

Hardness Results for the Shortest Path Problem under Partial Observability

Cenny Wenner

Examensarbete för 15 hp
Institutionen för datavetenskap, Naturvetenskapliga fakulteten, Lunds Universitet

Thesis for a diploma in computer science, 15 ECTS credits
Department of computer science, Faculty of science, Lund University

Abstract

We consider the problem of navigating a weighted random graph with the additional complication that an edge in the realization is concealed until one of its endpoints is visited. This problem and several of its variants have been studied in different fields and under different names, most notably the STOCHASTIC CANADIAN TRAVELLER PROBLEM and the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE. We show that these problems are **PSPACE**-complete, that a generalization of an **NP**-complete variant to Markov chains is **PSPACE**-hard to approximate within any exponential-polynomial factor, and that there are instances to the problems that have no descriptions of optimal policies which can be interpreted in polynomial time unless $\mathbf{PSPACE} \subseteq \mathbf{P/poly}$. These results are complete or partial answers to some open questions and contributes to the more interesting task of exploring the approximability of the problems.

Svårighetsresultat för vägsökningsproblem med begränsad synlighet

Vi presenterar ett flertal resultat relaterade till problemet att navigera i en viktad slumpgraf där kanter förblir dolda tills dess att en kants ändpunkter besöks. Detta problem och flertalet varianter är kända under olika namn i olika områden. Mera specifikt är problemet känt som the STOCHASTIC CANADIAN TRAVELLER PROBLEM (den kanadensiska resarens problem) och the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE (det stokastiska vägsökningsproblemet med omplanering). Vi visar att dessa problem är **PSPACE**-fullständiga, att en **NP**-fullständig variant generaliserad till Markov-kedjor är **PSPACE**-svår att approximera inom varje exponential-polynomial faktor, samt att det inte för alla instanser av problemen finns beskrivningar av optimala planer som kan läsas i polynomial tid om inte $\mathbf{PSPACE} \subseteq \mathbf{P/poly}$. Detta besvarar fullständigt eller delvis vissa öppna frågor och är ett steg närmare den mera intressanta frågan angående om problemet kan effektivt approximeras eller inte.

ACM Subject Descriptors

F.1.2 Modes of Computation – Alternation and nondeterminism, probabilistic computation; F.1.3 Complexity Measures and Classes – Reducibility and completeness, relations among complexity classes; F.2.2 Analysis of Algorithms and Problem Complexity, Nonnumerical Algorithms and Problems – Geometrical problems and computations; G.2.2 Discrete Mathematics, Graph Theory – Graph algorithms, path and circuit problems; and G.3 Probability and Statistics – Probabilistic algorithms, Markov and stochastic processes.

ASM Subject Classification

68Q17 Computational difficulty of problems, 68R10 Graph theory in relation to computer science, 68W20 Randomized algorithms, and 68W25 Approximation algorithms.

General terms

Algorithms, Theory.

Additional keywords

Approximation algorithms, complete problems, CANADIAN TRAVELLER PROBLEM, exploration-exploitation, games against nature, hardness of approximation, inapproximability, path finding and path planning, partial observability, policies and strategies, **PSPACE**, random graphs, randomness in computer science, route planning under uncertainty, satisfiability, shortest paths, STOCHASTIC SHORTEST PATH PROBLEM, QUANTIFIED SATISFIABILITY.

Contents

1	Introduction	1
1.1	A Natural Predicament	3
1.2	Outline	4
2	Background	7
2.1	Sets and Sequences	7
2.2	Graphs	8
2.3	Random Graphs	8
2.4	Computational Problems	10
2.5	Algorithms	10
2.6	Complexity Theory	12
2.7	Complexity Classes	13
2.8	Approximability	14
2.9	PSPACE -hard Satisfiability Problems	15
3	The Stochastic Canadian Traveller Problem	17
3.1	Problem Definition	17
3.2	Two Definitions of i-SSPPR	18
3.3	Previous Work	19
4	SCTP and i-SSPPR are PSPACE-complete	21
4.1	Overview	21
4.2	Proof Idea	22
4.3	Definition of the Reduction	22
4.4	PSPACE -hardness proof	26
5	Inapproximability of Markov-d-SSPPR	33
5.1	Definition of Markov-d-SSPPR	33
5.2	Overview	34

5.3	Reduction from QSAT to Markov-d-SSPPR	35
5.4	Hardness Proof	36
6	Inexistence of Polynomial-Time Policies	41
6.1	Outline	42
6.2	Restricted but Still Hard Problems	42
6.3	Definitions of Some Superinstances	43
6.4	A P/poly Algorithm	45
7	Discussion and Open Problems	49
7.1	Approximation	49
7.2	Natural Simplifications	50
7.3	Computational Hardness	51
7.4	Polynomial-Time Policies	51
7.5	Related Research Directions	53
	Bibliography	a

Acknowledgments

First and foremost, I am greatly indebted to my supervisor, Thore Husfeldt, who has an uncanny ability for inspiring students by making difficult things seem fun and really easy. A few years prior to this study, I was pondering over how one could produce a path-finding algorithm optimal not “only” in the worst case, but also optimal with respect to a probability distribution. At the time, my interest was in Artificial Intelligence and I approached our resident expert in Machine Learning, Jacek Malec, who promptly instead sent me to one of our resident experts in *algorithms*, Thore Husfeldt. Always eager to share “his art”, Thore made it clear that he was interested in the topic and pointed me towards several useful sources for approaching the problem formally. More than anything, I am grateful for this first meeting for awakening in me an interest in a host of topics that I today consider fundamental to my own research and even to my world view.

My thanks to Andrzej Lingas for brief discussions about a number of subjects. My apologies to the same for supervising this thesis for about two weeks before I gathered the courage to tell him that I really wanted to do the thesis under Thore’s supervision, whose area of expertise better suited the kind of results I was looking for.

This thesis has been handed to several persons for proofreading, but I would like to thank in particular Antonios Antoniadis, who I believe is the only one who has read it in detail and given ample of feedback on it. I thank the same for the many discussions and the activities that we have arranged in the past year – programming challenges, open problems nights, and thesis discussions. Confident that I’ve been labelled a nerd since grade school, I can safely share that some of the best times of my life have involved a blackboard, an open problem, and someone to discuss with, preferably a late friday night. Over the years, there’s another handful of people who have managed to endure my ludicrous behavior and provided a much-needed intellectual outlet. Although it is not much, I would like to recognize especially Niklas Citron, Ulf Kargén, Dennis Lindblad, and Mikael Örh.

I would like to thank several persons for taking the time to discuss this thesis in its early stages. Although I’m sure I’m missing a great many, I would like to extend my gratitude to Alex J. Champandard, Levente David, and Leszek Gasienic.

CHAPTER 1

Introduction

Computing optimal paths between two points in a given environment is a central problem in Computer Science with a great number of applications in several fields. Even so, efficient algorithms are today only known for certain classes of idealized environments and a large number of potential applications remain infeasible. Consider the following simplified natural tasks.

Having left a city familiar to yourself for some time, you arrive again with the aim of reaching a desired destination as quickly as possible. However, during the time you have been gone, some paths may have been closed off or your memory might even be playing tricks on you. To devise an optimal plan, you must account for the possibility of such surprises.

In Operations Research, a serious (sub)task is to route a single internet package to a destination while minimizing the time for the message to arrive. The world-wide network of servers defines an ever-changing graph which may very well differ from a broadcaster's current image of it. A policy must therefore be developed that allows the message to be dynamically rerouted as more of the network's present status is revealed.

Aiming to save yourself a couple of minutes driving time, you contemplate taking a shortcut through a forest region. You carry with yourself a map but you're not sure that it is entirely reliable. The road conditions might for instance not be the best and you could even end up having to backtrack to the very location that you started at.

In modern real-time strategy (RTS) games, regions not yet seen by a unit under a player's control are hidden to the player by something called "fog of war." A typical operation in such a game is to navigate, as quickly as possible, a given unit to a chosen location. In modern computer games,

path-finding algorithms are employed to solve such problems. This is done either by revealing the map to the unit for computing a plan or by producing an estimated plan by treating unknown regions as having a predefined cost to traverse. Consider now the task of optimally solving this problem given a probability distribution of what the hidden regions might be.

All of these problems can be phrased as navigation problems in graphs that are only partially known in advance and are revealed while navigating the graph. For the problems to be well-defined, however, one must choose a model for the unknown. Two of the most popular contemporary models in Computer Science are the *worst-case* and the *average-case models*. In the first, we evaluate our strategy by considering its performance in the case for which its performance is the worst. In the second, we assume, fix, or are given a distribution over the possible cases and evaluate the policy's expected performance. In this study, we shall concern ourselves primarily with the second kind. Under such assumptions, we will show that generalizations of the problems mentioned above are so called **PSPACE**-complete, and a host of similar results.

The problem of being given a graph as input, where each edge is associated with a probability of being in the realization independent of what other edges are in the realization is known as the stochastic version of the CANADIAN TRAVELLER PROBLEM (CTP) in Theoretical Computer Science and as the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE (i-SSPPR)¹ in Operations Research, Machine Learning, and Robotics [15, 10, 1]. We'll return to these problems to give a more thorough explanation in the following sections and in Chapter 3.

Even if a problem is not known to have an efficient algorithm today, one may very well be discovered in the future. In Computer Science, one is therefore not only interested in finding efficient algorithms but also to prove whether or not one actually exists, regardless of how we might find one.

SCTP is known to be $\mathbf{P}^{\#\mathbf{P}}$ -hard, which is widely believed to imply that there is no algorithm solving the problem and running in time bounded by a polynomial in the size of the input. On the other hand, the problem is also known to be **PSPACE**-easy and can therefore be solved by an algorithm with a polynomial amount of space and unbounded time. Whether or not the problem is $\#\mathbf{P}$ -easy, **PSPACE**-hard, or something inbetween, was previously not known. We settle this question by showing that the problem is **PSPACE**-hard. This makes it slightly less likely that our problem can be solved efficiently and may contribute to the more interesting unresolved question of whether or not there is an efficient approximation algorithm for the problem.

¹The (STOCHASTIC) SHORTEST PATH (PROBLEM) WITH RECOURSE has in the literature been acronymed SPR, SSPR, SPPR, and i-SSPPR.

Figure 1.1: A problem of partial observability: should we go from ‘s’ to ‘a’ or to ‘b’ in order to minimize our expected travel cost to reach ‘t’?

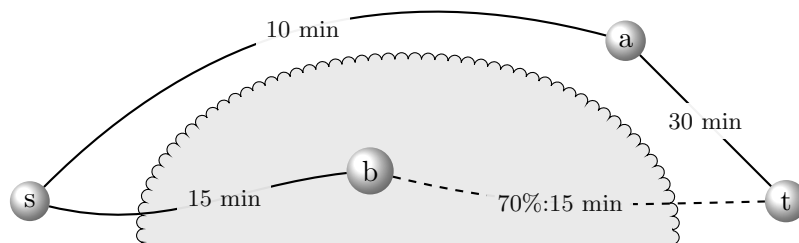
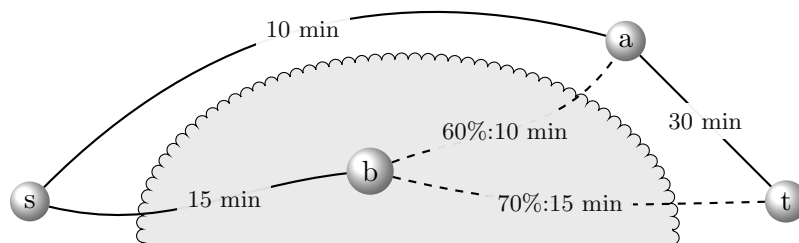


Figure 1.2: A more complicated problem of partial observability. The optimal continuation from ‘a’ depends on the particular walk taken to ‘a’ and the observations made along the way.



1.1 A Natural Predicament

Consider Figure 1.1. A *searcher* wishes to travel from city s to city t separated by a forest. The searcher is presented with a reliable path through a , and a less reliable path through b . The latter path may be more expensive than suspected, or blocked entirely, because, e.g. it is not known where the path leads, the path may occasionally be unavailable, or the road conditions vary from day to day.

We represent the unreliable road here as an edge that has a certain probability of being in the graph. The searcher won’t find out whether the edge is in the graph or not until she reaches one of the endpoints.

If the searcher chooses to take the road through the forest, she will first have to pay 15 minutes to arrive at b . Once at b , she will find out whether or not the edge $\{b, t\}$ is traversable. If the edge is in the graph, she completes her journey by paying another 15 minutes. If the edge is not traversable, the searcher will have to backtrack to s and take the path through a ; paying a total of $15 + 15 + 30 + 10 = 70$ minutes.

For this simple example, we find the solution by comparing the expected costs of the two alternatives. Going through a costs 40 minutes. Going through b costs, in expectation, $15 + 0.7 \cdot 15 + 0.3 \cdot (15 + 30 + 10) = 42$ minutes. Hence we conclude that the shortcut is not beneficial for a one-shot navigation.

Let us make the example slightly more complicated by introducing an unreliable edge between a and b . This road serves as a backup for b – if $\{b, t\}$ fails, then the searcher may, with probability 60%, go through a for a cost of $10 + 30 = 40$; a 15 minute discount over backtracking to the start.

The two examples given above are instances of the stochastic variant of the CANADIAN TRAVELLER PROBLEM (CTP), as well as the undirected variant of i-SSPPR. We will call this the STOCHASTIC CANADIAN TRAVELLER PROBLEM (SCTP) and it will be the chief topic of this study. In the stochastic variant, each edge is given a probability of being in the graph and the event that an edge is in the graph is independent of the events of all other edges.

1.2 Outline

The structure of the remainder of this thesis is as follows. In Chapter 2, we give an introduction to the topics that this thesis builds on. We define some simple relations and the notation that we will use. The last section of this chapter presents QUANTIFIED SATISFIABILITY and STOCHASTIC SATISFIABILITY, two problems that we will frequently refer to. This chapter has been written with the aim that undergraduate students should be able to follow the main ideas of the thesis. Complexity theorists can safely skip everything except the final section of this chapter.

In Chapter 3, we will define the problems that we will be studying and highlight some of the variants of them. In particular, we will argue that there are two different versions of the problem that go under the same name. We also summarize the work that has been done on the problems so far.

The three result chapters, Chapters 4 through 6, are considerably heavier (and more interesting) than the former ones and all follow the same template. We first present what it is that we want to show, some useful definitions, and what is currently known. This is followed by an intuitive idea of the proof, a definition of a reduction, some figures with concrete examples of what the reduction will produce, and finally a proof which involves the defined reduction.

In the first of the results chapters, Chapter 4, we show that SCTP and the two versions of i-SSPPR are **PSPACE**-complete. This proof, like the following ones, merely involves a careful choice of a reduction and extensive application of elementary algebra. The result, as well as the proof, should serve as an essential stepping stone for lower and upper bounds on approximability.

We mention a different variant of i-SSPPR in Chapter 5. This variant is known to be **NP**-hard. We consider the generalization of this problem to Markov chains and show that this problem can not be approximated within any exponential-polynomial function ($2^{n^{O(1)}}$). This should be somewhat bad

news as far as using SSPPR for e.g. routing goes. It may also provide some insight regarding whether or not SCTP is difficult to approximate.

The results chapters are concluded by Chapter 6, where we present a reduction which, under standard assumptions, rules out any descriptions for SCTP which can be interpreted in polynomial time. The idea of the proof in this chapter is to construct an algorithm for a **PSPACE**-hard language L by constructing, for each n , an SCTP-instance of size polynomial in n which contains every instance of L of size n as a subcase of the constructed instance. We believe this is a rather

In Chapter 7, we will reflect on our findings and the numerous questions that still remain unanswered. This chapter should contain many research directions, some of which might even be suitable undergrads.

CHAPTER 2

Background

We assume that the reader is familiar with elementary algorithm theory, complexity theory, and graph theory. We do try to review all of the necessary topics in this chapter, but the reader is strongly encouraged to consult the proper literature for a more thorough coverage. Complexity Theorists may safely skip this chapter at their own discretion.

2.1 Sets and Sequences

With (n) , integer $n \geq 0$, we refer to the set $\{1, \dots, n\}$. This is typically written $[n]$ but as this notation is also used for $\{0, \dots, n - 1\}$, we shall reserve (n) for the former meaning and $[n]$ for the latter.

For a finite set $A = (a_1, \dots, a_n)$, the *Cartesian product* A^k is the set of k -tuples with elements from A . Similarly, $A^{(k)}$ is defined as the set of subsets of A of size precisely k . The *Kleene star* operation on a set A is the union for all k of the Cartesian product, $A^* = \bigcup_k A^k$. If A is an alphabet, then A^* is the set of all strings of the language of finite length. The *powerset* of a set A , denoted $\mathcal{P}(A)$ or 2^A , is the set of all subsets of A .

With $\{f(i)\}_{i \in A}$, for some expression $f(i)$, we refer to the set of values that $f(i)$ attains for some value $i \in A$. We may omit A and assume it implicit, e.g. ranging over the set of values i for which $f(i)$ is defined. $\text{minarg}\{f(x) \mid x \in A\}$ shall be the set of elements of A that attains the smallest value of $f(x)$ amongst all values in A . minarg is also often defined as an arbitrary element in this set rather than the set itself.

In particular, the *empty set* $\{\}$ shall be denoted by \emptyset , the *set of nonnegative integers* $\{0, 1, 2, \dots\}$ by \mathbb{N} , the *set of integers* $\{0, 1, -1, 2, -2, \dots\}$ by \mathbb{Z} , and the *set of rationals* $\mathbb{Q} = \{p/q \mid p, q \in \mathbb{Z}\}$. $[x, y]$ shall denote the set

of reals inclusively from x to y .

2.2 Graphs

A *graph* G is a tuple (V, E) where V is the set of *vertices* and E the set of *edges*. For $G = (V, E)$ we shall let $V(G) = V$ be the vertex set and $E(G) = E$ the edge set. The notation of the edge set should not be confused with the expectation of random variables $E[X]$. The order of a graph, $|G|$, is its number of vertices and the size of a graph, $\|G\|$, is the number of edges it contains. Formally, $|G| = |V(G)|$ and $\|G\| = |E(G)|$. For a directed graph, $E \subseteq V^2$ and for an undirected simple graph, $E \subseteq V^{(2)}$. With $N_G(v)$, we refer to the set of vertices that v has an edge to and is called the *neighbours* of v .

Typical properties of graphs holds up to *isomorphism*. In other words, a typical property holds for G if and only if the property also holds in every graph G' that we get by, for each vertex of G , replacing it with some other object, no object being used to replace two distinct vertices. For this reason, one typically only specifies the order of the graph and allows the vertices to be some arbitrary set of size n , such as $[n]$.

The *empty graph of order n* is the graph on n vertices without any edges. The *empty graph* is the empty graph of order 0, (\emptyset, \emptyset) .

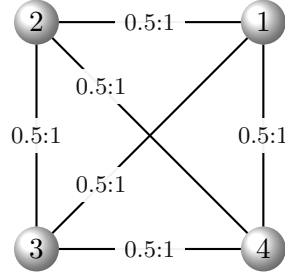
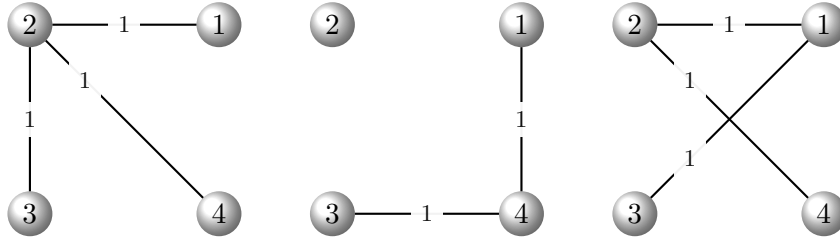
If $G' = (V', E')$ is a subgraph of $G = (V, E)$, written $H \subseteq G$, then $V' \subseteq V$ and $E' \subseteq E$ and all endpoints of E' are in V' . A *vertex-induced subgraph* G' of G is a subgraph which contains every edge of G such that both endpoints are in G' . A *spanning subgraph* is one where $V' = V$. Hence, only G itself is the spanning induced subgraph of G .

We are particularly interested in the family of graphs called paths. An (open) *walk* is a graph that can be written (x_1, \dots, x_k) for some x_1, \dots, x_k such that the vertices of the graph is $\{x_i\}_i$ and the set of edges is $\{x_1, x_2\}, \{x_2, x_3\}, \dots, \{x_{k-1}, x_k\}$. If, additionally, each vertex appears exactly once, then the walk is called an (open) *path*.

A *weighted graph* is a graph with a *weight function* associating edges to, in this study, nonnegative rationals. The weight of an edge, $w(u, v)$, is also called the *cost* of the edge. The cost of a path, $c((x_1, \dots, x_n))$, is the sum of edge weights and, following Karger and Nikolova [10], the *length* of a path is the number of edges it contains. We shall allow edges to have a weight of 0 but will not introduce new edges of this cost in our proofs.

2.3 Random Graphs

A (discrete) *probability space* consists of a countable set Ω , called the outcomes, and a function P , called the probability function. The probability function maps subsets of Ω to reals between 0 and 1 and it must hold that

Figure 2.1: *The random graph $\mathcal{G}_{4,0.5}$.***Figure 2.2:** *Three realizations of the random graph $\mathcal{G}_{4,0.5}$.*

$P(\Omega) = 1$, $P(\Omega \setminus A) = 1 - P(A)$, $0 \leq P(A) \leq 1$ for every $A \subseteq \Omega$, and $P(A + B) = P(A) + P(B)$ for disjoint sets $A, B \subseteq \Omega$.

A *random variable* is a mapping from the outcomes of a probability space to some set, typically the reals. In turn, one may speak one probabilities that the random variable attains a certain value, its mean (expectation), and so on.

A *random graph* of order n is a *probability space* that has as outcomes (also called *realizations*) undirected simple graphs on $(n]$. We shall call $e \in (n]^{(2)}$ for a *probabilistic edge* of a random graph if e has a non-zero probability of being in the realization. With, a *realized edge* or simply *an edge*, we shall refer to an edge in a particular realization. A weighted random graph is also called a *random* or *stochastic network* [1, 17].

A *random graph model* is a parameterized family of random graphs. The most well-known of these is the classical Erdős-Rényi model, $\mathcal{G}_{n,p}$. In the Erdős-Rényi model, each $e \in (n]^{(2)}$ has probability p of being in the graph, independent of all other elements of $(n]^{(2)}$.

An example of a random graph, $\mathcal{G}_{4,0.5}$, is seen in Figure 2.1. Each of the $\binom{4}{2} = 6$ probabilistic edges is independently in the graph with probability 50%. There are $2^{\binom{4}{2}} = 64$ possible realizations, each one equally likely. Three of them are given in Figure 2.2.

In the Bernoulli model, $\mathcal{G}(n, \{p_e\}_{e \in (n]^{(2)}}$, each probabilistic edge $e \in (n]^{(2)}$ has its own associated probability p_e . This is the first of the two random graph models that we will consider as inputs to our problems.

In the second model, the probability of each outcome with non-zero probability must be explicitly given. Note that this model is able to represent any distribution but may have a longer description of a distribution than the Bernoulli model has.

2.4 Computational Problems

A *decision problem* maps the set of inputs, or *instances*, to ‘yes’ or ‘no’, and is completely characterized by the set L of instances for which the answer is ‘yes’. If an instance $I \in \mathcal{I}$ is in L , then I is called a *positive instance* and otherwise a *negative instance*. The set L is called a *language*.

Typically, L is defined a subset of an *input language* where the input language is usually either the set of nonnegative integers, or the set Σ^* for some input alphabet Σ or a subset of the integers. If it is easy enough to distinguish whether $x \in \Sigma^*$ is in \mathcal{I} , then we may without further care call \mathcal{I} the input language. In other cases, that x is in \mathcal{I} is called a *promise*. For the problems we shall study, it is easy to check in polynomial time whether a given input encodes a valid instance and we shall therefore pay no respect to the input language.

A *search problem* (or *function problem*) allows a greater set of outputs than ‘yes’ and ‘no’. For instance, a decision version of the SHORTEST PATH PROBLEM is given an upper bound on the cost of the path and answers whether or not this bound is achievable. A search version of the same problem could instead have the task of outputting a shortest path.

2.5 Algorithms

An *algorithm* A describes operations that a hypothetical machine can execute in order to compute an answer $A(I)$ to an instance $I \in \mathcal{I}$. An algorithm A solves a problem P (decides a language L) iff, for every instance $I \in \mathcal{I}_P$, the algorithm takes some finite number of steps and produces the output $A(I) = P(I)$ (outputs ‘yes’ if $I \in L_P$).

Associated to an algorithm is a number of resource functions, or *complexity measures*, mapping instances to the amount of resources used by the machine in order to compute the answer. Of particular interest is the total time (number of elementary operations) used and the maximum used space (bits stored) at any point. We shall let $T_A(I)$ denote the total units of time used by algorithm A to compute the answer to instance I , called the *time complexity* of A , and $S_A(I)$ the greatest number of bits used at any time by algorithm A to compute the answer to instance I , called the *space complexity* of A .

To assist the analysis of algorithms, one chooses a suitable level of granularity of the functions describing complexity measures. The first typical

assumption is to study an algorithm's *worst-case complexity*. The worst-case complexity f_A of an algorithm A with complexity measure Φ (such as T_A) is defined as $f_A(n) = \max\{\Phi(I) : |I| \leq n\}$. The size $|I|$ of an instance $I \in \mathcal{I}_A$ can be defined, for example, as the number of bits used to encode the instance (preferred for complexity-theoretical results) or the number of vertices of the input graph (preferred for analyzing graph algorithms).

In addition to considering only the greatest amount of resources used by instances of a particular size, one typically restricts high-level studies to the growth of the resources used as a function of the size of the instance. For this, one usually employs *big-oh notation* for upper bounds or *big-omega notation* for lower bounds. We'll give an example of this simplification below.

The last simplification that will be of interest to us is to restrict the question of the efficiency of an algorithm to whether or not the algorithm runs, in the worst case, in polynomial time or polynomial space.

The task of the SHORTEST PATH PROBLEM is to compute the cost of the shortest path(s) in a given weighted graph from a given vertex to a given vertex. This can be phrased as a decision problem by asking whether the cost is less than or equal to a value.

One of the most well-known algorithms for computing the cost of a shortest path is *Dijkstra's algorithm*, paraphrased in Figure 2.3. In the uniform-cost model, we charge one unit of time for each elementary operation. A naïve implementation of the algorithm might have a time complexity of $T(I) = 4 + 6.5|V_I| + 1.5|V_I|^2 + 2|E_I|$. A worst-case complexity of $T(I)$ with respect to the number of vertices is $f(n) = 4 + 5.5n + 2.5n^2$ since $|E_I| = \binom{|V_I|}{2}$ in the worst case. The order of growth can be summarized as $O(n^2)$ and $\Omega(n^2)$. This order of growth is optimal with respect to the number of vertices. With respect to both the number of vertices and the number of edges, the complexity is $O(|V|^2 + |E|)$. By using a Fibonacci heap to implement the 'minarg{distance[u] | u ∈ U}' operation, the complexity drops to $O(|E| + |V| \log |V|)$.

Our study will interest itself with extending the SHORTEST PATH PROBLEM to random graphs where edges are revealed only upon reaching endpoints. In the SHORTEST PATH PROBLEM, one never needs to visit the same vertex twice; if one had, one could produce a path no longer by removing a subpath that starts and ends at such a vertex. In the partial observability case, an optimal strategy may involve visiting the same vertex more than once, assuming each visit has a different set of revealed edges. For this reason, the best strategy for going from s to t in a partially observable graph is not necessarily described by a path, but by a walk. Nevertheless, following the contemporary naming convention we shall not stress this distinction.

Figure 2.3: *Dijkstra's algorithm for computing the cost of shortest paths.*

```

DIJKSTRA-SHORTEST-PATH( $G, w, s, t$ )
1  for each  $v$  in  $V[G]$ 
2  do  $distance[v] \leftarrow \infty$ 
3   $distance[s] \leftarrow 0$ 
4   $U \leftarrow V(G)$ 
5  while  $|U| > 0$ 
6  do  $v \leftarrow \text{minarg}\{distance[u] \mid u \in U\}$ 
7      $U \leftarrow U - \{v\}$ 
8     for each  $u$  in  $N_G(v)$ 
9     do  $distance[u] \leftarrow \min(distance[u], distance[v] + w(v, u))$ 
10 return  $distance[t]$ 

```

2.6 Complexity Theory

In *Complexity Theory*, problems are categorized by their difficulty. A (*complexity*) *class* is a set of problems and, more specifically, usually sets of decision problems. Of particular importance is the class \mathbf{P} , the set of decision problems that can be solved in polynomial time by a *Turing machine* or a modern personal computer with unbounded memory.

The main tool for showing that a problem is easy or difficult is the *reduction*. A reduction of a problem P to a problem Q substitutes the task of solving an instance of P with the task of solving one or more instances of Q . If Q can be solved efficiently, then so can P . Likewise, if P is known to be difficult to solve and P is reducible to Q , then Q must also be difficult to solve.

A problem P is polynomial-time *Karp-reducible*, also called polynomial-time *many-one* reducible, to a problem Q if there exists a polynomial-time algorithm A which given an instance $I \in \mathcal{I}_P$ produces an instance $A(I) \in \mathcal{I}_Q$ such that the answer to the instance $A(I)$ is the same as the answer to Q . In other words, $Q(A(I)) = P(I)$ for every $I \in \mathcal{I}_P$ for some polynomial-time algorithm A . That P is many-one-reducible to Q is written $P \leq_m Q$. A class \mathcal{C} is *closed under reduction* \leq_R if that $Q \in \mathcal{C}$ and $P \leq_R Q$ implies $P \in \mathcal{C}$.

A polynomial-time *Turing-reduction*, also called a *one-to-one* reduction, from a problem P to a problem Q is a polynomial-time algorithm solving Q with access to a special operation that answers instances to P in constant time.

A search problem P is *self-reducible* if there is a polynomial-time Turing-reduction to P from the language of pairs of input and output of P . In other words, if P is self-reducible, then we can reduce the task of finding an acceptable output for an instance of P to the problem of deciding whether

a given output would be acceptable. The task of computing the cost of a shortest path, and the cost problems of our study, are self-reducible: the cost can be written as a fraction where the numerator and denominator are bounded by a polynomial times 2^N where N is the size of the input. We can for this reason find the cost of the shortest path with a binary search in polynomial time using the decision version: “is the cost less than x ?”

2.7 Complexity Classes

The class **NP** is the set of problems that can be solved in polynomial time by a machine that, in addition to the regular operations, in each step can “guess” a bit 0 or 1 and outputs ‘yes’ iff at least one assignment of the guesses makes the algorithm accept the instance. Equivalently, it is the set of problems that can be solved by first guessing a “proof” (guessing a *witness*) of polynomial length that the instance is positive and then verify the proof in polynomial time. More formally,

$$\mathbf{NP} = \{L \mid \exists_k \exists_{R \in \mathbf{P}} (x \in L \text{ iff } \exists_y |y| \leq |x|^k \text{ and } (x, y) \in R)\}.$$

A problem L is called \mathcal{C} -easy if the problem is in the class \mathcal{C} . If a problem L is \mathcal{C} -hard with respect to the reduction f , then every problem in \mathcal{C} is f -reducible to L , i.e. every problem in \mathcal{C} is a special case of L . A problem that is both easy and hard with respect to \mathcal{C} is called \mathcal{C} -complete. These are the “hardest” problems in \mathcal{C} since they are at least as hard as any other problem in \mathcal{C} .

A canonical **NP**-complete problem is the SATISFIABILITY problem (SAT). The Satisfiability Problem asks us to determine whether or not a set of variables can be set to ‘true’ or ‘false’ to make a given *quantifier-free boolean formula* evaluate to true. The formula is given in *conjunctive normal form*, i.e. the formula is a conjunction (“all must be true”) of disjunctive clauses (“at least one must be true”) of variables or the negation of variables. A variable that appears in a clause and which may be negated will be called a *literal*. The following boolean formula would be a positive (*true*) instance of this problem. Here, ‘ \exists ’ stands for choosing a value, ‘ \wedge ’ stands for “and”, ‘ \vee ’ for “or”, ‘ \neg ’ for “not”, and ‘ x_i ’ is a variable.

$$\exists_{x_1} \exists_{x_2} \exists_{x_3} (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge \neg x_2$$

A proof that this formula is positive would be to set x_3 to *true* and x_1 and x_2 to *false*. The proof can be verified in polynomial time by inserting and evaluating these values. We will present a generalization of this problem later in this chapter. This generalization shall play a significant role in this thesis.

The *complement class* **coC** of a class \mathcal{C} is the set of all languages L such that the complement of L is in \mathcal{C} . In other words, the positive and the

negative instances are interchanged. The complement of \mathbf{P} is \mathbf{P} itself since a polynomial-time algorithm can simply exchange the answers ‘yes’ and ‘no’. The same argument does not work for \mathbf{NP} as we in the definition said that the answer is ‘yes’ iff some assignment of the guesses produces a ‘yes’ answer. At the time of writing, it is not known whether or not $\mathbf{NP} = \mathbf{coNP}$. A canonical problem for \mathbf{coNP} is the Tautology Problem (also called $\forall\text{-SAT}_1$): given a boolean formula in prenex normal form with only universal quantifiers, does the expression evaluate to *true*? With a universal quantifier, we mean that every choice of the variable is tested and the formula is satisfiable if and only if every choice satisfies the formula.

PSPACE (*polynomial space*) is the set of problems that can be solved by a Turing machine restricted to a polynomial amount of memory and unbounded time. It is easy to show that the most amount of time needed is $2^{p(n)}$ if the space is bounded by the polynomial $p(n)$.

$\#\mathbf{P}$ (*counting polynomial time*), pronounced sharp-P, is the set of problems for which there is a nondeterministic Turing machine such that the number of accepting paths of the guesses equals the answer of the problem instance. Equivalently, if there is a k and $R \in \mathbf{P}$ such that the answer to our question equals the number of $y, |y| \leq n^k$, for which $(x, y) \in R$. $\#\mathbf{P}$ is a class of *function problems* or *search problems*. In other words, we do not restrict the output to ‘yes’ or ‘no’ but may instead demand answers such as a specific number or a graph with a certain property. To recast $\#\mathbf{P}$ as a class of decision problems, it can serve as a so called oracle for a polynomial time machine. This class is called $\mathbf{P}^{\#\mathbf{P}}$ and is a subset of **PSPACE**. One of the great results of computer science is that the entire so called polynomial hierarchy is contained in this class even if we’re only allowed to ask this oracle a single question!

The class $\mathbf{P/poly}$ (*polynomial time with polynomial advice*) is the set of problems that can be solved if you have a (possibly) different polynomial-time algorithm for every size of the input. This gives the model the power to even solve some otherwise unsolvable problems. Owing to the polynomially-bounded resources however, it is not even known whether this class contains \mathbf{NP} .

2.8 Approximability

If we know that we most likely won’t find an efficient enough algorithm that exactly solves one of our problems, we might instead turn our attention to algorithms that *approximately* solve the problem. In this study, we will consider so called ρ -approximation for minimization problems. We shall call an algorithm for an $f(n)$ -*approximation algorithm* if the algorithm runs in polynomial time and always produces a solution that costs at most $f(n)$ times the minimum cost among all solutions.

In addition to developing practical approximation algorithms, computer scientists are interested. For instance, having found a 3-approximation algorithm, is there also a 2-approximation or have we done as well as we can? Results that show that there are no approximation algorithms typically do so by showing that if the said algorithm existed, then it could be used for solving hard problems exactly. For instance, if we had a 2-approximation algorithm for the TRAVELING SALESPERSON PROBLEM, then we could solve HAMILTONIAN CYCLE by taking an arbitrary instance to HAMILTONIAN CYCLE and reducing it to an instance of TSP by making each edge in the graph cost one and add the missing edges with cost $n + 2$. If the input graph had a Hamiltonian cycle, then the cheapest TSP of the new graph would be n . A 2-approximation algorithm is guaranteed to return a solution within 2 times the optimal. However, using any of the edges that did not exist in the HAMILTONIAN CYCLE instance would produce a solution of cost at least $2n + 1$. We can therefore decide the existence of a Hamiltonian cycle solely on the answer of the approximation algorithm. A similar argument can be made for every constant strictly greater than zero which means that TSP cannot be approximated within any constant (greater than 1) unless $\mathbf{P} = \mathbf{NP}$.

2.9 PSPACE-hard Satisfiability Problems

We will now define some satisfiability problems that will be used in our reductions. The reductions will primarily be from QUANTIFIED SATISFIABILITY (QSAT) but may most naturally be seen as reductions from a variant of the STOCHASTIC SATISFIABILITY (SSAT) problem.

Informally, instances to QUANTIFIED SATISFIABILITY (QSAT), also known as TRUE QUANTIFIED BOOLEAN FORMULA (TQBF), are boolean formulas in conjunctive normal form over some number of variables. The variables can be seen as chosen by two competing agents. The first agent, call him *Existential*, tries to make the formula true and the second, *Universal*, tries to make the formula false. The variables are numbered and processed from the first to the last. For odd numbers, Existential chooses whether the variable is set to true or false, and for even numbers, Universal makes the decision. The two agents are aware of assignments the other agent has done before them. The QSAT instance has the answer ‘yes’ if and only if Existential has a strategy for setting the variables such that no matter how Universal chooses his variables, the boolean formula is satisfied. We call Existential’s choices for *existential quantifiers* (\exists) and call Universal’s for *universal quantifiers* (\forall).

Definition 2.1 (EXACTLY-3 QUANTIFIED SATISFIABILITY (E3QSAT)). An instance to QSAT consists of (in unary) n , the number of variables, m , the number of clauses, and $6m$ integers defining the clauses. We shall assume

that these $6m$ clauses are six-tuples $(c_{i,1}, c_{i,2}, c_{i,3}, s_{i,1}, s_{i,2}, s_{3,i})$ where $c_{i,j}$ denotes the index of the j :th literal in the i :th clause and $s_{i,j}$ is ‘0’ if the literal is positive and ‘1’ if it is negative. For instance, $(3, 1, 4, 0, 1, 1)$ could represent $\neg x_3 \vee x_1 \vee \neg x_4$, where ‘ \neg ’ stand for negation. As usual, the odd-indexed variables shall be chosen by existential quantifiers and the even-indexed by universal quantifiers.

OUTPUT: ‘accept’ if the encoded formula evaluates as true, and otherwise ‘reject’.

Without loss of generality, we shall assume that n is even for this and the following definitions of this section.

Fact 1. *E3QSAT is PSPACE-complete.*

Definition 2.2 (EXACTLY-3 STOCHASTIC SATISFIABILITY (E3SSAT)). This problem is similar to QSAT but instead of having universal quantifiers, where a “worst” choice is made, randomized quantifiers are used, where a random truth value is selected.

The task is to compute whether the *probability* that the formula is satisfiable is at least $1/2$. Note that probability is 1 if and only if the corresponding QSAT instance is satisfiable.

INSTANCE: the same as for QSAT but here even-indexed variables are chosen at random.

OUTPUT: ‘accept’ if the encoded formula evaluates as true, and otherwise ‘reject’.

Definition 2.3 (E3SSAT’). Exactly-3 SSAT’ [14] is a variant of E3SSAT. All quantifiers are existential but every second quantifier has *true* and *false* independently “disabled” with respective probability $1/2$ and cannot be chosen. If both choices are disabled, which happens with probability $1/4$, then the realized formula is unsatisfiable.

We see that an E3SSAT’ instance with n quantifiers has a satisfiability probability of at most $(3/4)^{\lfloor n/2 \rfloor}$, and the probability is $(3/4)^{\lfloor n/2 \rfloor}$ precisely when the corresponding QSAT instance is satisfiable.

In the original definition, an instance is positive if and only if the probability of a satisfying assignment is greater than $1/2$. We shall concern ourselves only with the question of whether or not the probability of a satisfying assignment is $(3/4)^{\lfloor n/2 \rfloor}$ which, because of the comment of the previous paragraph, corresponds to deciding whether QSAT instances are satisfiable and hence the problem is **PSPACE-hard**.

INSTANCE: the same as QSAT but now even-indexed variables are existential quantifiers where the choices 0 and 1 have independently been disabled with probability 0.5. If, for any variable, 0 and 1 have both been disabled, then the instance is treated as ‘not satisfied.’

The Stochastic Canadian Traveller Problem

In this chapter, we define the STOCHASTIC CANADIAN TRAVELLER PROBLEM, compare the two different definitions of the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE, and summarize the results known for these problems and the techniques that we will use.

3.1 Problem Definition

We define below the stochastic variant of the CANADIAN TRAVELLER PROBLEM, which is also one of the definitions of the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE on undirected graphs.

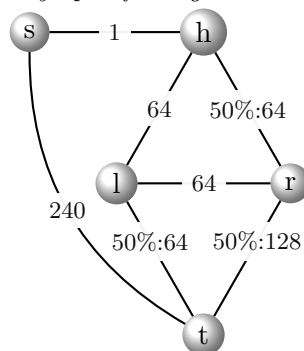
Definition 3.1 (STOCHASTIC CANADIAN TRAVELLER PROBLEM (SCTP)). An instance $I = (G, c, p, s, t)$ to SCTP consists of an undirected (random) graph $G = (V, V^{(2)})$ with edge costs $c : V^{(2)} \mapsto \mathbb{Q}$, edge probabilities $p : V^{(2)} \mapsto \mathbb{Q} \cap [0, 1]$, a source vertex $s \in V$, and a target vertex $v \in V$.

A policy, or strategy, to navigate the instance maps a location and observations made about which edges are in the realization to the vertex to go to next. Formally, $\pi : V \times \mathcal{P}(V^{(2)}) \times \mathcal{P}(V^{(2)}) \mapsto V$. The probability of observing an edge $\{v, w\}$ when arriving at a vertex v is 0 if v or w has been visited before but $\{v, w\}$ has not been observed, 1 if the edge has been observed, and otherwise $p_{\{v,w\}}$.

For an instance I and policy π , call the random path taken in instance I for $\mathbf{X}^\pi = (X_0^\pi, \dots, X_{T-1}^\pi, X_T^\pi, X_T^\pi, \dots)$ where T is the first time the target t is reached. To simplify analysis, we shall assume $T \in \mathbb{N}$ and that the path visits t indefinitely from time T and onwards for a cost of 0 per transition.

The cost of a policy is given by $c(\pi, I) = E[c(\mathbf{X}^\pi)] = E[w(X_0^\pi, X_1^\pi) + w(X_1^\pi, X_2^\pi) + \dots + w(X_{T-1}^\pi, X_T^\pi)]$ where $w(u, v)$ is the cost of the edge $\{u, v\}$.

Figure 3.1: An SCTP instance with no optimal policy from s to t that follows a tentative path and reroutes only upon finding that the path fails.



Let $\Pi_I^* = \{\pi^* \mid c(\pi^*, I) = \max_{\pi} c(\pi, I)\}$ be the set of optimal policies for I . The task of the STOCHASTIC CANADIAN TRAVELLER PROBLEM is to compute the expected cost of the optimal policies. The complexity of computing a description of an optimal policy will be addressed in Chapter 6.

With $(x_0, \dots, x_k)[i : j], 0 \leq i \leq j \leq k$, we shall refer to the subpath (x_i, \dots, x_j) . In particular, we have $\mathbf{X}^\pi = \mathbf{X}^\pi[0 : \infty]$.

3.2 Two Definitions of i-SSPPR

In the literature, there are two different definitions of the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE. This is probably because i-SSPPR is a generalization of the STOCHASTIC SHORTEST PATH PROBLEM and because of the techniques that were used for stochastic problems at the time the problem was introduced [1].

In the first definition of i-SSPPR [1], the searcher is required to choose a tentative probabilistic path and follow it until either the target is reached or one of the probabilistic edges of the path fails. Only upon failure of the tentative path is the searcher allowed to choose a new path. We assume that a probabilistic edge that fails should have had a prior non-zero probability. The other definition, used e.g. by Polychronopoulos and Tsitsiklis [17], allows the searcher to take into account new information at any point of the traversal. This latter definition agrees with the definition of (directed) SCTP.

Figure 3.1 proves that the two definitions of i-SSPPR are not equivalent. The optimal SCTP policy of this graph is to go from s to h , and continue to l if $\{h, r\}$ is in the graph and otherwise backtrack to s . The expected cost of this policy is $1 + 0.5(1 + 240) + 0.5 \cdot (64 + 64 + 0.25(128 + 64 + 1 + 240)) = 240 - 3/8$. A policy that may only reroute upon failure of a tentative path has a minimum cost of 240.

The remaining of our proofs will be given for the SCTP definition, but our results apply also for the path-rerouting definition. To see this, consider any of our proofs and

relatively cheap edges with relatively small probability to the vertices where we want decisions to be made. The failure of these edges guarantees that the choices can be made. For approximability and competitive-ratio analysis, one may have to pay greater care to respect this distinction.

Owing to the following lemma, we shall in this study not concern ourselves with whether or not the optimal policies we consider are of polynomial length.

Lemma 3.1. *There is a polynomial p such that for every SCTP instance I with edges of cost strictly greater than 0 and optimal policy of I , the target is reached within $p(|I|)$ steps. For instances with edges of cost 0 or greater, there is at least one optimal policy with this restriction.*

Proof. Consider the sequence in which vertices are visited for the first time. This sequence is of length at most $|V|$. Between two consecutive new vertices, a number of already visited vertices may be visited but no new observations are made. In particular, there is an optimal path from the first to the second vertex containing each vertex at most once. If each edge has a cost strictly greater than 0, then every optimal path contains every vertex at most once. Hence, the target can be reached within $|V|^3$ steps for an optimal policy. \square

3.3 Previous Work

Although we do not claim that CTP and SSPPR identifies core issues of computation like for instance satisfiability does, CTP and SSPPR are natural optimization problems that often surface in practice. This is hinted by the number of definitions and names that the problem is known as in different fields. An overview of some of the different versions of SSPPR is presented in and Provan [18]. Provan [18] also presents an efficient dynamic programming algorithm for the STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE AND RESETS (SSPPRR) (also called the stochastic version of the RECOVERABLE CANADIAN TRAVELLER PROBLEM or the STOCHASTIC CANADIAN TRAVELLER PROBLEM AND RESAMPLING). In this problem, the graph is resampled from the defined distribution after each of the searcher's steps. Owing to the memoryless property of this problem, one can reduce the problem to a Markov Decision Process with polynomial horizon and thereby show that SSPPRR is in \mathbf{P} since such MDP's can be solved in polynomial time with linear programming [19]. Such a reduction was done and empirically tested in Briggs, Scharstein, and Abbott [5] and [6], where SSPPRR goes under the name EXPECTED SHORTEST PATH. Bar-Noy and

Schieber [2] gave a particularly efficient algorithm for this problem in the, natural, special case where every vertex has a self-loop with probability 1 and no more expensive than any other edge adjacent to that vertex. In other words, instances where you can always “wait at a vertex” and “it is cheaper to wait than to go to another vertex.” Karger and Nikolova [10] [13] notes that, even without resampling, one may solve SSPPR in polynomial time for instance where each vertex is visited at most once and instances that consist of vertex-disjoint paths. The earliest stab at exact algorithms for i-SSPPR was, arguably, taken by Polychronopoulos and Tsitsiklis [17].

The most well-studied problem for path finding in random graphs is the STOCHASTIC SHORTEST PATH PROBLEM. In STOCHASTIC SHORTEST PATH PROBLEM, you must select a path to follow in a graph where each edge cost is a random variable. The goal can be for instance to minimize the expected cost or minimizing the probability that the cost is above some threshold [4]. i-SSPPR differs from this problem in that you are allowed to update the plan as you acquire information about the state of the graph.

In Operations Research, i-SSPPR was first introduced by Andreatta and Romeo [1] and subsequently studied in Bertsekas and Tsitsiklis [3]. In Theoretical Computer Science, the problem was defined by Papadimitriou and Yannakakis [15]. Computational hardness results for the problem and variants have primarily been pioneered by Papadimitriou and Yannakakis [15], Bar-Noy and Schieber [2].

The topic which probably has been awarded the most attention for SCTP has been the analysis of bounds on the competitive ratios for polynomial-time and unbounded policies. In the competitive ratio variant of the problem, one wishes to minimize the ratio of the cost of an (online) policy, which traverses the graph without (initially) knowing the realization, to the cost of an (offline) policy, which knows the realization in advance. A constant-factor approximation algorithm for such a problem would ensure that you’re always at most some c times worse than the optimal online policy, which in turn is some unknown number of times worse than the offline policy. The work done on SCTP in relation to this has been to explore parametrized upper and lower bounds ensuring the existence or inexistence of an online policy that is always at most some k times worse than the offline policy. Two very recent papers on this topic would be Westphal [20] and Xu, Hu, Su, Zhu, and Zhu [21].

CHAPTER 4

SCTP and i-SSPPR are PSPACE-complete

The competitive-ratio version of the CANADIAN TRAVELLER PROBLEM was proven PSPACE-complete in Papadimitriou and Yannakakis [15]. In fact, it was noted that it is, within every constant, PSPACE-hard even to approximate CTP restricted to edges of cost 1 (CARDINALITY CTP). In the same paper, proofs were given that SCTP and, implicitly, i-SSPPR are #P-hard.

No progress has been made on the computational hardness of the original problems since this publication. We present a logarithmic-space Karp-reduction from QSAT to SCTP such that the QSAT instance is satisfiable if and only if the expected cost of the SCTP instance is less than a particular value. A similar proof can be constructed for e.g. MAX-CLAUSE SSAT. The reduction is most easily seen as a reduction from QSAT to E3SSAT' (defined in Section 2.9) which, in turn, is reduced to SCTP. If the QSAT instance is satisfiable, then the “equivalent” E3SSAT' instance is satisfiable with probability $(3/4)^{\lfloor n/2 \rfloor}$, and if the QSAT instance is not, then the probability is at most $2^{-n}(3^{\lfloor n/2 \rfloor} - 1)$. We shall write $E3SSAT'(x)$ for an instance x to denote the probability that x is satisfiable.

4.1 Overview

In the next section, we give an intuitive idea about how the proof works. We define the reduction R in Section 4.3 and show that it can be done in logarithmic space and that the resulting instance is of polynomial size. Section 4.4 contains the primary parts of the proof, which we present in the form of a series of lemmas leading up to the main theorem.

4.2 Proof Idea

See Figure 4.1 for the reduction graph for the \forall -QSAT instance $\forall_{x_1} \exists_{x_2} \forall_{x_3} (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$. This instance can be turned into a \exists -QSAT instance – which is the kind we have discussed up to now – by adding a dummy quantifier. The graph has been split into five regions: A, B, C, D, and E. In our explanations, we shall refer to this graph in order to make the arguments easier to follow.

The idea behind the reduction is to construct a graph where *the searcher* will have to choose a truth assignment by, for each variable of the QSAT instance, taking one of three roughly equal paths (from x_i to 0, 1, or 2, where 0 denotes ‘false’, 1 denotes ‘true’, and 2 causes unsatisfiability). Each path will force the searcher to go through those vertices associated with the clauses containing that particular assignment of the variable (two vertices labelled ‘ \bar{x}_i ’, ‘ x_i ’, or two unlabelled). For each clause, there will be an edge leading to the target vertex with probability $7/8$ (edges labelled ‘ $p:\alpha$ ’). Whilst the searcher assigns the variables, such an edge will be too expensive to follow, but if a certain event occurs, then it will become desirable to use these edges (at d).

After each clause has been visited, the searcher will have a choice between going directly to the target, using a slightly more expensive edge (d to t), or going to the target through one of the clauses, using one of the above-mentioned edges (d to r).

The decision is constructed in such a way that it will only be beneficial to take the path through the clauses (d to r) if each clause is known to have an edge to the target. This in turn requires that each clause has been visited or, in other words, that an assignment satisfying the QSAT formula has been chosen.

The bound on the expected cost is chosen such that the bound is achievable if and only if the the clause alternative can be chosen $(3/4)^{-n/2}$ of the time, which is possible if and only if the QSAT instance is satisfiable.

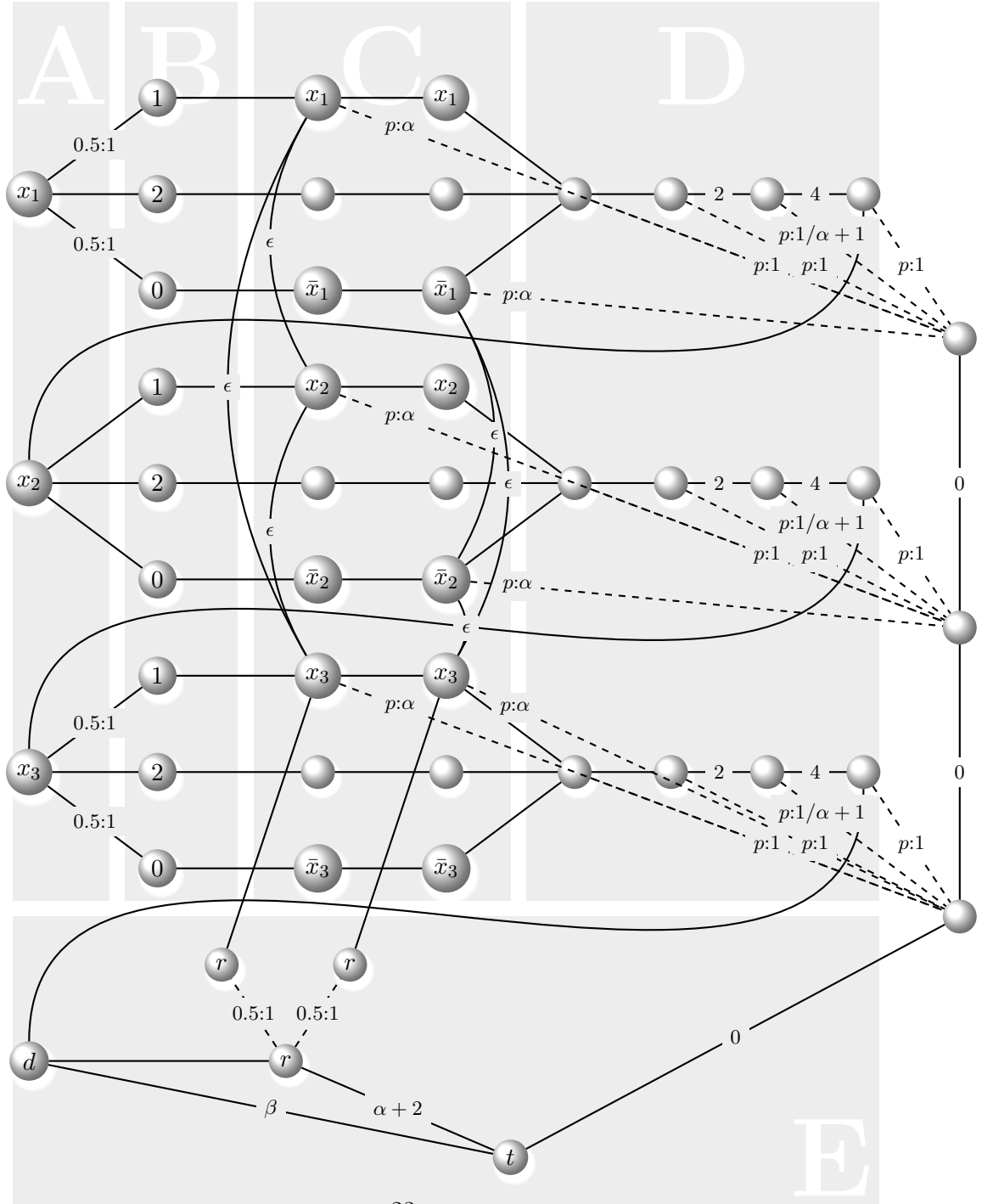
To devise this setup, we will rely on probabilistic cheap edges to the target (labelled ‘ $p:1$ ’). Such edges can force a searcher to take an expensive edge to a desired location. If the cheap edge isn’t in the graph, the searcher will have to continue with the best policy from the new location which, here, will not involve following the expensive edge back to the previous vertex.

4.3 Definition of the Reduction

We define a reduction R from E3SSAT’ to SCTP. For pragmatical reasons, the reduction is only presented with a high-level description.

Given an QSAT instance I , we first add the clauses $(x_1 \vee \neg x_1) \wedge \dots \wedge (x_n \vee \neg x_n)$, which does not change the satisfiability of the instance. Let the result-

Figure 4.1: Reduction graph for the QSAT instance $\forall_{x_1} \exists_{x_2} \forall_{x_3} (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$. Edge costs of 1 and edge probabilities of 1 have been omitted. Edge labels denote **probability:cost**. ϵ is $2^{-4m-7}(m+1)^{-6} = 1/23887872$, p is $1 - 2^{-m-2}(m+1)^{-2} = 1 - 1/144$, α is $(m+1)^2 = 9$, and $\beta = \alpha + 3 + 3\epsilon = (m+1)^2 + 3 + 3 \cdot 2^{-4m-7}(m+1)^{-6} = 12 + 1/17915904$. Additional vertices and edges of cost 0 added for clarity.



ing instance, I' , be $(n, m, c_{1,1}, c_{2,1}, c_{3,1}, c_{1,2}, \dots, c_{3,m}, s_{1,1}, s_{2,1}, s_{3,1}, s_{1,2}, \dots, s_{3,m})$. Call the constructed instance $R(I)$ to SCTP for (G, s, t, c, p, B) . To recapitulate, n is the number of variables, m the number of clauses, $c_{1,1}$ through $c_{3,m}$ describes the variables of the clauses and $s_{1,1}$ through $s_{3,m}$ describes what variables are negated. G is the graph we build, s the start vertex, t the target vertex, c maps edges to their costs, p is a function mapping an edge to its probability, and B is the bound on the expected cost.

Let $\alpha = (m+1)^2$, $h = \lceil \log_2(16 + 4\alpha) \rceil$, $p = 1 - 2^{-m-2}/\alpha$, $\bar{p} = 1 - p$, $\epsilon = 2\bar{p}^4\alpha = 2^{-4m+7}(m+1)^{-6}$, $\delta = \epsilon\bar{p} + 2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3$, and $\beta = \alpha + 3 + 3\epsilon$.

Vertices of G

The vertices of the undirected graph G are

- the “decision vertex” d , the clause choice vertex r , and the target vertex t (region E),
- r_j , clause alternative number $1 \leq j \leq m$ (region E),
- for each $1 \leq i \leq n$:
 - v_i , representing x_i , setting $s = v_1$ (region A),
 - $v_i^{(1)}$, $v_i^{(0)}$, and $v_i^{(2)}$, representing, respectively, setting x_i to true, false, and neither (region B),
 - $u_{i,j,y}$ for each $1 \leq j \leq m$ and $y \in \{0, 1, 2\}$ (region C),
 - and $H_{i,1}, \dots, H_{i,h}$, where $h = \lceil \log_2(16 + 4(m+1)^2) \rceil$ (region D),

We let $U_{k,j}$ stand for the vertex representing the k th literal of the j th clause, $u_{c_{k,j},j,s_{k,j}}$.

Edges of G , and associated costs c and probabilities p

Edges with cost 1 and probability 1 connect, where i ranges from 1 to n and y ranges from 0 to 2,

- if i is odd, v_i with $v_i^{(0)}$, $v_i^{(1)}$, and $v_i^{(2)}$ (region A to region B),
- $v_i^{(y)}$ with $u_{1,1,y}$ (region B to C),
- $u_{1,j,y}$ with $u_{1,j+1,y}$ for each $1 \leq j < m$ (region C to C),
- $u_{1,m,y}$ with H_i (region C to D),
- r_j with $U_{3,j}$ for every $1 \leq j \leq m$ (region D to A),
- $H_{h,i}$ with v_{i+1} unless $i = n$, and $H_{h,n}$ with d (region D to A and E).

In region E, add an edge from d to r with cost 1, an edge from r to t with cost $\alpha + 3 = (m+1)^2 + 3$, and an edge from d to t of cost $\beta = \alpha + 3 + 3\epsilon = \alpha + 3 + \epsilon\bar{p} + 1.5 \cdot 2^{-m}\bar{p}^3$. Connect r to r_j with cost 1 and probability 0.5 for each $1 \leq j \leq m$.

For $1 \leq i \leq n$,

- if i is even, then the variable-vertex v_i is connected to $v_i^{(0)}$, $v_i^{(1)}$, and $v_i^{(2)}$ with an edge of cost 1 and probability 0.5 (region A to B),
- for $1 \leq j < h$, $H_{i,j}$ is connected to $H_{i,j+1}$ with an edge of cost 2^j , and to t with cost 1 and probability $p = 1 - 2^{-m-2}(m+1)^{-2}$ (region D to D),
- and $H_{i,3}$ is connected to t with an edge of cost $\alpha + 1$, only to simplify the proof (region D to E).

For every clause $j \in (m]$,

- $U_{1,j}$, $U_{2,j}$, and $U_{3,j}$ are connected to one another with edges of cost $\epsilon = 2^{-m-1}\bar{p}^3 = 2^{-4m-7}(m+1)^{-6}$ (region C to C),
- t is connected to $U_{1,j}$, $U_{2,j}$, and $U_{3,j}$ with edges of cost $\alpha = (m+1)^2$ and probability 0.5 (region C to E),
- and finally is r connected to $U_{1,j}$ with cost 1 and probability 0.5 (region C to E).

Bound B

Let $q = \bar{p}^{nh}$ and $\theta = q + \frac{1-q}{p} \left(m + 2 + \frac{(1-2\bar{p})^h}{1-2\bar{p}} \right)$. We will, in Section 4.4, prove that q is the probability for an optimal policy to reach d before t , and that θ is the expected cost of the part of the taken path that does not pass d . The bound B is defined as $\theta + q(\alpha + 3 + 3\epsilon - (3/4)^{n/2}(1 - 2^{-m})\epsilon)$.

Lemma 4.1. *The function R can be computed in logarithmic space.*

Proof. Our description utilizes no more than a constant number of indexes, each one bounded linearly by n . To this comes a few implicit indexes and constant memory aside from values. All values that are used can be written as a rational numbers with absolute values of numerator and denominator bounded exponential in n . In fact, besides β and edges of cost powers of 2, the bound is polynomial. For β , the subtraction of 2^{-4m-1} can be done in logarithmic space and likewise can 2^j be written out with space logarithmic in j . All in all, this demands no more than logarithmic space. \square

Lemma 4.2. *$|R(I)|$ is $O(|I|^k)$ for some k .*

Proof. The reduction $R(x)$ only produces a polynomial number of vertices and, as said above, costs and probabilities, and the bound B can be written as fractions of exponentially bounded values. Since $|I|$ is $\Omega(n)$, the lemma follows. \square

4.4 PSPACE-hardness proof

We shall prove that every optimal policy visits the *decision vertex* d at time $\tau_d = (m + h + 2)n$ unless the target vertex is reached before this time. More specifically, we shall show that every optimal policy chooses one of $v_i^{(0)}$ and $v_i^{(1)}$ from v_i , and follows the path of length $m + h + 1$ to $H_{i,h}$, and then goes to v_{i+1} . The exceptions to this being when $i = n$, for which $H_{n,h}$ is followed by d instead, when an edge of cost 1 to t is available, and when, for some i , neither of $v_i^{(0)}$ and $v_i^{(1)}$ are neighbors of v_i , in which case the edge from v_i for the smallest such i is taken to $H_{i,1}$.

We will now show that every optimal policy follows, in instance I , the path

$$\begin{aligned} \mathbf{z}_{\mathbf{y},I} = & (v_1, v_1^{(y_1)}, u_{1,1,y_1}, u_{1,2,y_1}, \dots, u_{1,m,y_1}, H_{1,1}, \dots, H_{1,h}, \\ & v_2, v_2^{(y_2)}, u_{2,1,y_2}, u_{2,2,y_2}, \dots, u_{2,m,y_2}, H_{2,1}, \dots, H_{2,h}, \\ & \dots, \\ & v_n, v_n^{(y_n)}, u_{n,1,y_n}, u_{n,2,y_n}, \dots, u_{n,m,y_n}, H_{n,1}, \dots, H_{n,h}, d), \end{aligned}$$

for some $y_1, \dots, y_n \in \{0, 1, 2\}$ and where the index starts at 0. With \bar{p} we shall refer to $1 - p$, the complement probability.

Lemma 4.3. *For every instance I and optimal policy $\pi^* \in \Pi^*$, with probability 1 there are $y_1, \dots, y_n \in \{0, 1, 2\}$ such that $X^{\pi^*}[0 : \min(T - 1, \tau_d)] = \mathbf{z}_{\mathbf{y},I}[0 : \min(T - 1, \tau_d)]$.*

Proof. Assume false. Following the assumption, let π^* be an optimal policy and $\tau = i(m + h + 2) + j, 0 \leq j < m + h + 2$, be the smallest integer such that for every assignment of y_1, \dots, y_n to 0, 1, or 2, the probability $P(X_\tau^{\pi^*} = x_\tau \neq z_\tau)$ is non-zero for some $x_\tau \neq z_\tau$.

Call π' the policy that follows the path $\mathbf{z}_{\mathbf{y},I}$ with $y_1 = \dots = y_n = 2$ and goes from d to t . We will compare the expected costs of π' and π^* from the time $\tau - 1$. This cost is bounded from below by the cost of the shortest path to the target t assuming that all remaining probabilistic edges are in the graph.

There are four cases: an ϵ -edge was used to change layer ($2 < j \leq m + 2$), an edge was followed backwards (hence $x_\tau = z_{\tau-2}$), the edge $\{U_{3,i}, r_i\}$ was used for some i , or an edge was followed backwards from $H_{i,1}$ to the last x_i -clause of a vertex of “the other” truth assignment (for some $y, y', y \neq y', v_{\tau-2} = u_{i,n,y}, v_{\tau-1} = H_{i,1}$, and $x_\tau = u_{i,n,y'}$).

For all cases besides the first, we merely note that if the cost from $v_{\tau-1}$ in the best case is c , then the expected cost is $E[c(\mathbf{X}^{\pi'}[\tau - 1 : T])] \leq c + \bar{p}\alpha$ since the shortest path necessarily goes through $H_{i,j}$ for some i, j . On the other hand, $E[c(\mathbf{X}^{\pi^*}[\tau - 1 : T])] \geq c + 1 > c + \bar{p}\alpha$.

For the case of the ϵ -transitions, we note that

$$E[c(\mathbf{X}^{\pi^*}[\tau - 1 : T])] \geq m + 4 + \bar{p} + 2\bar{p}^2 + 4\bar{p}^3 + \epsilon - j.$$

By taking the simple strategy of testing $H_{i,1}$, $H_{i,2}$, and $H_{i,3}$ to see if there's an edge to t of cost 1 and taking the edge of cost $\alpha + 1$ from $H_{i,3}$ otherwise, we know that

$$E[c(\mathbf{X}^{\pi^*}[\tau - 1 : T])] \leq m + 4 + \bar{p} + 2\bar{p}^2 + 4\bar{p}^3 + \alpha\bar{p}^4 - j.$$

Since $\alpha\bar{p}^4 < \epsilon$, this would contradict the assumption that π^* is optimal. \square

$$\text{Call } q = \bar{p}^{nh} \text{ and } \theta = q + \frac{1-q}{p} \left(m + 2 + \frac{(1-2\bar{p})^h}{1-2\bar{p}} \right).$$

Lemma 4.4. *For every optimal policy π^* , the probability of reaching d before t is q and the expected cost of the subpath $X^{\pi^*}[0 : \tau_d]$ is θ .*

Proof. Using Lemma 4.3, we observe for the probability that

$$P(X_{\tau_d}^{\pi^*} = d) = P(\{H_{1,1}, t\}, \{H_{1,2}, t\}, \dots, \{H_{n,h}, t\} \notin E) = \bar{p}^{nh}.$$

\square

Proof. For the expected cost, consider an optimal policy and arbitrary $y_1, \dots, y_n \in \{0, 1\}$. Identifying terms, we have

$$\begin{aligned} E[c(\mathbf{X}^{\pi^*}[0 : \tau_d])] &= \sum_{k=0}^{\tau_d-1} P(T = k + 1)w(z_k, t) + \sum_{k=0}^{\tau_d-1} P(T > k + 1)w(z_k, z_{k+1}) \\ &= P(T \leq \tau_d) + \sum_{i=0}^{n-1} \sum_{j=0}^{m+h+1} P(T > (m+h+2)i + j)w(z_j, z_{j+1}) \\ &= q + \sum_{i=0}^{n-1} \bar{p}^{ih} \sum_{j=0}^{m+h+1} (2\bar{p})^{\max(0, j-m-2)} \\ &= q + p^{-1}(1-q) \left(m + 2 + (1-2\bar{p})^{-1}((1-2\bar{p})^h) \right) = \theta. \end{aligned}$$

\square

Lemma 4.5. *Consider the time τ_d when an optimal policy π^* reaches d for the first (and only) time. If, for every $1 \leq j \leq m$, at least one of the edges $\{U_{1,j}, t\}$, $\{U_{2,j}, t\}$, and $\{U_{3,j}, t\}$ has been observed being in the graph, then the successor vertex of the policy is r and otherwise t .*

Proof. The vertex d has as neighbors only r, t , and $H_{n,h}$. The cost of the edge $\{d, t\}$ is $\beta < \alpha + 4 \leq (m+1)^2 + 4 \leq w(H_{n,h}, H_{n,h-1})$, hence we may rule $H_{n,h}$ out as the next vertex.

By Lemma 4.3, at time τ_d , none of the probabilistic edges $\{\{r, r_j\}\}_j$ has been revealed for an optimal policy. Hence there's a probability of at least 2^{-m} that none of the edges $\{\{r, r_j\}\}_j$ is in the graph. The cost of going from d to r in this event is $\alpha + 3 + \delta$.

Assume, for the sake of contradiction, that an optimal policy π^* chooses to go to r even though, for some k , neither of $\{U_{1,k}, t\}$, $\{U_{2,k}, t\}$, and $\{U_{3,k}, t\}$ has been observed having an edge to t . On top of the event that none of the edges is available, there's a probability of 2^{-m} that $\{r, r_k\}$ is the only edge in the graph among $\{\{r, r_j\}\}_j$. Were this to happen, the cost of going from d to t through $U_{3,k}$ is at least $\alpha + 3 + \epsilon\bar{p} + 2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3 = \alpha + 3 + \delta$ since, in the best case, neither of $U_{1,k}$, $U_{2,k}$ and $U_{3,k}$ has been visited before and each one of the three edges has a probability p of being connected to t with an edge of cost 1. Were neither of the vertices to have an edge to t , then the policy must pay 1, in the best case, to follow an edge to a clause which is known to have such an edge to t .

Hence, the expected cost of going from d through r to t if not every clause is known to have an edge to t is at least $\alpha + 3 + 2^{-m+1}\delta = \alpha + 3 + 2 \cdot 2^{-m}(\epsilon\bar{p} + 2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3) \geq \alpha + 3 + 2 \cdot 2^{-m}\bar{p}^3$. This is not optimal since $\beta = \alpha + 3 + 3\epsilon = \alpha + 3 + 1.5 \cdot 2^{-m}\bar{p}^3$.

If, on the other hand, for every $1 \leq j \leq m$, at least one of $\{U_{1,j}, t\}$, $\{U_{2,j}, t\}$, and $\{U_{3,j}, t\}$ has been observed, then the following simple policy bounds from above the cost of going through r .

Upon reaching r , take any available edge to $U_{3,j}$ for some j or go to t if none is available. If $U_{3,j}$ has an edge to t , then take it. Otherwise, follow an ϵ -edge to whichever one of $U_{1,j}$ and $U_{2,j}$ that we know has an edge to t and continue there.

The probability that no edge is available from r is 2^{-m} and the probability that $U_{3,j}$ does not have an edge to t is at most \bar{p} . Note that this lower bound is independent of whether or not $U_{3,j}$ has been visited before time τ_d . The described policy, and subsequently an optimal policy, therefore has a cost from t of at most

$$\begin{aligned} & \alpha + 3 + (1 - 2^{-m})\epsilon\bar{p} + 2^{-m}\delta = \alpha + 3 + \epsilon\bar{p} + 2^{-m}(2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3) \\ & = \alpha + 3 + 2\bar{p}^5/\alpha + 2^{-m}(4\bar{p}^6/\alpha + (4\bar{p}^4/\alpha + 1)\bar{p}^3) \\ & \leq \alpha + 3 + 2^{-m-1}\bar{p}^4 + 2^{-m}(\bar{p}^3/8 + (17/16)\bar{p}^3) \\ & \leq \left\| m \geq 1, \bar{p} \geq 0.5 \right\| \leq \alpha + 3 + (23/16)2^{-m}\bar{p}^3 < \beta \end{aligned}$$

□

Lemma 4.6. *For every instance $I \in \mathcal{I}_{E3SSAT}$ and optimal policy π^* , the probability that, when reaching d for the first time, for every $1 \leq j \leq m$ at least one of $\{U_{1,j}, t\}$, $\{U_{2,j}, t\}$, and $\{U_{3,j}, t\}$ has been observed to be in the*

graph is at least $E3SSAT'(I)p^m$ and at most $E3SSAT'(I)(1 - \bar{p}^3)^m$, where $E3SSAT'(I)$ is the probability that I is satisfiable.

Proof. From Lemma 4.3, we know that the subpath from v_1 to d is fully specified by the variables $y_1, \dots, y_n \in \{0, 1, 2\}$. Since we have the clauses $(x_1 \vee \neg x_1) \wedge \dots \wedge (x_n \vee \neg x_n)$, choosing $v_i^{(2)}$ (i.e. $y_i = 2$) for any i means that the boolean formula is not satisfiable. The assignment of y_1, \dots, y_n therefore corresponds to the assignment of the variables x_1, \dots, x_n of the formula and the maximum probability of a satisfying assignment corresponds to the maximum probability of visiting at least one of $U_{1,j}, U_{2,j}$, and $U_{3,j}$ for every $1 \leq j \leq m$.

Having for every j visited one of the three, the probability of having seen an edge for each j for some i from $U_{i,j}$ to t is at least p^m and at most $(1 - \bar{p}^3)^m$. Multiplying this with the probability of satisfying the assignment gives the promised bounds. \square

Lemma 4.7. *For every $I \in \mathcal{I}_{E3SSAT'}$ and $\pi^* \in \Pi^*$, the optimal cost, $c(\pi^*, I)$, is at most B if and only if $E3SSAT'(I) = (3/4)^{n/2}$.*

Proof. Note that for every instance I , $E3SSAT'(I) \leq (3/4)^{n/2}$.

Assume that I is satisfiable with probability $2^{-n}3^{n/2}$ and let π^* be an arbitrary optimal policy. Combining Lemma 4.3 with Lemma 4.5, Lemma 4.6, and the policy of Lemma 4.5 implies that

$$\begin{aligned} c(\pi^*, I) &\leq \theta + q \left((1 - E3SSAT'(I)p^m)\beta + E3SSAT'(I)p^m(\alpha + 3 + (1 - 2^{-m})\epsilon\bar{p} + 2^{-m}\delta) \right) \\ &= \theta + q \left(\alpha + 3 + 3\epsilon - (3/4)^{n/2}p^m(3\epsilon - \epsilon\bar{p} - 2^{-m}(2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3)) \right). \end{aligned}$$

We wish to show that this is smaller than $B = \theta + q(\alpha + 3 + 3\epsilon - (3/4)^{n/2}(1 -$

$2^{-m}\epsilon$).

$$\begin{aligned}
c(\pi^*, I) &< B \\
&\iff p^m(3\epsilon - \epsilon\bar{p} - 2^{-m}(2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3)) > (1 - 2^{-m})\epsilon \\
&\iff \iff 2^{-m} = 4\bar{p}\alpha \text{ and } \epsilon = 2\bar{p}^4\alpha \iff \\
&\iff p^m(3\epsilon - \epsilon\bar{p} - 4\bar{p}\alpha(2\epsilon\bar{p}^2 + 2\epsilon\bar{p}^3) - 2\epsilon) > (1 - 4\bar{p}\alpha)\epsilon \\
&\iff p^m(3 - \bar{p} - 8\bar{p}^3\alpha - 8\bar{p}^4\alpha - 2) \geq 1 - 4\bar{p}\alpha \\
&\iff (1 - \bar{p})^m(1 - \bar{p} - 16\bar{p}^3\alpha) \geq 1 - 4\bar{p}\alpha \\
&\iff \iff (1 + x)^k \geq 1 + kx \text{ for } k \geq 0, x \geq -1 \iff \\
&\iff (1 - \bar{p}m)(1 - \bar{p} - 16\bar{p}^3\alpha) \geq 1 - 4\bar{p}\alpha \\
&\iff 1 - \bar{p} - 16\bar{p}^3\alpha - \bar{p}m + \bar{p}^2m + 16\bar{p}^4m\alpha \geq 1 - 4\bar{p}\alpha \\
&\iff 4\bar{p}\alpha + \bar{p}^2m + 16\bar{p}^4m\alpha \geq (m + 1)\bar{p} + 16\bar{p}^3\alpha \\
&\iff \bar{p}m^2 \geq 4\bar{p}^3\alpha \iff \iff \bar{p} \leq 1/32 \iff \\
&\iff 1024\bar{p}^3m^2 \geq 4\bar{p}^3\alpha \iff 256 \geq 4
\end{aligned}$$

It remains to show the lemma for instances that are satisfiable with probability less than $(3/4)^{n/2}$. Let I be an arbitrary instance of this kind. More specifically, we have $\text{E3SSAT}'(I) \leq (3/4)^{n/2} - 2^{-n}$ since each of the $4^{n/2}$ outcomes are equally likely. With reasoning similar to the above,

$$\begin{aligned}
c(\pi^*, I) &\geq \theta + q(\beta + \text{E3SSAT}'(I)(1 - \bar{p}^3)^m(\alpha + 3 + 2^{-m}\delta - \beta)) \\
&= \theta + q\left(\alpha + 3 + 3\epsilon - 2^{-n}(3^{n/2} - 1)(1 - \bar{p}^3)^m(3\epsilon - 2^{-m}(2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3))\right)
\end{aligned}$$

and

$$\begin{aligned}
c(\pi^*, I) &> B \\
&\iff (1 - (3/4)^{-n/2})(1 - \bar{p}^3)^m(3\epsilon - 2^{-m}(2\epsilon\bar{p}^2 + (2\epsilon + 1)\bar{p}^3)) \leq (1 - 2^{-m})\epsilon \\
&\iff \iff (1 - (3/4)^{-n/2}) \leq (1 - 2^{-m}) \text{ as } m \geq n \iff \\
&\iff (1 - 2^{-m})(3 - 4\bar{p} - 8\bar{p}^3 - 2\epsilon^{-1}\bar{p}^4) \leq 1 - 2^{-m} \\
&\iff 1 - 4\bar{p} - 8\bar{p}^3 \leq 1.
\end{aligned}$$

□

Theorem 1. *SCTP is **PSPACE**-complete.*

Proof. Lemmas 4.1, 4.2, and 4.7 shows that $\text{E3SSAT}'$ is log-space Turing-reducible to SCTP. Since $\text{E3SSAT}'$ is **PSPACE**-hard, so is SCTP.

To see that SCTP can be solved in polynomial space, one merely notes that the expected cost of an instance can be computed as the minimum

over a weighted average of some constant and the expected cost of up to a polynomial number of instances with strictly fewer probabilistic edges. As there are only a polynomial number of probabilistic edges and the cost can be represented with a polynomial number of bits, a polynomial amount of space is sufficient at any one point. \square

Inapproximability of Markov-d-SSPPR

A variant to giving each edge an independent probability is to give an explicit probability for each edge subset of an input graph. The number of such subsets is exponential in the number of vertices, but if we specify in the problem that subsets of probability 0 need not be described, then we get the problem DISTRIBUTION-SSPPR (d-SSPPR or R-SSPPR). Because of the less concise representation of the realizations of the graph, this problem turns out to be **NP**-complete [18, 16].

Another variant of SCTP is to resample the graph after each step the searcher takes. This problem is called the Recoverable Canadian Traveller Problem, the Stochastic Shortest Path Problem with Recourse and Resets, the Expected Shortest Path Problem, and others [2, 6]. By reducing Recoverable-CTP to a Markov Decision Process (MDP), we see that this problem is solvable in polynomial time.

We name, respectively, Markov-i-SSPPR and Markov-d-SSPPR the generalization to Markov chains of Recoverable-i-SCTP and Recoverable-d-SCTP, as well as the generalizations of i-SSPPR and d-SSPPR. In this chapter, we present a simple proof showing that for every polynomial p , it is **PSPACE**-hard to approximate an instance I of Markov-d-SSPPR within $2^{p(|I|)}$. If a similar result can be derived for Markov-i-SSPPR, we do not know.

5.1 Definition of Markov-d-SSPPR

Definition 5.1 (MARKOV DISTRIBUTION-STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE(Markov-d-SSPPR) (decision version)). An instance

$I = (n, w, e, m, P, \mathbf{p}_0, B)$ to Markov-d-SSPPR consists of the number of vertices of the graph, n , the number of states of the chain, m , an edge-weight function w , a function e mapping states to edge sets, an m by m transition matrix P , a distribution of the initial state, \mathbf{p}_0 , and the bound $B(n)$. The output is ‘yes’ if and only if it is possible to traverse the Markov chain from vertex 1 to vertex n with expected cost bounded by $B(n)$.

A policy for this problem maps from the current vertex and history of observations to the next vertex of the walk. As in d-SSPPR, the observations after each step is the set of adjacent probabilistic edges of the reached vertex and the subset of them that are in the current realization. Note that if more than one state has the same realized neighborhood of a particular vertex, then the particular state cannot be deduced from observing that neighborhood alone.

One may note that Markov-d-SSPPR is a particular kind of partially observable Markov decision process (POMDP) but a generalization of Markov decision processes. Furthermore, d-SSPPR is the special case of Markov d-SSPPR when the transition matrix is an identity matrix and Recoverable-d-SCTP is the special case of identical rows. We could also substitute the m states with $(n+1)^k m$ states if we would like the distribution of “the next state” to be influenced by the last k visited vertices.

For an instance I , let p' be the element of P or \mathbf{p}_0 and w' be the element of w requiring the most bits to be represent under a common scheme. We shall call the size of the input for $N = |I| \in \Theta(m(m+n) + m^2 \log(w'p') + \log B) \subseteq (n \log w'p'B)^{O(1)}$.

It does not seem obvious to us that even exponential space is enough for the representation of an optimal policy for Markov-SSPPR with polynomial-time interpretation. See Chapter 7 for some reflections and other unanswered questions.

5.2 Overview

To prove that Markov-d-SSPPR cannot be approximated in polynomial time within $2^{N^{O(1)}}$, we assume there exists such an algorithm and show that this implies a contradiction. We construct a reduction for a fixed function $f(N) \in 2^{N^{O(1)}}$ which, given an arbitrary instance to QSAT, produces an instance $R_f(I)$ to Markov-d-SSPPR such that the expected cost of every suboptimal policy is more than $f(n)$ times greater than the expected cost of an optimal policy for a satisfiable QSAT instance, $B(n)$. If the QSAT instance is not always satisfiable, then our reduction guarantees that the expected cost is more than $f(n)B(n)$.

By querying the approximation algorithm whether the cost is bounded by $f(n)B(n)$, we are guaranteed to get a ‘yes’ answer if there is a policy achieving a expected cost of at most $B(n)$ and a ‘no’ answer if every policy

costs more than $f(n)B(n)$. For positive QSAT instances, the former is guaranteed and for the negative instances, the latter. Hence we are able to answer the QSAT instance in polynomial time.

The idea of the reduction is to have the chain go through a number of states, forcing a policy to either set a variable to true or false at random or to make a choice of how to value should be set. Having set a variable, the policy must go through vertices representing the clauses containing those variables. Exactly one of the clauses will have an edge of cost 1 to the target t . Which clause is not deducible without visiting every clause and if the target hasn't been reached after setting all of the vertices, then the policy is forced to take an expensive edge to the target. More specifically, the edge will be so costly that if the policy is not guaranteed to always find the cheap edge, then the expected cost will be strictly greater than $f(n)B(n)$. The cost will, however, be in $2^{N^{O(1)}}$ and thereby representable in polynomial space.

5.3 Reduction from QSAT to Markov-d-SSPPR

Let A_f be a supposed $f(n)$ -approximation algorithm for Markov-d-SSPPR with $f(n) \in 2^{N^{O(1)}}$ and $f(n) \geq 1$ for every n . For a given QSAT instance I , we produce an instance $R_f(I)$ to Markov-d-SSPPR such that $QSAT(I) = A_f(R_f(I))$. We define the reduction and show that it is computable in polynomial time.

Let the positive variables of I be $v_1^{(0)}, v_2^{(0)}, \dots, v_n^{(0)}$, the negative variables $v_1^{(1)}, v_2^{(1)}, \dots, v_n^{(1)}$, and the clauses C_1, \dots, C_m . A clause such as “ v_2 or not v_5 or not v_6 ” shall be denoted with $C = \{v_2^{(0)}, v_5^{(1)}, v_6^{(1)}\}$. Note that $m \leq 8n^3$.

The reduction graph G_I contains $3n + 2m + 1$ vertices: $v_1, \dots, v_n, v_1^{(0)}, v_1^{(1)}, \dots, v_n^{(0)}, v_n^{(1)}, c_1, \dots, c_m, d_1^{(0)}, d_1^{(1)}, \dots, d_m^{(0)}, d_m^{(1)}, v_{n+1}, t$, respectively representing variable assignment vertices, assignment to true and false, clause vertices, dummy vertices when literals are not in the clause of the same index, a dummy vertex for convenience, and the target. The starting vertex is v_1 .

$R_f(i)$ has $(m + 1)(m + 3)n$ states: $W_{k,i,y}, w_{k,i}$, and $s_{k,i,j}$ for $k \in (m], i \in (n], y \in \{0, 1\}$, and $j \in (m]$. Most transitions of the chain are deterministic: the chain transitions from $W_{k,i,y}$ to $w_{k,i}$ to $s_{k,i,1}$ to $s_{k,i,2}$ to \dots to $s_{k,i,m}$. The chain transitions, for $i < n$, from state $s_{k,i,m}$ to $W_{k,i+1,0}$ or $W_{k,i+1,1}$ with respective probability 0.5. The states $s_{k,n,m}$ transition to themselves.

The edge sets for the respective states are as follows. For every state $W_{k,i,y}, w_{k,i}$, or $s_{k,i,j}$, there is an edge from c_k to t . For states $W_{k,i,y}$, there is an edge from v_i to $v_i^{(y)}$, denoting setting the variable v_i to true ($y = 1$) or false ($y = 0$). If i is odd, there is furthermore an edge from v_i to $v_k^{(1-y)}$, allowing an existential choice. From $w_{k,i}$ there is an edge from $v_i^{(0)}$ and $v_i^{(1)}$. The edge goes from $v_i^{(y)}$ to c_1 if $v_i^{(y)} \in C_1$ and otherwise to $d_1^{(y)}$. For states

$s_{k,i,j}$, $j < m$, there is, for each $y \in \{0, 1\}$, an edge from u to u' where u is c_j if $v_i^{(y)} \in C_j$ and otherwise $d_j^{(y)}$, and u' is c_{j+1} if $v_i^{(y)} \in C_{j+1}$ and otherwise $d_j^{(y)}$. From states $s_{k,i,m}$, there is an edge to v_{i+1} from c_m , $d_m^{(0)}$, and $d_m^{(1)}$; and from v_{n+1} to t . All edges cost 1 except the edge from v_{n+1} to t , which costs $\alpha \stackrel{\text{def}}{=} 2^{\lfloor n/2 \rfloor + 1} m f(n) B(n)$.

The starting distribution of the chain is uniform over the states $W_{1,1,1}, \dots, W_{k,1,1}$ and the bound is $B(n) = 4n^3$.

See Figure 5.1 for an example instance of the reduction.

Lemma 5.1. *For every function $f(n) \in 2^{N^{O(1)}}$ and instance I to QSAT, the size of the produced instance, $|R_f(I)|$, is polynomial in $|I|$.*

Proof. We have that $|R_f(I)| \in (n \log w' p' B)^{O(1)}$, where w' is the edge cost and p' the probability that uses the most number of bits to represent. Since all edges of $R_f(I)$ has cost 1 or $2f(n)B(n)$, the most demanding cost is $w' = 2f(n)B(n) \in 2^{|I|^{O(1)}}$. Likewise, $p' = 0.5 \in O(1)$. It follows that $|R_f(I)| \in (n|I|^{O(1)})^{O(1)} = |I|^{O(1)}$. \square

Lemma 5.2. *For every function $f(n) \in 2^{N^{O(1)}}$, the reduction R_f can be computed in polynomial time.*

Proof. In our description, we have only used a constant number of indices ranging over a set of polynomial cardinality. All mentioned numbers are representable in space polynomial in the size of the QSAT instance. \square

5.4 Hardness Proof

Let K be the random variable appearing in the starting state $W_{K,1,1}$.

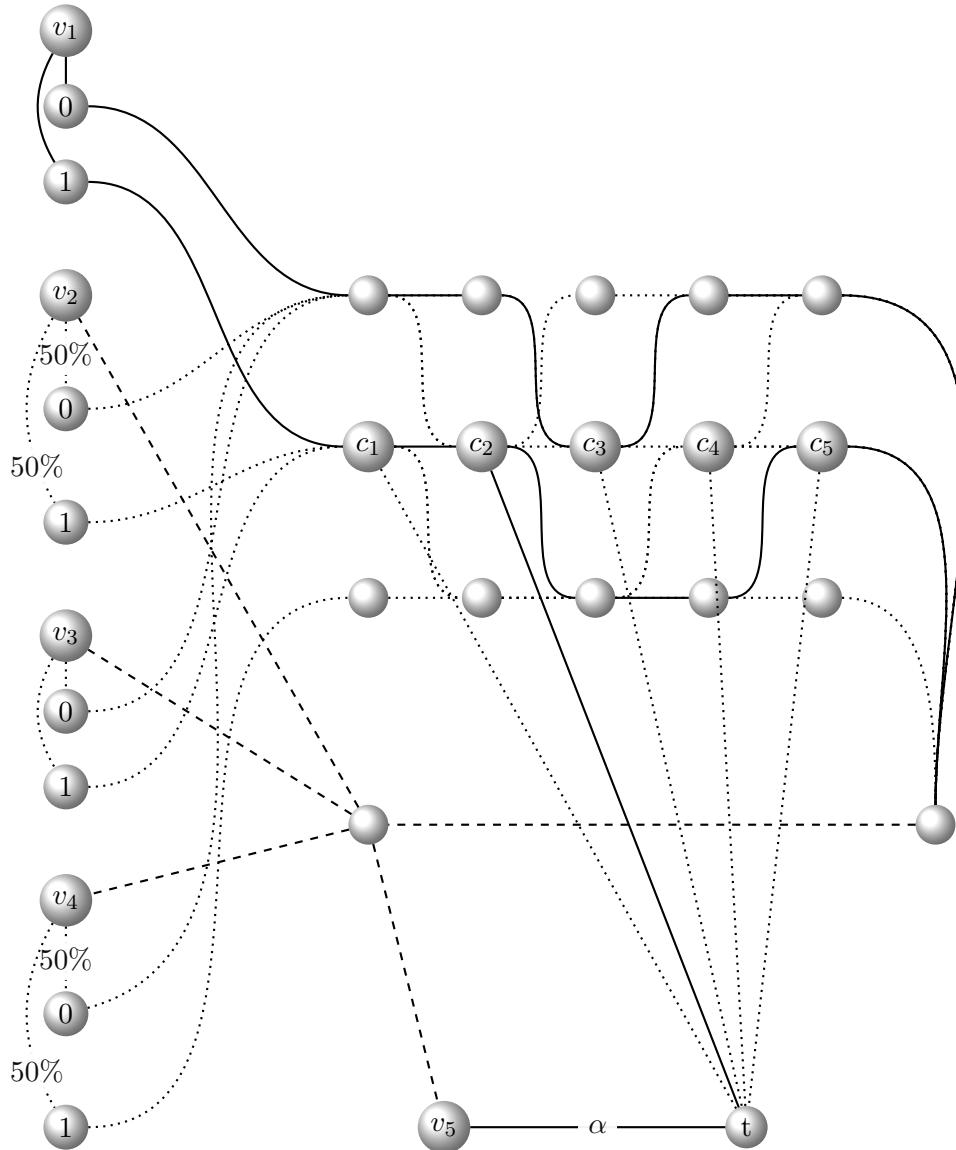
Lemma 5.3. *For every QSAT instance I on n variables and policy π , there is with probability 1 some $1 \leq k \leq m$ and $y_1, \dots, y_n \in \{0, 1\}$ such that the walk taken by π in $R(I)$ is a prefix of*

$$\mathbf{z} = v_1, v_1^{(y_1)}, g_{1,1}, \dots, g_{1,m}, v_2, v_2^{(y_2)}, g_{2,1}, \dots, g_{2,m}, \dots, v_n, \dots, g_{n,m}, t$$

and where $g_{i,j}$ is c_j if $v_i^{y_i} \in C_j$ and otherwise $d_j^{(y_i)}$. Furthermore, if in a realization, \mathbf{z} is not a subwalk of the walk, then the last vertex of the subwalk is c_K and the vertex following it is the target t .

Proof. The walk is more or less dictated by the progression of the states. The only available branches is the initial choice of k , decisions at v_i for some i , and from c_K . The former two are parametrized and taking the edge from c_K to t agrees with the claim. \square

Figure 5.1: Example reduction for the QSAT instance $\exists_{x_1} \forall_{x_2} \exists_{x_3} \forall_{x_4} (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_4)$. Whole-drawn lines indicate edges available for at least one of the the states $W_{2,1,1}, w_{2,1}, s_{2,1,1}, s_{2,1,2}, s_{2,1,3}, s_{2,1,4},$ and $s_{2,1,5}$ (i.e. the assignment of the first variable). Dotted lines lines indicate edges available for some state and dashed edges have been inserted in order to reduce clutter. The bound $B(4)$ is 256 and $\alpha = 2^{\lfloor n/2 \rfloor + 1} m f(n) B(n)$ is $10240f(4)$. Note how every suboptimal policy has an expected cost of at least $512f(4)$.



It follows that K, Y_1, \dots, Y_n are random variables which are invariants of the process. Furthermore, for every QSAT instance and policy for the instance, if we condition the policy's walk on $\{K = k, Y_2 = y_2, \dots, Y_{2\lfloor n/2 \rfloor} = y_{2\lfloor n/2 \rfloor}\}$ for some values, then there is a single realization of the walk with probability 1. We shall call this walk $\mathbf{x}^{(k)}$ and $T = \min\{\tau \mid x_\tau^{(k)} = t\}$ the first time the target is visited. Recall that we defined the walk of a policy such that the target is visited indefinitely from the time of the walk's first visit.

Lemma 5.4. *There is no QSAT instance I and policy with expected cost strictly greater than $B(n)$ and less than or equal to $f(n)B(n)$.*

Proof. Note that no policy can take more than $n(m+2) + 1 \leq B(n)$ steps because of the state transitions. Aside from $\{v_{n+1}, t\}$, no edge has a cost greater than one. Hence, if the probability is 0 that the policy takes this edge, then the expected cost is at most $B(n)$. Since no realization of the process has probability less than $2^{-\lfloor n/2 \rfloor}/m$, the minimum probability of taking $\{v_{n+1}, t\}$ which is greater than 0 is $2^{-\lfloor n/2 \rfloor}/m$. The cost of $\{v_{n+1}, t\}$ is $2^{\lfloor n/2 \rfloor + 1}mf(n)B(n)$ and so if the policy takes this edge with non-zero probability, then the expected cost of the policy is at least $2f(n)B(n) > f(n)B(n)$. \square

Lemma 5.5. *For every satisfiable QSAT instance I , there is a policy achieving an expected cost of at most $B(n)$.*

Proof. We know that if there is no realization in which a policy takes the edge $\{v_{n+1}, t\}$, then the expected cost is bounded by $B(n)$. In other words, we wish to show that $c_k \in \mathbf{x}^{(k)}$ for every $1 \leq k \leq m$ and $y_2, \dots, y_{2\lfloor n/2 \rfloor} \in \{0, 1\}$. Let σ be a satisfying conditional assignment of I and consider the following policy. Follow \mathbf{z} unambiguously by going to $v_{2i+1}^{\sigma(Y_1, \dots, Y_{2i})}$ upon reaching vertex $v_{2i+1}, i < n$. We claim that, for every realization, this policy visits each clause vertex c_i , thereby implying the existence of a policy that never has to take the edge from v_{n+1} to t . Let k, y_1, \dots, y_n be an arbitrary realization of the random variables K, Y_1, \dots, Y_n while following the described policy. We have the property that $y_{2i+1} = \sigma(y_1, \dots, y_{2i})$ and, because σ satisfies I , that $\forall_i \exists_j v_j^{(y_j)} \in C_i$. But if $Y_j = y_j$ for some j , then $v_j^{(y_j)} \in \mathbf{x}^{(k)}$ and if furthermore $v_j^{(y_j)} \in C_i$, then $g_{i,j} = c_i \in \mathbf{x}^{(k)}$. \square

Lemma 5.6. *For every QSAT instance I , if there is a policy for $R_f(I)$ with expected cost at most $B(n)$, then the instance I is satisfiable.*

Proof. Following the proofs above, a policy achieves a cost of $B(n)$ only if, for every $1 \leq k \leq m$ and $y_2, \dots, y_{2\lfloor n/2 \rfloor} \in \{0, 1\}$, the vertex c_k is in $\mathbf{x}^{(k)}$. We show that this implies that there is a policy such that, for arbitrary $y_2, \dots, y_{2\lfloor n/2 \rfloor} \in \{0, 1\}$, there is a k such that $c_i \in \mathbf{x}^{(k)}$ for every $1 \leq i \leq m$

Figure 5.2: Polynomial-time algorithm for QSAT assuming the existence of an effective $f(n)$ -approximation algorithm A_f for Markov- d -SSPPR.

```
POLYTIMEQSAT $_{f,A_f}(I)$ 
1  return  $A_f(R_f(I)) \leq f(n)B(n)$ 
```

and therefore that the QSAT instance is satisfiable. Formally, we wish to show that

$$\forall_I \exists_\pi \forall_{y_2, \dots, y_{2\lfloor n/2 \rfloor}} \forall_k c_k \in \mathbf{x}^{(k)} \implies \forall_I \exists_\pi \forall_{y_2, \dots, y_{2\lfloor n/2 \rfloor}} \exists_k \forall_i c_i \in \mathbf{x}^{(k)}.$$

Note that if we have a policy guaranteeing that $c_k \in \mathbf{x}^{(k)}$ for every k , then there also exists a policy such that $x_{T^{(k)}-1}^{(k)} = c_k$. Furthermore, there exists a policy with this property and the property that, for every $1 \leq k_1, k_2 \leq m$ with $T^{(k_1)} \leq T^{(k_2)}$, the two walks $\mathbf{x}^{(k_1)}$ and $\mathbf{x}^{(k_2)}$ share prefix in the sense that $\mathbf{x}^{(k_1)}[0 : T^{(k_1)} - 1] = \mathbf{x}^{(k_2)}[0 : T^{(k_1)} - 1]$. This follows from that the value of the random variable K only affects whether the edges of the form $\{c_i, t\}$ are in the graph and there are therefore no differences in observations between the two cases save for when c_K is reached.

Assume that the above consequent is false even though we assume that the antecedent is true. Consider an arbitrary instance for which it does not follow, a policy with the properties discussed in the preceding paragraph, and arbitrary falsifying $y_2, \dots, y_{2\lfloor n/2 \rfloor} \in \{0, 1\}$. Following the assumption, let $\phi(1), \dots, \phi(m)$ be such that for every k , $c_{\phi(k)} \notin \mathbf{x}^{(k)}$. We need to show that this implies a contradiction.

Following that $c_{\phi(k)} \notin \mathbf{x}^{(k)}$, we have that either $\phi(k) = k$ or $T^{(\phi(k))} < T^{(k)}$, and therefore that $c_{\phi(\phi(k))} \notin \mathbf{x}^{(k)}$. Every function $f : X \mapsto X$ for finite X eventually produces a cycle after iterated application and so there exists $j \in [n]$ and $\lambda \geq 1$ such that $\phi^\lambda(j) = j$, where e.g. $\phi^3(k) = \phi(\phi(\phi(k)))$. It follows that $j = c_{\phi^\lambda(j)} \notin \mathbf{x}^{(j)}$ which contradicts our assumption about the policy that $c_k \in \mathbf{x}^{(k)}$ for every k . \square

Lemma 5.7. *Assume the existence of a polynomial-time $2^{N^{O(1)}}$ -approximation algorithm A_f for Markov- d -SSPPR. In that case, Algorithm 5.4 returns ‘yes’ exactly for positive QSAT instances.*

Proof. Following Lemma 5.4 and Lemma 5.6, if a QSAT instance I is not satisfiable, then there is no policy for $R_f(I)$ achieving an expected cost of $f(n)B(n)$ or less. The approximation algorithm therefore returns a value greater than $f(n)B(n)$ and Algorithm 5.4 behaves as expected.

For every positive QSAT instance I , we know from Lemma 5.5 that there is a policy for $R_f(I)$ with expected cost at most $B(n)$ and so the approximation algorithm is guaranteed to produce an answer of at most $f(n)B(n)$ with the appropriate consequences for Algorithm 5.4. \square

Theorem 2. *Markov-d-SSPPR cannot be approximated within $2^{N^{O(1)}}$ unless $P = PSPACE$.*

Proof. Assume false and let A_f be an efficient $f(n)$ -approximation algorithm for Markov-d-SSPPR for some $f(n) \in 2^{N^{O(1)}}$. Consider the algorithm that, given $I \in \mathcal{I}_{QSAT}$, runs A_f on the Markov-d-SSPPR instance $R_f(I)$ and returns ‘yes’ if and only if the output is at most $f(n)B(n)$. From Lemma 5.7, we know that a positive answer to $R_f(I)$ implies that the optimal cost is at most $B(n)$, which in turn means that the output of the described algorithm for I corresponds to whether or not I is satisfiable. Since Lemma 5.2 shows that R_f can be computed in polynomial time, we have proved the existence of a polynomial-time algorithm for QSAT and, indeed, every PSPACE-easy problem. \square

Corollary 5.8. *Both versions of Markov-d-SSPPR are PSPACE-hard to approximate within $2^{N^{O(1)}}$.*

Proof. In Section 3.2, we proved that there are instances of Markov-d-SSPPR such that the expected cost of a policy that may reroute at any time is strictly better than every policy which may only reroute only upon an edge which the policy intended to take failed. Since this is the only difference between the two versions and we have proved that the former version is PSPACE-hard to approximate, we need only show that there is an optimal policy for the first version which is also a valid policy for the second. This is easy to see. Assume that we have a conditional satisfying assignment for QSAT and set the tentative walk to \mathbf{z} with y_{2k} set to the assignment of x_{2k} given $x_1 = y_1, \dots, x_{2k-1} = y_{2k-1}$ and setting y_{2k+1} to 0. Take a subwalk of this plan by, from each clause vertex c_i , plan to take a probabilistic edge to t , whenever there is one. \square

Inexistence of Polynomial-Time Policies

In this section, we show that unless $\mathbf{PSPACE} \subseteq \mathbf{P/poly}$, there is no polynomial p , such that for every instance $I \in \mathcal{I}_{SCTP}$, there is a *description* of an optimal policy to navigate I bounded in size and time to interpret $p(|I|)$. Instead of this longish statement, we shall simply say that SCTP does not have (or guarantee) polynomial-time policies unless $\mathbf{PSPACE} \subseteq \mathbf{P/poly}$. Note also that this statement says nothing about the resources needed to find the policies (if there are any) and we do not imply any bounds of this kind, not even computability.

This problem was briefly mentioned by Provan [18] for i-SPPRPR, and subsequently for i-SCTP in Karger and Nikolova [10]. In the latter paper, the question was posed as follows.

[...] Indeed, even simpler questions such as whether there is a polynomial size *description* of an optimal or approximately optimal policy, regardless of whether it can be constructed, remains unanswered.

The question about approximate policies we leave unanswered. Regarding optimal policies, we choose to interpret this statement as though there should be an algorithm which, given the description and the observation history, produces in polynomial time a next vertex minimizing expected travel cost. The polynomial growth is here with respect to the instance size, not the input to the policy. Were we not to place any time restrictions, a description of an optimal policy would simply be the problem instance, as it may be solved in polynomial space.

For many problems, one can argue that they do not have polynomial-time policies because one could then guess a policy and then guess an example

that shows that the policy does not achieve the desired bound. For many **PSPACE**-complete problems between two players, such an argument would rule the policies out unless Σ_2^P . For games involving randomness, so called *games against nature*, it is not clear how to do the verification in polynomial time, resulting in the weaker collapse $\mathbf{P}^{\#\mathbf{P}} = \mathbf{PSPACE}$ if such a game does indeed guarantee polynomial-time policies. We will address this point again in the closing discussion of this thesis.

The graphs that we will be studying in this chapter are extensions of the graphs used in Chapter 4. For this reason, the graphs are rather complicated and we will refrain from giving entirely accurate figures and instead only highlight the key idea.

6.1 Outline

The proof can be summarized as *if there is a polynomial-time optimal policy for every SCTP instance, then we can construct a \mathbf{P}/\mathbf{poly} -algorithm for a **PSPACE**-hard language L by taking as advice the optimal policy of a well-chosen polynomial-size instance $S(n)$. In particular, this instance will be chosen in such a way that every optimal policy must contain the answers to every instance of size n . The **PSPACE**-hard language that we will use for this is SCTP with a restricted set of edges. We begin by showing that this version is **PSPACE**-hard, then proceed by defining the instances S and a reduction R , and finally prove that the reduction is indeed polynomial-time and that the answer to $R(I)$ is the same as the answer to I .*

6.2 Restricted but Still Hard Problems

Lemma 6.1. *The language of true quantified boolean formulas (E3QSAT instances) with as many clauses as variables is **PSPACE**-complete.*

Proof. Given an arbitrary instance to E3QSAT with n variables and m clauses, if $m < n$, add $n - m$ trivially satisfiable clauses. If $m > n$, then add $m - n$ variables that do not appear in any clause. Since $m \leq n^3$, this constitutes a polynomial-time reduction from E3QSAT to n -Clause E3QSAT. **PSPACE**-easiness follows because the latter has as instances a subset of the former. \square

Definition 6.1 (ϵ_c -Card SCTP). We define the problem ϵ_c -Card SCTP as SCTP restricted to edges either with cost 1 and probability in $\{0, 0.5, 1\}$ or with cost 2^{-cn} and probability 1.

Lemma 6.2. *ϵ_4 -Card SCTP is **PSPACE**-complete.*

Proof. Consider the proof of Theorem 1 for instances to n -Clause E3QSAT. Each edge falls in one of the categories 1) edges of probability 1 and cost

1 or $\alpha = (m + 1)^2 + 2 = (n + 1)^2 + 2$, 2) edges of probability $p = 1 - 2^{-m-2}(m + 1)^{-2} = 1 - 2^{-n-2}$ and cost 1, α or $\alpha + 1$, and 3) edges of probability 1 and cost ϵ or $\alpha + 3 + 3\epsilon$. Begin by multiplying each of these costs (and the expected bound) by 128. Any edge with probability 1 and polynomial integer cost can be replaced by a polynomial number of cost 1 and probability 1 edges, leaving us with invalid edges of Categories 2 and 3. Instead of $p = 1 - 2^{-n-2}(n + 1)^{-2}$, we may use $p = 1 - 2^{-3n-2}$ with tedious but straight-forward changes of other constants. The desired probability is then achievable using $3n + 2$ parallel edges of probability 0.5, followed by a number of edges of probability 1 in series for the desired cost. Finally, we have that $2^{-cn} = 2^{-4n} = 128\epsilon$, which is just what we need in order to substitute the edges of Category 3 with series of edges of cost 2^{-cn} or 1. \square

6.3 Definitions of Some Superinstances

We will now define the SCTP instances $S : \mathbb{N} \mapsto \mathcal{I}_{SCTP}$ of which we take optimal polynomial-time policies as advice. Let h be $\lceil \log_2(1 + 4(m + 1)^2) \rceil$, N the size of $S(n)$, and $\mu = N^3 + 1$. Given n , construct the random graph $S(n)$ by taking the complete graph on $n(n + h + 3) + 1$ vertices and replacing each edge $\{u, v\}$ by a graph with vertices $\{u, v, uv_1, uv_2\}$ and probabilistic edges $\{u, uv_1\}, \{u, uv_2\}, \{uv_1, v\}$, and $\{uv_2, v\}$, where the former two have costs $\epsilon = 2^{-(c+1)n}N^{-3}/2$ and probabilities 1, the third has cost 1 and probability 0.5, and the fourth one has cost 2^{-cn} and probability 0.5. Let $A_1, \dots, A_{\kappa=n(n+1)/2}$ be an enumeration of the vertices called uv_1 for some u and v and B_1, \dots, B_κ an enumeration of the vertices called uv_2 such that the vertices A_i and B_i were introduced by the same edge substitution. Add to the graph vertices which we shall call $a_1, \dots, a_{\kappa+1}$, a vertex named $B_{\kappa+1}$, and a new target vertex t' . For each $1 \leq i \leq \kappa + 1$, add an edge of cost 0 and probability $\phi = 1 - 0.5\epsilon\mu^{-1}(N^3 + 1)^{-1}$ from a_i to t' and from b_i to t' . Furthermore, and an edge of cost $\kappa\mu = \kappa(N^3 + 1)$ between the same vertices by introducing dummy vertices). For each $1 \leq i \leq \kappa$, add an edge of cost 2μ and probability ϕ from a_i to a_{i+1} , an edge of probability 1 and cost μ from A_i to a_{i+1} , from $a_{\kappa+1}$ to B_1 , from B_i to B_{i+1} , from $B_{\kappa+1}$ to s , from t to t' , and finally an edge of cost μ and probability ϕ from a_i to A_i . We let the starting vertex be A_1 and the target t' .

See Figure 6.2 for how a superinstance could look like for the graphs of Figure 6.1. Note that, in our proof, the given instances are first transformed into the graphs that showed **PSPACE**-hardness. Furthermore, although we probably should show both, we only show one of the two kinds of vertices that substitute edges of the original graph.

Lemma 6.3. *The size of the SCTP instance $S(n)$ is polynomial in n .*

Proof. The lemma follows from that the number of vertices of $S(n)$ is polynomial in n and that the edge costs and probabilities of $S(n)$ can be represented

Figure 6.1: Two input graphs of size 3 with starting vertices s and targets t .

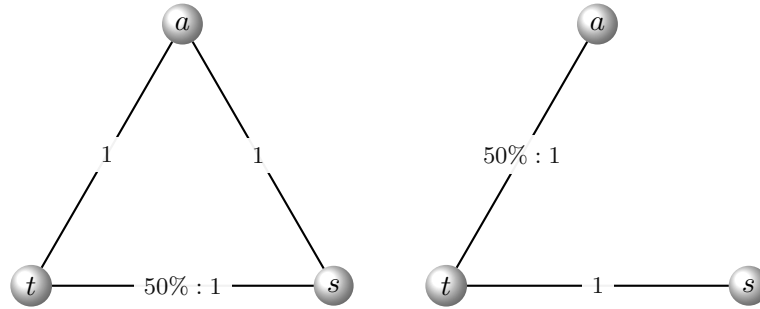
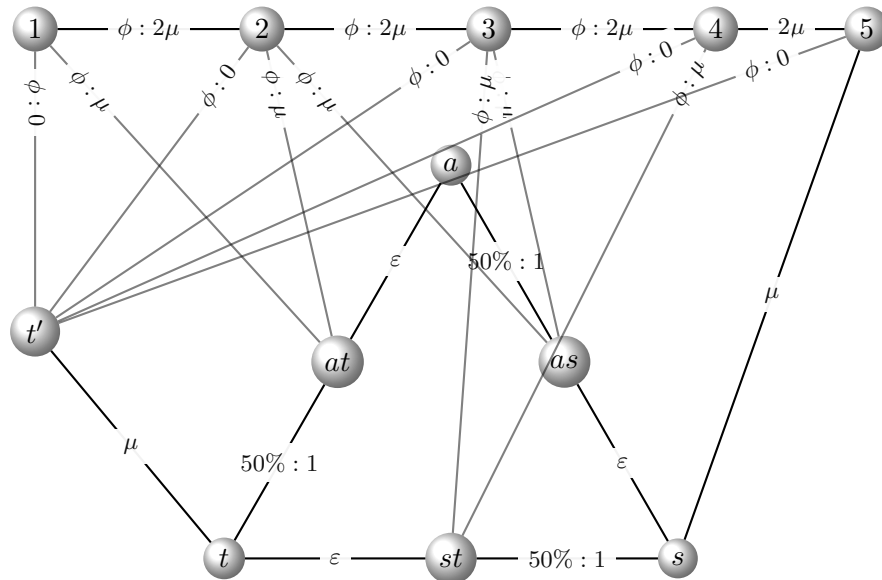


Figure 6.2: A “superinstance” for instances of size 3. The new start and end nodes are 1 and t' , respectively. To optimally solve this “superinstance”, you must also solve every instance of size 3.



with a polynomial number of bits. \square

6.4 A P/poly Algorithm

Let $C(I)$ for an n -Clause E3QSAT instance I be the “partial realization” (state with posterior distribution) of $S(n)$ that we next describe. Give $C(I)$ the same vertex set as $S(n)$. Let G be the reduction graph of I as constructed in the preceding section. Construct a bijection f between the vertex set of G and the vertices of the former complete graph of $S(N)$ by associating the starting vertex of G with s in $C(I)$, the target vertex of G with t in $C(I)$, and associating the remaining vertices arbitrarily. Let s be our starting vertex and t' the target. Add to $C(I)$ every probability edge of $S(n)$ that has probability one. For each pair of vertices $e_i = \{u, v\}$, $1 \leq i \leq \kappa$ in $C(I)$, do

- if the probability of the edge $\{f(u), f(v)\}$ in G has probability 0.5 (and hence cost 1), then add to $C(I)$ an edge of cost 2μ from a_i to a_{i+1} ,
- otherwise, if the cost of the edge $\{f(u), f(v)\}$ is 2^{-cn} , then add to $C(I)$ an edge of cost 2^{-cn} from uv_2 to v and an edge of cost μ from a_i to a_{i+1} .
- otherwise the edge has cost 1 and probability either 0 or 1. In either case, add to $C(I)$ an edge of cost μ from a_i to a_{i+1} .

Figure 6.3 and Figure 6.4 constitutes partial restrictions of Figure 6.2 such that the optimal policy from s to t in the resulting states corresponds to the optimal policies in the graphs of Figure 6.1. Note how considering the ϵ -edges to cost 0 and the α -edges to be impassable produces nearly the exact instances of Figure 6.2.

Lemma 6.4. *For every instance I of n -Clause E3QSAT, an optimal policy for $C(I)$ immediately takes the edge of cost $\mu + B(n) + 2N^3\epsilon$ from s to t' if and only if I is not satisfiable.*

Proof. From our construction and the proof of Theorem 1, we have that I is satisfiable if and only if G has a policy with expected cost at most $B(n)$. Note further that in every realization, there is a path from s to t in G (recall that G comes from our reduction of E3QSAT). Since μ is much larger than $B(n)$, no policy in $C(I)$ from s to t will contain any edges of cost μ . Hence, as we every optimal policy of G visits at most n^3 vertices, there must exist a policy for $C(I)$ from s to t of cost at most $B(n) + n^3\epsilon < B(n) + 2N^3\epsilon$. Since t and t' are connected by an edge of cost μ , it follows that satisfiability of I implies that the next action in $C(I)$ is not to take the edge to t' .

Figure 6.3: A restriction and optimal path prefix with optimal strategies corresponding to optimal strategies of the first graph given in Figure 6.1. The policy would in this state continue from vertex s .

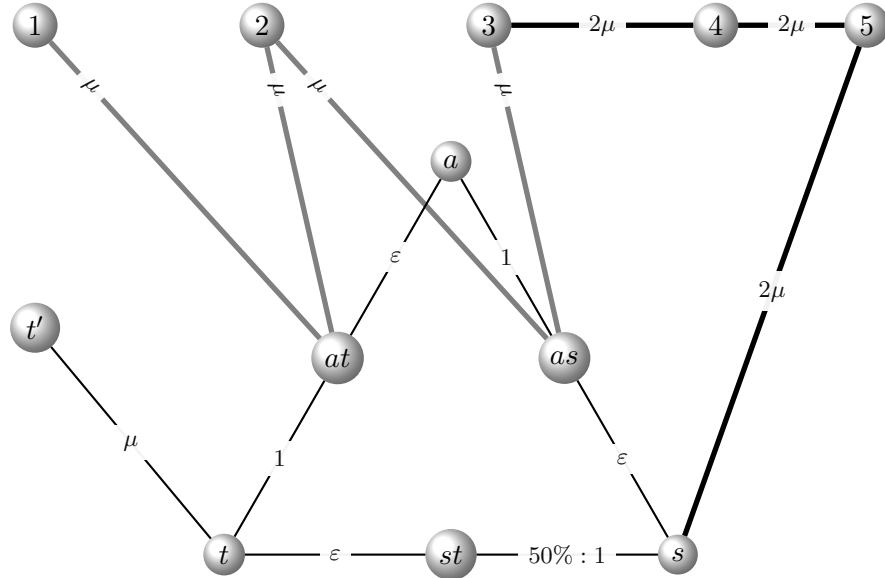
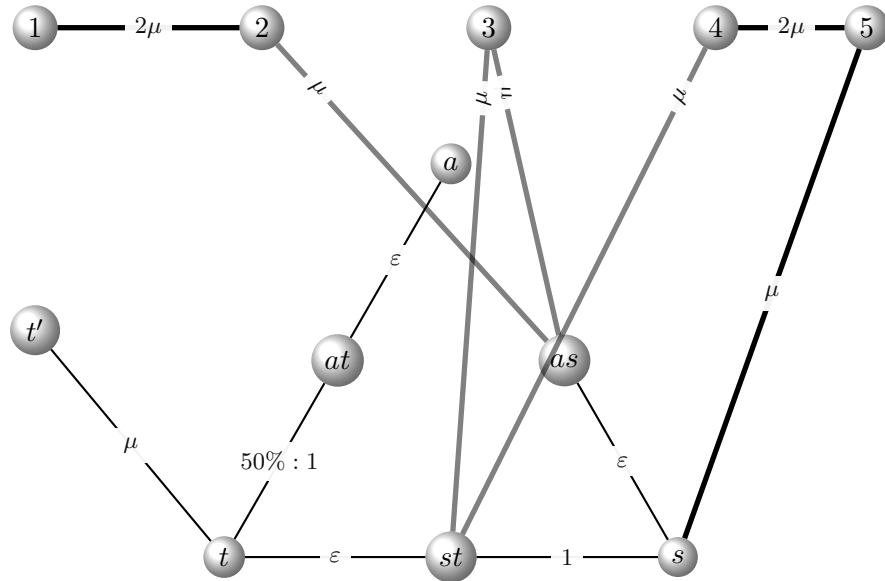


Figure 6.4: A restriction and optimal path prefix with the same optimal strategies as the second input graph of Figure 6.1.



Assume instead that I is not satisfiable, then the cost of every policy in G must be at least $B(n) + 2^{-n}\epsilon > B(n) + 2N^3\epsilon$, hence an optimal policy for $C(I)$ will indeed choose the edge from s to t' as its first action. \square

Lemma 6.5. *For every instance I of n -Clause E3QSAT and an optimal policy for $S(|I|)$, there is a non-zero probability that a state with the same posterior distribution as $C(I)$ appears.*

Proof. Consider an arbitrary realization H of $C(I)$ with non-zero probability. H is also a realization of $S(n)$ with non-zero probability. Hence, all we have to show is that every optimal policy navigating $S(n)$ in the (concealed) realization H will eventually reach a point where the known and unknown edges of the graph corresponds in distribution to $C(I)$.

We argue that every optimal policy will, in instance I with the hidden realization H , follow the path $a_1, \dots, a_{\kappa+1}, B_1, \dots, B_{\kappa+1}, s$ unless the first defective vertex is t' and with the exception that A_i may be visited after a_i before continuing on the path.

To see this, we note that the definition of $C(I)$ implies that for the statement to be false, the first defective vertex must either be the the vertex visited two steps earlier or be a vertex other than A_i, a_i, B_i , and t' for every i .

Consider the second case and let us call a first such vertex for v . Since u cannot be s , it must have been preceded by either A_i or B_i for some $i \leq \kappa$, call it u . From v , one can not reach t' without traversing an edge of cost μ and the cost of the edge $\{u, v\}$ is at least ϵ . Hence the expected cost to reach t' from v is at least μ . We proceed by showing that this is worse than following the proposed path. Having followed the given path up to u , there is an edge of cost α leading to either a_{i+1} or b_{i+1} which in turn has yet to be visited and therefore (conditioned on the observations made so far) has an edge to t' of cost 0 and probability \bar{p} . Consider the alternative policy that instead of going from u to v , goes to a_{i+1} or b_{i+1} and continues to t' regardless of whether the cost of going there is 0 (probability ϕ) or $\mu(N^3 + 1)$ (probability $1 - \phi$). The expected cost from u of this policy from is $\alpha + (1 - \phi)\mu(N^3 + 1) = \alpha + 0.5\epsilon < \alpha + \epsilon$.

The first case is pretty much the same, but the expected cost from the new state of the backwards-going policy is at least $2\mu + \epsilon$ and the expected cost of a policy following our path two more steps is at most $2\mu + 0.5\epsilon$. \square

Lemma 6.6. *For every instance I , a graph for $C(I)$ can be computed in time polynomial in the size of I .*

Lemma 6.7. *For every n -Clause E3QSAT instance I and optimal policy for $S(|I|)$, the policy's next action in the state that corresponds to $C(I)$ is to take the edge to t' if and only if I is not satisfiable.*

Proof. Assume false and consider an instance I and policy π for which the lemma does not hold. From Lemma 6.4, we know that every optimal policy in $C(I)$ goes directly to t' if and only if I is not satisfiable. It follows that a policy which does not abide to this is suboptimal. Since $C(I)$ appears with non-zero probability by Lemma 6.5, we can produce a better policy than the supposedly optimal policy π by changing the next action of the policy in the state $C(I)$. \square

Theorem 3. *There are instances of SCTP and i -SSPPR for which there is no polynomial-time optimal policies unless $\mathbf{PSPACE} \subseteq \mathbf{P/poly}$.*

Proof. Assume the existence of polynomial-time optimal policies for every SCTP instance and consider the following $\mathbf{P/poly}$ -algorithm. Let the advice associated with n be a (polynomial size) description of an optimal policy for the instance $S(n)$. Given an n -Clause E3QSAT instance I and the policy for $S(n)$, construct in polynomial time the state $C(I)$ and query the policy for the next vertex. If the next vertex is t' , return that the E3QSAT instance is not satisfiable and otherwise that it is. From the lemmas above, we know that the described algorithm runs in polynomial time and that the output does indeed correspond to the desired answer of the E3QSAT instance. \square

CHAPTER 7

Discussion and Open Problems

We have highlighted a number of variants of the STOCHASTIC CANADIAN TRAVELLER PROBLEM and the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE, most notably that there are two different versions of the latter problem. We have shown that the STOCHASTIC CANADIAN TRAVELLER PROBLEM, the EDGE-INDEPENDENT STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE, and some variants are **PSPACE**-complete, that the MARKOV DISTRIBUTION-STOCHASTIC SHORTEST PATH PROBLEM WITH RECOURSE is **PSPACE**-hard to approximate within any exponential-polynomial function, and that SCTP and others do not have optimal polynomial-time policies for every instance unless the polynomial hierarchy collapses. Considering SCTP's (supposed) frequent appearance, this is bad news.

The full report [16] of the paper by Polychronopoulos and Tsitsiklis [17] also contains several interesting directions not covered here.

7.1 Approximation

The approximability of SCTP and i-SSPPR is, however, yet to be untangled and the **PSPACE**-completeness result should be an excellent basis for further studies. From the relation $\mathbf{IP} = \mathbf{PSPACE}$, we know, for every $\varepsilon > 0$, that there is a **PSPACE**-hard language of SSAT instances such that every negative instance can be satisfied with probability less than 2^{-n^ε} [8]. This relation seems useful for SCTP but can unfortunately not be immediately applied to our proof because whether or not the instance has been satisfied by the path in the proof contributes to the expected cost with at most $3q\varepsilon = 3 \cdot 2^{-4m-7-(n+2)mh} (m-1)^{-6-2nh} = 2^{-O((n+m)\log n)}$. One could

probably consider a **PSPACE**-hard QSAT-language with a linear number of clauses, but this would not be very impressive either. The randomized debate systems of Condon, Feigenbaum, Lund, and Shor [7] might provide an interactive proof system more suitable for SCTP. It should be noted that it seems much easier to find (but we have not) some interesting lower bounds for the inapproximability of the directed version of SCTP, i.e. i-SSPPPR. For one, the **PSPACE**-hardness proof we provided would not need the $H_{i,j}$ -vertices, which in fact were introduced as gadgets for replacing directed edges. This would immediately drop the factor $q = \bar{p}^{nh}$ from the contribution of not having the path visit each clause, but we have still not managed to produce an inapproximability of even $n^{-\delta}$ for some δ . It seems feasible to produce such a bound if one could start the reduction with a **PSPACE**-hard satisfiability language where only a constant fraction of the clauses could be simultaneously satisfied for negative instances. Although we haven't pursued this line of thought, we presently do not know of any languages of that kind.

To develop an approximation algorithm for SCTP, if one does exist, one may do well to look into the heuristics that have so far been developed, as reviewed in Section 3.3, and to analyze why our **PSPACE**-hardness proof fails, and supposedly could not be modified, to provide inapproximability results.

7.2 Natural Simplifications

For practical applications, we typically only encounter a subfamily of networks. Perhaps our problems are in fact easy for these? There are two possibilities here, either to study distributions where the set of realizations belongs to a particular family, or to study distributions where the graph defined by the probabilistic edges belongs to a particular family. For upper bounds, the latter seems the easiest. Studying planar graph would be one natural assumption, some other ones would be to bound the degree of the graph, to consider grid graphs, to assume that the costs of available edges is a metric or to assume that all edges have a cost of 1 and/or that each of probabilities belongs to $\{0, 0.5, 1\}$ (the cardinality version).

Since both the planar and the cardinality version of the ST-RELIABILITY problem are **P^{#P}**-complete, it follows that the planar and the cardinality SCTP variants are **P^{#P}**-hard. Regarding the metric problem, it is less clear. There are metrics for which the problem is polynomial-time computable, e.g. if all distances are 0. For any l_p norm, the problem is at least **P^{#P}**-hard since one may reduce reliability to it and add a detour of an exponential-polynomial cost greater than the longest optimal path through the input graph divided by the realization of least non-zero probability. A similar construction can be made for grid graphs if we allow exponential-polynomial

edge costs.

For SCTP, the cost of each edge could be seen as a random variable taking on either a predetermined value or ‘ ∞ ’. With parallel edges or dummy vertices, the cost could be given by a discrete distribution with some q number of outcomes. This would however require $\Theta(q)$ space to represent in SCTP. For other problems, e.g. the STOCHASTIC SHORTEST PATH PROBLEM, it has been extensively studied what happens if we allow for other variables – this would certainly be a most interesting extension.

7.3 Computational Hardness

The difficulty of SSPPR-like problems depends on how succinct the description of the random graph model is. For instance, d-SSPPR requires each realization to be explicitly given and is solvable in non-deterministic polynomial time. Perhaps there is a representation which combines the generality of d-SSPPR with the succinctness of SCTP, thereby producing an SSPPR-problem which is considerably harder to approximate than SCTP or even harder than **PSPACE**. Is it possible to predict or bound the difficulty of the problem when considering a different distribution? One starting point would be to consider the universal distribution, which is not computable [11]. Such a study might also provide some insights into how to design a difficult polynomial-time computable distribution.

Is Markov-d-SSPPR **EXPTIME**-hard or can it perhaps be solved in polynomial space? We have not provided any results for Markov-i-SSPPR, i.e. where each edge in the graph is an independent Markov process. Naturally, the problem is at least **PSPACE**-hard. Perhaps one could for this produce gadgets that produce a behavior similar to the one in our proof for Markov-d-SSPPR by letting edges be absorbing two-state Markov chains which have a high expected absorption time. In that case, perhaps one could allow the policy to make a pass through the graph, follow a path of polynomial length, and return to most likely find predetermined edges having gone from passable to impassable and similar in the other direction, combined with edges in parallel and series for multiple detours returning to the same area.

7.4 Polynomial-Time Policies

We have shown that SCTP can not have any polynomial-time policies unless **PSPACE** \subseteq **P/poly**. Our proof constructs an instance of polynomial size such that all instances of a particular size can be answered by a policy for the constructed instance. This idea seems to work for any **PSPACE**-complete problem that we have considered and it feels as though we are missing a simple but fundamental fact. What we can argue is that if we have two

players that play a game of polynomial length and where the result can be determined after the exchange in polynomial time, then there is no policy for the first player unless $\Sigma_2^p = \mathbf{PSPACE}$. Likewise, if the game runs for k rounds for any $k \geq 3$, polynomial-time policies for the game would imply that $\Sigma_2^p = \Pi_2^p = \mathbf{PH}$. This can easily be seen by having a Turing machine guess a polynomial-time policy (which is of polynomial size) and switch into universal mode (i.e. accept iff every guess accepts) to verify that the policy achieves the desired result. The verification would be done by using the guessed policy for the choices of the first player and universal guesses the choices of the second player, and accepting iff the first player wins. It is not clear whether or not a similar argument could be made for $\mathbf{P/poly}$ instead of the second level of the polynomial hierarchy. Note that $\mathbf{PSPACE} \subseteq \mathbf{P/poly}$ is stronger than $\mathbf{PSPACE} \subseteq \Sigma_2^p$, as the former implies the second. It is not known whether or not the converse holds.

For stochastic problems, however, it is not quite as straightforward since computing the probability that a move will result in a win requires powers similar to those of $\#\mathbf{P}$. An argument analogous to the above therefore only yields that no game of polynomial length between a player and nature, as defined in Papadimitriou [14], guarantees a polynomial-time optimal policy unless $\mathbf{P}\#\mathbf{P} = \mathbf{PSPACE}$. In Liberatore [12], so called non-uniform compilability classes were employed to show that there are no polynomial-time policies for markov decision processes (MPD's) unless $\mathbf{NP} \subseteq \mathbf{P/poly}$. This also implies that partially observable markov decision processes (POMDP's) can not have polynomial-time policies unless the same collapse occurs. That SCTP and i-SSPPR can not have polynomial-time policies does not directly follow from this result since SCTP and i-SSPPR are subproblems of POMDP's, not superproblems.

We have not addressed whether or not a polynomial-time policy might exist for approximate policies or the relation between the existence of such policies and the existence of approximation algorithms for SCTP. We have here also considered policies of arbitrary representation, i.e. *any* data that could be used to compute a next action. For, e.g., Markov Decision Processes, there are so called succinct policies that follows more restricted schemes. Whether or not there exists policies of this kind for our problems, without the time restrictions, remains unanswered.

Something that we have not addressed in this paper is the issues that arise when there are realizations such that the goal can not be reached. Considering that this probability is the same regardless of the policy we consider, the issue is not a serious one. There is a number of different ways to deal with the issue in the literature but there are essentially four variants: prohibit such instances, minimize the expected cost for realizations that do have a path to the goal, allow the algorithm to stop and return that there is no path, or allow the algorithm to stop but only at the starting vertex. The second version allows for more distributions than the first and the distinction

could prove important for approximability results. The fourth version seems considerably more complicated than the third version which in turn seems considerably more complicated than the first and second. As far as we can tell, the relations between these variants have not been studied and inapproximability results may be considerably easier to derive for some of them. Note that the common practice of adding an expensive enough edge with probability one from the start to the target belongs to the fourth kind.

7.5 Related Research Directions

The intention behind our work was to study the problem of computing shortest paths in random graphs. We found that this was a diverse field of study with applications in many different fields and with varying conditions and goals. If one must choose and continue to follow a tentative path with the goal of minimizing the expected cost, then the problem is most well-known as the STOCHASTIC SHORTEST PATH PROBLEM, for which there is an extensive literature.

A paper which we believe could be of significant independent interest is Hassin and Zemel [9] where weighted random graphs of each edge uniform in $[0, 1]$ are studied. Among other things, it is shown that one may in $O(|E|) = O(n^2)$ time remove all but $O(n \log n)$ edges of the realization of the graph in such a way that almost surely¹ the costs of shortest paths and minimum spanning trees do not change. This is a very interesting trade-off between accuracy and efficiency that does not clearly correspond to average-case complexity.

¹Where “almost surely” means that the expected number of $n \in \mathbb{N}$ for which it does not hold is finite.

DISCUSSION AND OPEN PROBLEMS

Bibliography

- [1] G. Andreatta and L. Romeo. Stochastic shortest paths with recourse. *Networks*, 18(3):193–204, 1988.
- [2] A. Bar-Noy and B. Schieber. The canadian traveller problem. In *SODA*, pages 261–270, Philadelphia, PA, USA, 1991. Society for Industrial and Applied Mathematics.
- [3] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991. ISSN 0364-765X.
- [4] B. Bonet. Solving stochastic shortest-path problems with rtdp. Technical report, 2002.
- [5] A. J. Briggs, D. Scharstein, and S. D. Abbott. Reliable mobile robot navigation from unreliable visual cues. In *Algorithmic and Computational Robotics: New Directions*, pages 349–362, 2000.
- [6] A. J. Briggs, C. Detweiler, and D. Scharstein. Expected shortest paths for landmark-based robot navigation. *International Journal of Robotics Research*, 23:717–728, 2004.
- [7] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Random debaters and the hardness of approximating stochastic functions. In *SIAM Journal on Computing*, pages 280–293. IEEE Computer Society Press, 1997.
- [8] J. Feigenbaum. Complexity classes, and approximation algorithms. *Documenta Mathematica, Journal of the Deutsche Mathematiker-Vereinigung*, Extra Volume ICM, 1998.
- [9] R. Hassin and E. Zemel. On shortest paths in graphs with random weights. In *Mathematics of Operations Research*, volume 10, 1985.

BIBLIOGRAPHY

- [10] D. Karger and E. Nikolova. Exact algorithms for the canadian traveller problem on paths and trees. Technical report, MIT Computer Science & AI Lab, 2008. URL <http://hdl.handle.net/1721.1/40093>.
- [11] M. Li, P. M. B. Vit'anyi, and F. W. E. Informatica. Average case complexity under the universal distribution equals worst-case complexity. *Information Processing Letters*, 42:145–149, 1992.
- [12] P. Liberatore. The size of mdp factored policies. In *Eighteenth national conference on Artificial intelligence*, pages 267–272, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.
- [13] E. Nikolova and D. R. Karger. Route planning under uncertainty: The canadian traveller problem. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 969–974, 2008.
- [14] C. H. Papadimitriou. Games against nature. *J. Comput. Syst. Sci.*, 31(2):288–301, 1985. ISSN 0022-0000. doi: [http://dx.doi.org/10.1016/0022-0000\(85\)90045-5](http://dx.doi.org/10.1016/0022-0000(85)90045-5).
- [15] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *ICALP*, pages 610–620, London, UK, 1989. Springer-Verlag.
- [16] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. Technical report, MIT Lab. for Information and Decision Systems, 1993. URL <http://hdl.handle.net/1721.1/3319>.
- [17] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27:133–143, 1996.
- [18] J. S. Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003.
- [19] M. L. Puternam. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.
- [20] S. Westphal. A note on the k -canadian traveller problem. *Inf. Process. Lett.*, 106(3):87–89, 2008. ISSN 0020-0190.
- [21] Y. Xu, M. Hu, B. Su, B. Zhu, and Z. Zhu. The canadian traveller problem and its competitive analysis. *Journal of Combinatorial Optimization*, 2008.