

Proof. By rounding the solution, we lose at most a factor $1/\delta$ on the penalty cost. This means that the penalty cost is at most a factor $1/\delta$ times the penalty cost of the LP-relaxation. By Theorem 2.12, the connection and tour cost for tour-SRoB on T can be bounded by 5 times the optimal solution of its LP-relaxation. We obtain a feasible solution for this LP-relaxation by deleting the s_j's from the LP-relaxation of prize-collecting tour-SRoB and multiply all other variables by a factor $1/(1-\delta)$. Combining the two statements, we obtain that the connection and tour cost can be bounded by $5/(1-\delta)$ times the connection and tour cost of the optimal LP-solution. □

Note that by choosing $\delta = \frac{1}{6}$, we obtain a 6-approximation. Choosing δ at random improves this result. If we choose δ uniformly at random on $[0, \omega]$, with $0 < \omega \leq 1$ to be specified later (see [104], Section 5.7), we obtain the following result.

Lemma 2.14. *Randomization of the algorithm above gives a solution for the prize-collecting tour-SRoB such that the resulting tour plus connection cost is in expectation bounded by* $(5/\omega)\ln(1/(1-\omega))\,C_{LP}$ *and the resulting penalty cost is in expectation bounded by* $(1/\omega)\Pi_{LP}$.

Proof. The tour and connection costs are deterministically bounded by $5/(1-\delta)C_{\text{LP}}$. If we take the expected value with respect to δ, we get that the tour and connection costs are bounded by $\mathbb{E}\left[5/(1-\delta)\right]C_{\text{LP}}$ in expectation. Computing this expectation gives:

$$\mathbb{E}\left[\frac{5}{1-\delta}\right] = \int_0^{\omega} \frac{5}{1-x} \cdot \frac{1}{\omega} dx = -\frac{5}{\omega}\ln(1-x)\Big|_0^{\omega} = \frac{5}{\omega}\ln\left(\frac{1}{1-\omega}\right).$$

If $s_j^* < \omega$, then s_j^* gets rounded to 1 (j will not be visited) with probability s_j^*/ω. If $s_j^* \geq \omega$, then s_j^* gets rounded to 1 with probability 1, but here we have $s_j^*/\omega \geq 1$. So, we can bound the penalty cost by $(1/\omega)\sum_j s_j^*\pi_j \leq (1/\omega)\Pi_{\text{LP}}$. □

Note that the algorithm can be derandomized by checking all values $s_j^* \in [0, \omega]$ for δ, since the set of unvisited vertices does not change for values in between two consecutive values of s_j^*. So, by checking at most n values, we obtain a deterministic algorithm with the same guarantees. Choosing $\omega = 1 - e^{-1/5}$ gives the following approximation guarantee.

Theorem 2.15. *There is a* 5.52*-approximation algorithm for the prize-collecting tour-SRoB problem.*

2.6.2 Obtaining an (α, β)-TSP-approximator

In this subsection, it is shown how to obtain an (α, β)-TSP-approximator using the results for prize-collecting tour-SRoB. We first show how a priori TSP and tour-SRoB are related.

Lemma 2.16. *Any approximation algorithm for the tour-SRoB problem can be turned into an approximation algorithm for the a priori TSP in the independent decision model with loss of at most a factor 3 in the approximation.*

Proof. Given an instance of a priori TSP with edge weights c_e and probabilities p_i, we define an instance of tour-SRoB as follows. The edge weights are $c'_e = c_e \ \forall e$, $M = 1$ and at each vertex there is a client with demand $d_j = 2p_j$. Given any feasible solution for this instance we get a feasible solution for a priori TSP of at most the same cost as follows. Let T be the tour in the SRoB solution. For the a priori tour we take T and double all the edges from clients to facilities in the SRoB solution. It is easy to see that the expected length of the shortcut TSP solution is at most that of the SRoB solution. Let OPT_{TSP} and OPT_{SRoB} denote the optimal value of, respectively, the a priori TSP and the tour-SRoB instance. It remains to show that $\text{OPT}_{\text{SRoB}} \leq 3\text{OPT}_{\text{TSP}}$. Select vertex i with probability p_i and take an optimal tour on the set of selected vertices X. Let this be the tour for the SRoB solution. Connect all other vertices in the cheapest way to X. It follows from the analysis in [95], as summarized in Equation (2.1), that the cost of this SRoB solution is at most 3 times the optimal length of the a priori TSP instance, since the construction above is just their algorithm except for the fact that we take an optimal tour on X. Hence, $\text{OPT}_{\text{SRoB}} \leq 3\text{OPT}_{\text{TSP}}$. □

The theorem above applies as well in the k-client setting, both for regular α-approximations and for (α, β)-tour-SRoB-approximators. Recall that in the k-client tour-SRoB we want to find a minimum cost tour-SRoB solution connecting k clients.

Corollary 2.17. *In the independent decision model, we have that*

1. *If there is an α-approximation for the k-client tour-SRoB problem, then there is a 3α-approximation for the a priori k-TSP.*

2. *If there is an (α, β)-tour-SRoB-approximator, then there is an $(\alpha, 3\beta)$-TSP-approximator in the a priori setting.*

Proof. In both cases, we use the same transformation as in the proof of Lemma 2.16.

1. The proof works similarly, except that in the last step, we need to sample X from the vertices on the optimal k-tour, instead of sampling X from all vertices V. By Lemma 2.16, we get a 3α-approximation.

2. We sample X from the vertices on the minimum length tour on $(1-\epsilon)n$ vertices. Note that the number of visited vertices in the obtained TSP-solution solution is the same as in the optimal TSP-solution, so we do not lose anything there. Hence, by Lemma 2.16, we obtain an $(\alpha, 3\beta)$-TSP-approximator in the a priori setting.

□

Our final lemma shows that an (α, β)-tour-SRoB-approximator can be obtained using results from prize-collecting tour-SRoB.

Lemma 2.18. *If there is an α-approximation for prize-collecting tour-SRoB, then there is a $(2\alpha, 2\alpha)$-tour-SRoB-approximator.*

Proof. Consider an instance of tour-SRoB. Assume that there exists a solution T of cost at most L which visits at least $(1-\epsilon)n$ vertices. We show how to get a solution of length at most $2\alpha L$ that visits at least $(1-2\alpha\epsilon)n$ vertices. As noted in [21], we can perform a binary search on the optimal value of ϵ given L, if ϵ is not specified. Define an instance of prize-collecting tour-SRoB by giving each vertex a penalty $\pi = L/(\epsilon n)$. The optimal value of this instance is at most that of solution T which is $L + \epsilon n \pi \leq 2L$. Hence, any α-approximation for the prize-collecting tour-SRoB instance should return a solution that has tour and connection cost at most $2\alpha L$ and also a penalty cost of at most $2\alpha L$. The latter implies that it leaves at most $2\alpha L/\pi = 2\alpha\epsilon n$ vertices unvisited. □

Now, we finally get a constant-factor approximation algorithm for the a priori TRP in the uniform model.

Theorem 2.19. *There is an $O(1)$-approximation for the a priori traveling repairman problem in the uniform model.*

Proof. From Theorem 2.15 we get an α_0-approximation for the prize-collecting tour-SRoB, where $\alpha_0 = 1/(1-e^{-1/5})$. Combining this with Lemma 2.18, we obtain an $(2\alpha_0, 2\alpha_0)$-tour-SRoB-approximator. Using Corollary 2.17, we get an $(2\alpha_0, 6\alpha_0)$-TSP-approximator. Plugging this results into the result of Theorem 2.7, we obtain an algorithm with guarantee $2e\lceil 2\alpha_0 \rceil 6\alpha_0 + 1 < 2161$ for the a priori TRP in the uniform model. □

2.7 Conclusion

For the a priori traveling repairman problem we were only able to give a constant-factor approximation in the uniform model and the constant is still large. For the correctness of Theorem 2.6 and 2.7 the uniformity of the probabilities is essential. It is not clear how to reduce the case of independent probabilities to the uniform model. Therefore, the problem is wide open in the independent decision model with non-uniform probabilities. Also, it is not known if the uniform problem can be solved efficiently in case all points are on the line. If any optimal solution has the property that no point is passed without visiting it, like in the deterministic problem, then the problem may be solved by dynamic programming. However, a proof of this property is missing and we have shown that this property does not hold in the scenario model.

In our analysis we used the theory of (α, β)-TSP-approximators. Better approximations may be obtained by using the a priori k-TSP or k-client tour-SRoB. No constant-factor approximation is known for these problems.

Finally, it is good to note that there is still a lot to do in the scenario model. It would be interesting to see if this extra knowledge, i.e., an explicit list of scenarios, can help us to obtain stronger approximation results. The next chapter addresses the a priori TSP in the scenario model.

CHAPTER 3

The a priori traveling salesman problem

The results in this chapter were published in [38].

3.1 Introduction

In a priori routing, we extend our classical routing problems to the case that the set of clients is uncertain or changes regularly. Because reoptimizing over and over again might be inconvenient or impossible, we want to find a single tour. Given a tour and a set of clients, the active set, we shortcut the tour to the active set. In universal routing, the goal is to minimize the worst-case ratio of the value of the obtained solution and the deterministic optimal value. In a priori routing, we want to be good on average. The problem we consider in this chapter is formally defined as follows.

In the *a priori traveling salesman problem in the scenario model*, we are given a complete weighted graph $G = (V, E)$ and a set of scenarios $\mathcal{S}$ with $S_1, \ldots, S_m \subseteq V$. Scenario S_j has probability p_j of being the active set, where $\sum_j p_j = 1$. We begin by finding an ordering on V, called the first-stage tour. When an active set is released, the second-stage tour is obtained by shortcutting the first-stage tour on the vertices of the active set. The goal is to find a first-stage tour that minimizes the expected length of the second-stage tour. If $p_j = \frac{1}{m}$ for all j, then minimizing the expected length is equivalent to minimizing the sum of the tour lengths over the scenarios. Throughout this chapter, we assume that the edge weights obey the triangle inequality.

A problem related to the a priori TSP is the *universal TSP*. In this problem, one is given a TSP-instance. For a given first-stage tour τ, denote $c(\tau_A)$ for the length of τ restricted to A, and denote $\textsc{Opt}(A)$ for the optimal value of the TSP-instance restricted to A. The competitive ratio of the tour is now defined as the worst-case ratio of $c(\tau_A)$ and $\textsc{Opt}(A)$ over all $A \subseteq V$. The goal in universal TSP is to find a tour that minimizes the competitive ratio.

The a priori TSP has, for example, a direct application in photo-lithography processes used in semi-conductor manufacturing to transfer the geometric pattern of a chip onto a wafer [34]. This is done by putting UV light through a photomask on a photoresistant layer on top of the wafer. The entire wafer is not exposed at once, but one square at a time. If certain parts of the square do not need to be exposed, blades are moved in to block the UV light. Moving the blades is a time-consuming, and hence costly, process. Since it often influences the total processing time of a wafer in the lithography machine, minimizing the distance reduces the processing time. The

blading positions are defined in a file. The blading positions are obtained from this file by reading it from top to bottom and the positions are used by the machine in order of appearance. A product will visit the photolithography machine multiple times during its fabrication. Every time it will use the same file that defines its blading positions, but it will not use all blading positions defined in the file in every visit. For each visit, there is a given subset of the blading positions that has to be used. Hence minimizing the movement of the blades comes down to finding an ordering of the blading positions such that the sum over all visits of the total distance between the blading-positions is minimized. The authors of [34] show that this is precisely a form of the a priori TSP in the scenario model.

As already mentioned in Chapter 2, a priori TSP has already been considered in the literature in the independent decision and black-box model. In the independent decision model, vertex i is active with probability p_i, independent of the other vertices. Shmoys and Talwar [95] showed that a sample-and-augment approach gives a randomized 4-approximation, which can be derandomized to an 8-approximation algorithm. This factor was improved by Van Zuylen [105] to 6.5. In the black-box model, we have no knowledge on the probability distribution over the vertices, but we are able to sample from it. Schalekamp and Shmoys [94] showed that one can obtain a randomized $O(\log n)$-approximation even without sampling. A deterministic $O(\log^2 n)$-approximation can be obtained by using the result for universal TSP [59]. It was shown by [58] that there is an $\Omega(\log n)$ lower bound for deterministic algorithms on general metrics. By using the result of [62] and Theorem 3 in [58], it is also known that there is no deterministic algorithm with guarantee $o\left(\sqrt[6]{\log n/\log\log n}\right)$ for planar metrics. For randomized algorithms, no lower bound is known for the black-box model.

The above mentioned results give us the first results for a priori TSP in the scenario model. First of all, we inherit the randomized $O(\log n)$-approximation. Secondly, we know that a deterministic algorithm that does not use the information given in the scenarios will not achieve an approximation guarantee better than $O(\log n)$. The main question is whether we can use the scenarios to improve upon the $O(\log n)$ upper bound and which restrictions we can put on the scenarios in order to obtain constant-factor approximability. This question will be considered in this chapter.

The scenario model has not been studied extensively for other optimization problems. Immorlica et al. [67] investigated scenario versions of the vertex cover and the shortest path problem. Ravi and Sinha [89] also looked at these problems and also defined scenario versions of bin packing, facility location and set cover. The problems in [89] differ from our setting in the sense that the weights used in the instance differ between scenarios. Further, the authors of [26] investigate a two-stage stochastic scheduling problem, where the set of jobs to be processed is uncertain. Finally, in [44], the classical scheduling problem of minimizing the makespan on two machines is considered in the a priori model with scenarios.

A priori TSP can be considered as a stochastic version of TSP. Alternatively, one could consider a robust version where we want to minimize the maximum length over all scenarios. We will refer to this problem as *Min-Max TSP*. When applicable, we will state to which extend the theorems for a priori TSP also hold for the Min-Max TSP. An easy observation is that the approximation ratios for universal TSP carry over directly to MinMax-TSP. Hence, we have an $O(\log^2 n)$-approximation algorithm.

In this chapter, we will first examine the most natural lower bound that we call

the master tour lower bound. We use this lower bound to show that there exists a constant-factor approximation algorithm for the problem if the number of scenarios is fixed. However, we also show that this lower bound cannot be used to improve upon the $O(\log n)$-approximation. We then look at several natural restrictions on the scenarios, namely small, big and nested scenarios. For small scenarios, we give strong inapproximability results. After that, we analyze the performance of the optimal tour on V for big scenarios. For nested scenarios, we show that there exists a 9-approximation algorithm. Finally, we show that there exists an elegant connection to an a priori minimum spanning tree problem. We end with a discussion on some open problems.

3.2 Master tour lower bound

In this section, we explore the master tour lower bound. Here, we use that the contribution of scenario S_j to the objective value of an optimal solution, denoted by Opt, is at least $p_j T_j^*$, where T_j^* is the length of the optimal tour on S_j, so $\textsc{Opt} \geq \sum_j p_j T_j^*$. Two natural algorithms for a priori TSP in the scenario model are as follows. For each scenario, find an α-approximate tour, where α is the best approximation ratio available for TSP, and sort the scenarios on their resulting tour lengths T_j. Rename the scenarios such that $T_1 \leq T_2 \leq \ldots \leq T_m$. Now traverse the tours $1, 2, \ldots, m$, skipping already visited vertices, resulting in tour τ_1. Alternatively, rename the scenarios such that $p_1 \geq p_2 \geq \ldots \geq p_m$ and traverse the tours $1, 2, \ldots, m$, skipping already visited vertices, resulting in tour τ_2. We get the following result.

Theorem 3.1. *Tours τ_1 and τ_2 are $(2m-1)$-approximations for a priori TSP in the scenario model, where $m \geq 2$ is the number of scenarios.*

Proof. Let us analyze tour τ_1. Consider an arbitrary scenario S_j. Let D_j be the diameter of G restricted to S_j, so we have $T_j^* \geq 2D_j$. Note that when analyzing the contribution of scenario S_j, we only have to consider tours that contain vertices in S_j. Further, it might happen that two tours, say T_x and T_y, with $x, y < j$, $S_x \cap S_j \neq \emptyset$ and $S_y \cap S_j \neq \emptyset$, belong to disjoint scenarios. In this case, we have to go from T_x to T_y. If $d(A, B)$ denotes the maximum distance between a vertex in A and a vertex in B, then this move costs us at most an extra $d(S_x \cap S_j, S_y \cap S_j)$. In the worst case, all scenarios before S_j have a non-empty intersection with S_j. For $j = 1$, the contribution is just $p_1 T_1 \leq \alpha p_1 T_1^*$. For $j \geq 2$, the contribution of S_j to the objective value of our solution is at most

$$\begin{aligned} &p_j(T_1 + d(S_1 \cap S_j, S_2 \cap S_j) + T_2 + \ldots + d(S_{j-2} \cap S_j, S_{j-1} \cap S_j) + T_{j-1} + T_j) \\ \leq &p_j(jT_j + (j-2)D_j) \leq p_j\left(\alpha j T_j^* + (j-2)\frac{1}{2}T_j^*\right) = \left(\left(\alpha + \frac{1}{2}\right)j - 1\right)p_j T_j^*. \end{aligned}$$

Note that you do not have to incur an extra distance from S_{j-1} to S_j, since they have a non-empty intersection. In general, this holds for the last scenario that intersects with S_j. The objective value is at most

$$\alpha p_1 T_1^* + \sum_{j=2}^{m}\left(\left(\alpha + \frac{1}{2}\right)j - 1\right)p_j T_j^* \leq \left(\left(\alpha + \frac{1}{2}\right)m - 1\right)\textsc{Opt}.$$

Since $\alpha = 1.5$ [27], we get a $2m-1$-approximation algorithm. The analysis for τ_2 is similar and the proof is omitted here. □

Since in the proof of Theorem 3.1 we bound the length of each tour by $2m-1$ times the optimal tour for that scenario, it is obvious that τ_1 and τ_2 are also $(2m-1)$-approximations for Min-Max TSP.

It turns out that the master tour lower bound will not give a constant-factor approximation for a priori TSP on general metrics. This can be deduced from Theorem 2 in [58], which roughly states the following. Suppose you are given a d-regular Ramanujan graph G on n vertices with girth $g \geq \frac{2}{3}\log_{d-1} n$. Take a random walk of length $70g$ in G and let S be the vertices visited in this walk. Now, consider a TSP-tour on the vertices of G. Theorem 2 in [58] states that for each of the first $g/2$ steps of the tour restricted to S, the probability that the edge has length $\Omega(\log n)$ is bounded from below by a constant.

Theorem 3.2. *There is an instance of a priori TSP in the scenario model such that* $\textsc{Opt} = \Omega(\log n)\sum_j p_j T_j^*$ *and* $\textsc{Opt} = \Omega(\log m)\sum_j p_j T_j^*$.

Proof. We use Theorem 2 from [58] as discussed above. Let G be a d-regular Ramanujan graph on n vertices with girth $g \geq \frac{2}{3}\log_{d-1} n$. The set of scenarios is the set of all vertex sets of walks of length $70g$. The probability p_j of scenario S_j is equal to the probability that S_j is the vertex set of a random walk of length $70g$. For a fixed first-stage tour, Theorem 2 in [58] states that in each of the first $g/2$ steps of the second-stage tour, there is a constant probability that the second-stage tour uses an edge of length $\Omega(\log n)$. This implies that the expected length of the first $g/2$ steps of the tour have expected length $\Omega(\log n)$. Since $T_j^* = O(g)$, the first $g/2$ steps are a constant fraction of all the steps and so the lower bound also holds for the entire tour. Hence, we have an instance such that $\textsc{Opt} = \Omega(\log n)\sum_j p_j T_j^*$. The number of scenarios is equal to the number of possible walks of length $70g$. This is equal to $n \cdot d^{70g} = O(nd^{\log n}) = O(n^{\log d+1})$. Since d is a constant, this number is polynomially bounded. Hence, we have $\Theta(\log m) = \Theta(\log n)$, which gives us the second lower bound. □

A natural question one can ask is whether a given instance has an optimal value that is equal to the master tour lower bound. Stated differently, is there a tour such that if we shortcut on the vertices of a scenario, we get the optimal solution for that scenario? Deineko et al. [31] studied this problem for the case where every possible subset is a scenario. They called this the master tour problem and showed that it is polynomially solvable. We can reformulate the problem to the case where we are given a set of scenarios and we only have to be optimal for these scenarios. We shall prove in Chapter 4 that this problem is hard to solve by showing it is Δ_2^p-complete.

3.3 Small scenarios

We start with showing that a priori TSP is still NP-hard when all scenarios are very small. We reduce from the Max Cut problem [70]. Here, we are given a graph $G = (V, E)$ and our goal is to find a set $X \subseteq V$ such that $|\delta(X)|$ is maximized. Recall that $\delta(X)$ denotes the cut seperating X from $V \setminus X$ (page 7).

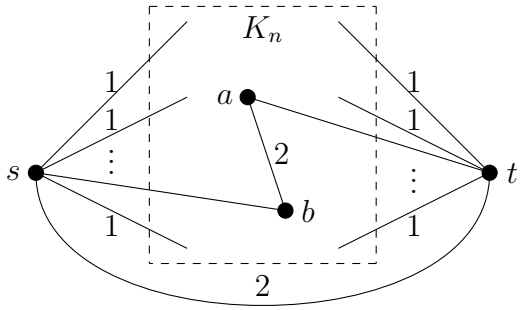


Fig. 3.1: Graph G' as in the proof of Theorem 3.3.

Theorem 3.3. *A priori TSP is NP-hard even when $|S_j| \leq 4$ for all j.*

Proof. We reduce from the Max Cut problem. Given an instance $G = (V, E)$ of Max Cut, we create an instance of a priori TSP by making a complete graph G' on $V \cup \{s, t\}$. All edges with s or t as endpoint, except edge (s, t), have length 1 and all other edges have length 2 (see Figure 3.1). For every edge $(a, b) \in E$, we create a scenario $\{a, b, s, t\}$. All scenarios have equal probability. Note that the second-stage tour on a scenario either has a length of 4 or length 6. We say that a scenario is satisfied if its resulting tour has length 4. Hence, minimizing the expected length is equivalent to maximizing the number of satisfied scenarios. We will show that $\text{Opt}_{\text{TSP}} = 6|E| - 2\text{Opt}_{\text{CUT}}$, where Opt_{TSP} and Opt_{CUT} are the optimal sum of tour lengths of a priori TSP in the created instance and the optimal value of Max Cut in the original instance respectively.

Suppose there is a cut of size at least k in G, say $Q \subseteq V$. First, visit s. Then, visit the vertices of Q in arbitrary order. After that, we visit t. Finally, we visit the vertices not in Q in arbitrary order. It is easy to see that every scenario corresponding to an edge in the cut has length 4, whereas other scenarios have length 6. Hence, there is a tour satisfying at least k scenarios.

On the other hand, suppose that we have a tour in G' satisfying at least k scenarios. Without loss of generality, the tour can be written as (s, R_1, t, R_2), where R_1 and R_2 are sequences of vertices. The only way to satisfy a scenario $\{a, b, s, t\}$ is by putting one vertex of $\{a, b\}$ in R_1 and one vertex in R_2. Hence, the k satisfied scenarios correspond to edges in the cut $R_1 \subseteq V$ which has size at least k. □

The proof above does not imply that the master tour problem is NP-complete for scenarios of size 4 since deciding whether there is a cut containing all edges is easy. The master tour problem with scenarios is still open for $|S_j| \leq 4$. By adjusting the proof of Theorem 3.3, we can prove that the master tour problem with scenarios is NP-hard when $|S_j| \leq 5$. This is done by reducing from Set Splitting instead of Max Cut and using that 3-Set Splitting is NP-complete [79]. In 3-Set Splitting, we are given n elements and a collection Σ of sets containing three distinct elements. The question is whether we can partition the elements such that each set is splitted, i.e., there is a partition $(X, V \setminus X)$ such that neither $\sigma_1, \sigma_2, \sigma_3 \in X$ nor $\sigma_1, \sigma_2, \sigma_3 \in V \setminus X$ for all $(\sigma_1, \sigma_2, \sigma_3) \in \Sigma$.

Theorem 3.4. *The master tour problem with scenarios is NP-complete when* $|S_j| \leq 5$ *for all* j*.*

Proof. The proof is similar to the proof of Theorem 3.3. Create graph G' as in the proof of Theorem 3.3 and create a scenario $\{\sigma_1, \sigma_2, \sigma_3, s, t\}$ for each set in Σ. Note that the shortest TSP-tour for a scenario has length 6. Suppose we have a yes-instance for 3-Set Splitting, say (Q_1, Q_2). Consider the tour that first visits s, then all vertices corresponding to elements of Q_1, then t and finally the vertices corresponding to elements of Q_2. In this tour, each scenario has a second-stage tour of length 6, which equals the minimum length tour on each scenario. On the other hand, suppose we have a master tour. We can rewite this as (s, R_1, t, R_2), where R_1 and R_2 are sequences of vertices. Since this tour is a master tour, we know that for every scenario neither $\sigma_1, \sigma_2, \sigma_3 \in R_1$ nor $\sigma_1, \sigma_2, \sigma_3 \in R_2$ holds. Hence, (R_1, R_2) is a yes-instance for 3-Set Splitting. □

This also shows that Min-Max TSP is NP-hard when $|S_j| \leq 5$ for all j. Moreover, when $|S_j| \leq 5$ for all j, it is NP-complete to decide whether an instance of Min-Max TSP has maximum length 6 or 8. Hence, we cannot approximate Min-Max TSP within a factor of $\frac{4}{3}$, unless P=NP.

Note that the graph we used in the proof of Theorem 3.3 is obtained by taking the metric completion of $K_{2,n}$. This graph is planar, bipartite and it has treewidth and pathwidth equal to 2. Deterministic TSP would be polynomially solvable on such a graph with bounded treewidth. Furthermore, there is a PTAS for deterministic TSP in planar graphs [4]. The next theorem shows that this is not the case for a priori TSP (since the proof uses the same graph as before, a metric completion of $K_{2,n}$). This theorem relies on the fact that maximum cut, given the *unique games conjecture* (UGC), cannot be approximated by a factor above the Goemans-Williamson [56] constant, i.e., approximately 0.878567, unless P=NP [73]. Without this conjecture, Håstad [64] showed that it cannot be approximated above a factor $\frac{16}{17}$, unless P=NP.

Since we will also use the unique games conjecture in other theorems, we will briefly explain what it says. The conjecture is concerned with the approximability of the *unique games problem*. Here, we are given a graph and a set of M labels. Moreover, there is a permutation of $\{1, \ldots, M\}$ on each edge, where edge (u, v) has permutation π_{uv}. Each edge only has one permutation, i.e., if π_{uv} exists, then π_{vu} does not exist. In the problem, one has to assign labels to the vertices. We say that an edge is satisfied if $\pi_{uv}(L(u)) = L(v)$, where $L(u)$ is the label assigned to vertex u. The goal is to maximize the number of satisfied edges. The unique games conjecture can be formulated as follows. Given any $\epsilon, \delta > 0$ (with $\delta < 1 - \epsilon$), there exists some $M > 0$ depending on ϵ and δ, such that for the unique games problem with M labels, it is NP-hard to distinguish between instances in which at least a $(1 - \epsilon)$ fraction of the edges can be satisfied, and instances in which at most a δ fraction of the edges can be satisfied.

Theorem 3.5. *There is no* 1.0117*-approximation for a priori TSP with* $|S_j| \leq 4$*, unless P=NP. Assuming UGC, there is no* 1.0242*-approximation, unless P=NP.*

Proof. Consider the reduction from the proof of Theorem 3.3. As a result, we have $\text{OPT}_{\text{TSP}} = 6|E| - 2\text{OPT}_{\text{CUT}}$. If we have an $(1 + \alpha)$-approximation algorithm, we get a

tour with total length at most $(1+\alpha)(6|E| - 2\text{OPT}_{\text{CUT}})$. This implies that there are at least η satisfied scenarios, where

$$\begin{aligned} 4\eta + 6(|E| - \eta) &= (1+\alpha)(6|E| - 2\text{OPT}_{\text{CUT}}) \\ -2\eta &= -2(1+\alpha)\text{OPT}_{\text{CUT}} + 6\alpha|E| \\ \eta &= (1+\alpha)\text{OPT}_{\text{CUT}} - 3\alpha|E|. \end{aligned}$$

These correspond to edges in the cut, hence we have

$$\begin{aligned} \text{Size of cut} &\geq (1+\alpha)\text{OPT}_{\text{CUT}} - 3\alpha|E| \\ &\geq (1+\alpha)\text{OPT}_{\text{CUT}} - 6\alpha\text{OPT}_{\text{CUT}} \\ &= (1-5\alpha)\text{OPT}_{\text{CUT}}, \end{aligned}$$

where the second inequality follows from $\text{OPT}_{\text{CUT}} \geq |E|/2$. Hence, assuming P≠NP, there is no $(1+\alpha)$-approximation for $1-5\alpha \geq \frac{16}{17}$, i.e., there is no 1.0117-approximation. If we also assume that the unique games conjecture holds, there is no $(1+\alpha)$-approximation for $1-5\alpha \geq 0.878567$, i.e., there is no 1.0242-approximation. □

Since graph G' in Figure 3.1 used in Theorem 3.5 is the metric completion of $K_{2,n}$, we get the following corollary.

Corollary 3.6. *A priori TSP in the scenario model on planar bipartite graphs does not admit a PTAS, unless P=NP.*

For instances with scenarios of size at most 6, we can slightly strengthen the result of Theorem 3.5, by reducing from Max E4-Set Splitting, which cannot be approximated with a factor above $\frac{7}{8}$, unless P=NP [64]. This gives an inapproximability of 1.0265 when $|S_j| \leq 6$.

One could also consider the path-version of a priori TSP. In fact, the application on photolithography is modeled as the path-version. It is easy to see that this problem is trivial when $|S_j| \leq 2$ for all j. If we delete t from the graph created in the reduction of Theorem 3.3, we can use this graph and the same reduction to show that the path-version of a priori TSP is NP-hard when $|S_j| \leq 3$. It is easy to see that this graph can be obtained by taking the metric completion of the star graph. Note that we can also adjust Theorem 3.5 to the path-version which will give the same inapproximability result, i.e., there is no 1.0117-approximation, unless P=NP, and there is no 1.0242-approximation if we also assume that the UGC holds.

We can strengthen the inapproximability of a priori TSP by using strong results on permutation constraint satisfaction problems [61]. We will define this class of problems after Theorem 3.7. The problem we need we will call *4-undirected cyclic ordering* (4-UCO). To the best of our knowledge, the problem has never been considered. In this problem, we are given a ground set N and a set of 4-tuples Δ using elements from N. Our goal is to construct an ordering on N that maximizes the number of satisfied 4-tuples. We say that 4-tuple (a,b,c,d) is satisfied if one of the following sequences is a subsequence of the total ordering: (a,b,c,d), (b,c,d,a) , (c,d,a,b), (d,a,b,c), (d,c,b,a), (c,b,a,d), (b,a,d,c), (a,d,c,b). In other words, we get a collection of cycles and we want to find a cyclic ordering maximizing the number of cycles that can be embedded in it. For completeness, we first show that deciding whether all 4-tuples

can be satisfied is NP-complete by using a reduction from Cyclic Ordering. In this problem, we are given a set of ordered triples Δ^{CO} of ground set N. The question is whether there exists a cyclic ordering (a directed tour) on all elements such that each triple is ordered in the right direction. This problem is NP-complete [49].

Theorem 3.7. *4-undirected cyclic ordering is NP-hard.*

Proof. Given an instance of Cyclic Ordering, we create elements ϕ_1 and ϕ_2 for every element $\phi \in N$ and three additional elements, x, y and z. For every element $\phi \in N$ we create ordered sets (x, y, ϕ_1, ϕ_2), (x, z, ϕ_1, ϕ_2) and (y, z, ϕ_1, ϕ_2). For every triple in Δ^{CO}, we create one set by splitting an arbitrary element. For example, we create set (a_1, b_1, b_2, c_1) for triple (a, b, c).

If there exists a cyclic ordering, say $(a, b, \ldots, q)$, we can construct the following satisfying solution for 4-UCO: $(x, y, z, a_1, a_2, b_1, b_2, \ldots, q_1, q_2)$.

On the other hand, suppose that we have a satisfying solution for 4-UCO. Without loss of generality, we may assume that (x, y, a_1, a_2) is visited in this direction. We will show that x, y and z are visited consecutively. Suppose this is not the case and x, y and z are placed at different positions on the solution. This splits the solution into three segments. It is easy to see that for any $\phi \in N$, we must have ϕ_1 and ϕ_2 in the same segment. Now, suppose that these elements are visited in the segment between x and y. This implies that the tour has to visit (x, ϕ_2, ϕ_1, y) in this order. However, this conflicts with scenario (y, z, ϕ_1, ϕ_2). Similarly, placing ϕ_1 and ϕ_2 between y and z implies visiting (y, ϕ_2, ϕ_1, z) in this order. This conflicts with scenario (x, y, ϕ_1, ϕ_2). Thus, we know that the solution visits x, y and z consecutively. We now fix the positions of ϕ_1 for all $\phi \in N$ and we move ϕ_2 to the position next to ϕ_1. This does not conflict with any of the scenario's. The resulting arrangement of the ϕ_1 elements corresponds to an arrangement consistent with Δ^{CO}. □

As said before, we can strengthen the inapproximability results. For that, we first define the class of permutation constraint satisfaction problems (permutation CSPs). A problem in this class with arity k is characterized by a set of permutations Π of $\{1, \ldots, k\}$. In a permutation CSP with arity k, we are given a set N of n elements and a collection of constraints, where each constraint is an ordered k-tuple of elements. The goal is to find an ordering on N that maximizes the number of constraints that is ordered as in Π. An example of a problem in this class is Cyclic Ordering. Here, $\Pi = \{123, 231, 312\}$ meaning that a constraint is satisfied as long as its elements are ordered in the right direction.

A random permutation of N results in satisfying $\frac{|\Pi|}{k!}$ constraints. In [61], it is shown that every permutation CSP of constant arity is approximation resistant. This means that, under the unique games conjecture, the best we can do is to construct a random ordering. In fact, they showed that for any $\epsilon > 0$ it is hard to distinguish between instances where at least a $(1 - \epsilon)$ fraction of the constraints can be satisfied from instances where at most a $(\frac{|\Pi|}{k!} + \epsilon)$ fraction of the constraints can be satisfied. This implies that it is hard to obtain an α'-approximation for Cyclic Ordering for any $\alpha' > \frac{1}{2}$.

It is easy to see that 4-UCO is also in the class of permutation CSPs (with arity 4). A corollary of the work of Guruswami et al. [61] is that there is no approximation algorithm with guarantee larger than $\frac{8}{24} = \frac{1}{3}$, assuming the unique games conjecture

is true. The natural generalization of 4-UCO is 5-UCO. For this problem, there is no algorithm having a guarantee larger than $\frac{10}{120} = \frac{1}{12}$. Given these bounds, we get the following results.

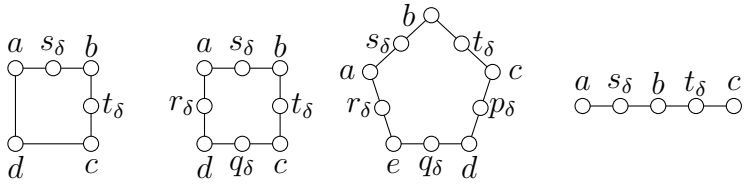


Fig. 3.2: Gadgets used in proofs of Theorem 3.8 and 3.9.

Theorem 3.8. *Under UGC, there is no α-approximation for a priori TSP with*

(a) $\alpha < \frac{10}{9}$ *when* $|S_j| \leq 6$,

(b) $\alpha < \frac{4}{3}$ *when* $|S_j| \leq 8$,

(c) $\alpha < \frac{41}{30}$ *when* $|S_j| \leq 10$,

unless P=NP.

Proof. (a) Given an instance of 4-UCO, we create $|N| + 2|\Delta|$ vertices, one for each element of N and two for each 4-tuple in Δ. We create edges that correspond to 4-tuples in Δ in the following way. For 4-tuple $\delta = (a, b, c, d)$, we have vertices a, b, c, d and vertices s_δ and t_δ. We create edges $(a, s_\delta), (s_\delta, b), (b, t_\delta), (t_\delta, c), (c, d)$ and (d, a), as in Figure 3.2. The scenarios correspond to these six vertices for every tuple. Finally, the distances correspond to the shortest path distances in the created graph. A tuple is satisfied if and only if the tour restricted to the scenario has length 6. A solution satisfying $\frac{1}{3}$ of the scenarios has value at least $\frac{1}{3} \cdot 6 + \frac{2}{3} \cdot 7 = \frac{20}{3}$. A solution satisfying all scenarios has a value of 6. Since it is hard to distinguish between these two cases, we obtain an inapproximability of $\frac{20}{18} = \frac{10}{9}$ for a priori TSP with $|S_i| \leq 6$.

(b) We use a similar reduction. Instead of adding two vertices per tuple, we create four new vertices. In Figure 3.2, these vertices are called $s_\delta, t_\delta, q_\delta$ and r_δ. The scenarios will therefore have size 8. Again, a tuple is satisfied if and only if the tour restricted to the scenario has length 8. However, if we restrict the tour to a scenario corresponding to a non-satisfied tuple, it must have length at least 12. A similar calculation gives an inapproximability of $(\frac{1}{3} \cdot 8 + \frac{2}{3} \cdot 12)/8 = \frac{4}{3}$.

(c) We now reduce from 5-UCO. We add 5 dummy vertices for each scenario and place them between consecutive elements on the cycle. The scenarios will therefore have size 10. Again, a tuple is satisfied if and only if the tour restricted to the scenario has length 10. If we restrict the tour to a scenario corresponding to a non-satisfied tuple, it must have length at least 14. A similar calculation gives an inapproximability of $(\frac{1}{12} \cdot 10 + \frac{11}{12} \cdot 14)/10 = \frac{41}{30}$.

□

For the path-version, we can strengthen previous results by using Betweenness. In this problem, we are given a set of triples Δ^{B} from elements of N. The triple (a, b, c) is satisfied if (a, b, c) or (c, b, a) is a subsequence of the total ordering. The goal is to find an ordering on N maximizing the number of satisfied triples. By [61], the best approximation ratio is $\frac{1}{3}$, assuming UGC. Without this conjecture, there is no approximation algorithm for Betweenness with a factor better than $\frac{1}{2}$, unless P=NP [7].

Theorem 3.9. *There is no $\frac{9}{8}$-approximation for a priori path-TSP with $|S_j| \leq 5$, unless P=NP. Assuming UGC, there is no $\frac{7}{6}$-approximation, unless P=NP.*

Proof. Given an instance of Betweenness, we create a graph with $|N| + 2|\Delta^{\mathrm{B}}|$ vertices. A scenario contains the elements used in a triple and two extra vertices. The edges are drawn in the following way. For triple $\delta = (a, b, c)$, we add edges $(a, s_\delta), (s_\delta, b), (b, t_\delta)$ and (t_δ, c) (Figure 3.2). A triple is satisfied if and only if the path restricted to the scenario has length 4. Assuming UGC, we get that there is no approximation algorithm with guarantee smaller than $(\frac{1}{3} \cdot 4 + \frac{2}{3} \cdot 5)/4 = \frac{7}{6}$ for a priori path-TSP with $|S_j| \leq 5$, unless P=NP. Without assuming UGC, there is no approximation algorithm with guarantee smaller than $(\frac{1}{2} \cdot 4 + \frac{1}{2} \cdot 5)/4 = \frac{9}{8}$, unless P=NP. □

Finally, we note that by using twice the diameter of a scenario as a lower bound, we can show that taking an arbitrary tour as a solution is a $\xi/2$-approximation when $|S_j| \leq \xi$. A random tour gives a value of at most $(\xi^2 - 3\xi + 4)/(2\xi - 2)$ times the optimal value in expectation. This factor approaches $\xi/2$ for ξ large. Similar results hold for the path-version.

3.4 Big scenarios

In this section, we investigate the special case of big scenarios, i.e., the case when each scenario has size at least $n - \xi$, for small ξ. One would expect that simply taking the optimal tour on the entire vertex set V would perform well on these instances. Here, we analyze this option. Let us denote $\textsc{Opt}(X)$ for the optimal value of a tour on X. Further, let $\textsc{Opt}(X)|_Y$ denote the value of the optimal tour on X shortcutted to Y. As before, let D_X denote the diameter of the graph restricted to X.

Lemma 3.10. *For $S \subset V$ and $1 \leq \xi \leq n/2$ such that $|S| = n - \xi$, we have*

$$\textsc{Opt}(V)|_S \leq \textsc{Opt}(S) + \xi D_S.$$

Proof. Suppose $S = V \setminus \{a_1, \ldots, a_\xi\}$. Let $\mathcal{D}_S^{a_i} = \min_{u \in S} c(u, a_i)$ for $i = 1, \ldots, \xi$. Since we can extend our tour on S to V by going back and forth to each a_i, we have

$$\textsc{Opt}(V) \leq \textsc{Opt}(S) + 2 \sum_{i=1}^{\xi} \mathcal{D}_S^{a_i}. \tag{3.1}$$

Furthermore, suppose w.l.o.g. that b_i and d_i are the two vertices in S that are visited before and after a_i in the optimal tour of V. If two consecutive vertices on the tour are not in S, then one can reconstruct the tour accordingly without increasing the length of the tour restricted to S. This is true since vertices not in S do not influence the

tour restricted to S and since $\xi \leq n/2$. Hence, we can assume that there are no two consecutive vertices on the tour that are not in S. Then

$$\begin{aligned}\textsc{Opt}(V) &= \textsc{Opt}(V)|_S + \sum_{i=1}^{\xi}(c(b_i,a_i)+c(a_i,d_i)-c(b_i,d_i)) \\ &\geq \textsc{Opt}(V)|_S + \sum_{i=1}^{\xi}(2\mathcal{D}_S^{a_i}-c(b_i,d_i)) \geq \textsc{Opt}(V)|_S - \xi D_S + 2\sum_{i=1}^{\xi}\mathcal{D}_S^{a_i}. \quad (3.2)\end{aligned}$$

Combining Equations (3.1) and (3.2) we get

$$\textsc{Opt}(V)|_S \leq \mathrm{OPT}(V) + \xi D_S - 2\sum_{i=1}^{\xi}\mathcal{D}_S^{a_i} \leq \textsc{Opt}(S) + \xi D_S.$$

□

The inequality is tight for the graph in Figure 3.3 with $\xi = 2$. We can generalize this tight instance for $\xi \leq n/2$ by adding more diagonal paths.

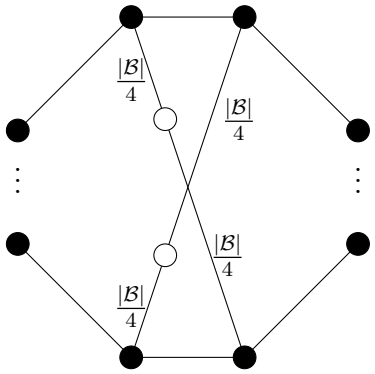


Fig. 3.3: Instance for which inequality of Lemma 3.10 is asymptotically tight for $\xi = 2$, where $\mathcal{B}$ is the set of black (non-white) vertices.

Theorem 3.11. *The optimal solution on V is a $(1+\frac{\xi}{2})$-approximation for a priori TSP with $|S_j| \geq n - \xi$, where $1 \leq \xi \leq \frac{n}{2}$.*

Obviously, these results extend to the Min-Max TSP.

3.5 Nested scenarios

Let us now consider the case of nested scenarios, i.e., $S_1 \subseteq S_2 \subseteq \ldots \subseteq S_m$. Here, the following algorithm gives a constant-factor approximation. First, compute a 1.5-approximate tour T_j for scenario S_j for all j. Let $\psi_1 = 1$. Next, for $h = 2, 3, \ldots$ let ψ_h be the largest number $k > \psi_{h-1}$ for which $T_k \leq 2T_{\psi_{h-1}}$. If no such k exists then let $\psi_h = \psi_{h-1} + 1$. The first-stage tour is obtained by visiting vertices in the order $T_{\psi_1}, T_{\psi_2}, \ldots$.

Theorem 3.12. *The algorithm above is a 9-approximation for nested scenarios.*

Proof. Consider scenario S_j. The last vertices of this scenario will be visited on the tour T_{ψ_h}, where h is the smallest index such that $\psi_h \geq j$. Note that for any $h \geq 2$, we have $T_{\psi_h} > 2T_{\psi_{h-2}}$. Hence, we can decompose the concatenated tour up to T_{ψ_h} into two parts which correspond to even and odd h respectively, such that both parts have geometrically increasing tour lengths. The length of the concatenated tour up to T_{ψ_h} is therefore at most

$$2T_{\psi_{h-1}} + 2T_{\psi_h}.$$

If $\psi_h = j$ then the length of the tour is at most $2T_{\psi_{h-1}} + 2T_{\psi_h} \leq 4T_{\psi_h} = 4T_j \leq 6T_j^*$.

If $\psi_h > j$ then we must have $T_{\psi_h} \leq 2T_{\psi_{h-1}}$ so the length of the tour is at most $2T_{\psi_{h-1}} + 2T_{\psi_h} \leq 6T_{\psi_{h-1}} \leq 9T^*_{\psi_{h-1}} \leq 9T_j^*$. □

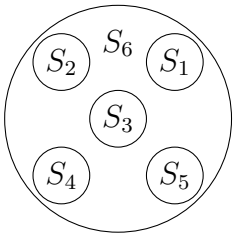


Fig. 3.4: Star-like instance with 6 scenarios.

Finding a constant-factor approximation is still open for laminar scenarios, i.e., when for each i, j, either $S_i \cap S_j = \emptyset$ or $S_i \subseteq S_j$ or $S_j \subseteq S_i$. It is even open in the case when the scenarios have the following star-like structure (illustrated in Figure 3.4).

$$S_i \cap S_j = \emptyset \text{ for } i \neq j, i, j = 1, \ldots, m-1, \text{ and } S_m = \bigcup_{j=1}^{m-1} S_j. \tag{3.3}$$

It would be interesting if one could get a constant-factor approximation for these instances. Finally, observe that the Min-Max TSP for laminar scenarios reduces to standard TSP since the largest scenario determines the value of the solutions.

3.6 Relation with minimum spanning tree problems

The deterministic TSP has a nice relation with the *minimum spanning tree problem* (MST). In this problem, we have to construct a tree connecting all vertices at minimum cost. Because deleting an edge from a tour results in a tree, we have that the optimal value of TSP is at least the optimal value for MST. We can construct a tour by doubling all edges used in the solution of MST, follow the resulting Eulerian tour and shortcut already visited vertices. This tour has a value of at most two times the optimal value of MST. Consequently, if we have an α-approximation for MST, we have a 2α-approximation for TSP. Since MST is polynomially solvable, we obtain a 2-approximation for TSP.

It is interesting to find out if a similar relation between a priori TSP and a priori MST as in the deterministic setting exists. We consider two versions of a priori MST. The first one is defined by Bertsimas [19], who called it *a priori MST*, although it

seems more natural to call it a priori Steiner tree. The second problem is defined by Boria et al. [24], who called it *probabilistic MST under closest ancestor* (PMST-CA). In both problems, we have a graph $G = (V, E)$ and a probability distribution over subsets of vertices. The second problem is only defined on complete graphs and has a root r that is always active. The root is optional in the first problem. The goal is to construct a tree on the entire vertex set in the first stage. A subset A of the vertices, drawn according to the probability distribution, is revealed in the second stage. In the a priori MST, the second-stage tree will be obtained by deleting inactive vertices, provided that the remaining tree stays connected. In the PMST-CA problem, the second-stage tree only contains active vertices. This is done by taking an edge between an active vertex and its closest active ancestor in the rooted first-stage tree. In both problems, the goal is to construct a first-stage tour that minimizes the expected weight of the second-stage tree.

Unfortunately, it turns out that the expected weight of the optimal a priori MST defined by Bertsimas is not smaller than the optimal a priori TSP in general. The gap between the optimal values of a priori MST and a priori TSP can be arbitrarily large.

Theorem 3.13. *There are instances such that the optimal value of the a priori MST-solution can be arbitrarily larger than the optimal value of the a priori TSP-solution.*

Proof. Take a 3-regular graph with girth g. Sachs [92] showed that these graphs exist. Define a scenario for each edge by the endpoints of the edge. All scenarios have the same probability. Any tour on this graph will be shortcutted to a tour of length 2 for each scenario, so the objective value of a priori TSP is 2. Consider the optimal a priori MST. Since this is a tree, it uses $n - 1$ edges. If an edge is in the tree, the corresponding scenario gets value 1. If an edge is not in the tree, the corresponding scenario gets value at least $g - 1$. Since there are $3n/2$ edges (and scenarios), we get at least objective value

$$\left(\frac{3n/2 - (n-1)}{3n/2}\right)(g-1) + \frac{n-1}{3n/2} = \frac{g+1}{3} + \frac{2g-4}{3n} \geq \frac{g+1}{3}.$$

Now, we can take g arbitrarily large, which makes the objective value arbitrarily large and hence the gap with the objective value of a priori TSP. □

Unlike the a priori MST, the PMST-CA problem can be used as a lower bound for a priori TSP. In fact, we only lose a factor 2. Note that the result below only works for the rooted version of a priori TSP, since PMST-CA is defined with a root vertex.

Theorem 3.14. *If there is an α-approximation for the PMST-CA, then there is a 2α-approximation for the a priori TSP, and vice versa.*

Proof. First, we show that the following inequalities are valid, where $\textsc{Opt}_{\text{MST}}$ and $\textsc{Opt}_{\text{TSP}}$ denote the optimal values of PMST-CA and a priori TSP respectively.

$$\textsc{Opt}_{\text{MST}} \leq \textsc{Opt}_{\text{TSP}} \leq 2\textsc{Opt}_{\text{MST}}.$$

The first inequality can be proven by taking the optimal a priori TSP-tour and deleting one edge. This gives a spanning tree on V, called T. If we look at a specific active set A, then the optimal a priori TSP-tour restricted to A will have exactly one edge less

than before. Namely, if we delete edge (a, b) from tour $(1, \ldots, a, b, \ldots, n)$, only edge $(\max\{k \in A : k \leq a\}, \min\{k \in A : k \geq b\})$ will disappear from the restricted tour on A. Note that for active set A, the tour without this edge is the same as T shortcutted to A. Hence, this is a feasible solution for PMST-CA with cost no larger than the optimal value of a priori TSP, and the first inequality has been proven.

The second inequality is proven by doubling the optimal tree and shortcutting the obtained Eulerian tour. In each scenario, the cost of the edges is at most twice the cost of the edges in the tree restricted to the scenario.

Now, if there is an α-approximation for PMST-CA, we double the tree and shortcut the Eulerian tour to obtain a tour on V. This tour has a value of at most

$$2\alpha \textsc{Opt}_{\text{MST}} \leq 2\alpha \textsc{Opt}_{\text{TSP}}.$$

Given an α-approximation for a priori TSP, we take the tour and delete one edge. The resulting tree has a value of at most

$$\alpha \textsc{Opt}_{\text{TSP}} \leq 2\alpha \textsc{Opt}_{\text{MST}}.$$

□

Recall that there is a randomized 3.5-approximation for a priori TSP in the independent decision model (as discussed in Section 2.2). There is also a deterministic 6.5-approximation [105] for this problem. Using Theorem 3.14, we obtain the following corollary.

Corollary 3.15. *There is a randomized* 7*-approximation and a deterministic* 13*-approximation for PMST-CA in the independent decision model. There is also a* $O(\log n)$*-approximation in the black-box model.*

Unfortunately, Theorem 3.14 does not imply a 2-approximation for a priori TSP, since we can prove that PMST-CA is NP-hard in the scenario model. For this, we need the following lemma. This lemma holds for both the scenario and the independent decision model.

Lemma 3.16. *If PMST-CA is NP-hard in the non-metric case, then it is NP-hard in the metric case.*

Proof. One can turn a graph into a graph satisfying the triangle inequality by adding a sufficiently large number M to all distances. In the PMST-CA, this affects every solution by an additive constant equal to $\sum_A p(A)(|A| - 1)M$, where $p(A)$ is the probability that set A is the active set. Hence, the complexity of the problem is preserved in the metric case. □

Boria et al. [24] showed that PMST-CA is NP-hard in the independent decision model, but only for the non-metric case. Using Lemma 3.16, we obtain the following corollary.

Corollary 3.17. *PMST-CA is NP-hard in the independent decision model, even if the triangle inequality is satisfied.*

Theorem 3.18. *PMST-CA in the scenario model is NP-hard.*

Proof. We reduce from the NP-complete problem Exact Cover by 3-Sets [70]. In this problem, we are given $3q$ elements, $X = \{x_1, \ldots, x_{3q}\}$, and m subsets, $Y = \{y_1, \ldots, y_m\}$, with $y_i \subseteq X$ and $|y_i| = 3$ for all i. The problem asks whether there are q sets that together cover all elements. Create the graph as in Figure 3.5. There are m scenarios with probability $1/m$. Define $S_i = X \cup \{r, s, y_i\}$.

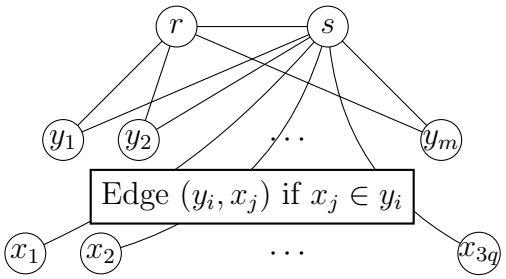


Fig. 3.5: Graph used in proof of Theorem 3.18. Edges (r, s) and (r, y_i) have length 0. Edges (s, y_i) and (y_i, x_j) have length 1. Edges (s, x_j) have length 2. All other edges have length M, where M is a large number.

If there is an exact cover, then construct the following solution. If set y_i is chosen in the cover, then use edge (s, y_i) and the edges from vertex y_i to the corresponding elements of y_i. If set y_i is not in the cover, then use edge (r, y_i). Finally, use edge (r, s). For any y_i in the cover, consider the subtree containing s, y_i and the x_j's corresponding to elements from subset y_i. In scenario S_i, the resulting subtree has value 4. In all other scenarios, vertex y_i will not be present and this subtree will contain three edges from s to the vertices of the elements. Hence, this solution has expected value equal to $q(1/m \cdot 4 + (m-1)/m \cdot 6) = q(6 - 2/m)$.

Note that an optimal tree will never use edges with weight M or a combination of edges that enforce using an edge of weight M in the shortcut solution. This leaves five ways of connecting a specific set vertex y_i and element vertex x_j, where j is in set i, to r and s. The five subtrees are depicted in Figure 3.6.

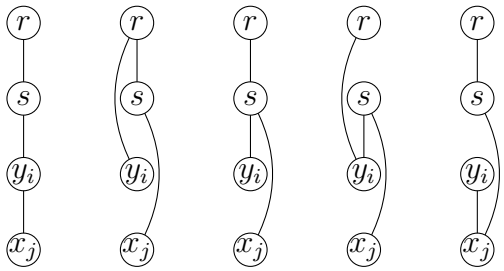


Fig. 3.6: Subtrees T_1 up to T_5.

Tree T_3 is dominated by T_1, since T_1 only has cost 2 for connecting x_j when y_i is inactive while T_3 always has cost 2. Similarly, T_4 is dominated by T_2 and T_5 is dominated by T_1. So, an optimal tree is a combination of T_1 and T_2. Suppose that the tree connects k set vertices to s which connect ℓ elements vertices. The other set vertices are connected to r whereas the other element vertices are connected to s.

Number the k set vertices connected to s as $1, \ldots, k$ and say that set vertex i connects ℓ_i element vertices. This tree has an expected value of

$$\frac{1}{m}\sum_{i=1}^{k}((\ell_i + 1) + 2(3q - \ell_i)) + \frac{m-k}{m}6q = 6q + \frac{1}{m}(k - \ell),$$

which is equal to $q(6 - 2/m)$ if and only if $k = q$ and $\ell = 3q$. Hence, there is a tree with expected value at most $q(6 - 2/m)$ if and only if there is an exact cover. Using Lemma 3.16 completes the proof. □

3.7 Conclusion

In this chapter, we showed how to get constant-factor approximation algorithms for some well-structured instances of the a priori TSP. An interesting question that remains unanswered is whether there exists a constant-factor approximation for a priori TSP with laminar scenarios. More specifically, it is still open whether we can do this on star-like scenarios as defined in Equation (3.3). Next to restricted scenarios we also considered restricted metrics. In Section 3.3 we showed that there is no PTAS for planar bipartite graphs. We do not have such results in the Euclidean plane. It would be interesting to settle the approximability of the problem in this metric. It is easy to construct examples where the optimal solution crosses itself and hence the non-crossing property does not hold. This property was a crucial ingredient of the PTAS by Arora [3] for the deterministic problem. So far, we have not been able to show any lower bound or improve the upper bound for this special case.

We did not succeed in improving the $O(\log n)$-approximation for the general problem. In fact, we conjecture that there is no $o(\log n)$-approximation algorithm for a priori TSP in the scenario model in the general case.

CHAPTER 4

The master tour problem

The results in this chapter were published in [40].

4.1 Introduction

In the *master tour problem* as defined in Deineko et al. [31], one is given n vertices with pairwise distances. A master tour is a tour (an ordering) on the vertices with the property that, for all subsets of the vertices, restricting the tour to the subset results in the optimal tour on that subset. Now, the question is whether a given instance has a master tour. More formally, let τ denote the tour on all vertices, τ_S the tour τ restricted to S, $c(\tau_S)$ the value of τ_S, and $\textsc{Opt}(S)$ the value of the optimal tour on set S. Given vertices V, with $|V| = n$, and distances c_{ij} for $i, j \in V$, does there exist a tour τ such that $c(\tau_S) = \textsc{Opt}(S)$ for all $S \subseteq V$? In [31] it is shown that this decision problem is polynomially solvable.

This problem is related to the field of universal and a priori optimization. In *universal TSP*, one has to construct a tour τ and the quality of this tour is measured by the worst case ratio of $c(\tau_S)$ and $\textsc{Opt}(S)$ over all $S \subseteq V$. In [59] it is shown that one can always find a tour with worst case ratio $O(\log^2 n)$. It was shown in Gorodezky et al. [58] that there are instances such that all tours have a worst case ratio of $\Omega(\log n)$. The master tour problem is the problem of deciding whether there exists a tour τ with worst case ratio equal to 1. The stochastic alternative of universal TSP is *a priori TSP* which is the subject of Chapter 3.

As discussed in Chapter 3, the approximability of a priori TSP in the scenario model with no restrictions on the scenarios or metric space is still open. One approach for proving approximation results is using the master tour lower bound, i.e., to compare the value of the solution with $\sum_j p_j T_j^*$, where p_j and T_j^* are respectively the probability and the optimal tour length for scenario j. The master tour problem can now be viewed as the question whether the optimal value is tight with this lower bound. This was our main motivation to look at the master tour problem in the scenario model.

In this chapter, we extend the master tour problem as defined by Deineko et al. [31] to the *master tour problem with scenarios*. In this model we are given n vertices with pairwise distances and, additionally, a set $\mathcal{S}$ of scenarios and the question becomes whether there exists a tour τ such that $c(\tau_S) = \textsc{Opt}(S)$ for all $S \in \mathcal{S}$. We also investigate other problems in the master setting, namely the Steiner tree problem and the maximum weighted satisfiability problem. In the *master tree problem with*

scenarios, one is given a weighted graph G on n vertices and a set of scenarios $\mathcal{S}$, where the scenarios are subsets of vertices. The question is whether there exists a spanning tree T of G such that $c(T_S) = \textsc{Opt}(S)$ for all $S \in \mathcal{S}$. Here, restricting a tree on a scenario S means that vertices not in S will be deleted provided that this does not disconnect the tree. The way of restricting the tree is the same as in the a priori MST defined by Bertsimas [19] as discussed in Section 3.6. In the *master maximum satisfiability problem with scenarios*, one is given a Boolean formula consisting of clauses $C_1, \ldots, C_m$ with weights $w_1, \ldots, w_m$ using variables $x_1, \ldots, x_n$ and a set of scenarios $\mathcal{S}$, where the scenarios are subsets of clauses. The question is whether there exists a truth-assignment to $x_1, \ldots, x_n$ such that this assignment is optimal for every scenario in $\mathcal{S}$. In this chapter, we show that the master tree problem with scenarios is polynomially solvable if $\mathcal{S} = 2^V$, as is the case for the master tour problem. Further, we show that all three problems are Δ_2^p-complete in general.

4.1.1 The class Δ_2^p

The complexity class Δ_2^p is defined as the class of problems that are polynomially solvable on a deterministic Turing machine augmented with an oracle for an NP-complete problem. In other words, it contains the problems that are efficiently solvable if we have an oracle that gives us the answer to an NP-complete problem in constant time. The class is also known as P^{NP} or $\mathrm{P}^{\mathrm{SAT}}$. It was defined by Stockmeyer [98] as part of the polynomial hierarchy.

To understand the polynomial hierarchy, we first have to define coNP. This is the class of decision problems for which there is a polynomially bounded certificate for a no-answer that can be checked in polynomial time. It can be shown that $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}$ and that an NP-complete problem is not in coNP (and vice versa), unless NP=coNP, which is generally believed to be false. The polynomial hierarchy generalizes the classical complexity classes P, NP and coNP to so called oracle machines. We indicate classes using an oracle by using superscripts. For example, the class P^{C} is the class of decision problems that can be solved in polynomial time if we have an oracle for some C-complete problem. Classes NP^{C} and $\mathrm{coNP}^{\mathrm{C}}$ are defined similarly.

The bottom level of the polynomial hierarchy is defined as $\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathrm{P}$. For $i \geq 1$, the ith level of the polynomial hierarchy consists of $\Delta_i^p = \mathrm{P}^{\Sigma_{i-1}^p}$, $\Sigma_i^p = \mathrm{NP}^{\Sigma_{i-1}^p}$ and $\Pi_i^p = \mathrm{coNP}^{\Sigma_{i-1}^p}$. For example, $\Delta_1^p = \mathrm{P}^{\mathrm{P}} = \mathrm{P}$, $\Sigma_1^p = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP}$ and $\Pi_1^p = \mathrm{coNP}^{\mathrm{P}} = \mathrm{coNP}$. Now, the polynomial hierarchy can be defined as $\mathrm{PH} = \cup_{i=0}^{\infty} \Sigma_i^p$. It follows that we have the inclusions $\Sigma_i^p \subseteq \Delta_{i+1}^p$, $\Pi_i^p \subseteq \Delta_{i+1}^p$ and $\Delta_i^p \subseteq \Sigma_i^p \cap \Pi_i^p$. It is generally believed that these inclusions are strict. If this is not true, one would either get $\Sigma_k^p = \Sigma_{k+1}^p$ or $\Sigma_k^p = \Pi_k^p$ for some k, which would imply a collapse of the hierarchy, i.e., $\mathrm{PH} \subseteq \Sigma_k^p$.

In this chapter, we are interested in $\Delta_2^p = \mathrm{P}^{\mathrm{NP}}$, the class of problems that can be solved in polynomial time using an oracle for an NP-complete problem. It contains both NP and coNP and is contained in $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$ and $\Pi_2^p = \mathrm{coNP}^{\mathrm{NP}}$. A Δ_2^p-complete problem is not contained in NP or coNP, unless NP=coNP. The polynomial hierarchy is illustrated in Figure 4.1. Note that the polynomial hierarchy is contained in PSPACE, the class of decision problems that can be solved using polynomial space. If PH=PSPACE, then the polynomial hierarchy collapses. Hence, it is generally believed that the structure of these classes is correctly depicted in Figure 4.1.

The first natural complete problem for Δ_2^p was found by Papadimitriou [83]. He

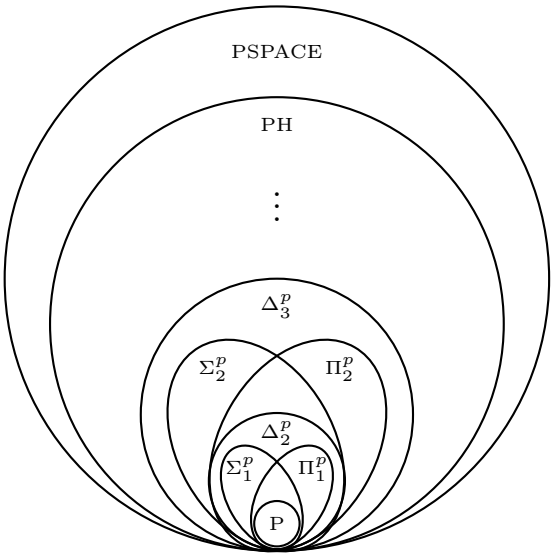


Fig. 4.1: Polynomial hierarchy and PSPACE.

showed Δ_2^p-completeness for the *unique optimum traveling salesman problem*, the problem of deciding whether the optimal TSP-tour is unique. Later, Krentel [75] showed that deciding whether the optimal value is equivalent to 0 mod k is Δ_2^p-complete for the maximum weighted satisfiability problem, the traveling salesman problem, integer programming, and the knapsack problem. He also showed that the problem of deciding whether $x_n = 1$ in the lexicographically maximum assignment satisfying all clauses is also complete in this class. The next lemma shows that the problems we consider fall in the complexity class described above.

Lemma 4.1. *The following problems are contained in* Δ_2^p.

- *The master tour problem with scenarios*
- *The master tree problem with scenarios*
- *The master maximum satisfiability problem with scenarios*

Proof. We prove the result for the master tour problem with scenarios. The other two results are proven similarly. We start with solving TSP for each scenario using a TSP-oracle. This gives an optimal value k_i for scenario S_i. Now, the problem becomes to decide about the existence of a tour on V such that the tour restricted to S_i has value k_i for all i. This problem is in NP, hence it can be solved by an NP-oracle. Concluding, the master tour problem with scenarios is solvable in polynomial time on a deterministic Turing machine with an NP-oracle. □

Other problems that fall naturally in this class are problems on unique optimal solutions. As mentioned before, Papadimitriou showed that the unique optimum traveling salesman problem is contained in this class. A similar proof can be used to show that the *unique optimum Steiner tree problem*, the problem of deciding whether the set of vertices used by the optimal solution is unique, is also in Δ_2^p. The same holds for the *unique optimum maximum weighted satisfiability problem*. In this chapter, we also show that these problems are also Δ_2^p-complete.

4.1.2 Outline

In the next section, we show Δ_2^p-completeness for the master maximum satisfiability problem with scenarios and the unique optimum maximum weighted satisfiability problem. The former is used to show completeness of the master tour problem with scenarios in Section 4.3. After that, we will discuss the complexity of the master tree problem with scenarios. It will be shown that the problem is polynomially solvable when each subset of V is a scenario. Here, it is also shown that the problem is Δ_2^p-complete in general as is the unique optimum Steiner tree problem. We conclude with a summary and discussion.

4.2 The master maximum satisfiability problem

Unlike the master tour problem, the master maximum satisfiability problem is not polynomially solvable when all possible subsets are a scenario, unless P=NP. In this case every single clause appears as a scenario. This means that the assignment should satisfy each clause in order to be optimal for all scenarios. Hence, the problem is equivalent to the satisfiability problem and thus NP-complete. For general scenarios, we obtain the following theorem. For this, we use that it is Δ_2^p-complete to decide whether variable x_n is set to true in the lexicographically maximum satisfying assignment of a given satisfiable formula φ [75]. Here, the ordering on the assignments is defined by the lexicographically ordering of the binary strings $(x_1, x_2, \ldots, x_n)$. We denote this decision problem by *lexicographically maximum SAT*.

Theorem 4.2. *The master maximum satisfiability problem with scenarios is Δ_2^p-complete.*

Proof. By Lemma 4.1, the problem is contained in Δ_2^p. To show hardness, we make a reduction from lexicographically maximum SAT. Given is a Boolean formula φ consisting of m clauses $C_1, \ldots, C_m$ using n variables $x_1, \ldots, x_n$. We create an instance for the master maximum satisfiability problem on the same variables and we create $m+n$ clauses. The first m clauses are C_1 up to C_m. These clauses get weight 2^n. The last n clauses correspond to the variables, i.e., clause $m+i$ is clause (x_i). Clauses $m+i$ will get a weight of 2^{n-i}. We define two scenarios. The first one contains all $m+n$ clauses, the second one only contains clause $m+n$. It is easy to see that, by the choice of the weights, the maximum weight assignment in the new instance corresponds to the lexicographically maximum assignment in the original instance. We conclude with observing that there is a master assignment if and only if $x_n = 1$ in the lexicographically maximum assignment. □

The theorem above will be needed to show completeness in Δ_2^p for the master tour problem with scenarios. Next, we will show that the unique optimum version of this problem is also Δ_2^p-complete. This problem was never considered before. A problem that was considered in the literature is the *unique satisfiability problem*, the problem of deciding whether there is exactly one assignment satisfying all clauses. It was shown that this problem is in D^p [85], the class of languages that are an intersection of an NP-language with a coNP-language. The class is equivalent to BH_2, the second level of the Boolean hierarchy [101]. In [20], it was shown that the unique satisfiability problem

is not D^p-complete under regular reductions. Valiant and Vazirani [100] showed that the problem is D^p-complete under randomized reductions. For the weighted version, we get the following result.

Theorem 4.3. *The unique optimum maximum weighted satisfiability problem is Δ_2^p-complete.*

Proof. The problem is contained in Δ_2^p, as we already observed in the introduction. We reduce this problem from lexicographically maximum SAT. Given is a Boolean formula φ consisting of m clauses $C_1, \ldots, C_m$ using n variables $x_1, \ldots, x_n$. We create the following instance for the maximum weighted satisfiability problem. We use variables $x_1, \ldots, x_n$ and a new variable d. There will be $m + n + 1$ clauses. Again the first m correspond to the m clauses of φ and the next n correspond to the variables of φ. Clause $m + n + 1$ is clause $(\overline{x_n} \vee d)$. The weight of the first m clauses will be 2^{n+1}, the clause (x_i) gets weight 2^{n+1-i} and clause $m+n+1$ gets weight 1. It is easy to see that, by the choice of the weights, the maximum weight assignment in the new instance corresponds to the lexicographically maximum assignment in the original instance. If $x_n = 1$ in the lexicographically maximum assignment, the optimal solution for maximum weighted satisfiability will set $x_n = 1$ and $d = 1$ to get the unique optimal solution. On the other hand, if $x_n = 0$, the lexicographically maximum assignment together with either $d = 1$ or $d = 0$ gives an optimal solution. Hence, the optimal solution is not unique in this case. Thus, we have proven that $x_n = 1$ in the lexicographically maximum assignment if and only if the optimal solution for maximum weighted satisfiability is unique. □

4.3 The master tour problem

As mentioned in the introduction, it was shown in [31] that the master tour problem is polynomially solvable if $\mathcal{S} = 2^V$. The authors showed that an instance has a master tour if and only if the distance matrix is a permuted Kalmanson matrix. The main result in [31] was that permuted Kalmanson matrices can be recognized in polynomial time, hence the master tour problem is polynomially solvable in this case. Before we discuss the general case, we note that the previous result extends beyond the case $\mathcal{S} = 2^V$. A closer examination of the proof shows that you need at least all subsets of V of size 4 to get the characterization above. Since [31] also showed how to find a master tour, we get the following result.

Theorem 4.4 ([31])**.** *Finding a master tour (if one exists) can be done in polynomial time if the set of scenarios $\mathcal{S}$ contains at least each subset of V of size 4.*

We now show that the problem is Δ_2^p-complete for general $\mathcal{S}$.

Theorem 4.5. *The master tour problem with scenarios is Δ_2^p-complete.*

Proof. By Lemma 4.1, the problem is contained in Δ_2^p. To show hardness, we reduce the master maximum satisfiability problem to the master tour problem. The reduction is an extension of the reduction of Krentel [75] and uses ideas from Papadimitriou [83]. We are given an instance of the master maximum satisfiability problem. So we have a Boolean formula φ containing clauses $C_1, \ldots, C_m$ with weights $w_1, \ldots, w_m$ using variables $x_1, \ldots, x_n$ and a set of scenarios $\mathcal{S}$. Each scenario is a set of clauses. We will

show the reduction for the case when all clauses have three literals. The proof naturally extends to the general case.

We construct the graph in Figure 4.2. For each variable, we create two vertices connected by two edges. One edge corresponds to setting the variable to true, the other edge corresponds to false. These variable gadgets are connected in a chain. For each clause, we create two vertices and four edges. The first three edges correspond to the three literals in the clause. The fourth edge corresponds to not satisfying the clause. The clauses will also be connected by a chain, where there is an extra *intermediate* vertex between two clause gadgets. We also add edges between the intermediate vertices in the chain of clause gadgets. These edges ensure that the shortcutting of the tour is done correctly. Finally, the chain of variable gadgets and the chain of clause gadgets are connected using two edges.

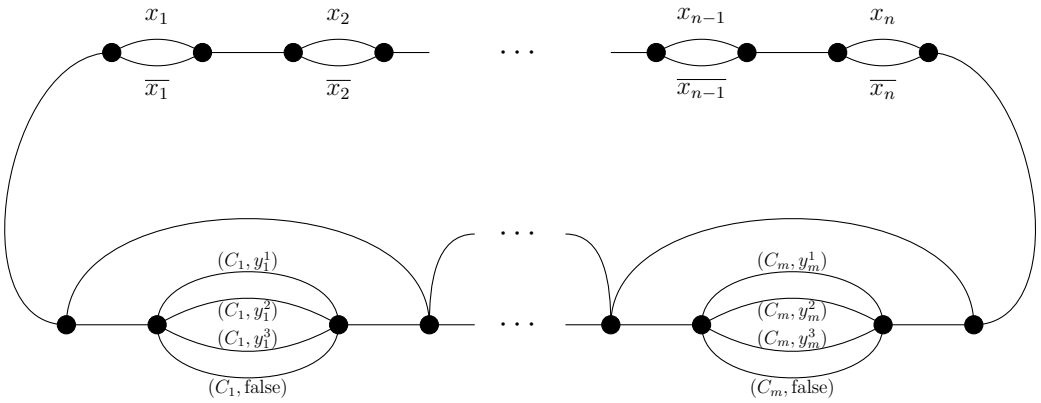


Fig. 4.2: Created instance for the master tour problem (without NAND-gadgets). Edge (C_j, y_j^i) corresponds to the ith literal of clause j.

The edges in the variable gadgets and the clause gadgets are connected through NAND-gadgets [83]. These gadgets make sure that the two edges it connects are not both used in the solution. There will be a NAND-gadget between variable edge x_i and every clause edge corresponding to literal $\overline{x_i}$. Similarly, there will be a NAND-gadget between variable edge $\overline{x_i}$ and every clause edge corresponding to literal x_i. To illustrate how the NAND-gadget works, we depicted an OR-gadget in Figure 4.3. A tour can only get through this gadget by entering it at one endpoint of an edge and leave at the other endpoint. In this way, exactly one of the edges will be used in the solution. The NAND-gadget is an extension of this OR-gadget. It uses more vertices which ensure that at most one of these edges will be used. For a further description of the NAND-gadgets and its proof of correctness, the reader is referred to [83]. Note that the NAND-gadgets also ensure that there are no multiple edges in the graph. In order for the NAND-gadgets to work here, we also need to add some edges between gadgets. This will make sure that the shortcutting works appropriately.

We use the following edge weights. All edges in the graph constructed, except edges corresponding to not satisfying the clause, have length 0, whereas non-edges have length M, where M is a large number. The edge corresponding to not satisfying C_i will get weight w_i. Note that the optimal traveling salesman tour in the created graph corresponds to a maximum weight assignment of φ, i.e., the tour minimizes the

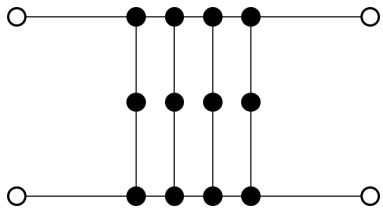


Fig. 4.3: OR-gadget. The top two white vertices and the bottom two white vertices are both connected by an edge in the original graph. The black vertices and their incident edges are added such that any tour should use one of the two original edges, but not both.

weight of the unsatisfied clauses. The result of this theorem still holds in the metric case, since adding a large weight to all distances does not change the complexity of the problem.

We now create the following $|\mathcal{S}|$ scenarios for the master tour problem. Each scenario contains all vertices in the variable chain and all intermediate vertices in the clause chain. Moreover, the scenario corresponding to the jth scenario of the master maximum satisfiability problem contains the vertices corresponding to the clauses in scenario j. It also contains the vertices of the relevant NAND-gadgets.

Suppose there is an assignment that is optimal for all scenarios. Use this assignment to make the following tour. For each variable gadget, use edge x_i if x_i is set to true and vice versa. If a clause is satisfied, choose one of the edges corresponding to a satisfied literal. If a clause is not satisfied, you have to use the corresponding edge. Note that if a clause is not in the scenario, the tour shortcuts the clause gadget through the edge between the intermediate vertices. Now, the tour restricted to a scenario corresponds to the maximum weight assignment, so it is the optimal tour for that scenario. Hence, we have a master tour. On the other hand, suppose that there is a master tour. Then this tour has to correspond to an assignment being optimal for all scenarios. □

The master maximum satisfiability problem with scenarios is already Δ_2^p-complete for two scenarios, which implies that this is also the case for the master tour problem with scenarios.

4.4 The master tree problem

Recall that in the master tree problem with scenarios, one is given a weighted graph G on n vertices and a set of scenarios $\mathcal{S}$, where the scenarios are subsets of vertices. The question is whether there exists a spanning tree T of G such that $c(T_S) = \textsc{Opt}(S)$ for all $S \in \mathcal{S}$, where $c(T_S)$ is the weight of the tree restricted to S and $\textsc{Opt}(S)$ is the optimal tree on S. Here, restricting a tree on a scenario S means that vertices not in S will be deleted provided that this does not disconnect the tree.

We start with showing that the master tree problem with scenarios is polynomially solvable when $\mathcal{S} = 2^V$. In that case V is one of the scenarios, which implies that the master tree is a minimum spanning tree. Therefore, no edge (i, k) with $c_{ij} + c_{jk} < c_{ik}$ for some j will be used. So, without loss of generality we assume that our instance is metric, i.e., our distances satisfy the triangle inequality. Further, observe that every

pair of vertices is a scenario. This implies that the master tree should contain a shortest path for each pair of vertices. We get the following characterization. Note that a metric space is a tree metric if there exists a tree such that the distances in the metric space correspond to the shortest path distances in the tree.

Theorem 4.6. *An instance of the master tree problem with $\mathcal{S} = 2^V$ has a master tree if and only if the graph distances form a tree metric.*

Proof. If the graph distances form a tree metric, one can choose the tree as solution. It is easy to see that every subset is connected in the cheapest way. Now, suppose that the graph distances do not form a tree metric. This means that there is a cycle in the underlying graph with the property that each edge in this cycle is cheaper than the sum of edge weights of the other edges in the cycle. Otherwise, this edge could be deleted which would break the cycle. This implies that each edge in this cycle is the shortest path for its endpoints. Hence, every edge in the cycle should be included to get a master tree. But if they are all included, there is a cycle. So, there is no master tree. □

Hence, the master tree problem with scenarios is solvable in polynomial time when $\mathcal{S} = 2^V$. Note that this result implies that the master tree problem with scenarios is polynomially solvable if $\mathcal{S}$ contains all pairs of vertices and V. As stated in the introduction, it turns out that the general problem is Δ_2^p-complete. The reduction is a slight variant of a standard reduction from the satisfiability problem [65].

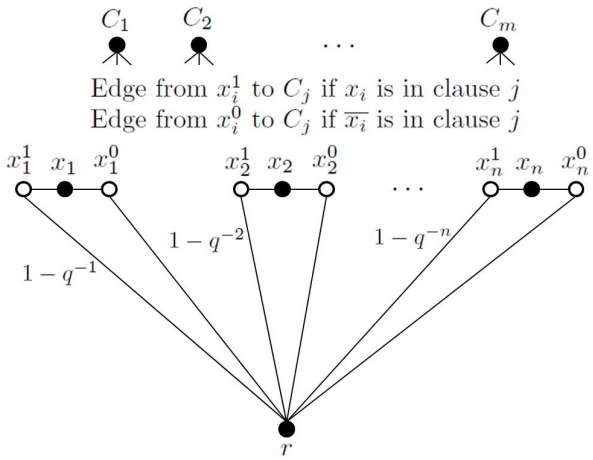


Fig. 4.4: Created instance for Steiner tree.

Theorem 4.7. *The master tree problem with scenarios is Δ_2^p-complete.*

Proof. By Lemma 4.1 the problem is in Δ_2^p. To show hardness, we reduce the problem from lexicographically maximum SAT. Given an instance of this problem, we construct the graph in Figure 4.4. All edge weights in the figure are one except for edges (r, x_i^1) which have weight $1 - q^{-i}$. Edge weights of non-present edges get a weight equal to the shortest path distance in the graph. The black vertices correspond to terminals, whereas the white vertices correspond to Steiner vertices. Further, we have that $q \geq 2$. We create two scenarios. The first scenario contains all terminals, the second scenario

only contains r and x_n. Note that any Steiner tree contains at least one of x_i^1 and x_i^0 for all i. Given that there is a satisfying assignment, the optimal Steiner tree will use exactly one out of x_i^1 and x_i^0 for all i. Because $q^{-i} > \sum_{j=i+1}^{n} q^{-j}$, the optimal solution will correspond to the maximum satisfying assignment. If the maximum satisfying assignment has $x_n = 1$, then there is an optimal Steiner tree using edges (r, x_n^1) and (x_n^1, x_n), so there is a master tree. If there is a master tree, it should use x_n^1 and since it only uses the true-vertex or the false-vertex, the optimal Steiner tree contains the true-vertex and $x_n = 1$ in the maximum satisfying assignment. □

We note that this reduction can also be used to show that computing the optimal Steiner tree is OptP-complete [75]. This basically means that computing the optimal value for the Steiner tree problem is as hard as computing the optimal values for TSP, maximum weighted satisfiability, integer programming, or the knapsack problem. The reduction can also be used to show that deciding whether a given Steiner vertex is in the optimal solution is Δ_2^p-complete. Finally, we will show that if one adjust the reduction slightly, it turns out that deciding whether the set of vertices used by the optimal Steiner tree is unique is also Δ_2^p-complete.

Theorem 4.8. *The unique optimum Steiner tree problem is Δ_2^p-complete.*

Proof. As observed in the introduction, the problem is in Δ_2^p. Given an instance of lexicographically maximum SAT, we construct the graph in Figure 4.5. This is the same graph as constructed in the proof of Theorem 4.7, except that instead of edge (r, x_n^0) with weight 1, there are edges (r, d) and (d, x_n^0) with a weight of $\frac{1}{2}$ each. If $x_n = 1$, the optimal solution uses vertex x_n^1 and the set of vertices used by the optimum is unique. If $x_n = 0$, then there are two sets of vertices resulting in an optimal Steiner tree: one using vertex d and one which does not. □

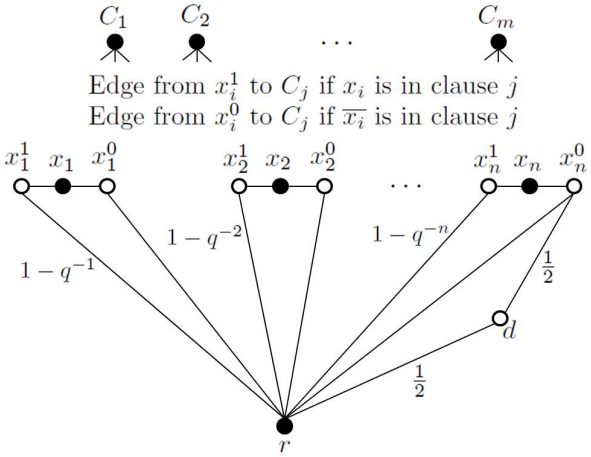


Fig. 4.5: Created instance for Steiner tree.

4.5 Conclusion

In this chapter, we showed that five new problems are complete for Δ_2^p. This was shown for the master tour problem with scenarios, the master tree problem with scenarios,

the master maximum satisfiability problem with scenarios, the unique optimum Steiner tree problem, and the unique optimum maximum weighted satisfiability problem. We also showed that the master tree problem is polynomially solvable when all subsets of V are a scenario. In the introduction, it was discussed that master problems have a natural connection to universal optimization. We think that the main open question in the field of universal and a priori optimization is the approximability of the a priori TSP in the scenario model. Here, we showed that it is Δ_2^p-complete to decide whether the optimal value is tight with the master tour lower bound. Further research should be performed in order to investigate the approximability of this problem.

CHAPTER 5

The Canadian traveler problem

5.1 Introduction

In this chapter, we consider the *Canadian traveler problem* (CTP). Here, we are given a weighted graph $G = (V, E)$ with two specified vertices s and t, and a probability distribution $p : 2^E \to [0, 1] \cap \mathbb{Q}$. This function gives for each $A \subseteq E$ the probability that A is the active set of edges. Only the active edges are present. We need to construct a walk from s to t. However, we only know whether an edge is present when we visit one if its incident vertices. The problem is to find a policy that minimizes the expected length of our walk. Here, a policy may use the observed realizations as input to decide where to go next. We consider the problem in the independent decision model and in the scenario model. In the independent decision model, each edge has a probability of being present and the event of an edge being present is independent of the other edges. In the scenario model, we are given an explicit list of scenarios $\mathcal{S}$, where each scenario describes which edges are present. The probability distribution p is known in advance.

It was shown in [88] that CTP is NP-hard in the scenario model. They also showed that the problem is polynomially solvable if the number of scenarios is bounded by a constant. In the independent decision model, the problem is PSPACE-complete [48] and it is #P-hard to compute the expected length [86, 88]. It is even not possible to describe an optimal policy, unless PSPACE $\subseteq$ P/poly [102]. Here, PSPACE is the class of problems that can be solved using polynomial space (see Figure 4.1 for its relation with PH) and #P is the class of counting problems corresponding to problems from NP. To see that the statement PSPACE $\subseteq$ P/poly is unlikely, it is sufficient to note that this would imply that PSPACE $= \Sigma_2^p \cap \Pi_2^p$ (see Section 4.1 for a discussion on the polynomial hierarchy). On the other hand, the problem is polynomially solvable on disjoint-path graphs [22]. Until now, the computational complexity of the problem was still open on series-parallel graphs. The complexity of CTP on this class of graphs was mentioned as one of the major open problems in [82]. Fried [47] considered CTP on graphs that become trees after deleting t. He conjectured that CTP is intractable in this case. It is easy to see that the class of graphs considered by Fried is a subset of the class of series-parallel graphs. A consequence of our work is that CTP is indeed NP-hard in both cases.

The problem also has an adversarial version, i.e., there is no probability distribution, but there is an adversary that chooses the edges that fail. In this problem, we compare

the length of the walk with the offline optimum, i.e., the optimal solution if we had full knowledge. The worst-case ratio between these values is called the competitive ratio. It is reasonable to consider the restriction that at most k edges fail. This variant is called *k-CTP* and was introduced by Bar-Noy and Schieber [9]. It was shown in [103] that the Backtrack-algorithm that repeatedly chooses the shortest path and returns when the path is blocked, is $2k+1$-competitive. Westphal [103] also showed that no algorithm can beat this bound. As a consequence, we obtain that the Backtrack-algorithm is an $O(n)$-approximation in the independent decision and scenario model, since we have full information about the graph after at most n turns. The Backtrack-algorithm is also an $O(|\mathcal{S}|)$-approximation in the scenario model, since we know which scenario is active after at most $|\mathcal{S}|$ turns. The main open problem is to improve these approximability results.

An important modeling issue, when considering approximation algorithms, is how to deal with an st-disconnected graph. In this case, no walk can reach t. In the independent decision model, this is usually solved by adding an edge from s to t that is present with probability one and has an arbitrarily large length. This modeling choice does not influence the computational complexity of the problem. However, it does influence the analysis of approximation algorithms. To get a sensible model from this perspective, we choose to minimize the conditional expectation of the length given that G contains an st-path. Equivalently, the value of a walk is zero whenever A induces an st-disconnected graph. This way, we get an objective value equal to the conditional expected length divided by the probability of having an st-connected graph. For the independent decision model this is a stronger formulation in the sense that if we have an α-approximation in this formulation, we also have an α-approximation in the former one (with the extra edge), but not vice versa. In the scenario model we can simply avoid this issue by deleting scenarios that induce an st-disconnected graph and normalize the remaining probabilities.

Before giving results on CTP, we consider its relation with the *multi-target graph search problem* (multi-target GSP), a generalization of the *graph search problem* (GSP). In the GSP [74, 6], we are given an edge-weighted graph, a root vertex s and a probability distribution over its vertices. The distribution specifies the probability that the target is at the corresponding vertex. The goal is to find a walk along the vertices that minimizes the expected length until finding the target. When all probabilities are equal (or polynomially bounded), the problem reduces to the traveling repairman problem (TRP), also known as the minimum latency problem. On general metrics, the current best approximation guarantee for TRP is 3.59 [25] building on techniques from [21] and [54] (see Section 2.2 for a brief introduction to these techniques). The problem is even NP-hard on weighted trees [96], but admits a PTAS in the Euclidean plane and on weighted trees [97]. In [6], the authors gave a 40-approximation for GSP by using similar techniques.

Here, we generalize the problem to the case where there can be multiple targets. For this, we again consider the independent decision model and the scenario model. In the independent decision model, each vertex has a probability of having a target and the event of having a target at a vertex is independent of the presence of targets at other vertices. In the scenario model, we are given an explicit list of scenarios $\mathcal{S}$, where each scenario describes at which vertices a target is present. Now, we want to minimize the expected length of the walk until a target has been found, given that there is at

least one target. The following theorem states how the multi-target GSP is related to the CTP.

Theorem 5.1. *The multi-target GSP is equivalent to a special case of the CTP.*

Proof. We proof the theorem for the independent decision model. The proof for the scenario model is similar and omitted here. Given is an instance of multi-target GSP, i.e., an edge-weighted graph $G = (V, E)$ with root s and a probability p_i for each vertex i. Create an instance of CTP by copying the graph and taking s as the start vertex. Add a new vertex t and add edges between t and all vertices in V. Edge (i, t) has weight zero and is active with probability p_i. All edges in E are active with probability one. The reduction is illustrated in Figure 5.1. Now, there is walk in the multi-target GSP-instance of expected length at most B if and only if there is a walk in the created CTP-instance of expected length at most B. □

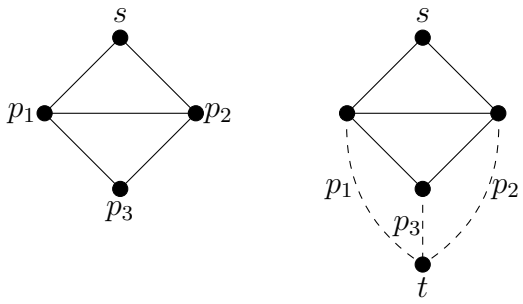


Fig. 5.1: An instance of multi-target GSP (left) and the same instance shown as a special case of CTP (right). Here, dashed edges have length zero and solid edges have length one. The p_e next to a dashed edge denotes the probability of this edge; solid edges have probability one.

We show that we can use the techniques from TRP-algorithms [21, 54] to obtain constant-factor approximations for the multi-target GSP in the independent decision model. More precisely, we give a $(3.59 + \epsilon)$-approximation for tree metrics and a $(14.4 + \epsilon)$-approximation for general metrics. For the scenario model, we show that the problem is NP-hard on a star, even when each scenario contains only two targets. For CTP, we investigate the adaptivity gap in the independent decision model. This gap measures the loss when restricting ourselves to non-adaptive solutions. We show that this is strictly greater than one on trees.

We also show that multi-target GSP in the independent decision model is NP-hard on trees. From this fact it follows that CTP is NP-hard on series-parallel graphs. For the scenario model, we strengthen the result by [88], by showing that CTP is even NP-hard on disjoint-path graphs and cactus graphs. The CTP in the independent decision model is solvable on these graph types [22]. The NP-hardness of CTP on general graphs in the scenario model easily follows from the hardness of GSP and the relation between these two problems, as stated in Theorem 5.1. In the next section, we will discuss results on the independent decision model and in Section 5.3 we will treat all results concerning the scenario model.

5.2 Independent decision model

5.2.1 Multi-target graph search

We start by showing that multi-target GSP is strongly NP-hard on trees by a reduction from TRP on trees [96]. The idea is to assign a probability p to each vertex, where p is small. We then prove that it is sufficient to choose p only polynomially small. Remember that we defined the objective function as the conditional expected length divided by the probability of having an *st*-connected graph.

Theorem 5.2. *Multi-target graph search in the independent decision model is strongly NP-hard on trees.*

Proof. We reduce from TRP on trees where the edge lengths are polynomially bounded. As shown in [96], this problem is still strongly NP-hard. Suppose that the sum of the edge lengths W is bounded by n^ℓ for some constant ℓ. Since edge weights are assumed to be integers, we have $W \geq n-1$. Now, create an instance of multi-target GSP by assigning a probability $p < 1 - e^{-1/n^{2\ell+1}}$.

If we denote C_i for the latency of vertex i in an arbitrary solution of TRP, it is easy to see that if we use this solution until we find a target, we have a walk for multi-target GSP with expected length

$$\mathcal{C} = pC_1 + (1-p)pC_2 + \ldots + (1-p)^{n-1}pC_n.$$

Hence,

$$\mathcal{C}/p = \sum_{i \in V} C_i + \sum_{k=1}^{n-1}((1-p)^k - 1)C_{k+1}.$$

The second summation in the equation above is always negative if $0 < p < 1$. Hence, if we take the optimal TRP-solution as our multi-target GSP-solution, we have $\mathcal{C}/p < \sum_{i \in V} C_i^*$, where C_i^* is the latency of vertex i in the optimal solution. If we choose p small enough such that the aforementioned summation is also larger than -1 for any TRP-solution, we can say the following. If we take a suboptimal TRP-solution, we have $\mathcal{C}/p \geq \sum_{i \in V} C_i^* + 1 + \sum_{k=1}^{n-1}((1-p)^k - 1)C_{k+1} > \sum_{i \in V} C_v^*$. Hence, if we choose p such that $\sum_{k=1}^{n-1}((1-p)^k - 1)C_{k+1} > -1$ for every TRP-solution, the optimal walk of multi-target GSP coincides with the optimal TRP-solution.

We can now show that $p < 1 - e^{-1/n^{\ell+1}}$ implies $\sum_{k=1}^{n-1}((1-p)^k - 1)C_{k+1} > -1$ for every TRP-solution. Namely,

$$\begin{aligned}
& p < 1 - e^{-1/n^{2\ell+1}} \\
\Rightarrow & 1 - p > e^{-1/n^{2\ell+1}} \\
\Rightarrow & 1 - p > \left(1 - \frac{1}{n^{2\ell}}\right)^{1/n} \\
\Rightarrow & (1-p)^n > 1 - \frac{1}{W^2} \\
\Rightarrow & ((1-p)^{n-1} - 1)W^2 > -1
\end{aligned}$$

$$\Rightarrow \sum_{k=1}^{n-1}((1-p)^{n-1}-1)C_{k+1} > -1$$
$$\Rightarrow \sum_{k=1}^{n-1}((1-p)^{k}-1)C_{k+1} > -1,$$

where in the second implication, we used $(1-\frac{1}{n})^n \leq e^{-1}$ for $n \geq 1$. To show strong NP-hardness, we need that p is polynomially bounded. For this, we need that $e^{-x} < 1 - x + x^2/2$. We get that

$$1 - e^{-1/n^{2\ell+1}} > \frac{1}{n^{2\ell+1}} - \frac{1}{2n^{2(2\ell+1)}} \geq \frac{1}{2n^{2\ell+1}} = \Omega\left(\frac{1}{n^{2\ell+1}}\right).$$

This completes the proof. □

The proof above also shows that multi-target GSP on general graphs is at least as hard to approximate as TRP. Since TRP has no constant-factor approximation algorithm if the distances violate the triangle inequality [93], we assume in the remainder of this section that the distances form a metric space. To get approximability results, we take the general approach in TRP-literature. For this, we define the following problem.

Definition 5.3. *Suppose we are given an edge-weighted graph $G = (V, E)$ with root s, probability p_i for each vertex i and a quota P. In the probability quota TSP (PQ-TSP), the goal is to find a tour of minimum length on set $X \ni s$ such that $1-\prod_{i\in X}(1-p_i) \geq P$, i.e., the probability of finding a target in X is at least P.*

Note that a polynomial time algorithm for the rooted version implies a polynomial time algorithm for the unrooted version by trying all n possible choices for the root. Assume for now that we have a β-approximation for this problem. We now want to solve the opposite problem, i.e., find a tour of length at most L that maximizes the probability of finding the target. Using binary search, we can use the β-approximation for PQ-TSP to solve the former problem with length bound βL.

Our algorithm now proceeds as follows. Let $L_0 = 2\gamma^U$, where U is uniformly distributed on $[0, 1]$ and γ is the inflation parameter. Further, let $L_j = \gamma^j L_0$. Now, for each L_j use the procedure above to obtain tour $T(L_j)$. Concatenate these tours, where already visited vertices are shortcutted. For each tour, choose the direction uniformly at random. We will show that if we take γ to be the root of $x\ln(x) = x + 1$, which is approximately 3.59, we get a 3.59β-approximation for multi-target GSP. First, we need to prove the following lemma.

Lemma 5.4. *Given that there is a target in $T(L_j)$, the expected time between starting $T(L_j)$ and finding a target is at most $L_j/2$, if we take the direction uniformly at random.*

Proof. For each non-empty realization, it is easy to see that if we sum the time until finding a target of the left and right direction, we get at most L_j. Since we take the direction uniformly at random, we know that the expected time between starting $T(L_j)$ and finding a target is at most $L_j/2$. □

Theorem 5.5. *Given a β-approximation for PQ-TSP, our algorithm is an 3.59β-approximation for the multi-target GSP.*

Proof. Let C_p^* be the length of the optimal tour until probability p is reached. By abusing notation, we will also use C_p^* to denote the optimal solution. Denote by OPT the value of the optimal solution. Note that $\text{OPT} = \int_0^{\tilde{p}} C_p^* dp$, where $\tilde{p} = 1 - \prod_{i=1}^n (1-p_i)$. Similarly, we define C_p^{ALG} and note that the objective value of the tour produced by the algorithm is $\text{ALG} = \int_0^{\tilde{p}} C_p^{\text{ALG}} dp$.

Suppose that $C_p^* = \rho\gamma^\ell$, where $1 \leq \rho < \gamma$. We distinguish two cases: $\rho < \gamma^U$ and $\rho \geq \gamma^U$.

If $\rho < \gamma^U$, then there exists a path with length at most $\gamma^U\gamma^\ell$ that reaches a probability of p. This means that $T(L_\ell)$ has a probability of at least p and length at most $2\beta\gamma^U\gamma^\ell$. Using Lemma 5.4, we know we reach this probability on average after time at most $\beta(\sum_{j=0}^{\ell-1} L_0\gamma^j + \frac{1}{2}L_0\gamma^\ell) \leq \beta\gamma^\ell L_0 \left(\frac{\gamma+1}{2(\gamma-1)}\right)$.

If $\rho \geq \gamma^U$, we still have $\rho < \gamma \leq \gamma^U\gamma$. Hence, there is a path with length at most $\gamma^U\gamma^{\ell+1}$ reaching probability p. This means that $T(L_{\ell+1})$ reaches probability p after length at most $2\beta\gamma^U\gamma^{\ell+1}$. Now, again by Lemma 5.4, we know we reach this probability on average after time at most $\beta(\sum_{j=0}^{\ell} L_0\gamma^j + \frac{1}{2}L_0\gamma^{\ell+1}) \leq \beta\gamma^{\ell+1}L_0 \left(\frac{\gamma+1}{2(\gamma-1)}\right)$. In the first case, we have $\log_\gamma \rho \leq U \leq 1$, and in the second case, we have $0 \leq U \leq \log_\gamma \rho$. Taking expectations over U gives

$$\begin{aligned} C_p^{\text{ALG}} \leq & \int_{\log_\gamma \rho}^1 \left(\beta L_0\gamma^\ell \left(\frac{\gamma+1}{2(\gamma-1)}\right)\right) dU \\ & + \int_0^{\log_\gamma \rho} \left(\beta L_0\gamma^{\ell+1} \left(\frac{\gamma+1}{2(\gamma-1)}\right)\right) dU \\ = & \frac{\gamma+1}{\ln\gamma}\beta C_p^*. \end{aligned}$$

Optimizing over γ gives $\gamma = 3.59$, the unique root of $x\ln x = x+1$, which gives a loss of 3.59β in the length. Hence, we obtained that $\text{ALG} \leq 3.59\beta\text{OPT}$. Similar to Theorem 2.7, we can derandomize this algorithm by using techniques from [54]. □

We now show that PQ-TSP is NP-hard, even on stars, but that we can obtain a FPTAS for this problem on tree metrics. In the following reduction, we reduce from PARTITION. In this problem, we are given n integers $a_1, \ldots, a_n$, and the question is whether there exists a set $X \subseteq \{1, \ldots, n\}$ such that $\sum_{i\in X} a_i = \frac{1}{2}\sum_{i=1}^n a_i$. This problem is weakly NP-hard [70].

Theorem 5.6. *PQ-TSP is NP-hard on stars.*

Proof. Given is an instance of PARTITION. Let $B = \frac{1}{2}\sum_{i=1}^n a_i$. Construct a star graph with n leafs, where each leaf is associated with an integer from the PARTITION-instance. Assign a weight of a_i to the edge connecting the root with the leaf corresponding to integer a_i, and set the probability of finding a target at this leaf equal to a_i/K, where $K > 2^n(\max_i a_i)^2$. We will show that it is NP-hard to decide whether there is a tour of length at most $2B$ with probability larger than $(B-1)/K$. For this, we need that for every $X \subseteq \{1, \ldots, n\}$ we have $\sum_{i\in X} a_i/K - 1 + \prod_{i\in X}(1 - a_i/K) < 1/K$. This is true since

$$\sum_{i\in X} a_i/K - 1 + \prod_{i\in X}(1-a_i/K) = \sum_{X'\subseteq X, |X'|\geq 2} (-1)^{|X'|} \prod_{i\in X'} (a_i/K) \leq 2^n (\max_i a_i)^2/K^2 < 1/K.$$

Suppose we have a yes-instance for PARTITION, say X. The tour on the vertices corresponding to X has length $2B$ and probability

$$1 - \prod_{i\in X}(1 - a_i/K) > \sum_{i\in X} a_i/K - 1/K = (B-1)/K.$$

If we have a no-instance for PARTITION, we either can choose a set Y such that $\sum_{i\in Y} a_i > B$ or $\sum_{i\in Y} a_i < B$. In the first case, the corresponding tour would have length larger than $2B$. In the second case, the probability will be

$$1 - \prod_{i\in Y}(1 - a_i/K) \leq \sum_{i\in Y} a_i/K \leq (B-1)/K.$$

Since K is exponential in the size of the PARTITION-instance, we have shown that PQ-TSP is weakly NP-hard. □

Theorem 5.7. *There is a FPTAS for PQ-TSP on tree metrics.*

Proof. We first give a dynamic programming algorithm that, for each tour length, finds the solution with maximum probability. This algorithm runs in pseudopolynomial time. Then, we show how to round the lengths of the edges such that we lose at most a factor $(1+\epsilon)$ in the length of the tour. Note that the solution is a double tree, hence it is sufficient to find a tree.

First, assume that we are given a binary tree with positive probabilities only at the leafs. This can be assumed without loss of generality by adding vertices with probability zero and edges with length zero. We also guess the farthest vertex from the root visited by our tour, and remove all vertices at a larger distance from the root. Call the corresponding distance D and note that, given that our guess is correct, we have $\text{OPT} \geq D$, where OPT is the value of the optimal solution of PQ-TSP. Our dynamic programming formulation now has a state for each vertex-length pair, denoted by (i, L). We define $f(i, L)$ as the maximum probability of finding a target in the subtree rooted at i using a tree of length at most L. By definition, $f(i, L) = 0$ whenever $L < 0$. If we denote the left and right child of i as $\hat{i}$ and $\tilde{i}$ respectively and if we use $c(\cdot, \cdot)$ for the edge lengths, we can compute this value as follows.

$$f(i, L) = \max_{0\leq\lambda\leq L} \{1 - (1 - f(\hat{i}, \lambda - c(i, \hat{i})))(1 - f(\tilde{i}, L - \lambda - c(i, \tilde{i})))\}.$$

The optimal solution can be computed by applying binary search on $f(s, L)$ for different L. It is easy to see that, for each guess of the farthest vertex, this algorithm runs in $O(n^3 D^2(\log n + \log D))$. To get a polynomial time algorithm, we pick $\epsilon > 0$ and we round each of the edge lengths up to its nearest multiple of $\epsilon D/n$. This way, we lose at most $\epsilon D \leq \epsilon\text{OPT}$ and hence a factor $(1+\epsilon)$ in the length of the tour, but we have reduced the running time to $O(n^5(\log n + \log D)/\epsilon^2)$. Since the total running time is now $O(n^6(\log n + \log D)/\epsilon^2)$, we have obtained a fully polynomial time approximation scheme for PQ-TSP on tree metrics. □

Note that to obtain approximation results for multi-target GSP using Theorem 5.5, it is sufficient to approximate the opposite problem of PQ-TSP. The proof above shows that for a given L, we can find a tour of length at most $(1+\epsilon)L$ and probability P if there exists a tour of length at most L and probability P in $O(n^6/\epsilon^2)$ time.

Corollary 5.8. *There is a $(3.59+\epsilon)$-approximation for multi-target GSP on tree metrics.*

Before discussing the approximability of multi-target GSP on general metrics, we first generalize PQ-TSP. We define the *polymatroid quota TSP* as follows. Here, a function q is polymatroid [80] if it satisfies

- $q(\emptyset) = 0$,
- $q(\mathcal{X}) \leq q(\mathcal{Y})$ for all $\mathcal{X} \subseteq \mathcal{Y}$,
- $q(\mathcal{X} \cup \{z\}) - q(\mathcal{X}) \geq q(\mathcal{Y} \cup \{z\}) - q(\mathcal{Y})$ for all $\mathcal{X} \subseteq \mathcal{Y}$ and all $z \notin \mathcal{Y}$.

Definition 5.9. *Suppose we are given an edge-weighted graph $G = (V, E)$ with root s, polymatroid function $q : 2^V \to \mathbb{Q}$ and a quota Q. In the polymatroid quota TSP, the goal is to find a tour of minimum length on set $X \ni s$ such that $q(\bar{X}) \leq Q$, where $\bar{X}$ is the complement of X.*

PQ-TSP is a special case of polymatroid quota TSP. This can be seen by taking $q(X) = 1 - \prod_{i \in X}(1-p_i)$ and setting $Q = 1 - \frac{\prod_i (1-p_i)}{1-P}$. Note that it also generalizes k-TSP [52], the problem of finding a tour on k vertices with minimum length, since $q(X) = |X|$ is polymatroid and $|X| \geq k$ is equivalent to $|\bar{X}| \leq n - k$.

To get approximation results for polymatroid quota TSP, we use results for the *prize-collecting TSP with polymatroid penalties*. In this problem, we can decide not to visit certain vertices. The penalty cost of not visiting these vertices is determined by a polymatroid function π. Now, we have to find a tour on a subset X of the vertices such that the sum of tour cost $L(X)$ and penalty cost $\pi(\bar{X})$ is minimized.

When $\pi(\bar{X}) = \sum_{i \in \bar{X}} \pi_i$, where π_i is the penalty paid for not visiting vertex i, the problem is known as the *prize-collecting TSP*. For this problem, Goemans and Williamson [55] showed that their algorithm produces a solution that visits X such that

$$L(X) + \pi(\bar{X}) \leq 2 \sum_{X \subseteq V \setminus \{s\}} y_X, \tag{5.1}$$

where $\sum_X y_X$ is the objective value of the dual of the LP-relaxation of the prize-collecting TSP. Goemans and Kleinberg [54] observe that the proof of [55] even shows that

$$L(X) + 2\pi(\bar{X}) \leq 2 \sum_{X \subseteq V \setminus \{s\}} y_X,$$

if $\pi(\bar{X}) = \sum_{i \in \bar{X}} \pi_i$. On the other hand, it was observed in [66] that the Goemans-Williamson algorithm [55] is a 2-approximation for prize-collecting Steiner tree with polymatroid penalties. One can extend this result to the prize-collecting TSP with polymatroid penalties. These results hold since (5.1) also holds if π is a polymatroid function. Moreover, we can easily extend this result to the following lemma, again using the proof of [55].

Lemma 5.10. *The Goemans-Williamson algorithm ([55], Section 4.3) produces a solution that visits set X with the property*

$$L(X) + 2\pi(\bar{X}) \leq 2 \sum_{X \subseteq V \setminus \{s\}} y_X,$$

where $\sum_X y_X$ is the objective value of the dual of the LP-relaxation of the prize-collecting TSP.

This lemma can be used to obtain a 5-approximation for polymatroid quota TSP using Lagrange relaxation as demonstrated in [28]. By using techniques from [5], we can improve this factor to $4+\epsilon$. Hence, we have a $(4+\epsilon)$-approximation for PQ-TSP on general metrics and hence, by Theorem 5.5, a $(14.4+\epsilon)$-approximation for multi-target GSP. However, since we perform binary search on the probability quota, the algorithm runs logarithmically in the ratio of the maximum and minimum non-zero difference between the probabilities of two sets. This gap is bounded by a single exponential function of the input size when the probabilities are polynomially bounded.

Corollary 5.11. *There is a $(14.4 + \epsilon)$-approximation for multi-target GSP when the probabilities are polynomially bounded.*

5.2.2 Canadian traveler problem

In this subsection, we discuss the complexity and approximability of CTP in the independent decision model. For the complexity, it follows from Theorem 5.1 and 5.2 that it is NP-hard on series-parallel graphs.

Corollary 5.12. *The CTP in the independent decision model is NP-hard on series-parallel graphs.*

Since CTP in the independent decision model is easy on disjoint-path graphs, this basically settles the computational complexity of the problem. For the approximability, we look at the power of being adaptive. To investigate this, we define non-adaptive policies and the adaptivity gap. In order to have a correct definition, we assume that all edges incident to t have weight zero. This assumption is without loss of generality since we can extend each edge to t with a new edge of weight zero and probability one.

Definition 5.13. *We say that a policy is non-adaptive if it has a fixed ordering of the vertices and it always tries to reach the first unvisited vertex in this order. If the vertex is disconnected with the root, i.e., we cannot reach it, the vertex is discarded and the next vertex in the ordering is considered. If it reaches one of the vertices adjacent to t and the edge to t is present, it visits t next.*

Definition 5.14. *Let $\textsc{Opt}(I)$ and $\mathrm{NA}(I)$ be the value of the optimal solution and the value of the optimal non-adaptive policy for instance I of CTP respectively. Then, the adaptivity gap is defined as*

$$\sup_I \frac{\mathrm{NA}(I)}{\textsc{Opt}(I)}.$$

On trees, i.e., when the graph without t is a tree, a non-adaptive policy is just a permutation of the vertices, where we can skip a subtree if the top edge of the subtree is not present. The instance in Figure 5.2 shows that we have to be adaptive in order to be optimal.

Theorem 5.15. *The adaptivity gap of independent-CTP on trees is larger than 1.*

Proof. Consider the instance in Figure 5.2. There are three non-adaptive solutions that have to be considered. These are $Y_1 = (s, e, a, b, d, c)$, $Y_2 = (s, a, b, d, c, e)$ and $Y_3 = (s, a, d, b, c, e)$. We get the following expected values, denoted by $C(\cdot)$.

$$\begin{aligned} C(Y_1) &= 0.5 \cdot 100 + 0.5(0.1 \cdot 2 + 0.1 \cdot 0.9 \cdot 6 + 0.1 \cdot 0.9^2 \cdot 16) \approx 51.018 \\ C(Y_2) &= 0.1 \cdot 2 + 0.1 \cdot 0.9 \cdot 6 + 0.1 \cdot 0.9^2 \cdot 16 + 0.9^3 \cdot 0.5 \cdot 124 \approx 47.234 \\ C(Y_3) &= 0.1 \cdot 4 + 0.1 \cdot 0.9 \cdot 8 + 0.1 \cdot 0.9^2 \cdot 14 + 0.9^3 \cdot 0.5 \cdot 122 \approx 46.723 \end{aligned}$$

However, an adaptive solution is allowed to condition on the presence of edge (s, e). If this edge is active, it will do (s, a, b, c, d, e). Otherwise, it will do (s, a, b, d, c). This gives the following value of adaptive solution Ad.

$$\begin{aligned} C(Ad) = \; & 0.5(0.1 \cdot 4 + 0.1 \cdot 0.9 \cdot 8 + 0.1 \cdot 0.9^2 \cdot 14 + 0.9^3 \cdot 122) \\ & + 0.5(0.1 \cdot 2 + 0.1 \cdot 0.9 \cdot 6 + 0.1 \cdot 0.9^2 \cdot 16) \approx 46.614 \end{aligned}$$

Hence, the adaptivity gap of independent-CTP on trees is at least 1.0023. □

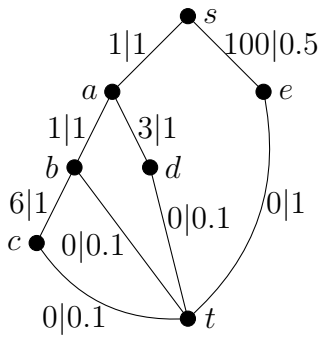


Fig. 5.2: Instance with adaptivity gap greater than 1. Labels denote "length|probability".

In order to achieve larger lower bounds, one has to try other approaches than used above by the following reasoning. In the instance in Figure 5.2, the optimal TSP-solution for the subgraph induced by $\{a, b, c, d\}$ is (a, d, b, c), while the optimal TRP-solution is (a, b, d, c). The adaptive algorithm is stronger since it can choose which of these solutions to use depending on the presence of edge (s, e). Extending this approach to get better lower bounds will not result in satisfying results. This is because one can approximate TSP and TRP simultaneously. To see this, one could take the algorithm by Goemans and Kleinberg [54] for TRP using Garg's 2-approximation for k-TSP [52] as its subroutine. This algorithm is a 7.2-approximation for TRP, but

it is also an 8-approximation for TSP. Hence, in order to achieve significantly larger lower bounds, one has to use more elaborate approaches.

Let us end this section with a discussion on Definition 5.13. On trees, it perfectly captures the intuition of a non-adaptive policy: a permutation of the vertices. It is debatable what a correct description of a non-adaptive policy is on general graphs. Definition 5.13 says that a non-adaptive policy will always try to reach the next unvisited vertex, but it is not specified how to do this. A policy could for example use the shortest path with respect to edge lengths to the next vertex. It is also allowed to use the path that maximizes the probability of reaching the next vertex. Here, we want to emphasize that it is arguable what the correct definition of a non-adaptive policy is.

5.3 Scenario model

We now consider multi-target GSP and CTP in the scenario model. We first discuss the complexity of the former problem on star metrics, and show that it is NP-hard when each scenario contains only two targets. As a consequence, by Theorem 5.1, CTP is NP-hard on disjoint-path graphs. Finally, we show that CTP in the scenario model on cactus graphs (graphs with the property that each edge is in at most one cycle) is also NP-hard.

5.3.1 Multi-target graph search

First observe that multi-target GSP in the scenario model is a generalization of GSP, since this is the case when $|S| = 1$ for all $S \in \mathcal{S}$. On star metrics, this problem is easily solvable by visiting the leafs in non-decreasing order of the ratio of the probability and the length of the edge to the leaf.

Let us now discuss the case where $|S| = 2$ for all $S \in \mathcal{S}$, called the two-target GSP. We will give a reduction from the min sum vertex cover problem (MSVC). In this problem, we are given a graph $G = (V, E)$ and we have to find a linear ordering of the vertices, i.e., a bijection $f : V \to \{1, \ldots, n\}$. The cover time of edge (u, v) is defined as the minimum of $f(u)$ and $f(v)$. The goal is to construct f such that the sum of cover times of the edges is minimized. In [43], it was shown that the problem is NP-hard.

Theorem 5.16. *Two-target GSP is NP-hard on stars.*

Proof. Given an instance of MSVC, i.e., a graph $G = (V, E)$, we create the following instance for two-target GSP. First, construct an unweighted star with $|V|$ leafs. Secondly, we create a scenario $\{u, v\}$ for each edge $(u, v) \in E$. Each scenario has probability $1/|E|$. Now, we will show that there is a solution for MSVC in the original instance with value z if and only if there is a solution for two-target GSP in the created instance of value $\frac{1}{|E|}(2z - |E|)$.

We can use the linear ordering as our tour for our instance of two-target GSP and vice versa. Now, we have found the target after walking $2h-1$ distance (and not before $2h - 1$) if the active scenario corresponds to an edge that is covered after h steps in the ordering. Hence, there is a solution for MSVC in the original instance with value

z if and only if there is a solution for two-target GSP in the created instance of value

$$\frac{1}{|E|}\sum_{(u,v)\in E}(2\min\{f(u),f(v)\}-1)=\frac{1}{|E|}(2z-|E|).$$

□

5.3.2 Canadian traveler problem

For the scenario model, we can prove NP-hardness for two extreme cases of series-parallel graphs, namely disjoint-path graphs and cactus graphs. That this is true for the former graph type follows immediately from Theorem 5.1 and 5.16, since adding edges from the leafs of the star to a new vertex t leads to a disjoint-path graph ($K_{2,n}$).

Corollary 5.17. *CTP in the scenario model is NP-hard on disjoint-path graphs.*

For cactus graphs, we need to do some more work. We reduce from Exact Cover by 3-Sets (X3C) [70]. In this problem, we are given $3q$ elements and m sets, each containing three distinct elements. The question is whether there are q sets such that all elements are covered.

Theorem 5.18. *CTP in the scenario model is NP-hard on cactus graphs.*

Proof. Starting from an instance of X3C, we construct the following graph. For this, we need the gadgets in Figure 5.3, denoted by gadget Z_1 and gadget Z_2. Assume without loss of generality that $3q+4$ is a power of 2. We take $m+3q+3$ times gadget Z_1 and concatenate these. This is done by identifying the rightmost vertex of a Z_1-gadget to the leftmost vertex of the next Z_1-gadget. The first m gadgets correspond to sets, the next $3q$ gadgets to elements, and the final three to dummies. Take the leftmost vertex of the first gadget as s. Then, take $\log_2\sigma$ times gadget Z_2, where $\sigma=3q+4$. Concatenate these by identifying the the rightmost vertex of a Z_2-gadget to the leftmost vertex of the next Z_2-gadget. Then, identify the rightmost vertex of the first chain with the leftmost vertex of the second chain. Finally, we take the rightmost vertex of the second chain as t. Note that the resulting graph is a cactus graph.

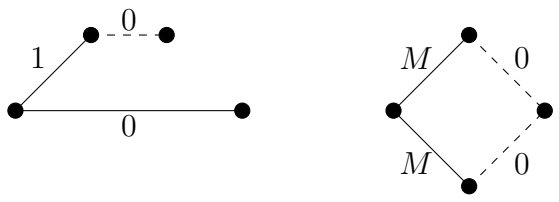


Fig. 5.3: Gadget Z_1 (left) and Z_2 (right), where dashed edges are uncertain.

The idea of the reduction is that we cannot afford to make a mistake in the Z_2-chain, that means, we need to know exactly which scenario is active before reaching the Z_2-chain. Learning which scenario is active is done by moving to the uncertain edges in the Z_1-chain. We say that we 'try out' the gadget. If there is an exact cover then learning can be done cheaper than when there is no exact cover.

We define σ scenarios, where each scenario corresponds to an element or to a dummy or neither. All scenarios have equal probability $\frac{1}{\sigma}$. A scenario corresponding to an element contains, next to all certain edges, the uncertain edge in the gadget corresponding to the element and in the gadgets corresponding to sets containing this element. A scenario corresponding to a dummy contains, next to all certain edges, the uncertain edge in the gadget corresponding to that dummy. There is one scenario that corresponds to neither of them. This scenario contains none of the uncertain edges in the first chain of gadgets. Moreover, we assign a unique path in the second chain to each scenario. This can be done as follows. For scenario j, the top edge of gadget z is active (and the bottom edge is not), if the zth digit in the binary representation of $j-1$ is a 1. Otherwise, the bottom edge is available.

Finally, we let B, the bound on the objective value of CTP, equal $M \log_2 \sigma + (3q^2 + 21q + 18)/\sigma$. By setting $M > 3q^2 + 21q + 18$, we know we will exceed B if we make guesses in the second chain, since each guess costs at least an extra M/σ. Therefore, one has to learn the scenario in the first chain of gadgets.

Assume there is an exact cover and let Y be a solution. Then, first we try out the gadgets corresponding to sets in Y. If an uncertain edge turns out to be active then we try out either one or two of the three corresponding element-gadgets until we know which scenario is active, and then move to the Z_2-chain. If no gadget from Y is active, we try out the three dummies until either the active one is found or none of them is active, and then move to the Z_2-chain. This policy has an expected length of

$$M \log_2 \sigma + \underbrace{\frac{1}{\sigma}(4+6+6)}_{\text{contribution first set}} + \ldots + \underbrace{\frac{1}{\sigma}(2q+2+2q+4+2q+4)}_{\text{contribution } q\text{th set}}$$
$$+ \underbrace{\frac{1}{\sigma}(2q+2+2q+4+2q+6)}_{\text{contribution dummies}} + \frac{1}{\sigma}(2q+6) = B.$$

Now, suppose that we have a yes-instance for CTP, i.e., we have a policy with an expected length of at most B. We need to show that this implies that we have a yes-instance for X3C.

As observed before, we need to learn the scenario on the first part. Moreover, once we see an active set, we will try out either one or two of the corresponding elements and take the active path. In general, the policy looks as follows. We will first try out k sets with three new elements. Then, we will try out ℓ sets with two new elements. Finally, we will try out the remaining elements and dummies. This policy has an expected value of

$$\begin{aligned} M \log_2 \sigma &+ \frac{1}{\sigma}(4+6+6) + \ldots + \frac{1}{\sigma}(2k+2+2k+4+2k+4) \\ &+ \frac{1}{\sigma}(2(k+1)+2+2(k+1)+2) + \ldots + \frac{1}{\sigma}(2(k+\ell)+2+2(k+\ell)+2) \\ &+ \frac{1}{\sigma}(2(k+\ell+1) + \ldots + 2(k+\ell+3q-3k-2\ell+3)) \\ &+ \frac{1}{\sigma}(2(k+\ell+3q-3k-2\ell+3)) \end{aligned}$$

$$
\begin{aligned}
&= M \log_2 \sigma + \frac{1}{\sigma}(9q^2 + 6k^2 + 2\ell^2 - 12qk - 6q\ell + 6k\ell + 27q - 6k - 4\ell + 18) \\
&= B + \frac{1}{\sigma}(6q^2 + 6k^2 - 12qk + 2\ell^2 + 6q - 6k - 6q\ell + 6k\ell - 4\ell) \\
&= B + \frac{1}{\sigma}(6(q-k)^2 + (6 - 6\ell)(q-k) + 2\ell^2 - 4\ell) \\
&= B + \frac{1}{\sigma}(2(\sqrt{3}(q-k) - \ell)^2 + 6(q-k) + (4\sqrt{3} - 6)\ell(q-k) - 4\ell) \\
&\geq B + \frac{1}{\sigma}(2(\sqrt{3}(q-k) - \ell)^2 + (4\sqrt{3} - 6)\ell(q-k)),
\end{aligned}
$$

where the last inequality follows from $\ell \leq \frac{3}{2}(q-k)$, since this is the maximum number of sets with two new elements you can choose after picking k disjoint 3-sets. Now, since $q \geq k$ and $\ell \geq 0$, this policy has expected value greater than or equal to B. Moreover, the expected value of the solution is equal to B if and only if $q = k$ and $\ell = 0$. Hence, we have a yes-instance for X3C. □

5.4 Conclusion

In this chapter, we considered the Canadian traveler problem and the multi-target graph search problem. For the CTP in the independent decision model, we showed NP-hardness on series-parallel graphs. This followed immediately after showing that multi-target GSP in the independent decision model is NP-hard on trees. For the multi-target GSP in the independent decision model we gave a $(3.59 + \epsilon)$-approximation for trees and a $(14.4 + \epsilon)$-approximation on general metrics. We also showed that one should be adaptive in order to be optimal for CTP.

For the scenario model, we gave an NP-hardness proof for two-target GSP on stars. As a consequence, CTP in the scenario model is NP-hard on disjoint-path graphs. Finally, we showed that this problem is also NP-hard on cactus graphs.

The main open problem in this field remains the approximability of CTP in the independent decision model. As a start, one could try to prove an upper bound on the adaptivity gap. We conjecture that this is bounded by a small constant on trees. In general, an approach could be to investigate good adaptive algorithms.

CHAPTER 6

The lost cow problem

An article based on this chapter appeared in [36].

6.1 Introduction

In this chapter, we study the *average cow problem* [8], also known as the *linear search problem* [18]. The problem can be seen from the cow's perspective as follows. Suppose she is standing before an infinitely long fence with one gate. She wants to get to the other side through the gate. However, she only has a probability distribution of the location of the gate. Moreover, it is foggy and she needs to walk along the fence (with unit speed) in order to find the gate. Her goal is to minimize the expected distance traveled until finding the gate. An alternative way to look at the problem follows the farmer's perspective. The farmer has lost one of his cows and he needs to find it somewhere along an infinitely long road.

Before discussing the results, we will give an overview of the literature which is divided into two fields. The first stream of literature uses the name average cow problem or cow-path problem. The second stream uses the name linear search problem.

6.1.1 Literature

Average cow problem The *cow-path problem* was introduced by [8]. The problem is interesting to study since it is a natural model for searching in an unknown environment. Here, a seeker has to find a target. In the cow-path problem, one does not know the distribution, but an adversary picks a distribution that maximizes the competitive ratio, i.e., the ratio between the length of the walk and the distance to the target. Note that the adversary will always pick one location with probability 1. This is a classical problem in the area of competitive analysis. It was shown by Baeza-Yates et al. [8] that the geometric algorithm achieves a competitive ratio of 9, i.e., if the target is located at distance d from our starting point r, we will find the target within $9d$ time, assuming the seeker moves at unit speed. The algorithm achieving this ratio directs the cow to the right for one unit of distance. It then returns to the starting point and walks two units to the left. By repeatedly doubling the distance, we get the desired competitive ratio. They also showed that we cannot achieve a lower competitive ratio. It was later shown by Kao et al. [69] that a randomized algorithm can achieve a competitive ratio of 4.6.

Baeza-Yates et al. [8] also studied the stochastic version of this problem, which they called the average cow problem. They claimed that it is optimal not to turn under the uniform distribution. We say that one does not turn if one does not make any additional turns other than at the border of the domain. We may also refer to this strategy as exhaustive search. The authors of [8] also gave a sufficient condition for which distributions it is optimal to turn infinitely many times. This result implied that for the triangular distribution (Figure 6.1) it is optimal to turn infinitely many times.

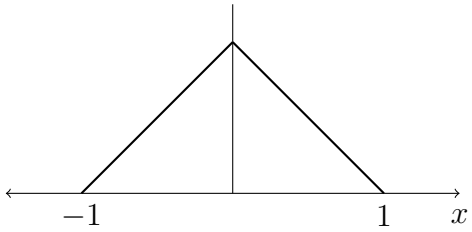


Fig. 6.1: Instance with triangular probability density function, i.e., $F'(x) = 1 - |x|$ for $x \in [-1, 1]$.

Linear search problem The problem we want to study is also known as the linear search problem. It was introduced in 1963 by Bellman [18]. The first papers [11, 12, 46] investigated the existence of optimal solutions and whether the minimum value can be attained. Beck [12] concluded that when a solution is allowed to start with an infinitesmall oscillation around the origin, an optimal solution exists for every distribution with a finite first moment.

Later, Beck and Newman [16] studied the problem in which the searcher does not know the distribution, but an adversary picks one that maximizes the competitive ratio. They showed that the geometric algorithm is 9-competitive and the randomized geometric algorithm is 4.6-competitive. Note that this is exactly the cow-path problem and that these results were reproduced by [8] and [69] 20 years later.

In the 1980's, the behavior of several distribution functions were studied. It was shown that it is optimal not to turn for the uniform distribution [13]. Baston and Beck [10] gave a characterization of the distributions for which it is optimal to turn infinitely many times. Of course, the normal distribution has also been investigated [14]. Kella [72] gave a sufficient condition for distributions for which it is optimal not to turn.

Since the problem is still unsolved in general, one could consider approximation algorithms. It was shown in the book by Alpern and Gal [2] that for any $\epsilon > 0$, one can find a solution with an expected value of at most a factor $1+\epsilon$ times the value of the optimal solution. The dynamic programming algorithm they designed is very similar to the algorithm of Afrati et al. [1] for line-TRP. This can be explained by observing that the linear search problem with a bounded discrete distribution is equivalent to *line-TRP*. In line-TRP, we are given an vertex- and edge-weighted line graph with root r, where vertex i has weight w_i, and edge (i, j) has weight $c(i, j)$. Now, line-TRP is TRP, defined in Chapter 1, where the input graph is a line graph. This problem can be solved using dynamic programming in time $O(n^2)$ [1], where n is the number of vertices. In [50], it was shown that one can improve the running time to $O(n)$.

The dynamic programming algorithm uses the observation that a vertex is visited whenever the walk passes the vertex for the first time (it never skips a vertex). Hence, the set of already visited vertices in a partial solution is defined by the left-most and right-most vertex visited. Let the vertices on the right of the origin r be denoted as $x_1, \ldots, x_R$ and the vertices on the left as $y_1, \ldots, y_L$. We assume w.l.o.g. that the numbering satisfies $0 < c(r, x_1) < \ldots < c(r, x_R)$ and $0 < c(r, y_1) < \ldots < c(r, y_L)$. Define $f(x_i, y_j)$ as the minimum cost caused by serving vertices $\{x_1, \ldots, x_i\} \cup \{y_1, \ldots, y_j\}$, ending at vertex x_i. Cost $f(y_j, x_i)$ is defined similarly. The optimal solution can now be found by using the recursive relation

$$f(x_i, y_j) = \min\{f(x_{i-1}, y_j) + \left(\sum_{q=i}^{R} w_q + \sum_{q=j+1}^{L} w_q\right) c(x_{i-1}, x_i),$$
$$f(y_j, x_{i-1}) + \left(\sum_{q=i}^{R} w_q + \sum_{q=j+1}^{L} w_q\right) c(y_j, x_i)\},$$

where initially $f(r, r) = 0$. The running time of this algorithm is indeed $O(n^2)$ [1].

When the distribution of the linear search problem is discrete and has a bounded domain, we can describe this input as n points, where point i has probability p_i. Now, the tour that minimizes the expected length of the walk is the same tour that minimizes the objective of line-TRP which has vertices at locations with positive probability and with vertex weights equal to the corresponding probabilities. Hence, this special case can be solved in polynomial time. To get a $(1 + \epsilon)$-approximation for the linear search problem for non-discrete distributions, one has to figure out how to round a continuous distribution to a discrete one and how to deal with an unbounded domain. This was done by Alpern and Gal [2].

Finally, the problem has been generalized to different objective functions. For example, one could use a convex function of the distance traveled [15, 17] or charge the searcher for making turns [32].

6.1.2 Outline

In this chapter, we will try to characterize for which distributions it is optimal not to turn. We managed to do this for symmetric distributions (Section 6.2). In Section 6.3, we also give a similar characterization for star search, i.e., there are k directions instead of two. In both sections, we will link our results with Kella's Theorem [72]. Finally, we give a worst-case performance bound for not turning and show results on the performance of not turning for several distributions.

6.2 Exhaustive search

We will now introduce our notation. Let (D, Z) be the random variable of the location of the target, which takes values $(d, z) \in \mathbb{R}^+ \times \{1, 2\}$. Here, $z = 1$ denotes the event of the target being on the left of the origin. On the other hand, when $z = 2$ the target is located at the right of the origin. Let $F_1(x) = \mathbb{P}(0 < D \leq x, Z = 1)$ and $F_2(x) = \mathbb{P}(0 < D \leq x, Z = 2)$ for $x \in \mathbb{R}^+$. Further, define $p_\ell = \mathbb{P}(Z = \ell)$ for $\ell = 1, 2$.

Note that $p_1 + p_2 = 1$. Let $b_\ell = \sup\{x : F_\ell(x) < p_\ell\}$ for $\ell = 1, 2$. Finally, we denote $\mathbb{E}[D]$ for the expected distance between the origin and the target.

An interesting question is whether we can characterize the distributions for which it is optimal not to turn. A sufficient condition was given by Kella [72]. Since we are going to investigate this question, we first give Kella's theorem. Define $K_\ell(x) = xp_\ell \left(\frac{1}{F_\ell(x)} - 1\right)$ on $x \in (0, b_\ell)$ for side ℓ. The theorem also holds for searching on stars, i.e., we have k directions instead of two directions.

Theorem 6.1 ([72])**.** *If $K_\ell(x)$ is non-increasing on $(0, b_\ell)$ for all ℓ, then it is optimal not to turn.*

Note that for the uniform distribution on $[-1, 1]$, we have $K_\ell(x) = 1 - \frac{1}{2}x$ for both ℓ. Hence, the result of Kella [72] has the result of Beck and Beck [13] as a special case.

We restrict ourselves to symmetric distributions with bounded domain. Hence, we have $F_1(x) = F_2(x) = F(x)$ for $x \in [0, 1]$. Moreover, we have $p_1 = p_2 = \frac{1}{2}$ and $b_1 = b_2 = 1$. We denote $cow(F)$ for the optimal solution for F.

Let us first find an expression for the optimal value. Suppose that $cow(F)$ turns m times, and its turning points are denoted by $0 < t_1 < \ldots < t_m < 1$. That $t_i < t_{i+1}$ for all i is intuitively clear, but technical to proof [13]. Without loss of generality, we assume the seeker turns at $t_1, t_3, \ldots$ on side 1 and at $t_2, t_4, \ldots$ on side 2. Equation (6.1) explicitly considers the distance traveled until reaching a point. Define, $t_{-1} = t_0 = 0$ and $t_{m+1} = t_{m+2} = 1$. Now, if t_i is one the left side and $(x, 2)$ lies in $(t_{i-1}, t_{i+1}]$ on the right side, it will be visited at time $2\sum_{j=1}^{i} t_j + x$. Integrating over all $x \in (t_{i-1}, t_{i+1}]$ (with respect to F) and summing over all intervals, we get that the expected length of the walk $cow(F)$ is equal to

$$L(t_1, \ldots, t_m) = \sum_{i=0}^{m+1} \int_{t_{i-1}}^{t_{i+1}} \left(2\sum_{j=1}^{i} t_j + x\right) dF(x). \tag{6.1}$$

An alternative way of computing the optimal value charges each turn t_i the delay it causes. We start with $\mathbb{E}[D]$. If we turn at t_i, the points in $(t_i, 1]$ will be delayed an extra $2t_i + 2t_{i+1}$. Hence, we can compute the value of $cow(F)$ as follows.

$$L(t_1, \ldots, t_m) = \mathbb{E}[D] + \sum_{i=0}^{m} (2t_i + 2t_{i+1}) \left(\frac{1}{2} - F(t_i)\right). \tag{6.2}$$

For example, if we turn once we get an objective value of

$$\begin{aligned} &\mathbb{E}[D] + (2 \cdot 0 + 2t_1) \left(\frac{1}{2} - F(0)\right) + (2t_1 + 2 \cdot 1) \left(\frac{1}{2} - F(t_1)\right) \\ &= \mathbb{E}[D] + t_1 + (2t_1 + 2) \left(\frac{1}{2} - F(t_1)\right). \end{aligned}$$

Here, the second term says that all locations on the left side are delayed by $2t_1$. The third term says that the locations to the right of t_1 are delayed with $2t_1 + 2$.

We can now characterize the symmetric distributions for which it is optimal not to turn. For this, we need the following three lemmas.

Lemma 6.2. *It is not optimal not to turn if $F(y) > y/(y + 1)$ for some $y \in (0, 1)$.*

Proof. The solution that does not turn has value $\mathbb{E}[D] + 1$. If we turn at y, we get a walk of expected length $\mathbb{E}[D] + y + (2y+2)(\frac{1}{2} - F(y))$. If we assume that it is optimal not to turn, we have that for all $y \in (0,1)$,

$$\begin{aligned} y + (2y+2)\left(\frac{1}{2} - F(y)\right) &\geq 1 \\ 2y &\geq (2y+2)F(y) \\ F(y) &\leq \frac{y}{y+1}. \end{aligned}$$

Hence, if there is a $y \in (0,1)$ such that $F(y) > y/(y+1)$, then it is not optimal not to turn. □

Note that $F(x) = x/(x+1)$ is a twice differentiable function. The derivative is equal to $F'(x) = 1/(x+1)^2$ and its second derivative is equal to $F''(x) = -2/(x+1)^3$. We will use this in the proof below.

Lemma 6.3. *If $F(x) = x/(x+1)$, then cow(F) does not turn.*

Proof. If we take the derivative of $L(t_1, \ldots, t_m)$ with respect to variable t_i, we get the following expression.

$$\begin{aligned} \frac{\partial}{\partial t_i} L(t_1, \ldots, t_m) &= \frac{\partial}{\partial t_i}\left((2t_{i-1} + 2t_i)\left(\frac{1}{2} - F(t_{i-1})\right) + (2t_i + 2t_{i+1})\left(\frac{1}{2} - F(t_i)\right)\right) \\ &= 2\left(\frac{1}{2} - F(t_{i-1})\right) + 1 - 2F(t_i) - 2(t_{i+1} + t_i)F'(t_i) \\ &= 2 - 2F(t_{i-1}) - 2F(t_i) - 2(t_i + t_{i+1})F'(t_i). \end{aligned}$$

Now, note that

$$\begin{aligned} \frac{\partial^2}{\partial t_m^2} L(t_1, \ldots, t_m) &= -2F'(t_m) - 2((t_m+1)F''(t_m) + f(t_m)) \\ &= -4F'(t_m) - 2(t_m+1)F''(t_m). \end{aligned}$$

Using that $F'(x) = 1/(x+1)^2$ and $F''(x) = -2/(x+1)^3$, we get

$$\begin{aligned} \frac{\partial^2}{\partial t_m^2} L(t_1, \ldots, t_m) &= \frac{-4}{(t_m+1)^2} + \frac{4(t_m+1)}{(t_m+1)^3} \\ &= \frac{-4}{(t_m+1)^2} + \frac{4}{(t_m+1)^2} = 0. \end{aligned}$$

Hence, L is linear in the direction of t_m. This means that we can set t_m equal to 1 or t_{m-2} without increasing the expected length of the walk. Therefore, it is not worse to turn one time less. So, the optimal value is non-decreasing in the number of turning points and we have obtained that it is optimal not to turn. □

Lemma 6.4. *If it is optimal for F not to turn and $\tilde{F}(x) \leq F(x)$ for all $x \in (0,1)$, then it is optimal not to turn for $\tilde{F}$.*

Proof. Let $\mathbb{E}[D]$ and $\mathbb{E}[\tilde{D}]$ denote the expected distance between the origin and the target corresponding to F and $\tilde{F}$ respectively. For the sake of contradiction, suppose that it is not optimal not to turn for $\tilde{F}$. This means that there is an $m \in \mathbb{N} \cup \{\infty\}$ such that

$$\mathbb{E}[\tilde{D}] + \sum_{i=0}^{m} (2t_i + 2t_{i+1}) \left(\frac{1}{2} - \tilde{F}(t_i) \right) < \mathbb{E}[\tilde{D}] + 1$$
$$\sum_{i=0}^{m} (2t_i + 2t_{i+1}) \left(\frac{1}{2} - \tilde{F}(t_i) \right) < 1.$$

Because $\tilde{F}(x) \leq F(x)$ for all $x \in (0, 1)$, we have that

$$\sum_{i=0}^{m} (2t_i + 2t_{i+1}) \left(\frac{1}{2} - F(t_i) \right) < 1$$
$$\mathbb{E}[D] + \sum_{i=0}^{m} (2t_i + 2t_{i+1}) \left(\frac{1}{2} - F(t_i) \right) < \mathbb{E}[D] + 1,$$

which would mean that it is not optimal for F not to turn. Hence, we have a contradiction and we have shown that it is optimal for $\tilde{F}$ not to turn. □

The next theorem follows from the three previous lemmas.

Theorem 6.5. *It is optimal not to turn for F if and only if $F(x) \leq H(x)$ for all $x \in (0, 1)$, where $H(x) = x/(x + 1)$.*

Proof. If $F(x) \leq H(x)$ for all $x \in (0, 1)$, we know, by Lemma 6.4, that it is optimal not to turn for F if it is optimal not to turn for H. Since $H(x) = x/(x + 1)$, Lemma 6.3 tells us that this is the case. If $F(y) > H(y)$ for some $y \in (0, 1)$, we know that it is not optimal not to turn for F by Lemma 6.2. □

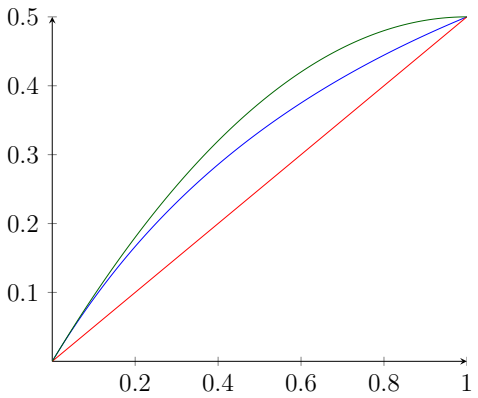


Fig. 6.2: Functions $H(x) = x/(x + 1)$ (blue), uniform distribution $F(x) = x/2$ (red) and triangular distribution $G(x) = x - x^2/2$ (green).

As can be seen from Figure 6.2, this theorem can be used to demonstrate that it is indeed optimal not to turn for the uniform distribution. For the triangular distribution, it is not optimal not to turn. In fact, the function corresponding to this distribution is above $H(x)$ for all $x \in (0, 1)$. This means that every solution with one turning point outperforms not turning at all.

6.2.1 Relation with Kella's Theorem

For the general (non-symmetric) problem Kella [72] showed that if the function $K_\ell(x) = xp_\ell\left(\frac{1}{F_\ell(x)} - 1\right)$ is non-increasing for each side ℓ, then it is optimal to do an exhaustive search. For the symmetric case, we can restate Theorem 6.5 in terms of $K(x)$. Namely,

$$\begin{aligned} F(x) &\leq \frac{x}{x+1} \\ \frac{1}{F(x)} &\geq \frac{x+1}{x} \\ \frac{1}{F(x)} - 1 &\geq \frac{1}{x} \\ x\left(\frac{1}{F(x)} - 1\right) &\geq 1 \\ K(x) &\geq \frac{1}{2}. \end{aligned}$$

Hence, exhaustive search is optimal if and only if $K(x) \geq \frac{1}{2}$. Now, since $K(1) = \frac{1}{2}$, it follows naturally that demanding that $K(x)$ is non-increasing is a sufficient condition for optimality. To show that this is not a necessary condition, consider the following example. Let $f(x)$ be $1/2$ on $(0, \frac{1}{2}]$, 0 on $(\frac{1}{2}, \frac{3}{4}]$ and 1 on $(\frac{3}{4}, 1]$, where f is the derivative of F. As can be seen in Figure 6.3, $K(x)$ is at least $\frac{1}{2}$, but $K(x)$ is increasing on $(\frac{1}{2}, \frac{3}{4}]$.

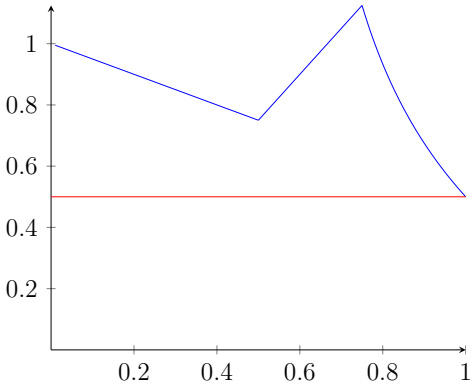


Fig. 6.3: $K(x)$ of described $F(x)$.

6.3 Exhaustive search on stars

We can generalize the results of Section 6.2 to search on stars. In this setting, we have k directions to search in. Our notation is as follows. Let (D, Z) be the random variable of the location of the target, which takes values $(d, z) \in \mathbb{R}^+ \times \{1, \ldots, k\}$. For any $\ell \in \{1, \ldots, k\}$, let $F_\ell(x) = \mathbb{P}(0 < D \leq x, Z = \ell,)$ for $x \in \mathbb{R}^+$. Further, define $p_\ell = \mathbb{P}(Z = \ell)$ for $\ell = 1, \ldots, k$. Note that $\sum_{\ell=1}^{k} p_\ell = 1$. Let $b_\ell = \sup\{x : F_\ell(x) < p_\ell\}$ for $\ell = 1, \ldots, k$. Finally, we denote $\mathbb{E}[D]$ for the expected distance between the origin and the target.

Again, we restrict ourselves to symmetric distributions with bounded domain. Hence, for all ℓ we have $F_\ell(x) = F(x)$ for $x \in (0,1)$. Moreover, we have $p_\ell = \frac{1}{k}$ and $b_\ell = 1$. We denote $cow(F)$ for the optimal solution for F.

Let us first find an expression for the optimal value. Suppose that $cow(F)$ turns m times, and its turning points are denoted by $\{t_i\}$, with $0 < t_i \leq t_{i+1} < 1$ for all i. That $t_i \leq t_{i+1}$ for all i is intuitively clear, but technical to prove. Hence, the proof is omitted. A consequence is that the cow turns at t_i on side ℓ if $i \bmod \ell = 0$. The optimal value can be obtained by charging each turn t_i the delay it causes. If we turn at t_i, the points in $(t_i, 1]$ in the same direction will be delayed an extra $2t_i + \ldots + 2t_{i+k-1}$. Hence, we can compute the value of $cow(F)$ as follows. For completeness, we set $t_0 = 0$ and $t_{m+j} = 1$ for $j = 1, \ldots, k-1$. The following equation generalizes Equation (6.2).

$$L(t_1, \ldots, t_m) = \mathbb{E}[D] + \sum_{i=0}^{m} \left(\sum_{j=i}^{i+k-1} 2t_j \right) \left(\frac{1}{k} - F(t_i) \right).$$

We can now characterize the symmetric distributions for which it is optimal not to turn. For this, we need the following three lemmas.

Lemma 6.6. *It is not optimal not to turn if $F(y) > y/(y+k-1)$ for some $y \in (0,1)$.*

Proof. The solution that does not turn has value $\mathbb{E}[D] + k - 1$. If we turn at y, we get a walk of expected length

$$\begin{aligned}
&\mathbb{E}[D] + \frac{1}{k}(2y + (2y+2) + \ldots + (2y + 2(k-2))) + \left(\frac{1}{k} - F(y) \right)(2y + 2(k-1)) \\
=&\mathbb{E}[D] + \frac{2y(k-1)}{k} + \frac{(k-1)(k-2)}{k} + \left(\frac{1}{k} - F(y) \right)(2y + 2(k-1)).
\end{aligned}$$

If we assume that it is optimal not to turn, we have that for all $y \in (0,1)$,

$$\begin{aligned}
\frac{2y(k-1)}{k} + \frac{(k-1)(k-2)}{k} + \left(\frac{1}{k} - F(y) \right)(2y + 2(k-1)) &\geq k - 1 \\
\left(\frac{1}{k} - F(y) \right)(2y + 2(k-1)) &\geq \frac{2(1-y)(k-1)}{k} \\
F(y) &\leq \frac{y}{y+k-1}.
\end{aligned}$$

Hence, if there is a $y \in (0,1)$ such that $F(y) > y/(y+k-1)$, then it is not optimal not to turn. □

Again we use that $F(x) = x/(x+k-1)$ is twice differentiable, which enables us to use calculus arguments. Denote F' and F'' for the first and second derivative of F respectively.

Lemma 6.7. *If $F(x) = x/(x+k-1)$, then $cow(F)$ does not turn.*

Proof. If we take the second derivative of $L(t_1, \ldots, t_m)$ with respect to variable t_m, we get the following expression.

$$\begin{aligned}\frac{\partial^2}{\partial t_m^2} L(t_1, \ldots, t_m) &= \frac{\partial^2}{\partial t_m^2}\left(2(t_m + k - 1)\left(\frac{1}{k} - F(t_m)\right)\right) \\ &= \frac{\partial}{\partial t_m}\left(2\left(\frac{1}{k} - F(t_m)\right) - 2(t_m + k - 1)F'(t_m)\right) \\ &= -2F'(t_m) - 2((t_m + k - 1)F''(t_m) + f(t_m)) \\ &= -4F'(t_m) - 2(t_m + k - 1)F''(t_m).\end{aligned}$$

Using that $F'(x) = 1/(x + k - 1)^2$ and $F''(x) = -2/(x + k - 1)^3$, we get

$$\begin{aligned}\frac{\partial^2}{\partial t_m^2} L(t_1, \ldots, t_m) &= \frac{-4}{(t_m + k - 1)^2} + \frac{4(t_m + k - 1)}{(t_m + k - 1)^3} \\ &= \frac{-4}{(t_m + k - 1)^2} + \frac{4}{(t_m + k - 1)^2} = 0.\end{aligned}$$

Hence, L is linear in the direction of t_m. This means that we can set t_m equal to 1 or t_{m-2} without increasing the expected length of the walk. Therefore, it is not worse to turn one time less. So, the optimal value is non-decreasing in the number of turning points and we have obtained that it is optimal not to turn. □

Lemma 6.8. *If it is optimal for F not to turn and $\tilde{F}(x) \leq F(x)$ for all $x \in (0, 1)$, then it is optimal not to turn for $\tilde{F}$.*

Proof. Let $\mathbb{E}[D]$ and $\mathbb{E}[\tilde{D}]$ denote for the expected distance between the origin and the target corresponding to F and $\tilde{F}$ respectively. For the sake of contradiction, suppose that it is not optimal not to turn for $\tilde{F}$. This means that there is an $m \in \mathbb{N} \cup \{\infty\}$ such that

$$\begin{aligned}\mathbb{E}[\tilde{D}] + \sum_{i=0}^{m}\left(\sum_{j=i}^{i+k-1} 2t_j\right)\left(\frac{1}{k} - \tilde{F}(t_i)\right) &< \mathbb{E}[\tilde{D}] + k - 1 \\ \sum_{i=0}^{m}\left(\sum_{j=i}^{i+k-1} 2t_j\right)\left(\frac{1}{k} - \tilde{F}(t_i)\right) &< k - 1.\end{aligned}$$

Because $\tilde{F}(x) \leq F(x)$ for all $x \in (0, 1)$, we have that

$$\begin{aligned}\sum_{i=0}^{m}\left(\sum_{j=i}^{i+k-1} 2t_j\right)\left(\frac{1}{k} - F(t_i)\right) &< k - 1 \\ \mathbb{E}[D] + \sum_{i=0}^{m}\left(\sum_{j=i}^{i+k-1} 2t_j\right)\left(\frac{1}{k} - F(t_i)\right) &< \mathbb{E}[D] + k - 1,\end{aligned}$$

which would mean that it is not optimal for F not to turn. Hence, we have a contradiction and we have shown that it is optimal for $\tilde{F}$ not to turn. □

The next theorem follows from the previous three lemmas.

Theorem 6.9. *It is optimal not to turn for F if and only if $F(x) \leq H(x)$ for all $x \in (0,1)$, where $H(x) = x/(x+k-1)$.*

Proof. If $F(x) \leq H(x)$ for all $x \in (0,1)$, we know, by Lemma 6.8, that it is optimal not to turn for F if it is optimal not to turn for H. Since $H(x) = x/(x+k-1)$, Lemma 6.7 tells us that this is the case. If $F(y) > H(y)$ for some $y \in (0,1)$, we know that it is not optimal not to turn for F by Lemma 6.6. □

6.3.1 Relation with Kella's Theorem

Recall that Kella [72] showed that if the function $K_\ell(x) = xp_\ell\left(\frac{1}{F_\ell(x)} - 1\right)$ is non-increasing for each direction ℓ, then it is optimal to do an exhaustive search. For the symmetric case, we can restate Theorem 6.5 in terms of $K(x)$. Namely,

$$\begin{aligned} F(x) &\leq \frac{x}{x+k-1} \\ \frac{1}{F(x)} &\geq \frac{x+k-1}{x} \\ \frac{1}{F(x)} - 1 &\geq \frac{k-1}{x} \\ x\left(\frac{1}{F(x)} - 1\right) &\geq k-1 \\ K(x) &\geq \frac{k-1}{k}. \end{aligned}$$

Again observe that $K(1) = (k-1)/k$. Hence, demanding that $K(x)$ is non-increasing on $(0,1)$ is a sufficient condition for the optimality of not turning. For general distributions, the observation above could lead to the conjecture that it is optimal not to turn if and only if $K_\ell(x)$ is greater than or equal to $K_\ell(b_\ell)$, i.e., $K_\ell(x) \geq b_\ell(1-p_\ell)$.

Conjecture 6.10. *It is optimal not to turn if and only if $K(x) \geq K(b_\ell)$ for all $x \in (0, b_\ell)$ and for all ℓ.*

However, we have not been able to prove this for the general case. In the next section, we will prove the conjecture for a more general case.

6.4 General distributions

For a restricted class of distributions, that contains the symmetric distributions, we will prove that our conjecture is true. In this section, we assume that $b_i/p_i = b_j/p_j$ for all i, j. Now, we will prove that it is optimal not to turn if and only if $F_\ell(x) \leq \frac{x}{x+b_\ell(1-p_\ell)/p_\ell}$ for all ℓ. This is equivalent to $K_\ell(x) \geq b_\ell(1-p_\ell)$ for all ℓ. Because the proof is similar to the symmetric case, we only give a sketch of the proof.

Theorem 6.11. *It is optimal not to turn for $\{F_\ell\}_{\ell=1}^k$ with $b_i/p_i = b_j/p_j$ for all $i \neq j$ if and only if $F_\ell(x) \leq H_\ell(x)$ for all $x \in (0, b_\ell)$ and all ℓ, where $H(x) = x/(x + b_\ell(1-p_\ell)/p_\ell)$.*

Proof sketch. Again, we start by stating that it should be better not to turn than to turn once. Not turning has an expected value of

$$\mathbb{E}[D] + 2b_1(1-p_1) + 2b_2(1-p_1-p_2) + \ldots + 2b_{k-1}(1-p_1-\ldots-p_{k-1}),$$

whereas turning at y on direction 1 (similar proof works for other directions) results in an expected value of

$$\begin{aligned}\mathbb{E}[D] &+ 2y(1-F_1(y)) + 2b_2(1-F_1(y)-p_2) + \ldots + 2b_{k-1}(1-F_1(y)-p_2-\ldots-p_{k-1}) \\ &+ 2b_k(p_1-F_1(y)).\end{aligned}$$

Now, after rearranging terms, it is better not to turn than to turn once if

$$\begin{aligned} 2b_1(1-p_1) - 2p_1(b_2+\ldots+b_k) &\leq 2y(1-F_1(y)) - 2F_1(y)(b_2+\ldots+b_k) \\ 0 &\leq 2y - 2yF_1(y) - 2F_1(y)b_1(1-p_1)/p_1 \\ F_1(y)(y + b_1(1-p_1)/p_1) &\leq y \\ F_1(y) &\leq \frac{y}{y+b_1(1-p_1)/p_1}. \end{aligned}$$

We now have to show that when $F_\ell(x) = x/(x+b_\ell(1-p_\ell)/p_\ell)$ for all ℓ, it is optimal not to turn, like in Lemma 6.7. Now, it is not that easy anymore to write down a simple closed-formula for the optimal value. However, if we fix some of the configurations of the optimal solution and try all of them it can work out fine. Assume that the last turning point, t_m, is on direction 1. Further, assume that after turning at t_m we fully search all directions in $S \subseteq \{2,\ldots,k\}$ before finishing direction 1. Since we are going to take the second derivative of the optimal value with respect to t_m, we are only interested in terms that are non-linear in t_m. The only term we have to consider is $2(t_m + \sum_{\ell\in S} b_\ell)(p_1 - F_1(t_m))$. Now,

$$\begin{aligned} \frac{\partial^2}{\partial t_m^2} L(t_1,\ldots,t_m) &= \frac{\partial^2}{\partial t_m^2}\left(2(t_m + \sum_{\ell\in S} b_\ell)(p_1 - F_1(t_m))\right) \\ &= -2(t_m + \sum_{\ell\in S} b_\ell)F''(t_m) - 4F'(t_m) \\ &= \frac{4(t_m + \sum_{\ell\in S} b_\ell)b_1(1-p_1)/p_1}{(t_m + b_1(1-p_1)/p_1)^3} - \frac{4b_1(1-p_1)/p_1}{(t_m + b_1(1-p_1)/p_1)^2} \\ &\leq \frac{4(t_m + b_1(1-p_1)/p_1)b_1(1-p_1)/p_1}{(t_m + b_1(1-p_1)/p_1)^3} - \frac{4b_1(1-p_1)/p_1}{(t_m + b_1(1-p_1)/p_1)^2} = 0. \end{aligned}$$

Here, the last inequality follows from $\sum_{\ell\in S} b_\ell \leq \sum_{\ell=2}^{k} b_\ell = b_1(1-p_1)/p_1$, since we assumed that $b_\ell/p_\ell = b_1/p_1$. Hence, whatever configuration has been chosen, the objective value is concave in t_m. Thus, by similar arguments as before (Lemma 6.7), it is optimal not to turn.

Finally, we have to show that a distribution that is dominated in each direction by a distribution for which it is optimal not to turn, also has an optimal solution that does not turn. Again, by fixing a configuration, we see that we can write down the optimal value as a sum of terms, where each term depends on $(p_i - F_i(t_j))$ for some i and j. Now, it is easy to see that the proofs of Lemma 6.4 and 6.8 extend to our cases. This completes the proof. □

6.5 Approximation

In this section, we investigate the worst-case performance of not turning. We start by giving a lower bound on the optimal value OPT for symmetric distributions that relates it to $\mathbb{E}[D]$, the expected distance between origin and target. The approximation ratio then follows easily. We restrict ourselves to the case of only two directions and probability distributions with domain $[0, 1]$ on each side.

Lemma 6.12. *For symmetric distributions,* $\text{OPT} \geq 2\mathbb{E}[D]$.

Proof. For each x, there is a symmetric point at the same distance in the other direction, say x'. The best way to visit just these two points, i.e., such that the average arrival time is minimized, is to walk to x and then to x' (or vice versa). In this way, you visit x at time x and x' at time $3x$, so on average at time $2x$. Hence, we can restrict ourselves to one direction, where each probability is doubled and each arrival time is also doubled. Since this is true for all $x > 0$, we see that OPT is at least

$$\int_0^\infty 4x dF(x) = 2\mathbb{E}[D].$$

□

Theorem 6.13. *Not turning is a* $\frac{1}{2}\left(1 + \frac{1}{\mathbb{E}[D]}\right)$*-approximation.*

Proof. The algorithm of not turning produces a walk of expected length $\mathbb{E}[D]+1$. Since $\text{OPT} \geq 2\mathbb{E}[D]$, we get a ratio of $\frac{1}{2}\left(1 + \frac{1}{\mathbb{E}[D]}\right)$. □

Note that for symmetric density functions that are concave on $(0, 1)$, we have $\mathbb{E}[D] \geq \frac{1}{3}$. This is true since the extreme case, the triangular distribution, has $\mathbb{E}[D] = \frac{1}{3}$. Hence, not turning is a 2-approximation. Finally, we show in Table 6.5 how well this approach performs compared to the optimal solution. Here, the first four distributions are of the form $1 - x^\alpha$. Since $\mathbb{E}[D]$ approaches $\frac{1}{4}$ for α small, not turning is a 2.5-approximation for these distributions. The last three distributions are of the form $e^{-\lambda x}$. If $\lambda \to \infty$, then $\mathbb{E}[D] \to 0$ and the approximation ratio gets arbitrarily large. The first column presents an upper bound on $\mathbb{E}[D] + 1$. In the second column of the table, a lower bound on the optimal value is given. This is obtained by discretizing the probability distribution and solve the problem with dynamic programming.

	UB	LB	Ratio
$1 - x^2$	1.375	1.270	1.083
$1 - x$	1.334	1.179	1.132
$1 - \sqrt{x}$	1.3	1.083	1.201
$1 - x^{1/100}$	1.252	0.942	1.330
e^{-10x}	1.100	0.383	2.873
e^{-50x}	1.020	0.069	14.783

Tab. 6.1: Approximation ratio's for not turning.

6.6 Conclusion

We gave a characterization of symmetric distributions for which it is optimal not to turn. This was also generalized to the case of multiple directions. It is still open whether we can also characterize asymmetric distributions with this property. If we look at the relation with Kella's theorem, one can see that for symmetric (and slightly more general) distributions it is optimal not to turn if and only if $K(x) \geq K(b_\ell)$ for all $x \in (0, b_\ell)$ and for all ℓ. We conjecture that for general distributions it is optimal not to turn if and only if $K(x) \geq K(b_\ell) = b_\ell(1 - p_\ell)$ for all $x \in (0, b_\ell)$, but we were not able to prove this.

CHAPTER 7

Graphs of bounded starwidth

The results in this chapter were published in [37].

7.1 Introduction

The traveling repairman problem is a very interesting problem in computational complexity. It is NP-hard on weighted trees [96] and polynomially solvable on line graphs [1] and star graphs. The latter result is folklore, and even holds when vertices have weights and our goal is to minimize the weighted sum of arrival times. An algorithm that solves the TRP on star graphs is the following. Sort the leafs in non-increasing order of the ratio between the weight of the vertex and the length of the edge. Then, visit the leafs in this order. It is unclear how the TRP can be solved on slightly more general graphs, like spiders of depth 2.

Intrigued by the complexity of the traveling repairman problem, we consider the case where the given graph looks like a star graph. In this chapter, we introduce the graph parameter starwidth. It naturally measures how much the graph looks like a star graph in the sense that it is the star-equivalent of treewidth [23]. A star decomposition of a graph and its starwidth are defined as follows.

Definition 7.1. *A star decomposition of a graph $G = (V, E)$ is a star with core X_0 and leafs $X_1, \ldots, X_m$, where each X_i is associated with a subset of V, satisfying the following properties:*

1. *The union of all sets X_i equals V.*
2. *For every edge (v, w) in G, there is a subset X_i that contains both v and w.*
3. *For $i \neq j, i, j \geq 1$, if X_i and X_j both contain vertex v, then X_0 contains v as well.*

The width of a star decomposition is defined as $\max_i |X_i| - 1$. The starwidth $sw(G)$ of graph G is the minimum width among all possible star decompositions of G.

An example of a star decomposition of a graph is illustrated in Figure 7.1. Note that star graphs have starwidth equal to 1. Also note that the starwidth of a graph is larger than or equal to its treewidth, since every star decomposition is also a tree decomposition. It is interesting to study this parameter, because it might be that certain

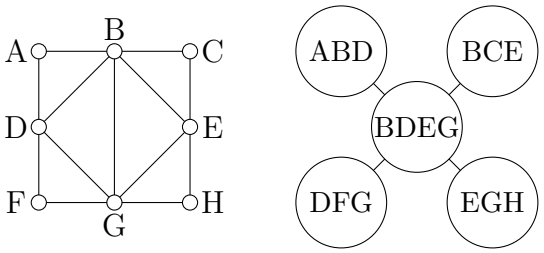


Fig. 7.1: Example graph and a star decomposition with width 3.

problems which are intractable on graphs restricted to some parameter being bounded, like bounded treewidth graphs, become tractable on bounded starwidth graphs. Unfortunately, we did not succeed in finding such a problem. However, this does not mean that such a problem does not exist and we think that the results obtained in this work can be interesting in their own right.

We start with characterizing the class of graphs of starwidth equal to 1 by a set of forbidden minors. We also show that the obstruction set for bounded starwidth graphs contains at least a path and a galaxy (collection of stars). Then, we show that computing the starwidth of a given graph is an NP-hard problem. It is fixed parameter tractable when parametrized by its outcome. We also show how the parameter relates to tree-depth and vertex cover number. Finally, we discuss the complexity of some problems on bounded starwidth graphs.

7.2 Characterization

We start with stating that the class of bounded starwidth graphs is closed under taking minors. Here, a graph H is a minor of G if H can be obtained from G by deleting vertices, by deleting edges and by contracting edges. It is easy to see that this is true, since deletion of vertices and edges or contraction of edges does not increase the starwidth of a graph.

Lemma 7.2. *If H is a minor of G, then $sw(H) \leq sw(G)$.*

This enables us to use the Graph Minor Theorem [91], which states that every class of minor-closed graphs can be characterized by a finite set of forbidden minors, also known as the obstruction set. We now show that the obstruction set of the graphs of starwidth 1 is given by the graphs in Figure 7.2.

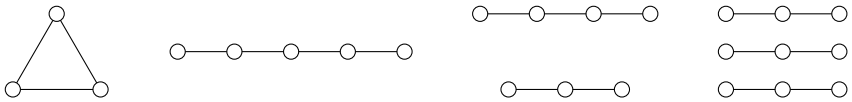


Fig. 7.2: K_3, P_5, $P_4 + P_3$ and $3P_3$.

Theorem 7.3. *Graph G has $sw(G) = 1$ iff G does not have K_3,P_5,$P_4 + P_3$ and $3P_3$ as minor.*

Proof. It is easy to see that K_3, P_5, $P_4 + P_3$ and $3P_3$ have starwidth 2. Since the class of graphs of starwidth equal to 1 is closed under taking minors, these graphs are forbidden minors.

Graphs that do not have K_3 as a minor are forests. Moreover, graphs that do not have P_5 as a minor have a diameter of at most 3 in each component. If there is a component with diameter 3, all other components are isolated edges or vertices. Otherwise, $P_4 + P_3$ is a minor. Similarly, if there is no component with diameter 3, the graph contains at most two stars with more than two vertices, and the other components are isolated edges or vertices. Otherwise, $3P_3$ is a minor. The theorem follows because the graphs described, i.e., graphs where at most two vertices have degree at least 2, have starwidth equal to 1. □

We now discuss the obstruction set for graphs of starwidth p, for some fixed p. Before proceeding to the next theorem, we define the graph class *galaxy*. A graph is called a galaxy if each component is a star. In particular, we define the (n, k)-galaxy to be a graph having n components, where each component is a star on k vertices.

Theorem 7.4. *The obstruction set characterizing graphs of starwidth k contains at least one line graph and one galaxy.*

Proof. First, we show that a line graph with n vertices has starwidth $\Omega(\sqrt{n})$. The optimal way to construct a star decomposition for P_n is to pick ℓ vertices whose deletion splits the path into $O(n/\ell)$ approximately equally sized segments, and put them in the core. This decomposition has starwidth $O(\min\{\ell, n/\ell\})$, which is minimized for $\ell = O(\sqrt{n})$. Hence, we have that $sw(P_n) \in \Omega(\sqrt{n})$. Thus, in order to get bounded starwidth, we need at least one path in the obstruction set.

Secondly, it is easy to see that the (n, n)-galaxy has starwidth exactly $n - 1$. This can be obtained by taking all centres in the core. Leaving one out leads to having a leaf containing n vertices. Hence, we need at least one galaxy in the obstruction set. □

It remains open whether these two forbidden minors define the class of bounded starwidth graphs, i.e., whether the class of bounded starwidth graphs is equivalent to the minor-closed class of graphs with an obstruction set containing at least one line graph and one galaxy.

7.3 Complexity

In this section, we first show that computing the starwidth of a given graph is NP-hard. Secondly, we will show that the problem parametrized by its outcome is fixed parameter tractable and even solvable in linear time.

To show that computing starwidth is NP-hard, we reduce from the *vertex cover problem* [70]. Here, we are given a graph $G = (V, E)$ and an integer k. The question is whether there is a subset of the vertices of size at most k such that every edge is covered, i.e., it has at least one endpoint in this set. The problem is still NP-complete in some restricted setting.

Lemma 7.5. *The vertex cover problem is still NP-complete when each vertex has degree at most $k - 1$, and there are no two vertices u, v with degree $k - 1$ such that $(u, v) \in E$ and $N(u) \setminus \{v\} = N(v) \setminus \{u\}$, where $N(u)$ denotes the neighborhood of vertex u.*

Proof. First of all, the problem is still NP-complete when each vertex has degree less than k. If vertex v has degree at least k, it is easy to verify whether the neighbors of v form a vertex cover. If this turns out to be false, we include v in the vertex cover, remove it from G and lower k by 1. Secondly, the problem is NP-complete when there are no two vertices of degree $k-1$ connected with an edge sharing the same $k-2$ neighbors. If an instance does have such a pair, one can do the following preprocessing step. If exactly one of the vertices is included in the vertex cover, then all the other $k-2$ neighbors should be included in it as well. This will result in a yes-instance if the graph without these vertices and incident edges is a star. If this is not the case for both vertices, one can include them in the vertex cover, delete them from G and lower k by 2. □

Theorem 7.6. *Computing starwidth is NP-hard.*

Proof. We are given a graph G and an integer k with the properties stated in Lemma 7.5. Create graph G' by copying G and adding $k - deg(v) - 1$ vertices for each vertex v, where $deg(v)$ denotes the degree of vertex v in G. Add edges such that each vertex forms a clique with its added vertices.

Suppose we have a vertex cover in G of size k. Then, we can create the following star decomposition of G'. The core consists of all vertices in the vertex cover. Each leaf corresponds to either a vertex in the remaining independent set with its neighbors or a vertex in the cover with its added vertices. It can be verified that each vertex is in the core or in a leaf, each edge is in the core or a leaf and that vertices that are in multiple leafs are also contained in the core. It is also easy to see that the core and each leaf has size at most k. Hence, G' has starwidth at most $k-1$.

Now, suppose that we a have a star decomposition of G' with width $k-1$. We claim that the core corresponds to a vertex cover in G of size k. For the sake of contradiction, suppose that this is not the case. This implies that there must be an edge (u, v) such that u and v are not in the core. This means that u and v are in the same leaf together with all their neighbors. If one of them has degree smaller than $k-1$ in G, then this leaf contains more than k vertices. This is also the case when both vertices have degree $k-1$ in G, since u and v do not share all their $k-2$ neighbors. Hence, we have a contradiction and the vertices in the core are a vertex cover of size k in G. □

Next, we consider the parametrized complexity of determining the starwidth of a graph. In this area of research, a problem is given by an input and a separate parameter k. A problem is called fixed parameter tractable (FPT) when there exists an algorithm that runs in $O(f(k) \cdot n^{O(1)})$ time. Here, n is the input size and f is some computable function. For more details on parametrized complexity, the reader is referred to [33].

In the k-starwidth problem, we are given as input a graph $G = (V, E)$ and a parameter k. The problem is to decide whether $sw(G) \leq k$. To show that k-starwidth is in FPT, we use the fact that a graph of starwidth at most k can be described by a finite set of forbidden minors [91]. Since verifying that a given graph has a specific minor can be done in time $O(f(h) \cdot n^3)$ [90], where h is the number of vertices of the minor, our problem is in FPT.

To improve this result, we show that the problem is MSO_1-expressible, where MSO_1 stands for first-order monadic second order logic, which is defined as follows. One is given a graph as a set of vertices V and an adjacency relation $adj(u, v)$. A graph

property is MSO_1-expressible if it can be described as a logic formula using sets of vertices. For example, one can describe non-connectivity by

$$\exists S(\exists x \in S \wedge \exists y \notin S \wedge (\forall u, v \in S(u \in S \wedge adj(u, v) \rightarrow v \in S))).$$

In words, there is a proper subset S of V such that vertices in S are only connected to vertices in S. Courcelle's Theorem [30] now states that every graph property that can be expressed as a MSO_1 formula of fixed size, can be recognized in linear time on bounded treewidth graphs. This enables us to prove the following theorem.

Theorem 7.7. *k-starwidth can be decided in $O(f(k) \cdot n)$ time.*

Proof. First, we give an alternative characterization of starwidth. A graph has starwidth at most k if there is a core S, containing at most $k+1$ vertices, such that for each vertex not in the core, there is a leaf containing this vertex, all its neighbors and at most $k+1$ vertices in total. This can be formulated in MSO_1 logic as follows.

$$\exists S(|S| \leq k+1 \wedge \forall v \in (V \setminus S)(\exists X(v \in X \wedge |X| \leq k+1 \\ \wedge \forall u \in (X \setminus S)(\forall w \in V(adj(u, w) \rightarrow w \in X))))).$$

Note that it is not allowed to use cardinality constraints. However, these can be avoided by additional strings of length $O(k)$. Namely, we can replace $\exists S(|S| \leq k+1)$ with

$$\exists S, s_1, \ldots, s_{k+1}(\forall v \in V(v = s_1 \vee \ldots \vee v = s_{k+1} \vee v \notin S)).$$

Hence, since we have fixed k, we derived that k-starwidth is MSO_1-expressible. Using the fact that every graph with bounded starwidth has bounded treewidth, we can apply Courcelle's Theorem in order to get a linear time algorithm. □

7.4 Relation with other parameters

Before relating starwidth to the vertex cover number, we discuss an equivalent graph completion problem. In graph completion problems, we are given a graph and we have to add edges such that the resulting graph is in some class of graphs while minimizing some objective function. It is well-known [23] that computing the treewidth is equivalent to finding the chordal completion with minimum clique size. Here, a graph is chordal if every cycle of length at least four has a chord, i.e., an edge between two vertices in the cycle that are not adjacent in the cycle. Also, computing pathwidth is equivalent with finding the interval completion with minimum clique size. An interval graph is the intersection graph of a family of intervals on the real line, where the vertices correspond to intervals.

We show a similar result for starwidth. For this, let graph class $\mathcal{H}$ contain all chordal graphs such that it has a star as optimal tree decomposition. These graphs can also be defined as chordal graphs with the property that there is one clique C_0 such that each pair of subsets of vertices that induce cliques $C_i \neq C_0$ and $C_j \neq C_0$ has the property $C_i \cap C_j \subseteq C_0$. This class is a subset of the chordal graphs and a superset of split graphs. Here, split graphs are defined as graphs that can be partitioned into a clique and an independent set. More generally, the class $\mathcal{H}$ is also a superset of k-starlike graphs, which are defined as follows. We define a graph to be k-starlike if

it is chordal and it has a star as optimal tree deomposition. Moreover, the number of vertices that is in a clique corresponding to a leaf in the star decomposition, but not in the clique corresponding to the core, is at most k. Note that the class of split graphs is equivalent to the class of 1-starlike graphs. We call the graphs in $\mathcal{H}$ the ∞-starlike graphs, since it is equivalent with the definition of k-starlike graphs, but without restricting cliques that correspond to leaves to have at most k new vertices. We can now prove the following theorem. Here, $\omega(G)$ denotes the size of the maximum clique of G. Further, we say that $H = (V_H, E_H)$ is a completion of $G = (V, E)$, denoted by $H \supseteq G$, if $V_H = V$ and $E_H \supseteq E$.

Theorem 7.8. *$sw(G) = \min\{\omega(H) - 1 | H \supseteq G, H \in \mathcal{H}\}$, where $\mathcal{H}$ is the class of ∞-starlike graphs.*

Proof. Suppose that we have a star decomposition of width $k-1$. For each node X_i of the decomposition, add edges to G such that the vertices of X_i form a complete subgraph. The resulting graph has a star as tree decomposition, namely the one we started with, and it has clique number at most k, since the core and every leaf has at most k vertices.

Now, suppose that G is contained in graph $H \in \mathcal{H}$ with clique number at most k. Take its tree decomposition, which is a star decomposition that is also valid for G. Since the clique number of H is at most k, each node has at most k vertices. Hence, the star decomposition has width $k-1$. □

We can now relate starwidth to the vertex cover number $vc(G)$. Here, we need that vertex cover number relates to finding the split completion of minimum clique size.

Corollary 7.9. *$vc(G) \geq sw(G)$.*

Proof. This follows from the fact that

$$\min_{H \supseteq G, H \in \mathcal{S}} \omega(H) \leq vc(G) + 1,$$

where $\mathcal{S}$ is the class of split graphs. This fact can be proven as follows. Given is a graph and its vertex cover. Now, add edges such that the vertices from the cover form a clique. It is easy to see that we have obtained a split completion. The size of the maximum clique in this graph is at most the size of the vertex cover plus 1. The extra vertex appears in this clique when it is adjacent to all vertices in the cover, but it is not in the cover itself.

Since split graphs form a subclass of ∞-starlike graphs,

$$\begin{aligned} vc(G) &\geq \min_{H \supseteq G, H \in \mathcal{S}} \omega(H) - 1 \\ &\geq \min_{H \supseteq G, H \in \mathcal{H}} \omega(H) - 1 = sw(G). \end{aligned}$$

□

Hence, when a graph has bounded vertex cover number, it also has bounded starwidth. It is easy to construct examples that have an unbounded vertex cover number, but bounded starwidth.

On the other hand, we can upper bound the class of bounded starwidth graphs by the class of bounded tree-depth graphs [81]. A tree-depth decomposition of component $G_i = (V_i, E_i)$ of $G = (V, E)$ is a rooted tree on V_i such that vertices connected by an edge have an ancestor-child relation in the tree. The tree-depth of a component is the minimum depth among all tree-depth decompositions and the tree-depth of a graph is the maximum tree-depth over all components.

Theorem 7.10. *If a graph has bounded starwidth, it has bounded tree-depth.*

Proof. Given a star decomposition of width k, one can construct the tree depth decomposition depicted in Figure 7.3. We start by making a path on the vertices in the core. Then, for each leaf, we consider the vertices of the leaf without vertices of the core and attach a path on these vertices to the bottom vertex of the first path. In the resulting tree, there is a ancestor-child relation between every pair of vertices, unless the vertices are in different leafs and not in the core. Since these vertices do not have an edge connecting them, we obtained a feasible tree-depth decomposition. This decomposition has tree-depth at most $2k + 2$. □

Observe that a galaxy has bounded tree-depth, but unbounded starwidth. This shows that these classes have a strict inclusion.

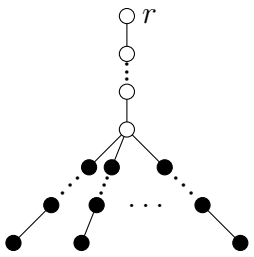


Fig. 7.3: Tree-depth decomposition with root r obtained from star decomposition. White vertices correspond to the vertices of X_0. Black vertices correspond to the vertices of $X_1 \setminus X_0, X_2 \setminus X_0, \ldots, X_m \setminus X_0$.

7.5 Problems on bounded starwidth graphs

The main open question is whether there is a problem that is intractable when considered on bounded treewidth graphs (or even bounded tree-depth graphs), but solvable on bounded starwidth graphs. This would increase the importance of the parameter. Unfortunately, we did not find such a problem yet. We now discuss the complexity of some problems on bounded starwidth graphs.

7.5.1 Traveling repairman problem

In this section, we consider again the traveling repairman problem (TRP) with vertex weights. Here, we are given a rooted graph with vertex weights and edge lengths and the goal is to find a tour that minimizes the sum of weighted latencies. It was shown in [96] that the problem is NP-hard on trees by a reduction from 3-Partition [51]. Here,

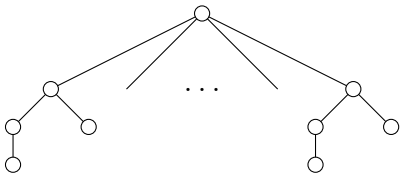


Fig. 7.4: Tree used in reduction for TRP

we show how a similar reduction shows that TRP is NP-hard on the tree depicted in Figure 7.4. Since this tree has starwidth 4, we know that TRP is still NP-hard on trees of starwidth 4.

Note that we can reduce an instance of TRP to an instance with unit vertex weights by replacing a vertex with weight w by w vertices at distance zero from each other. This works as long as the vertex weights are polynomially bounded. In this way, one can see that TRP is even NP-hard if the vertex weights are equal to one. However, since this construction increases the starwidth dramatically, we cannot make a similar statement about the complexity of TRP on bounded starwidth graphs.

We will reduce TRP from PARTITION [70] and redo the proof of [96]. First, we need some additional notation. Consider the optimal tour T and let $t_0 = 0, t_1, \ldots, t_k$ be the moments at which the repairman is in the root, where t_k equals the length of T. If we define T_i to be the subtour of T between time t_{i-1} and t_i, and denote $|T_i|$ and W_i for the total length and total weight of T_i respectively, then we have the following lemma.

Lemma 7.11. *[96] For any optimal tour T the following holds.*

1. $\frac{W_1}{|T_1|} \geq \frac{W_2}{|T_2|} \geq \ldots \geq \frac{W_k}{|T_k|}$.
2. *If $\frac{W_i}{|T_i|} = \frac{W_j}{|T_j|}$ for some $i, j \in \{1, \ldots, k\}$, then the total latency remains the same if the subtours T_i and T_j swap their position on T.*

We are now able to prove NP-hardness by using a reduction from PARTITION. In this problem, we are given n integers $p_1, \ldots, p_n$ and a number $P = \frac{1}{2}\sum_i p_i$. The question is whether there exists a set of indices $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} p_i = P$. This problem is weakly NP-complete [70].

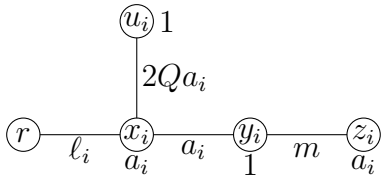


Fig. 7.5: Branch i in reduction for TRP.

Theorem 7.12. *TRP is (weakly) NP-hard on trees with starwidth 4.*

Proof. Given an instance of PARTITION, we define $a_i = Kp_i$ for all i and $Q = KP$, where $K = 2Pn^2$. Hence, $Q = \frac{1}{2}\sum_i a_i$. We construct the following starwidth 4 tree on $4n + 1$ vertices. Let r be the root of the tree.

For each i, we create a branch containing the path (r, x_i, y_i, z_i) and the edge (x_i, u_i) (Figure 7.5). The length of edges $(r, x_i), (x_i, y_i), (y_i, z_i)$ and (x_i, u_i) are equal to ℓ_i, a_i, m and $2Qa_i$ respectively, where ℓ_i and m are specified later. Vertices x_i and z_i get weight a_i, whereas vertices y_i and u_i get weight 1.

We can now choose lengths ℓ_i and m sufficiently large so that for each i, we only traverse edge (r, x_i) exactly once in each direction and we always visit vertex u_i before z_i. Furthermore, if we set ℓ_i equal to $M(2a_i + 2) - (2Q + 1)a_i - m$, where M is a big number, we have all $\frac{W_i}{L_i}$ equal, where W_i and L_i denote the total weight and total length of branch i respectively.

Note that there are only two ways to visit the clients in branch i. Either we do (r, x_i, u_i, y_i, z_i), which we call a 1-tour, or we do (r, x_i, y_i, u_i, z_i), which we call a 2-tour. Fix an optimal solution. Renumber the branches such that in the optimal tour branches $1, \ldots, n'$ are traversed as a 1-tour and branches $n' + 1, \ldots, n$ are traversed as a 2-tour. By Lemma 7.11, we know that branches $1, \ldots, n'$ are visited before branches $n' + 1, \ldots, n$.

We are now going to compare the optimal tour to the solution where each branch is traversed as a 1-tour. By Lemma 7.11 and the choices above, we may assume that the order in which the branches are served is the same for both tours. We start with the all 1-tour solution and modify it to obtain the optimal solution, i.e., we traverse branches $n' + 1, \ldots, n$ as a 2-tour instead of a 1-tour.

If branch i becomes traversed as a 2-tour, the following changes happen. The latency of vertex y_i decreases by $4Qa_i$. However, the latencies of u_j and z_j for $j \geq i$ and x_j and y_j for $j \geq i + 1$ each increase by $2a_i$. The total (gross) delay of vertices u_j and y_j is denoted by R. Note that there are at most $2n$ vertices that all have weight 1 and each of them has a delay of at most $2(n-1)Q$. Hence, $R < 4Qn^2$. We can now make the following derivation. Let C^{OPT} and C denote the total latency of the optimal and all 1-tour solution respectively.

$$\begin{aligned} C^{OPT} &= C + R + \sum_{i=n'+1}^{n} \left(2a_i^2 - 4Qa_i + \sum_{j=i+1}^{n} 4a_i a_j \right) \\ &= C + R + 2 \left(\sum_{i=n'+1}^{n} a_i \right)^2 - 4Q \sum_{i=n'+1}^{n} a_i. \end{aligned}$$

If we have a yes-instance for Partition, we can associate the branches that are served as a 1-tour and the ones served as a 2-tour with the sets that induce the partition respectively. Then, we get $\sum_{i=n'+1}^{n} a_i = Q$ and

$$\begin{aligned} C^{OPT} &= C + R + 2Q^2 - 4Q^2 \\ &= C + R - 2Q^2 \\ &< C + 4Qn^2 - 2Q^2 \\ &= C + 2K^2 - 2Q^2. \end{aligned} \tag{7.1}$$

Since $\sum_{i=n'+1}^{n} a_i$ and Q are both a multiple of K, we know that

$$\left(\sum_{i=n'+1}^{n} a_i - Q \right)^2 < K^2$$

if and only if $\sum_{i=n'+1}^{n} a_i = Q$. Equivalently,

$$2\left(\sum_{i=n'+1}^{n} a_i\right)^2 - 4Q\sum_{i=n'+1}^{n} a_i < -2Q^2 + 2K^2$$

if and only if $\sum_{i=n'+1}^{n} a_i = Q$. Therefore, if we have a no-instance for PARTITION, we have

$$\begin{aligned} C^{OPT} &= C + R + 2\left(\sum_{i=n'+1}^{n} a_i\right)^2 - 4Q\sum_{i=n'+1}^{n} a_i \\ &\geq C - 2Q^2 + 2K^2. \end{aligned}$$

Combing this with (7.1) completes the proof. Since the constructed tree is the tree in Figure 7.4 with starwidth 4, we have shown that TRP is NP-hard for trees with starwidth 4. □

Note that the proof above also shows that TRP is NP-hard on trees with diameter 6. In [21], it was shown that TRP is polynomially solvable on graphs of diameter 3. It would be interesting to close this gap. Also note that it follows from [21] that TRP is solvable on (connected) starwidth 1 graphs.

7.5.2 Other problems

In the *weighted diameter problem* (WDP), we are given a graph $G = (V, E)$ and $|E|$ integers. The goal is to find an assignment of the integers as lengths to the edges such that the diameter of the graph is minimized. It was shown to be NP-hard on trees in [87] by a reduction from PARTITION. The authors also note that a similar reduction from 3-PARTITION shows that the problem is still NP-hard on the tree in Figure 7.6. Hence, the WDP is NP-hard on trees of starwidth 3. Note that in 3-PARTITION, we are given $3m$ integers and the question is whether we can partition these into m sets of equal weight.

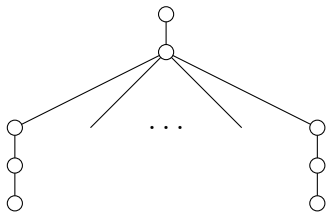


Fig. 7.6: Tree used in reduction for WDP.

In the *graph motif problem* (GMP), we are given a vertex-colored graph and a multiset of colors. The question is whether there is a connected subgraph that contains exactly the colors from the given multiset. Lacroix et al. [77] show that the problem is NP-complete on trees by a reduction from EXACT COVER BY 3-SETS. Since the tree they created (Figure 7.7) has starwidth 4, the GMP is still NP-complete on trees of starwidth 4.

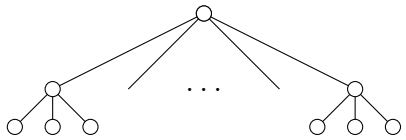


Fig. 7.7: Tree used in reduction for GMP.

Another problem that was shown to be NP-hard on trees, even on bounded diameter trees, is the *harmonious coloring problem* [35]. In this problem, we are given a graph and the goal is to properly color the vertices, i.e., such that every two vertices connected by an edge are colored differently, using the minimum number of colors. However, for each pair of colors, there should be at most one edge that has these colors as endpoints. The reduction used in [35] constructs a graph starting with a galaxy. Hence, this tree does not have bounded starwidth. It remains an open question whether the harmonious coloring problem, or its counterpart *achromatic number*, i.e., the problem of finding a proper vertex coloring, such that every pair of colors is used at least once, using the maximum number of colors, is solvable on bounded starwidth graphs.

7.6 Conclusion

We introduced the graph parameter starwidth and have shown preliminary results on characterization, complexity and the relation with other parameters. The main open question is whether there is a problem that is intractable on bounded treewidth graphs (or even on bounded tree-depth graphs), but solvable on bounded starwidth graphs.

Finally, we would like to mention that the TRP has only been shown to be polynomially solvable if the underlying graph is a star or a line (path) graph. Two extensions for which the complexity is still open, even in the case of uniform vertex weights, are depicted in Figure 7.8.

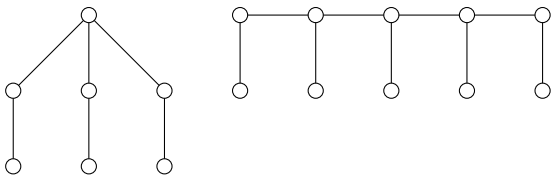


Fig. 7.8: Left: spider of depth two, right: centipede graph.

The first is a spider of depth 2, which is defined as a tree having exactly one vertex with degree greater than 2 and all other vertices are at distance at most 2 from this vertex. Note that this graph has starwidth equal to 2. If one is able to determine which legs have to be visited in two visits, one can use the algorithm for star graphs to solve TRP on this type of graph. However, an efficient way to do this is missing. Hence, the TRP is still open on spiders of depth 2. Another case which is still open is the TRP on caterpillar graphs. Formally, a caterpillar is a tree such that deletion of the leafs will result in a line graph. One could also consider it to be a path with edges sticking out. Extending the dynamic programming algorithm for TRP on line graphs

does not seem to work. Note that the TRP is even open on the centipede graph, a caterpillar with only one edge sticking out of each vertex of the central path. Also note that this models the TRP on the line with processing times, which is a long-standing open problem [74].

Overview own publications

Academic publications

Martijn van Ee, Some notes on bounded starwidth graphs. *Information Processing Letters*, 125:9−14, 2017.

Martijn van Ee and René Sitters, On the complexity of master problems. In G.F. Italiano, G. Pighizzini, and D.T. Sannella, editors, *Proceedings of the 40th Symposium on Mathematical Foundations of Computer Science*, volume 9235 of *Lecture Notes in Computer Science*, pages 567−576. Springer, 2015.

Martijn van Ee and René Sitters, Routing under uncertainty: The *a priori* traveling repairman problem. In E. Bampis and O. Svensson, editors, *Approximation and Online Algorithms: 12th International Workshop, Revised Selected Papers*, volume 8952 of *Lecture Notes in Computer Science*, pages 248−259. Springer, 2015.

Martijn van Ee and René Sitters, The *a priori* traveling repairman problem. To appear in Algorithmica.

Martijn van Ee, Leo van Iersel, Teun Janssen and René Sitters, *A priori* TSP in the scenario model. In K. Jansen and M. Mastrolili, editors, *Approximation and Online Algorithms: 14th International Workshop, Revised Selected Papers*, volume 10138 of *Lecture Notes in Computer Science*, pages 183−196. Springer, 2017.

Other

Martijn van Ee, De verdwaalde koe. *STAtOR*, 18(2):16-19, 2017.

Under review

Martijn van Ee, Leo van Iersel, Teun Janssen and René Sitters, *A priori* TSP in the scenario model. *Submitted for publication.*

Martijn van Ee and René Sitters, Approximation and complexity of multi-target graph search and the Canadian traveler problem. *Submitted for publication.*

Bibliography

[1] Foto Afrati, Stavros Cosmadakis, Christos H. Papadimitriou, George Papageorgiou, and Nadia Papakostantinou. The complexity of the travelling repairman problem. *RAIRO Informatique théorique*, 20(1):79–87, 1986.

[2] Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[3] Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[4] Sanjeev Arora, Michelangelo Grigni, David R. Karger, Philip N. Klein, and Andrzej Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 33–41. ACM/SIAM, 1998.

[5] Sanjeev Arora and George Karakostas. A $2 + \epsilon$ approximation algorithm for the k-MST problem. *Mathematical Programming*, 107(3):491–504, 2006.

[6] Giorgio Ausiello, Stefano Leonardi, and Alberto Marchetti-Spaccamela. On salesmen, repairmen, spiders, and other traveling agents. In G.C. Bongiovanni, G. Gambosi, and R. Petreschi, editors, *Proceedings of the 4th Italian Conference on Algorithms and Complexity*, volume 1767 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2000.

[7] Per Austrin, Rajsekar Manokaran, and Cenny Wenner. On the NP-hardness of approximating ordering-constraint satisfaction problems. *Theory of Computation*, 11:257–283, 2015.

[8] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J.E. Rawlins. Searching with uncertainty. In R.G. Karlsson and A. Lingas, editors, *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory*, volume 318 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 1988.

[9] Amotz Bar-Noy and Baruch Schieber. The Canadian traveller problem. In *Proceedings of the 2nd Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 261–270. ACM/SIAM, 1991.

[10] Vic Baston and Anatole Beck. Generalizations in the linear search problem. *Israel Journal of Mathematics*, 90:301–323, 1995.

[11] Anatole Beck. On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, 1964.

[12] Anatole Beck. More on the linear search problem. *Israel Journal of Mathematics*, 3(2):61–70, 1965.

[13] Anatole Beck and Micah Beck. Son of the linear search problem. *Israel Journal of Mathematics*, 48(2-3):109–122, 1984.

[14] Anatole Beck and Micah Beck. The linear search problem rides again. *Israel Journal of Mathematics*, 53(3):365–372, 1986.

[15] Anatole Beck and Micah Beck. The revenge of the linear search problem. *SIAM Joural of Control and Optimization*, 30(1):112–122, 1992.

[16] Anatole Beck and D.J. Newman. Yet more on the linear search problem. *Israel Journal of Mathematics*, 8:419–429, 1970.

[17] Anatole Beck and Peter Warren. The return of the linear search problem. *Israel Journal of Mathematics*, 14:169–183, 1973.

[18] Richard Bellman. Problem 63-9, an optimal search. *SIAM Review*, 5(3):274, 1963.

[19] Dimitris Bertsimas. *Probabilistic combinatorial optimization problems*. PhD thesis, Massachusetts Institute of Technology, 1988.

[20] Andreas Blass and Yuri Gurevich. On the unique satisfiability problem. *Information and Control*, 55(1-3):80–88, 1982.

[21] Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 163–171. ACM, 1994.

[22] Zahy Bnaya, Ariel Felner, and Solomon Eyal Shimony. Canadian traveler problem with remote sensing. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 437–442, 2009.

[23] Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–21, 1993.

[24] Nicolas Boria, Cécile Murat, and Vangelis Paschos. On the probabilistic min spanning tree problem. *Journal of Mathematical Modelling and Algorithms*, 11(1):45–76, 2012.

[25] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 36–45. IEEE, 2003.

[26] Lin Chen, Nicole Megow, Roman Rischke, and Leen Stougie. Stochastic and robust scheduling in the cloud. In N. Garg, K. Jansen, A. Rao, and J.D.O. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015*, volume 40 of *LIPIcs*, pages 175–186. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[27] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[28] Fabian A. Chudak, T. Roughgarden, and David P. Williamson. Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation. *Mathematical Programming*, 100(2):411–421, 2004.

[29] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[30] Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[31] Vladimir G. Deineko, Rüdiger Rudolf, and Gerhard J. Woeginger. Sometimes travelling is easy: The master tour problem. *SIAM Journal of Discrete Mathematics*, 11(1):81–93, 1998.

[32] Erik D. Demaine, Sándor S. Fekete, and Shmuel Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2-3):342–355, 2006.

[33] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[34] Jan Driessen and Teun Janssen. Minimizing the blading in lithography machines: an application of the *a priori* TSP problem. Unpublished manuscript, 2016.

[35] Keith Edwards and Colin McDiarmid. The complexity of harmonious colouring for trees. *Discrete Applied Mathematics*, 57(2-3):133–144, 1995.

[36] Martijn van Ee. De verdwaalde koe. *STAtOR*, 18(2):16–19, 2017.

[37] Martijn van Ee. Some notes on bounded starwidth graphs. *Information Processing Letters*, 125:9–14, 2017.

[38] Martijn van Ee, Leo van Iersel, Teun Janssen, and René Sitters. *A priori* TSP in the scenario model. In K. Jansen and M. Mastrolili, editors, *Approximation and Online Algorithms: 14th International Workshop, Revised Selected Papers*, volume 10138 of *Lecture Notes in Computer Science*, pages 183–196. Springer, 2017.

[39] Martijn van Ee and René Sitters. The *a priori* traveling repairman problem. To appear in Algorithmica.

[40] Martijn van Ee and René Sitters. On the complexity of master problems. In G.F. Italiano, G. Pighizzini, and D.T. Sannella, editors, *Proceedings of the 40th Symposium on Mathematical Foundations of Computer Science*, volume 9235 of *Lecture Notes in Computer Science*, pages 567–576. Springer, 2015.

[41] Martijn van Ee and René Sitters. Routing under uncertainty: The *a priori* traveling repairman problem. In E. Bampis and O. Svensson, editors, *Approximation and Online Algorithms: 12th International Workshop, Revised Selected Papers*, volume 8952 of *Lecture Notes in Computer Science*, pages 248–259. Springer, 2015.

[42] Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoß, and Guido Schäfer. Connected facility location via random facility sampling and core detouring. *Journal of Computer and System Sciences*, 76(8):709–726, 2010.

[43] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[44] 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 Z. Cai, A. Zelikovsky, and A.G. Bourgeois, editors, *Proceedings of the 20th International Conference on Computing and Combinatorics*, volume 8597 of *Lecture Notes in Computer Science*, pages 559–571. Springer, 2014.

[45] Lester R. Ford Jr. and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[46] Wallace Franck. An optimal search problem. *SIAM Review*, 7(4):503–512, 1965.

[47] Dror Fried. *Theoretical Aspects of the Generalized Canadian Traveler Problem.* PhD thesis, Ben-Gurion University of the Negev, 2013.

[48] Dror Fried, Solomon Eyal Shimony, Amit Benbassat, and Cenny Wenner. Complexity of Canadian traveler problem variants. *Theoretical Computer Science*, 487:1–16, 2013.

[49] Zvi Galil and Nimrod Megiddo. Cyclic ordering is NP-complete. *Theoretical Computer Science*, 5(2):179–182, 1977.

[50] Alfredo Garcia, Pedro Jodrá, and Javier Tejel. A note on the traveling repairman problem. *Networks*, 40(1):27–31, 2002.

[51] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[52] Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proceeding of the 37th Annual ACM Symposium on the Theory of Computing*, pages 396–402. ACM, 2005.

[53] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the 19th Symposium on Discrete Algorithms*, pages 942–951. SIAM, 2008.

[54] Michel X. Goemans and Jon Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1-2):111–124, 1998.

[55] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[56] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[57] Michael T. Goodrich and Roberto Tamassia. *Algortihm design: foundations, analysis, and internet examples*. Wiley, 2002.

[58] Igor Gorodezky, Robert D. Kleinberg, David B. Shmoys, and Gwen Spencer. Improved lower bounds for the universal and *a priori* TSP. In M.J. Serna, R. Shaltiel, K. Jansen, and J.D.P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010*, volume 6302 of *Lecture Notes in Computer Science*, pages 178–191. Springer, 2010.

[59] Anupam Gupta, Mohammad Taghi Hajiaghayi, and Harald Räcke. Oblivious network design. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 970–979. ACM, 2006.

[60] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM Journal on Computing*, 40(5):1361–1401, 2011.

[61] Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.

[62] Mohammad T. Hajiaghayi, Robert 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 Algorithms*, pages 649–658. ACM, 2006.

[63] T.E. Harris and F.S. Ross. Fundamentals of a method of evaluating rail net capacities. Research Memorandum RM-1573, The RAND Corportation, Santa Monica, California, October 24, 1955.

[64] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[65] Mathias Hauptmann. *Approximation complexity of optimization problems: Structural foundations and Steiner tree problems*. PhD thesis, Rheinischen Friedrich-Wilhelms-Universität Bonn, 2004.

[66] Ara Hayrapetyan, Chaitanya Swamy, and Éva Tardos. Network design for information networks. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 933–942. SIAM, 2005.

[67] Nicole Immorlica, David R. Karger, Maria Minkoff, and Vahab S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 691–700. SIAM, 2004.

[68] Patrick Jaillet. *Probabilistic traveling salesman problems*. PhD thesis, Massachusetts Institute of Technology, 1985.

[69] Ming-Yang Kao, John H. Reif, and Stephen R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

[70] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[71] Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81(8):1665–1677, 2015.

[72] Offer Kella. Star search - a different show. *Israel Journal of Mathematics*, 81:145–159, 1993.

[73] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

[74] Elias Koutsoupias, Christos Papadimitriou, and Mihalis Yannakakis. Searching a fixed graph. In F. Meyer auf der Heide and B. Monien, editors, *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 280–289. Springer, 1996.

[75] Mark W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[76] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research*, 2:83–97, 2017.

[77] Vincent Lacroix, Chirstina G. Fernandes, and Marie-France Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):360–368, 2006.

[78] Timo Leipälä. On the solutions of stochastic traveling salesman problems. *European Journal of Operational Research*, 2(4):291–297, 1978.

[79] László Lovász. Coverings and colorings of hypergraphs. In *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 3–12, 1973.

[80] László Lovász. *Mathematical Programming The State of the Art*, chapter Submodular functions and convexity. Springer, 1983.

[81] Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, 2006.

[82] Evdokia Nikolova and David R. Karger. Route planning under uncertainty: The Canadian traveller problem. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 969–974, 2008.

[83] Christos H. Papadimitriou. On the complexity of unique solutions. *Journal of the ACM*, 31(2):392–400, 1984.

[84] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc, 1998.

[85] Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[86] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

[87] Yehoshua Perl and Shmuel Zaks. On the complexity of edge labelings for trees. *Theoretical Computer Science*, 19:1–16, 1982.

[88] George H Polychronopoulos and John N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, 1996.

[89] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization. *Mathematical Programming*, 108(1):97–114, 2006.

[90] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[91] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[92] Horst Sachs. Regular graphs with given girth and restricted circuits. *Journal of the London Mathematical Society*, 1(1):423–429, 1963.

[93] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[94] Frans Schalekamp and David B. Shmoys. Algorithms for the universal and *a priori* TSP. *Operations Research Letters*, 36(1):1–3, 2008.

[95] David B. Shmoys and Kunal Talwar. A constant approximation algorithm for the *a priori* traveling salesman problem. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization*, volume 5035 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2008.

[96] René Sitters. The minimum latency problem is NP-hard for weighted trees. In W.J. Cook and A.S. Schulz, editors, *Proceedings of the 9th International Conference on Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 230–239. Springer, 2002.

[97] René Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 604–616. SIAM, 2014.

[98] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[99] Chaitanya Swamy and Amit Kumar. Primal–dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[100] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(3):85–93, 1986.

[101] Gerd Wechsung. On the Boolean closure of NP. In L. Budach, editor, *Proceedings of the International Conference on Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 485–493. Springer, 1985.

[102] Cenny Wenner. Hardness results for the shortest path problem under partial observability. Bachelor thesis, Department of computer science, Faculty of science, Lund University, 2009.

[103] Stephan Westphal. A note on the k-Canadian traveller problem. *Information Processing Letters*, 106(3):87–89, 2008.

[104] David P. Williamson and David B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.

[105] Anke van Zuylen. Deterministic sampling algorithms for network design. *Algorithmica*, 60(1):110–151, 2011.

Summary

In this thesis, we study the approximability and complexity of routing problems under uncertainty. In routing problems, we have to walk through a graph while optimizing some objective. For example, in the *traveling salesman problem* (TSP) we have to visit all vertices in the graph using a tour of minimum length. Another example is the *traveling repairman problem* (TRP), where we want to minimize the total latency. In this thesis, we consider two types of routing problems under uncertainty: *a priori routing* and *graph search*.

In a priori routing only a subset of the vertices of the graph needs to be visited but this subset is unknown. However, we need to make just one tour on all vertices called the first-stage tour. After that, the set of vertices to be visited, the active set, is revealed. The second-stage tour can now be derived by shortcutting the first-stage tour to the active set. If we are given a probability distribution over the active sets, we can compute the expected cost of the second-stage tour. Now, our goal is to construct a tour minimizing this expectation. We consider two objectives, namely the length (as in TSP) and the total latency (as in TRP) of a tour. We also consider two models for the probability distribution. In the independent decision model, each vertex is active with its own probability independent of other vertices. In the scenario model, we are given a list of possible active sets, called scenarios, with a probability distribution over the scenarios.

The *graph search problem* is defined as follows. There is a target hidden at a vertex in the graph and one has to find it. We are given a probability distribution and our goal is to find a walk that minimizes the expected length until finding the target. When all probabilities are equal, the graph search problem reduces to the TRP. We consider generalizations to multiple targets, failing edges and non-discrete probability distributions.

In Chapter 2 we study the *a priori traveling repairman problem* in the independent decision model. Using a series of reductions to other routing problems, we were able to give a constant-factor approximation for the uniform case, i.e., when all probabilities are equal. Then, in Chapter 3, we discuss the *a priori traveling salesman problem* in the scenario model. We show that there is a constant-factor approximation when the scenarios are small, big or nested or when the number of scenarios is bounded. By linking the problem to the so-called permutation constraint satisfaction problems, we show strong inapproximability results. We also look into approximation algorithms using the master tour lower bound, i.e., the weighted sum of optimal values for the scenarios. However, we show that this lower bound will not help us to improve on prior work. In Chapter 4, we show that determining if a given graph and a set of scenarios has a master tour, a tour such that each second-stage tour has the same length as the optimal tour on just that scenario, is Δ_2^p-complete. We also show this for the master

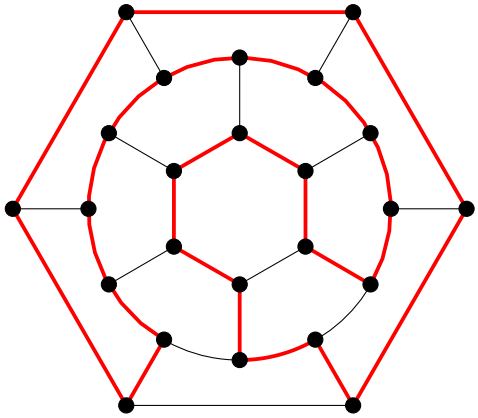
versions of satisfiability and Steiner tree. The main open question in a priori routing is whether we can improve upon the current best $O(\log n)$-approximation for a priori TSP in the scenario model.

The *Canadian traveler problem* is investigated in Chapter 5. In this problem, we have to walk from s to t in a graph. However, only edges in the active set (again using a probability distribution) are present. Moreover, we only know whether an edge is present after visiting one of its endpoints. The goal is to construct a policy that minimizes the expected length of the walk. We study both the independent decision and the scenario model. For the independent decision model, we show NP-hardness for series-parallel graphs, solving an open question posed in the literature. Moreover, we investigate what it means for an algorithm to be non-adaptive and show results on the implications of being non-adaptive through analyzing the adaptivity gap. For the scenario model, we show NP-completeness on disjoint-path graphs and cactus graphs. Although this chapter is named after the Canadian traveler problem, it mainly concerns a special case called the *multi-target graph search problem*. For this problem, in the independent decision model, we show NP-completeness on trees and give a 14.4-approximation for general metrics. In the scenario model, the problem is already NP-complete on stars.

Chapter 6 deals with the *lost cow problem*. This problem can be seen as a non-discrete generalization of the graph search problem on the line. For symmetric probability distributions, we characterize whether it is optimal not to turn, i.e., to walk to the end of the line on one side and then to the other end. We generalize this result to more than two directions and also consider non-symmetric distribution functions. In Chapter 7, we introduce the graph parameter *starwidth*. It naturally measures to what extend a graph looks like a star graph in the sense that it is the star-equivalent of treewidth. Our main goal in introducing this parameter was to extend the polynomial solvability of TRP on stars to graphs that look like stars. Unfortunately, TRP turns out to be NP-complete on trees of starwidth 4. Nonetheless, this parameter might be interesting in another setting. In this chapter, we consider characterizations of graphs with small starwidth, the complexity of computing the starwidth of a given graph and the relation between other graph parameters. Finally, we show three problems that are NP-complete on trees and remain NP-complete on trees with small starwidth.

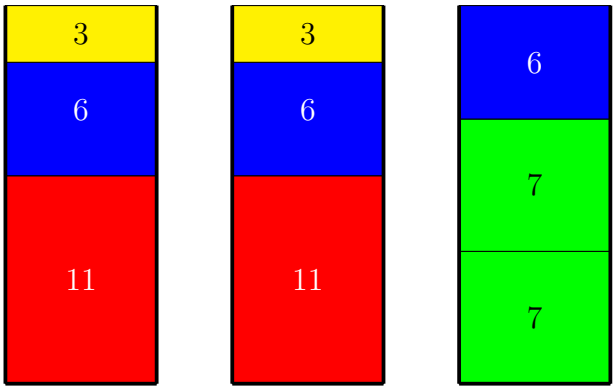
Solutions to the exercises

Hamiltonian cycle The first exercise was to determine whether the graph in Figure 1.1 contains a Hamiltonian cycle, a tour that uses only edges to go from one vertex to another, and visits each vertex exactly once. This graph, the $(12, 2)$-Petersen graph, is Hamiltonian. For example, the following tour can be drawn.



Actually, there are 34 Hamiltonian cycles in this graph. The problem of deciding whether a given graph has a Hamiltonian cycle is an NP-complete problem [70].

Bin packing The reader was challenged to solve the instance of the bin packing problem with bin capacity 20 and item sizes 11, 11, 7, 7, 6, 6, 6, 3 and 3. A solution to the bin packing problem can be visualized as in the figure below, where heights correspond to bin capacity or item sizes. The solution depicted below is the only solution using three bins. It is clear that this is optimal, since the items would not fit into two bins. The bin packing problem, pack the items into a minimum number of bins, is also an NP-hard problem (its decision version is NP-complete) [51].



Assignment problem Here, we were looking for the optimal assignment of swimmers to disciplines for a 4×50 meters relay medley. There are actually three solutions, with value 125, as shown in the tables below, where the chosen entries of the matrix are colored red.

Sol. 1	A	B	C	D
Back	39	33	38	35
Brea	34	41	34	33
Butt	29	30	33	31
Free	28	27	30	29

Sol. 2	A	B	C	D
Back	39	33	38	35
Brea	34	41	34	33
Butt	29	30	33	31
Free	28	27	30	29

Sol. 3	A	B	C	D
Back	39	33	38	35
Brea	34	41	34	33
Butt	29	30	33	31
Free	28	27	30	29

This problem, also known as the weighted bipartite matching problem, is polynomially solvable. One algorithm that solves the problem is the Hungarian method [76]. It starts with the matrix of values. Since in any assignment each discipline will take at least the minimum time for that discipline, we can subtract this minimum for each row in the matrix and obtain an equivalent instance. So,

$$\begin{pmatrix} 39 & 33 & 38 & 35 \\ 34 & 41 & 34 & 33 \\ 29 & 30 & 33 & 31 \\ 28 & 27 & 30 & 29 \end{pmatrix} \rightarrow \begin{pmatrix} 6 & 0 & 5 & 2 \\ 1 & 8 & 1 & 0 \\ 0 & 1 & 4 & 2 \\ 1 & 0 & 3 & 2 \end{pmatrix}$$

We also know that in any assignment each swimmer will take at least its minimum time over the disciplines. Hence, we can also subtract this minimum over all columns in the matrix (only useful in column 3).

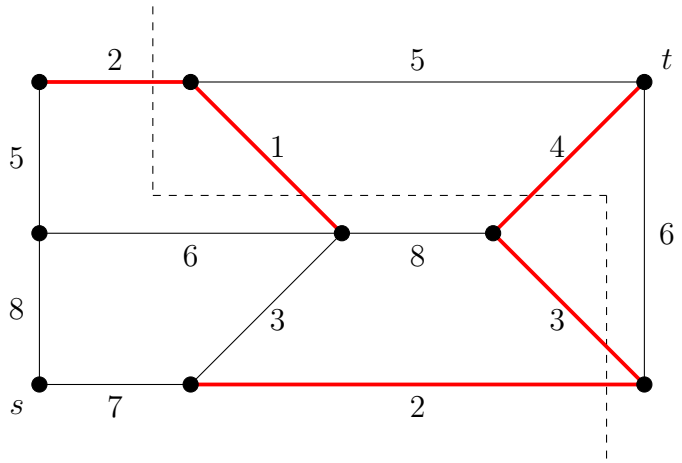
$$\begin{pmatrix} 6 & 0 & 5 & 2 \\ 1 & 8 & 1 & 0 \\ 0 & 1 & 4 & 2 \\ 1 & 0 & 3 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 6 & 0 & 4 & 2 \\ 1 & 8 & 0 & 0 \\ 0 & 1 & 3 & 2 \\ 1 & 0 & 2 & 2 \end{pmatrix}$$

We would like to make an assignment using only zeros. Unfortunately, this is not possible yet. Since this is not possible, we could cover all zeros in the matrix by covering less than four rows or columns. In this case, we can cover all zeros by covering the first two columns and the second row. The Hungarian method now looks at the minimum uncovered element, subtracts it from all uncovered elements and adds it to all double covered elements. This gives

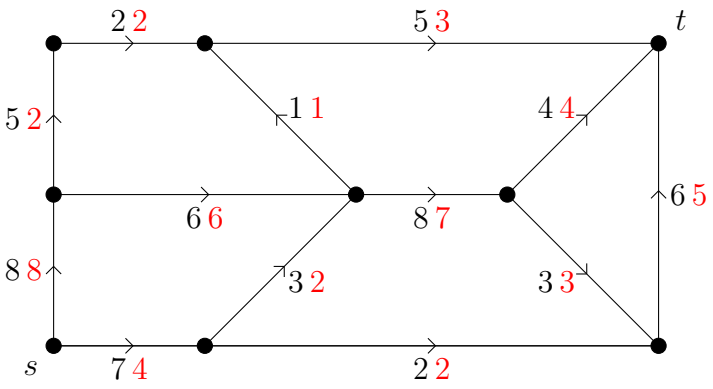
$$\left(\begin{array}{|c|c|cc} \hline 6 & 0 & 4 & 2 \\ \hline 1 & 8 & 0 & 0 \\ \hline 0 & 1 & 3 & 2 \\ 1 & 0 & 2 & 2 \\ \hline \end{array}\right) \rightarrow \begin{pmatrix} 6 & 0 & 2 & 0 \\ 3 & 10 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Now, we can make an assignment using only zeros. In fact, there are three ways to do this, like stated before. One could understand the last step of the algorithm if one understands the problem more thoroughly. For this, we advise the reader to read a basic textbook in discrete mathematics [84].

Minimum cut As a final exercise, the reader could try to find the minimum cut in the graph in Figure 1.4. The solution is illustrated in the figure below, where red edges are edges chosen in the cut. The minimum cut has weight 12. In the figure below, there is also a dashed line which shows how the graph is cut into two components, one containing s and one containing t.



The minimum cut problem is also polynomially solvable. Moreover, it has a nice relation with the maximum flow problem. In the latter problem, one is given a graph with edge capacities and two specified vertices s and t. The goal is to send as much flow as possible from s to t, while not exceeding the edge capacities. Here, a flow can be thought of as liquid flowing through pipelines. A flow should obey flow conservation in every vertex other than s and t, meaning that the amount of flow entering the vertex is equal to the amount of flow leaving the vertex. A consequence of this restriction is that the amount of flow that flows trough a cut is the same for every cut. Now, it is easy to conclude that the maximum amount of flow is at most the weight of the minimum cut. More surprisingly, it holds that these quantities are always the same [45]. To show that the solution for the minimum cut problem with weight 12 is optimal, it is sufficient to show that there is a flow of value 12, which is shown in the figure below. Red numbers show how much flow is sent through the edge and the arrow shows in which direction the flow is sent.



Dankwoord

Let me start by thanking the reading committee consisting of Nikhil Bansal, Nicole Megow, Guido Schäfer, Marc Uetz and Tjark Vredeveld. I am honored that my thesis has been approved by these great researchers. Since all other people I wish to thank are Dutch, I will now switch to the Dutch language.

Allereerst ben ik veel dank verschuldigd aan mijn begeleider en copromotor René Sitters. Sinds mijn komst naar de VU in 2012 heeft hij mij begeleid bij mijn onderzoek. Eerst voor mijn masterscriptie en daarna vier jaar als aio, resulterend in dit boek. Samen hebben we mooie artikelen gepubliceerd. Het was fijn om met iemand met zoveel expertise te werken. Ook was het fijn dat we dezelfde fascinatie voor TRP-achtige problemen hebben. Ik hoop ook in de toekomst nog regelmatig samen naar een whiteboard te staren en mooie artikelen te schrijven. Ook ben ik mijn promotor Leen Stougie erg dankbaar. Naast het onderzoek ging zijn aandacht ook naar mijn persoonlijke omstandigheden en mijn toekomstplannen. Leen, bedankt voor al je adviezen. Het was fijn door twee personen met totaal verschillende persoonlijkheden begeleid te worden. Samen hebben jullie mij gevormd tot de wetenschapper die ik vandaag ben.

Verder wil ik al mijn coauteurs bedanken. Speciale dank voor Teun Janssen en Leo van Iersel met wie wij samen aan hoofdstuk 3 hebben gewerkt. Ook wil ik alle mensen bedanken met wie ik samen op conferenties ben geweest. Dank voor al het plezier en alle inspiratie die ik heb opgedaan in met name Pittsburgh (ISMP 2015), La Roche (MAPSP 2015), Seeon-Seebruck (MAPSP 2017) en natuurlijk Lunteren.

Ook binnen de VU zijn er mensen die ik wil bedanken. Ik vond het fijn om te werken op de afdeling Econometrie & OR, ondanks de roerige tijden. Ik wil in het bijzonder Hedda bedanken voor haar ondersteunde diensten en al onze gesprekken. Het was fijn om met iemand buiten de wetenschap te praten om voor een beetje perspectief te zorgen. Verder wil ik mijn kamergenoten en mede-aio's bedanken, in het bijzonder Joost, Suzanne, Annelieke en Thomas.

Ik had natuurlijk nooit een PhD-traject overwogen zonder de inspirerende colleges van mijn UvA-docenten. In het bijzonder wil ik Maurice Koster, Cees Diks, Erik van der Sluis en Sindo Núñez bedanken. Zij hebben mij geholpen bij mijn eerste stappen als docent en hebben mij altijd aangemoedigd een PhD-traject in te gaan, op de UvA of op de VU.

Laat ik eindigen met het bedanken van mijn vrienden en familie. Speciale dank gaat naar Robert en Jeffrey, die tijdens de verdediging mij als paranimfen zullen ondersteunen. Bedankt hiervoor en bedankt voor jullie onvoorwaardelijke steun in het algemeen. Naast Robert en Jeff wil ik graag Thimo, Kundu, Co en Kelly bedanken voor de leuke tijden op het korfbalveld, op de fiets en in de Balu. Ik bedank Pieter voor het zijn van grote broer, die mij liet nadenken over het praktisch nut van mijn werk. Mijn ouders wil ik bedanken voor het vertrouwen en de vrijheid die ze mij altijd hebben

gegeven, want zoals het gezegde luidt: “Een arend kan niet vliegen in een kippenhok”. Verder hebben uitspraken als “Een 7 is ook een mooi cijfer” en “Barcelona is te ver fietsen” mij altijd gemotiveerd om hoge doelen te stellen en die met hard werken te halen.

Als laatste wil ik Fiona bedanken voor het compleet maken van mijn leven. Sinds 2,5 jaar ben ik de gelukkigste man op aarde en hoop nog vele jaren met jou te kunnen zijn. Bedankt voor het aanhoren van mijn verhalen over verdwaalde koeien, Canadese reizigers, lekkende emmers, NP-compleetheid en 2200-approximaties. Ik hou van jou, i.e., $x<3$.

Martijn van Ee
Amsterdam, September 2017

ABRI DISSERTATION SERIES

1. Drees, J.M. (2013). The polycentricity of expansion strategies: Beyond performance as a main driver.
2. Arzlanian, S. (2014). Social networks and firm performance: Examining the relation between dimensions of social capital, social network perception and firm performance.
3. Fleisher, C. (2014). The contemporary career navigator: Individual and organizational outcomes of self-directed career management.
4. Wruck, S. (2014). Warehouse operations revisited - Novel challenges and methods.
5. Volk-Makarewicz, W. (2014). Advances in derivative estimation: Ranked data, quantiles, and options.
6. Van Anholt, R. (2014). Optimizing logistics processes in cash supply chains.
7. Polat, T. (2015). Active aging in work: Motivating employees to continue working after retirement.
8. Ossenkop, C. (2015). What you see is what you get!? Looking into ethnic diversity and professional careers in Dutch organizations.
9. Engel, Y. (2015). Venturing into the unknown, but not for the first time: An examination of firm-founders careers & entrepreneurial decision-making under uncertainty
10. Kolbe, L. (2015). The mindset of the R&D professional: Decision making in innovative context
11. Pachidi, S. (2015). Crunching the numbers: Studying the enactment of analytics in an organization.
12. El Baroudi, S. (2015). Shading a fresh light on proactivity research: Examining when and how proactive behaviors benefit individuals and their employing organizations.
13. Eijdenberg, E. L. (2016). Small business growth in east African least developed countries: Unravelling the role of the small business owners.
14. Lysova, E.I. (2016). What does your career mean to you? Understanding individual career and work behaviors through the prism of the meaning of career.
15. De Mol, E. (2016). Heart and brain: The influence of affective and rational determinants in new venture teams: An empirical examination.
16. Daubner-Siva, D. (2016). Dealing with dualities: A paradox perspective on the relationship between talent management and diversity management.

17. Berkhout, J. (2016). Topics in Markov Chain Theory and Simulation Optimisation.

18. Van Werven, R. (2017). Acquiring resources for a new venture: A study of the micro-level linguistic practices of startup entrepreneurs.

19. Prats Lopez, M. (2017) Managing Citizen Science in the Humanities: The challenge of ensuring quality.

20. Kaandorp, M.S. (2017) Creating from within: A study on interpersonal networking approaches and intuition in task-related interaction.

21. Van Grinsven, M. (2017) A patient is not a car; Lean in healthcare: Studying agency in the translation of management concepts.

22. Muleta Eyana, S. (2017) Entrepreneurial behaviour and firm performance of Ethiopian tour operators.