Theoretical Aspects of the Generalized Canadian Traveler Problem

Thesis submitted in partial fulfillment
of the requirements for the degree of
"DOCTOR OF PHILOSOPHY"

by
Dror Fried

Submitted to the Senate of Ben-Gurion University of the Negev

July 2013
Be'er-Sheva

Theoretical Aspects of the Generalized Canadian Traveler Problem

Thesis submitted in partial fulfillment
of the requirements for the degree of
"DOCTOR OF PHILOSOPHY"

by
Dror Fried

Submitted to the Senate of Ben-Gurion University of the Negev

Approved by the Advisor, Prof. Eyal shimony	[illegible]	MAY 28, 2014
Approved by the Dean of the Kreitman School of Advanced Graduate Studies		

July 2013
Be'er-Sheva

This work was carried out under the supervision of

Professor Eyal Shimony

In the Department of Computer Science
Faculty of Natural Sciences

<u>Research-Student's Affidavit when Submitting the Doctoral Thesis for Judgment</u>

I Dror Fried, whose signature appears below, hereby declare that
(Please mark the appropriate statements):

☑ I have written this Thesis by myself, except for the help and guidance offered by my Thesis Advisors.

☑ The scientific materials included in this Thesis are products of my own research, culled <u>from the period during which I was a research student</u>.

___ This Thesis incorporates research materials produced in cooperation with others, excluding the technical help commonly received during experimental work. Therefore, I am attaching another affidavit stating the contributions made by myself and the other participants in this research, which has been approved by them and submitted with their approval.

Date: 27.5.2014 Student's name: Dror Fried Signature: [illegible]

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Eyal Shimony, for his continuous support in my Ph.D. studies and research. His many advices about my research, and about scientific research in Computer Science have been invaluable, and I will carry these with me. I appreciate all his contributions of time, energy, and funding, to make my Ph.D. research productive. I am positive that his experience and guidance helped me a lot to grow as a research scientist.

In addition I would like to thank my former M.Sc. advisors, Professor Uri Abraham and Professor Matatyahu Rubin, for their ongoing care and support throughout my M.Sc. and my Ph.D. studies.

I would like to thank the Israel Science Foundation grant 305/09, and the Lynn and William Frankel Center for Computer Science for their support throughout these years.

I would like to thank my collaborators for their help and contribution in my research: Gal Amram, Amit Benbassat, Zahy Bnaya, Prof. Ariel Felner, Yaron Gonen, Dr. Yuval Itan, Olga Maksin, Cenny Wenner, Dr. Gera Weiss, and Doron Zarchy.

In addition, I would like to thank all my colleagues and friends from the Department of Computer Science, who have helped me a lot throughout these years. Specifically I would like to thank Dr. Gila Morgenstern for her help, and to Udi Apsel, for all the invaluable brainstorming we have had.

I would like to thank my family. My dear parents, Israel Fried and Rachel Fried, who supported me throughout my life, and taught me not to be afraid of asking the hard questions; to my sister Yael Fried for her care, and to

my mother and father in-law Shlomit and Itzhak Gilboa for their care and support. Finally I would like to thank my dear wife Sagit, who put a tremendous effort to pull this cart up the hill (sometimes with me in it), and to my dearest Itay, Nir, and Roei, who went to bed too many times waiting for their dad to come home. Well kids, this is what it was all for.

To Sagit and to my parents,

Contents

Abstract

A basic question in navigation is how can the planner devise a path to a destination when parts of the roads might be blocked, and the planner knows whether a road is blocked only when actually reaching that road. A possible formalization of this problem is the *Canadian Traveler Problem* (CTP) in which a traveling agent is given a weighted graph with a given source and destination. Although the graph is known, each edge may be blocked with a known probability; the agent learns that an edge is blocked only upon reaching a vertex incident on that edge. The problem is to find a policy for the agent to travel from the source to the destination, which minimizes the expected travel cost.

In this work we study theoretical aspects, and various variants, of the CTP. As the CTP is a problem in decision-making under uncertainty, we model the CTP as a Partially Observable Markov Decision Process (POMDP). Using this model we can observe and analyze various policies, and by doing that we construct optimal policies that solve the CTP.

Originally stated by Papadimitriou and Yannakakis [31], the adversarial version of the CTP was shown to be PSPACE-complete, with the stochastic version shown to be in PSPACE and #P-hard. We first show that the stochastic CTP is also PSPACE-complete: initially proving PSPACE-hardness for the dependent version of the stochastic CTP (called CTP-Dep), and proceeding with gadgets that allow us to extend the proof to the independent case. This result, published in [14], settles a question that was open for two decades.

A common approach in Computer Science, called "divide and conquer", is to find an optimal solution to a problem by decomposing the problem into sub-problems, and finding an optimal solution to each sub-problem. In this work, we indeed suggest a decomposition method for the CTP. Since the "divide and conquer" approach does not necessarily achieve optimal policies on general CTP-graphs, we define specific constraints that every policy has to meet in order to use the "'divide and conquer"' approach. We introduce a CTP variant, in which every policy for a CTP instance "must" solve certain CTP sub-instances as well. By defining the *factored-cost* of CTP sub-instances, we introduce the *partition framework* through which a CTP instance can (not always efficiently) be decomposed into sub-instances. Then, a general optimal solution can be efficiently found by finding an optimal solution to each sub-instance.

Another CTP variant introduced in this work, called CTP-Tree, is the CTP on a tree-like structure. CTP-Tree is a generalization of the CTP on a disjoint-paths graph, as appeared in [6]. We define the concept of *committing vertex* in which the agent is bound to explore an entire subtree with a given vertex being the root. Using the partition framework, we first provide an algorithm that yields polynomial time solution to CTP-Tree in which all vertices are committing. Using this result we provide an efficient dynamic programming algorithm for CTP-Tree in which all vertices but one (with unblocked outgoing edges) are committing. In addition, we provide a polynomial time solution to a specific CTP-Tree, called EFC-CTP-Tree, in which all the factored-cost of subtrees of the same height are equal. We test empirically how well such solutions to EFC-CTP-Tree approximate optimal solutions to the more general CTP-Tree.

Finally, in many realistic settings, the CTP needs to be solved for a group of agents moving sequentially, requiring minimization of the combined travel cost of all agents. For example, think of an owner of a fleet of trucks who has to move the trucks, one after the other, from a single source to a single destination. We introduce a multi-agent variant of the CTP, called the Repeated-CTP, in which an agent moves only after its predecessor has reached the destination. We provide efficient optimal solutions to the Repeated-CTP on disjoint-path graphs. This result appeared in [7].

Keywords: Canadian Traveler Problem, Navigation under Uncertainty, Stochastic Shortest Path with Recourse.

Chapter 1

Introduction

The Canadian Traveler Problem (CTP) is a problem in navigation under uncertainty. Given a graph, an agent is initially posed at a start vertex. By performing move actions along the edges, the agent has to reach a goal vertex. Suppose that the agent has complete knowledge of the graph structure and the cost of the edges. However, some of the edges might be blocked with a certain probability, and the agent observes that an edge is blocked only when the agent reaches a vertex incident on that edge. The task is to minimize the travel cost from the start to the goal. Since some of the graphs edges may be blocked, a simple search for a path does not work; a solution is a policy that has the smallest expected traversal cost.

Motivation for the CTP comes from problems in real life. Consider, for example, the following navigation problem. The planner might be familiar with the map (e.g., Canada), and with the cost of the roads, whether the cost is the length of the road or the time to traverse the road; still, the planner has only limited knowledge concerning the current status of the roads. A certain road might be blocked (e.g., snow), and the planner has no way of knowing it, before actually reaching that road. Hence search algorithms that find the shortest path in a graph might be useless. The question the planner asks is what is the plan for choosing the roads that ensures the minimum expected cost to reach its destination.

In this work we discuss some of the deep theoretical challenges in the

CTP. The exact complexity class of the CTP has remained unsolved for more than two decades. We settle the issue in this dissertation, proving that the CTP is PSPACE-complete. As the CTP is a classical problem in decision making under uncertainty, we model the CTP as a Partially Observable Markov Decision Process (POMDP). That way, we can carefully construct policies, and define various variants of the CTP. We then use these variants to define "divide and conquer" methods for the CTP. Later, we implement these methods in a special tree-like structure CTP called CTP-Tree. In addition, we introduce several variants of the CTP, which we analyze theoretically, and provide polynomial time algorithms to specific CTP instances.

Dissertation structure. This dissertation is organized as follows. In Chapter 2 we provide notation and background on models of decision making under uncertainty. In Chapter 3 we discuss the complexity class of the CTP and show that the CTP is PSPACE-complete. In Chapter 4 we discuss decompositions of the CTP, and introduce the so called "partition framework". In Chapter 5 we introduce CTP-Tree, and implement some of the techniques gained in Chapter 4 to provide optimal solutions for special CTP-Tree instances. In Chapter 6 we introduce a variant of multi-agent CTP called Repeated-CTP, and provide an optimal solution for Repeated-CTP with a disjoint paths graph.

Chapter 2

Background

2.1 Notation

Graphs. A graph G is an ordered pair (V, E) where V is the set of vertices, and $E \subseteq V \times V$ is the set of edges. A graph $G' = (V', E')$ is called a subgraph of $G = (V, E)$ if $V' \subseteq V$, and $E' \subseteq E$. A weighted graph is a graph with a (non-negative) weight function $w : E \to \Re^{\geq 0}$ over the edges. We denote the set of edges incident on a vertex v by E_v.

Trees. A (rooted) tree $T = (V, E)$ is a connected acyclic graph, with a designated vertex $r \in V$ called the *root*. The sequence of vertices that form a path from the root to v is called the *trunk* of v. That is, $H_v = (v_0, \cdots v_l)$ is the trunk of v if H_v form a simple path in T, $v_0 = r$, and $v_l = v$. The *splitting vertex* of vertices v and u with trunks $H_v = (v_0, \cdots v_l)$, and $H_u = (u_0, \cdots u_j)$ is the maximum $k \geq 0$ such that $v_k = u_k$.

For a tree T and $u, v \in V$, if $u \neq v$, and v is in the trunk of u, then v is called an *ancestor* of u, and u is called a *descendant* of v. We also say u is *reachable* from v. If v is an ancestor of u and $(v, u) \in E$, then v is called the *parent* of u, and u is called a *child* of v. Vertices in T that have the same parent are called *siblings*. If u is a child of v, and z is a child of u, then z is called a *grandchild* of v, and v is the *grandparent* of z. The parent of u is denoted by $Parent(u)$. If $v = Parent(u)$, then (v, u) is called an outgoing edge of v and an incoming edge of u. An *intermediate vertex* in T is a vertex with at least 2 outgoing edges. We define a partial order $\preceq_T$ over $V \times V$ such that $u \preceq_T v$ if u is a descendant of v. If $u \preceq_T v$ then the *distance* from v to u is the number of edges in the simple path from v to u.

A *leaf* in a tree T is a vertex without children. The *depth* of a vertex $v \in T$, denoted by $depth(v)$, is the number of vertices in the trunk of v minus 1. Note that $depth(r) = 0$. $Depth(T)$, the depth of the tree, is defined to be $max_{v \in T} depth(v)$. The *height* of a vertex v, denoted by $height(v)$, is the largest distance from v to a leaf l such that $l \preceq_T v$. The height of T, $Height(T)$, is defined to be the height of the root, $height(r)$. $Rank(v)$ is the set of all vertices $u \in T$ for which $depth(u) = depth(v)$. A *balanced* tree is a tree in which every two vertices of the same depth have the same height. A *cut* in T is a set $S \subseteq T$ such that $u \not\preceq v$ and $v \not\preceq u$ for every $u, v \in S$. S is a *maximal cut* in T, if S is a cut in T, and every vertex $v \notin S$ is either an ancestor or a descendant of a vertex in S. For a vertex $v \in V$, $T(v) = \{u | u \preceq v\}$ is the subtree of T with a root v. If $v \neq r$, then the subtree $T^{Par}(v)$ is defined to be $T(v)$ with an additional vertex $Parent(v)$ and an additional edge $(Parent(v), v)$. The size of the tree T is the number of vertices in T.

Functions. Given functions f, f' from A to $\Re$, we say that $f' \leq f$ is $f'(a) \leq f(a)$ for every $a \in A$. For a function $f : A \to B$, and $A' \subseteq A$, the *restriction* of f to A' is the function $f \upharpoonright A' : A' \to B$ where $(f \upharpoonright A')(a) = f(a)$ for all $a \in A'$.

2.2 The Canadian Traveler Problem

The *Canadian Traveler Problem* (CTP), first defined by Papadimitriou & Yannakakis [31], is a tuple (G, s, t, p, w), where $G = (V, E)$ is a finite connected undirected weighted graph, with a source vertex s, and a target vertex t. Every edge $e \in E$ has a non-negative cost $w(e)$, and a probability (independent for all the edges) $p(e)$ of being blocked. The probability that e is unblocked is denoted by $q(e) = 1 - p(e)$. Starting at s, an agent can traverse unblocked edges for a cost of $w(e)$. The status of an edge (*blocked*,*unblocked*) is revealed to the agent only when the agent arrives at a vertex incident on that edge, and this status of the edge remains fixed subsequently. The goal of the agent is to reach t while minimizing the total travel cost, which is the sum of the cost of the edges that the agent traversed to reach vertex t. As the exact travel cost is uncertain until t is reached, the task is to devise a policy that minimizes the expected travel cost. Such policy is called an *optimal policy*

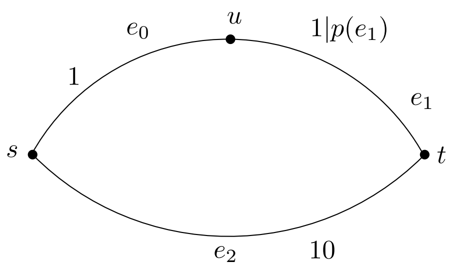


Figure 2.1: A simple CTP instance. $w|p$ denotes cost | blocking probability.

For example, Figure 2.1 depicts a simple CTP instance. The agent at s has two "reasonable" policies to choose from: the first policy, π_1, is to reach t by traversing the unblocked edge e_2 for a cost of 10. The expected cost of π_1 is 10. The second policy, π_2, is to traverse e_0 and observe e_1; if e_1 is unblocked, traverse e_1 and reach t. However, if e_1 is blocked, then traverse e_0 back to s and reach t by traversing e_2. The expected cost of π_2 is

$$1 + (1 - p(e_1)) + p(e_1)(1 + 10)$$

Other policies in which the agent traverses e_0 back and forth regardless of whether e_0 is unblocked, are clearly not optimal. This example is a simple case of the CTP on disjoint path graphs, discussed in Section 2.5. In this example π_1 is optimal if and only if

$$10 \leq 1 + (1 - p(e_1)) + p(e_1)(1 + 10)$$

that is if and only if $p(e_1) \geq 0.8$.

Since the size of an optimal policy is potentially exponential in the size of the problem description, we state that the objective in the CTP is finding the first move in an optimal policy. The CTP can also be stated as the *CTP decision problem* stated as follows. Given an instance of the CTP, and an edge e incident on s, does there exist an optimal policy where traversing e is the first move?

The Canadian Traveler Problem is essentially a problem of sequential decision making under uncertainty. Therefore we next give definitions of models for decision making under uncertainty, followed by the description of the CTP as such a model.

2.3 Decision making under uncertainty

We repeat the definitions of *Markov Decision Process* (MDP), and Partially Observable Markov Decision Process (POMDP) [36, 19, 1]. We then give the definition of Deterministic-POMDP (Det-POMDP), which is a special case of POMDP [26, 8].

2.3.1 Markov Decision Process (MDP)

A *Markov Decision Process* (*MDP*) is a specification of a sequential decision problem for a fully observable environment with a Markovian transition model, and additive rewards. Formally, an MDP M is a tuple (S, A, T, R) defined as follows. S is a (finite) set of configurations of the environment called *states*. A is a (finite) set of *actions*, which can be performed at various states. $T : S \times A \times S \to [0, 1]$ is the *transition function*, where $T(s, a, s')$ is the probability of reaching state s' if action a is performed in state s. We assume that the transitions are *Markovian*, in the sense that the probability of reaching s' from s depends only on s and a, and not on a history of earlier states. Finally, $R : S \times A \times S \to \Re$ is called a *reward function*, where $R(s, a, s')$ is the reward obtained when reaching state s' from state s by performing action a. Note that this reward can be either negative or positive, but must be bounded. The initial state of the environment is denoted by s_0.

A solution to an MDP, called a *policy*, is a function $\pi : S \to A$, which is a specification of what action the agent should perform in every possible state. We call the sequence of states derived by actions performed so far an *environment history* of the policy. The value of a policy π is measured by the expected utility over the possible environment histories generated by π, starting from s_0. An *optimal policy* π^*, is a policy that yields the highest expected utility. Sometimes the reward function is a negative *cost function*, and then an optimal policy is defined to be a policy that yields the *lowest* expected cost, called the *optimal cost*.

For an MDP M, we define a *horizon* as the number of time steps until M terminates, where every action performed by the agent can be considered as a time step. If the horizon is bounded, we say M is a *finite horizon* MDP. In a finite horizon MDP, the optimal action in a given state is time dependent. If, on the other hand, the horizon of M is unbounded, M is called an *infinite horizon* MDP. In an infinite horizon MDP, there is no reason to act differently in the same state at different times. Hence the optimal action in every given state depends only on the current state.

A special case of an infinite horizon MDP is where every optimal policy terminates after a finite number of states with probability 1, but the number of steps until termination is uncertain, and unbounded [1]. Such MDP is called an *indefinite horizon* MDP. In an indefinite horizon MDP, terminal states are defined. A set of *terminal states* is a subset $K \subseteq S$ in which for every $k \in K$, and $a \in A$, we have $T(k, a, k) = 1$, and $R(k, a, k) = 0$. The Canadian Traveler Problem is a special case of an indefinite horizon MDP.

Given a policy π, the utility of a state sequence $[s_0, s_1, \cdots]$ is

$$U^\pi([s_0, s_1, \cdots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \tag{2.1}$$

where $\gamma \in [0, 1]$. This utility is called *discounted reward* and has a *discount factor* γ. For $\gamma = 1$ the utility is called *additive (undiscounted) reward.* Then given a policy π for an MDP, The utility for every state, denoted by $U^\pi(s)$, can be computed by using the Bellman equation [4], and we have:

$$U^\pi(s) = \sum_{s'} \Big(T(s, \pi(s), s') \big(R(s, \pi(s), s') + \gamma U(s') \big) \Big) \tag{2.2}$$

The utilities of the states are assigned iteratively, and are updated until they finally converge to a unique solution, $V^{\pi}(s)$, for every state. The expected cost of π is defined to be $V^{\pi}(s_0)$, and is denoted by $C(\pi)$. Then a policy π^* for an MDP M is an optimal policy if and only if $C(\pi^*) \geq C(\pi)$ for every policy π for M. In case of a cost function, π^* is optimal if and only if $C(\pi^*) \leq C(\pi)$ for every policy π for M.

When computing the utility of a state, or of a state sequence, the discount factor is usually 1 in a finite horizon MDP, and less than 1 in an infinite horizon MDP - so the utility computations in Equations (2.1), and (2.2) converge. However, in an indefinite horizon MDP, the discount factor can be 1, as we are interested only in policies that terminate after a finite time; thus the utility computations converge.

2.3.2 Partially Observable Markov Decision Process (POMDP)

In MDP we assume that the environment is fully observable; that is - the agent always knows the state of the environment. Combined with the Markovian assumption, an optimal policy depends only on the current configuration of the environment. However, when the environment is partially observable, the agent does not necessarily know the state of the environment; therefore a different type of model is needed.

A *Partially Observable Markov Decision Process* (POMDP) , is a specification of a sequential decision problem, much like MDP, with several additions. Formally a POMDP is a tuple $M = (S, A, T, R, Z, O, b_0)$, where (S, A, T, R) is an MDP called the *underlying MDP* of M. In addition, Z is a (finite) set of elements called *observations*, with an *observation distribution* function $O : S \times A \times Z \rightarrow [0, 1]$ such that $O(s, a, o)$ is the probability of receiving observation o when the state s is reached **after** performing action a.

By receiving observations after performing an action, the agent obtains some knowledge about the true state of the environment. This knowledge is represented as a *belief state*, which is a probability distribution over the set of states S. For a belief state b, and a state s, $b(s)$ represents the probability that the environment is in state s. The initial belief state is denoted by b_0. The belief state space of M is denoted by B_M. As the agent does not know the exact state of the environment, he must perform actions based on his current belief state of the environment.

Let $p(o|a,b)$ be the probability of receiving observation o, once action a was performed in belief state b. Then

$$p(o|a,b) = \sum_{s' \in S} \Big(O(s',a,o) \sum_{s \in S} \big(T(s,a,s') b(s) \big) \Big) \tag{2.3}$$

Given a belief state b, and an observation o received after performing action a, the new belief state, denoted by b_a^o , can be computed as follows. If $p(o|a,b) = 0$, then $b_a^o(s') = 0$. Otherwise for every state s',

$$b_a^o(s') = \frac{1}{p(o|a,b)} O(s',a,o) \sum_{s \in S} \big(T(s,a,s') b(s) \big) \tag{2.4}$$

Note that b_a^o is a probability distribution over S, and therefore is a belief state as well.

It is important to note that in an optimal policy for a POMDP M, the optimal action depends only on the agent's current belief state [36]. Hence an optimal policy can be described as $\pi : B_M \rightarrow A$, and the process can be specified as a new MDP N, called the *belief-state MDP* of M, by using B_M, as the state space for N. The initial state for N is therefore b_0. Thus we see that solving a POMDP can be reduced to solving an MDP; the utility of a policy π at a given belief state b is $V^\pi(b)$, and the cost of π, $C(\pi)$, is $V^\pi(b_0)$. However, as B_M is a set of probability distributions over S, the state space of N can be significantly large, hence MDP algorithms are not efficient on POMDPs. The problem of solving POMDPs, and even finding approximately optimal policies, is intractable in the size of the POMDP.

2.3.3 Deterministic-POMDP

A special case of POMDP, called Deterministic POMDP (Det-POMDP), is when the actions and the observations of the POMDP model are both deterministic. First introduced by Littman [26], Det-POMDP captures many important problems, one of which is the CTP. The following formal definition of Det-POMDP is due to Bonet [8].

A *Deterministic POMDP* is a POMDP $M = (S, A, T, R, Z, O, b_0)$ with the following restrictions.

1. The transition function T is deterministic. That is, there is a unique state that is reached after an action a is performed in a state s. Formally, for every state s, and action a, there is a state s' such that $T(s, a, s') = 1$.

2. The observation function O is deterministic. That is, there is a unique observation that is received after reaching a state s by performing action a. Formally, for every state s, and an action a, there is an observation o such that $O(s, a, o) = 1$.

3. M has an indefinite horizon. The set of terminal states of M is denoted by K.

In a Det-POMDP a cost function is used instead of a reward function. Note that as the actions and observations in Det-POMDP are deterministic, the only source of uncertainty in Det-POMDPs comes from the initial belief state. However, the belief-state MDP representation of a Det-POMDP is no longer deterministic.

For a belief state b, and an action a, we define a distribution b_a, called an *intermediate belief state* such that for every $s' \in S$,

$$b_a(s') = \sum_{s \in S} T(s, a, s') b(s) \tag{2.5}$$

No actions are preformed in intermediate belief states as they are changed when observations are received. As the observations are deterministic as well, then for a state s and an observation $o \in Z$, we have:

$$b_a^o(s) = \begin{cases} 0, & \text{if } p(o|a,b) = 0 \text{ or } O(s,a,o) = 0; \\ \frac{b_a(s)}{p(o|a,b)}, & \text{otherwise.} \end{cases} \tag{2.6}$$

Let $sup(b)$ be the *support* of a belief state b, meaning $sup(b) = \{s \in S \mid b(s) > 0\}$. We denote the set of actions that the agent can perform in state s by $A_s \subseteq A$. For a belief state b we define $A_b = \bigcap_{s \in sup(b)} A_s$ as the set of actions that can be performed in belief state b.

We say a belief state b' is *reachable* from a belief state b if b' is reached from a consecutive series of actions and observation that starts at b. We say b' is reachable from a set of belief states B_1 if there is a belief state $b \in B_1$ such that b' is reachable from b. We say b' is reachable in a policy π from b, if b' is reachable from b through a series of actions and observation that starts at b, in which every belief state is reached after performing an action in π, and receiving a certain observation. We denote by $B_M(b)$ the set of belief states in B_M that are reachable from a belief state b by performing only actions from A. Similarly, we denote the set of belief states that are reachable in a policy π from the belief state b, by $B_M(b, \pi)$. We denote a partial order $\leq_\pi$ on $B_M(b_0, \pi)$ such that $b' \leq_\pi b$ iff $b' \in B_M(b, \pi)$, that is b' is reachable from b in π. Finally, we say belief states b, b' are *separated* in π if $b \not\leq_\pi b'$ and $b' \not\leq_\pi b$.

2.3.4 Weighted AND/OR Trees

A *Weighted AND/OR tree* (W-AND/OR tree) is a weighted directed tree $T = (V, E, c, p, r)$. V is set of *nodes* in T, and $E \subseteq V \times V$ is a set of *arcs* in T[1]. In addition we have the following notations.

1. $V = V_{AND} \cup V_{OR}$, where V_{AND} (called *AND-nodes*), and V_{OR} (called *OR-nodes*) are finite, and disjoint sets of nodes. $r \in V_{OR}$ is the root of T.

2. $E = E_{AND} \cup E_{OR}$ where $E_{AND} \subseteq (V_{AND} \times V_{OR})$ (called *AND-arcs*),

[1]Note that the graph elements in the CTP graph are called "vertices" and "edges".

and a $E_{OR} \subseteq (V_{OR} \times V_{AND})$ (called *OR-arcs*) are finite, and disjoint, sets of arcs.

3. c is a non-negative cost function defined over the OR-arcs.

4. p is a probability function defined over the AND-arcs, such that for every $n \in V_{AND}$ we have:

$$\sum_{(n,n') \in E_{AND}} p((n, n')) = 1$$

Note that this definition of W-AND/OR trees resembles Expectimax trees, in which the AND nodes are called *chance nodes*; see [36].

A Det-POMDP $M = (S, A, T, R, Z, O, b_0)$ can be described as a labeled W-AND/OR tree $T_M = (V, E, c, p, r, L)$, such that (V, E, c, p, r) is a W-AND/OR tree and L is a label function from V to B_M, and from E to $A \cup Z$ as follows. $L(v) \in B_M$ for every node v, $L(e) \in A$ for every OR-arc e, and $L(e) \in Z$ for every AND-arc e. T_M is constructed as follows:

- $L(r) = b_0$.
- If $v \in V_{OR}$ then the outgoing arcs of v are OR-arcs stated as follows. For every action $a \in A_b$, where $b = L(v)$, there is a single outgoing OR-arc $e = (v, v')$, such that $v' \in V_{AND}$, $L(e) = a$, and $L(v') = b_a$.

  The OR-arcs are also called *action-arcs*, or *arcs for action* a when specifically related to an action a. We set

$$c((v, v')) = \sum_{i,i' \in S} b(i)R(i, a, i')T(i, a, i') \tag{2.7}$$

- If $v \in V_{AND}$ then the outgoing arcs of v are AND-arcs stated as follows. For every observation o that can be received at $L(v) = b_a$, there is a single outgoing AND-arc $e = (v, v')$, such that $v' \in V_{OR}$, $L(e) = o$, and $L(v') = b_a^o$. The AND-arcs are also called *observation arcs*, or *arcs for observation* o when specifically related to an observation o. We set $p(e) = p(o|a, b)$.

Next, we define the policy tree T_π for a policy π for M. A subtree T_π of T_M describes a policy π for M if:

- $r \in T_\pi$.
- If $n \in T_\pi$ is an AND node, then all the outgoing arcs of n are in T_π.
- If $n \in T_\pi$ is an OR node, then exactly one of the outgoing arcs of n is in T_π. If $L(n) = b$ then the outgoing arc of n is an arc for the action $\pi(b)$.

As the sets of observations and actions are finite, we have that T_π is well defined. All the policies throughout this work terminate after finite time; therefore T_π is considered finite [2]. We now describe the expected cost $C(\pi)$ of π. For every node v in T_π, $V^\pi(v)$ is defined recursively as follows:

$$V^\pi(v) = \begin{cases} 0, & \text{if } v \text{ is a leaf} \\ \sum_{(v,u)\in E}(p((v,u))V^\pi(u)), & \text{if } v \in V_{AND} \\ c((v,u)) + V^\pi(u), & \text{if } v \in V_{OR} \end{cases} \tag{2.8}$$

Then $C(\pi) = V^\pi(r)$.

Finally, we provide a formal definition to a "partial policy". Recall that $T(v)$ is the subtree of T with a root v. For a Det-POMDP $M = (S, A, T, R, Z, O, b_0)$ and $b \in B_M$, let M_b be the Det-POMDP (S, A, T, R, Z, O, b). Note that M_b is indeed a Det-POMDP, $B_{M_b} = B_M(b)$, and if $L(v) = b$ then $T_{M_b} = T(v)$. Now assume that $b \in B_M(b_0, \pi)$ for a given policy π for M. Then $B_M(b, \pi) \subseteq B_{M_b}$, and $T_\pi(v)$ describes a policy π_b for M_b. π_b is called a *partial policy* of M. From Equation (2.8), we have that $C(\pi_b) = V^\pi(v)$.

Although two distinct nodes v, v' can be labeled by the same belief state b, it follows by construction that $T(v)$ and $T(v')$ are identical, as well as $T_\pi(v)$ and $T_\pi(v')$; therefore $V^\pi(v) = V^\pi(v')$. For this reason, and as T_π is finite, we may assume that if $u, v \in T_\pi$ and u is a descendant of v then $L(u) \neq L(v)$.

[2] Policies in the CTP that do not terminate are clearly not optimal.

Claim 2.3.1 *If π^* is an optimal policy for a Det-POMDP M, and $b \in B_M(b_0, \pi^*)$, then π^*_b is an optimal policy for M_b.*

The proof of claim 2.3.1 follows easily from Equation 2.8. In Appendix A we give a detailed proof in which we use backward induction along the trunk of a vertex $v \in T_\pi$ with $L(v) = b$. This technical method appears in several proofs throughout this work; we refer the reader to this proof for the exact technical details.

2.4 Det-POMDP definition for the CTP

Given a CTP instance I, we define the following Det-POMDP $M_I = (S, A, T, R, Z, 0, b_0)$ as follows:

- The set of states S is defined to be $V \times \prod_{e \in E}\{blocked, unblocked\}$. For every edge e, $Y_e : S \rightarrow \{blocked, unblocked\}$ is a function in which $Y_e(l)$ denotes the status of edge e in state l. If $Y_e(l) = blocked$ (respectively $Y_e(l) = unblocked$), we say that e is *blocked* (respectively *unblocked*) in l. In addition, $loc : S \rightarrow \{v_1, \cdots, v_n\}$ is a function in which $loc(l)$ denotes the location of the agent in state l. If $loc(l) = v$, we say that the agent is *at vertex v* in state l. A terminal state, henceforth called a *goal state*, is defined to be any state in which the agent is at vertex t.

- For each edge $e \in E$, we define an action $Move(e)$. Given states l, l', and an action $a = Move(e)$, where $e = (v, w)$, we define the transition function, T, as follows. $T(l, a, l') = 1$ if e is unblocked at l, $Y_{e'}(l) = Y_{e'}(l')$ for every edge $e' \in E$, and in addition $loc(l) = v$ and $loc(l') = w$. Otherwise $T(l, a, l') = 0$.

- Given states l, l', and an action $a = Move(e)$ for some $e \in E$, let $R(l, a, l') = w(e)$ in all cases where $T(l, a, l') = 1$, and 0 otherwise.

- The set of observations Z is a set of subsets of $\{(e, i) \mid e \in E, i \in \{blocked, unblocked\}\}$. For a state l, a *move* action a, and an observation o, the observation distribution $O(l, a, o)$ is :

$$O(l,a,o) = \begin{cases} 1, & \text{if } loc(l) = v \text{, and } o = \{(e,i) \mid e \in E_v, i = Y_e(l)\} \\ 0, & \text{otherwise.} \end{cases}$$

- The initial belief state, b_0, is defined as follows. If $loc(l) \neq s$ then $b_0(l) = 0$. Otherwise, let $E_b = \{e \in E \mid Y_e(l) = 0\}$ and $E_{ub} = \{e \in E \mid Y_e(l) = 1\}$. Then

$$b_0(l) = \prod_{e \in E_b} p(e) \prod_{e \in E_{ub}} (1 - p(e)) \tag{2.9}$$

We abuse notation by denoting the elements of a Det-POMDP M_I, for a CTP instance I, as elements of I. For example, we denote the belief state space of M_I by B_I rather than B_{M_I}.

An edge e is *blocked* (respectively *unblocked*) in a belief state b if e is blocked (respectively unblocked) in l for every state $l \in sup(b)$. An edge that is neither blocked nor unblocked in b is *unknown* in b. Likewise, we say the agent is *located* in v in b if $loc(l) = v$ for every state $l \in sup(b)$. We define a partial function $Loc : B_I \to V$ such that $Loc(b) = v$ when the agent is located in v in b. If $Loc(b) = t$ then $b \in B_I$ is called a *terminal belief state*. As a terminal belief state is reached after performing an action, we have that all terminal belief states are intermediate belief states. Therefore, if T_π is a policy tree for a policy π for I, then the leaves of T_π are all AND-nodes. Unless mentioned otherwise, the edges incident on s are unblocked in every CTP instance.

For example, Figure 2.2 describes a Det-POMDP representation for the CTP instance I in Figure 2.1, in a form of a Weighted And/OR Tree. The square nodes are OR-nodes, the round nodes are AND-nodes. Every terminal belief state (in which the agent is for certain in t) is in bold. Two actions are allowed at the initial belief state b_0: $move(e_0)$ and $move(e_2)$. If the action $move(e_0)$ is performed, for a cost of $w(e_0)$, then two observations are obtained: o_1: "e_1 is unblocked", and o_2: "‘e_1 is blocked". o_1 is obtained with probability $1 - p(e_1)$, and o_2 with probability $p(e_1)$, and so on. Figure 2.3 describes the following policy for I: traverse e_0 and observe e_1; if e_1 is unblocked, traverse e_1 and reach t. However, if e_1 is blocked, then traverse e_0 back to s and then reach t by traversing e_2.

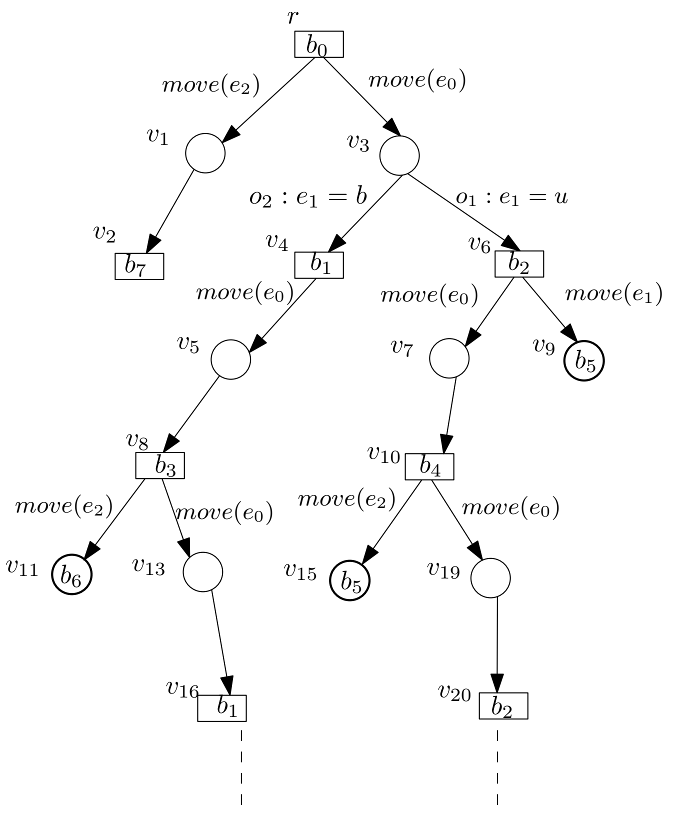


Figure 2.2: A Det-POMDP representation of the CTP instance in Figure 2.1. The square nodes are OR-nodes, the round nodes are AND-nodes.

Note that every state includes the specific location of the agent and the exact status of each edge. Therefore the size of the state-space of the CTP is at most $V \times 2^{|E|}$.

Alternative representations for belief states of the CTP. The status (*blocked*,*unblocked*,*unknown*) of an edge e in b is denoted by $b|e$ and is called the belief status of e. Given the status of the edges at a belief state b, we can compute $b(l)$, for a state l, as follows. $b(l) = 0$ if at least one of the following holds:

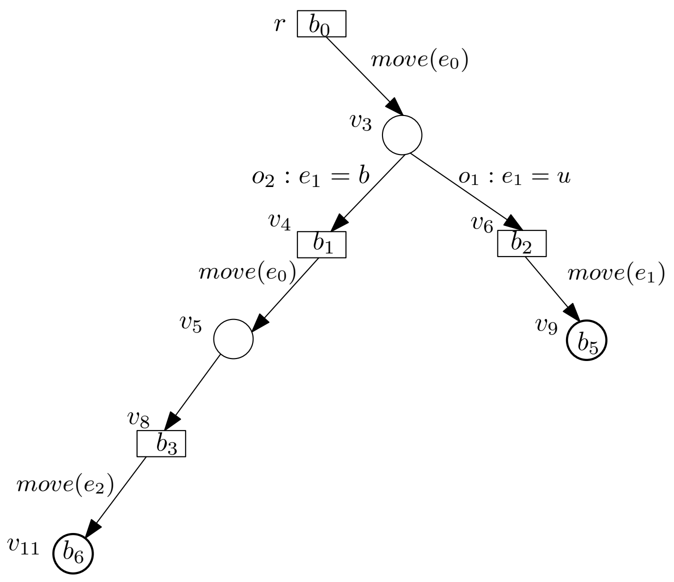


Figure 2.3: A policy description for the CTP instance I.

- $loc(l) \neq Loc(b)$.
- $b|e = blocked$, and $Y_e(l) = unblocked$.
- $b|e = unblocked$, and $Y_e(l) = blocked$.

Otherwise, let $E_1 = \{e \in E \mid b|e = unknown \text{ and } Y_e(l) = blocked\}$, and $E_2 = \{e \in E \mid b|e = unknown \text{ and } Y_e(l) = unblocked\}$. Then we have

$$b(l) = \prod_{e \in E_1} p(e) \prod_{e \in E_2} (1 - p(e)) \tag{2.10}$$

Note that Equation 2.9 is a special case of Equation 2.10.

Lemma 2.4.1 *Let I be a CTP instance, and let $b, b' \in B_I$ be belief states such that $Loc(b) = Loc(b')$. Suppose that $b|e = b'|e$ for every edge $e \in E$. Then $b = b'$.*

Proof: Let $b, b' \in B_M$ where $Loc(b) = Loc(b')$. Then from equation 2.10, $b(l) = b'(l)$ for every state l in which $loc(l) = Loc(b)$, and otherwise $b(l) = b'(l) = 0$. Therefore $b = b'$.

$\square$

Using Lemma 2.4.1, we can now uniquely describe every belief state b as a tuple of size $|E|+1$, which contains the belief status of every edge, plus $Loc(b)$. This tuple is henceforth called the *variables-status representation* of a belief state. Therefore the size of the belief state space is at most $V \times 3^{|E|}$.

As the status of every edge remains unchanged, once its status is revealed, we make the following claim:

Claim 2.4.2 *Let π be an optimal policy, with OR-nodes $z_1, z_2 \in T_\pi$, such that $z_1 \not\preceq z_2$ and $z_2 \not\preceq z_1$. Let $b_1 = L(z_1)$ and $b_2 = L(z_2)$. Then $b_1 \neq b_2$.*

Proof: Let $z \in T_\pi$ be the splitting node for z_1, z_2 (note that z must be an AND-node). Denote $L(z)$ by b_a for a belief state b and an action a. Therefore there are observations $o_1 \neq o_2$ received in b_a, for which b_1 is reachable in π from $b_a^{o_1}$, and b_2 is reachable in π from $b_a^{o_2}$; see Figure 2.4. As $o_1 \neq o_2$, there is an edge $e \in E_{Loc(b_a)}$ for which e is blocked in o_1 and is unblocked in o_2. As the status of e remains unchanged, we have that $b_1|e \neq b_2|e$, which implies from Lemma 2.4.1 that $b_1 \neq b_2$.

□

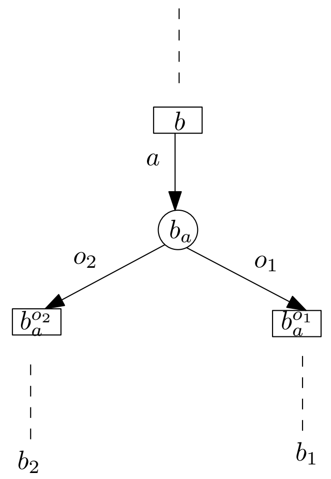


Figure 2.4: As $o_1 \neq o_2$ we have that b_1 and b_2 do not have the same variables-status representation.

The status of the edges at a belief state b can also define a blocking probability function p_b over the set of edges as follows. If an edge e is blocked in b, then $p_b(e) = 1$. If e is unblocked in b, then $p_b(e) = 0$. If e is unknown in b, then $p_b(e) = p(e)$. For the initial belief state b_0 we have $p_{b_0}(e) = p(e)$ for every $e \in E$. Therefore for a CTP $I = (G, s, t, p, w)$, every belief state b admits a CTP instance $I_b = (G, Loc(b), t, p_b, w)$ in which the graph layout of I and I_b is the same, and the only difference is in the location of the agent and the blocking probability function.

Weather configuration of CTP . A *macro-action* is defined as a conditional sequence of actions that the agent can perform. Following [12], a possible status description (*blocked* or *unblocked*) of the entire set of edges, is called a *weather*. Denote the set of all possible weathers by W, and the probability that weather w occurs by p_w. Therefore by defining $C(\pi, w)$ to be the cost of π given a specific weather w, we can compute $C(\pi)$ as follows.

$$C(\pi) = \sum_{w \in W} p_w C(\pi, w) \tag{2.11}$$

This alternative description of the cost of a policy is used throughout this work.

Another important observation made by [12] is as follows. We say a vertex v is *explored in a belief state* b if E_v, the incident edges on v, are all known in b. Note that the only uncertainty in the status of the edges is in the edges that are incident on vertices that are not yet explored. The *explored-neighborhood* of b, denoted by $Nex(b)$, is the set of all vertices u that have the following property. *There is an unblocked path in b from s to u, in which all the vertices apart from u, are explored in b.* Note that u itself can remain unexplored. The *fringe* of b, denoted by $\partial(b)$, is defined to be the set of all non-explored vertices in $Nex(b)$.

Therefore every optimal policy for the CTP can be defined as a set of macro-actions; each is defined as follows. At every belief state b with $Loc(b) = v$, a macro-action $Traverse(u)$ for $u \in \partial(b)$, is a consecutive series of actions $Move(e)$ along the shortest unblocked path $\{v, v_1, \cdots, v_l\}$, where $v_l = u$, in which $v_i \in Nex(b) \backslash \partial(b)$ for every $i < l$. See Figure 2.5 for an example.

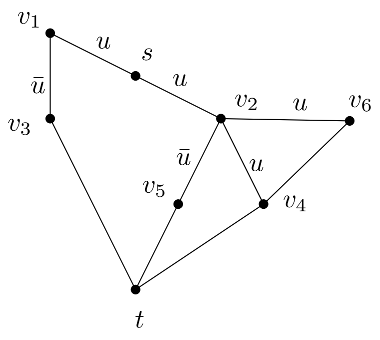


Figure 2.5: A CTP instance I in a belief state b for M_I. Edges with label u are unblocked edges in b; edges with label $\bar{u}$ are blocked edges in b. Unlabeled edges are unknown edges in b. Then $Nex(b)$ is $\{s, v_1, v_2, v_4, v_6\}$. $\partial(b)$ is $\{v_4, v_6\}$. Assume $Loc(b) = v_2$, then the possible macro actions at b are $Traverse(v_4)$, and $Traverse(v_6)$.

Default path As throughout this work we are interested in the expected travel cost, we face a problem of defining the objective when all the paths from s to t are found blocked. Namely, the expected travel cost can be infinite. This problem can be handled through one of the following approaches. First, we can assume that every instance has an unblocked traversable path, called the *default path*, from s to t, usually with a very large cost. This default path is traversed if and only if all other paths from s to t are found blocked. This approach can be thought of as a rescue choice, such as a *call for a helicopter*, that the agent must take if no regular path to the goal exists. A second approach is to assume that such a large cost default path exists from *every* vertex to target. Note, however, that these two approaches yield totally different optimal policies, as in the first approach one has to keep in mind the cost of retracing to the source vertex before traversing the default path. A third approach is to consider only instances in which such an unblocked path from the source to the target exists. Unless mentioned otherwise, we use the first approach.

2.5 CTP in a disjoint paths graphs

We follow the work of [6], and show a polynomial time solution for the Canadian Traveler Problem on disjoint paths graphs (CTP-DISJ). Although this variant is a limited case of multi-agent CTP on disjoint path graph (see Section 6.2), it is important to be familiar with CTP-DISJ at this stage, since the solution for CTP-DISJ underlies many results presented throughout this work. A detailed proof for the main theorem in this section, Theorem 2.5.3, can be found in Section 6.2.

A CTP-DISJ instance is a CTP with a graph constructed from $k \geq 1$ paths, denoted by $I_0, \cdots, I_{k-1}$ (see Figure 2.6). Apart from s and t, in which all the paths meet, all the paths are vertex-disjoint. We assume w.l.o.g. that at least one path is known to be traversable. Otherwise, we can add a traversable default path consisting of a single unblocked edge between s and t with a finite, but very large, cost, which is traversed if and only if all other paths are blocked.

The length r_i of each path I_i is the number of the edges of I_i. The edges of path I_i starting from s are denoted by $e_{i,j}$ for $0 \leq j < r_i$.

For a path I_i, and an edge $e_{i,j}$, let $W_{i,j} = \sum_{l<j} w(e_{i,l})$ be the cost of the path I_i up to edge $e_{i,j}$ not including $e_{i,j}$. Let $W_i = W_{i,r_i}$ be the cost of the entire path I_i. Define Q_i to be the probability of path I_i being unblocked; (thus $Q_i = \prod_{l<r_i} q(e_{i,l})$, where $q(e_{i,l}) = 1 - p(e_{i,l})$), and let $P_i = 1 - Q_i$ be the probability that I_i is blocked.

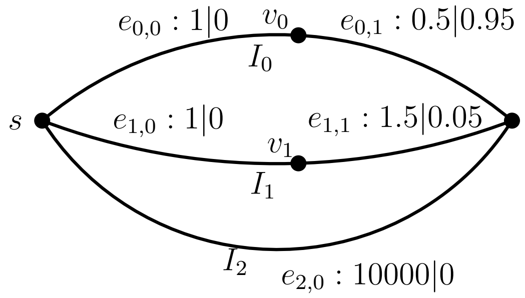


Figure 2.6: CTP with disjoint path graph.

We define two macro-actions, through which all optimal policies on disjoint-paths graphs can be specified. Both macro actions are defined for an agent situated at s.

Definition 2.5.1 *For a path I_i, macro action $TRY(i)$ is to move forward along I_i until reaching t; if a blocked edge is encountered, the agent returns along the path and stops at s. An agent performing a TRY action on a path is said to be* trying *the path.*

Definition 2.5.2 *For a path I_i and an edge $e_{i,j}$, macro-action $INV(i,j)$ (meaning* investigate*) is to move forward along I_i until reaching (without crossing) $e_{i,j}$, or a blocked edge, whichever occurs first. In either case the agent returns to s.*

For a path I_i, denote by $BC(i)$ the random variable representing the backtracking cost of I_i: that is, the cost of traversing I_i, finding I_i blocked, and returning to s. Since I_i can be blocked anywhere, and the cost to reach edge $e_{i,j}$ is $W_{i,j}$, the expected backtracking cost is:

$$E[BC(i)] = 2\sum_{j<r_i} W_{i,j}p(e_{i,j})\prod_{l<j} q(e_{i,l}) \tag{2.12}$$

Denote the expected cost of $TRY(i)$ by $E[TRY(i)]$. Then

$$E[TRY(i)] = Q_iW_i + E[BC(i)] \tag{2.13}$$

A policy that consists only of TRY actions, but never uses INV actions (that is, only backtracks if a blocked edge is revealed), is called *committing*. In Chapter 4 we see a generalization of this definition of commitments in the CTP.

Since a TRY macro action either reaches t or finds a path blocked, it never makes sense to try the same path more than once, and thus all such committing policies can be represented by an order of paths to be tried. Let M be an instance of CTP-DISJ, and let π^*_M be a committing policy for M in which the agent tries the paths in a non-decreasing order of $D_i = \frac{E[TRY(i)]}{Q_i}$. Assume without loss of generality that the D_i are all different, and thus π^*_M is unique. Then the following theorem [6] holds:

Theorem 2.5.3 π^*_M *is an optimal policy for* M.

Proof outline:

We first show that π^*_M is optimal among all committing policies for M. As every committing policy is a permutation of TRY actions, we see that given two committing policies, π and π' for M, such that π' is obtained from π by switching $TRY(i)$ with $TRY(i+1)$, then $C(\pi) < C(\pi')$ if and only if $D_i < D_{i+1}$.

Next, let ν^* be an optimal, but not committing policy, for M. We assume w.l.o.g. that ν^* is an optimal policy with a minimal number of INV-edges in T_{ν^*} among all the optimal policies for M. We can then show that T_{ν^*} contains a subtree, T, with only one INV-edge, and define a policy ν' that is obtained from ν^* by replacing T with another tree, T', which has no INV-edges at all. We then show that $C(\nu') \leq C(\nu^*)$, contradicting the minimal number of INV-edges in T_{ν^*} among the optimal policies of M. $\square$

Chapter 3

Complexity of the Canadian Traveler Problem

3.1 The CTP is PSPACE-complete

When originally introduced in [31], two variants of the CTP were examined: the adversarial variant and the stochastic variant. Both variants were shown to be in PSPACE. The adversarial variant was shown to be PSPACE-hard by reduction from QBF. For the stochastic version only #P-hardness was established by reduction from the *st-reliability* problem (as stated in [31]), leaving the question of PSPACE-hardness open. Apparently proving the stronger result requires some form of dependency between the edges, achieved "through the back door" in the adversarial variant. This chapter, published in [14], settles the question, showing that the CTP is indeed PSPACE-complete. In fact we show that it is PSPACE-hard to solve the CTP decision problem, stated as follows: Given an instance of the CTP, and an edge e incident on s, does there exist an optimal policy where traversing e is the first move? Membership of this problem in PSPACE was shown in [31], by the general argument that the CTP is a "game against nature" for which PSPACE algorithms exist. A more detailed argument appears in [14].

We begin with a preliminary known variant, of CTP with dependent directed edges, *CTP-Dep*, which allows for a simple proof (first shown in

[14]) of PSPACE-hardness by reduction from QBF. Then, we proceed with a PSPACE-hardness proof for the "standard" stochastic CTP. Although the latter result subsumes the PSPACE-hardness of CTP-Dep, proving the dependent CTP result first greatly simplifies the intuition behind the proof of the standard case.

3.2 Dependent directed CTP is PSPACE-hard

The dependent CTP is a generalization of the CTP where edge blocking probabilities can be dependent. Therefore, instead of the function p in the definition of the CTP, we have a general probability distribution in the dependent version. In order to make the reduction simpler, we define the problem over a directed graph.

Formally, the *dependent CTP* (called CTP-Dep) is a 5-tuple (G, c, s, t, BN) with $G = (V, E)$ a directed graph, a cost function $c : E \to \Re^{\geq 0}$, $s, t \in V$ are the source and target vertices, respectively, and a distribution model BN over binary random variables describes the dependency of the blocking probabilities the edges of E.

As in the CTP, the problem is to find a policy that minimizes the expected traversal cost from s to t. We assume that BN is specified as a Bayes network (see [32]) as follows. The Bayes network (Y, A, P) consists of a set of directed arcs A between a set of binary random variables Y, so that (Y, A) is a directed acyclic graph. P describes the conditional probability tables, one for each $y \in Y$. Note that it is sufficient to show that if the in-degree in the Bayes network graph (Y, A) is bounded by a constant, then the size of an explicit representation of a Bayes network, as well as the time to generate it, are low-order polynomial in the number of random variables. The bounded in-degree is guaranteed by construction in the proof below.

Theorem 3.2.1 *Determining whether there exists an optimal policy for the CTP-Dep in which a given action is the first move is PSPACE-hard.*

Proof. We prove Theorem 3.2.1 by reduction from the PSPACE-complete problem QBF [16]. Recall that QBF is the language of all *true* quantified Boolean formulas in prenex normal form, $\Phi = \forall x_1 \exists x_2 ... \varphi(x_1, x_2, ..., x_n)$, where φ is a Boolean formula in conjunctive normal form, with n variables and m clauses. Every clause contains literals, each consisting of either a variable or a negated variable. We assume that each clause has at most 3 literals (see [16]). Given a QBF instance Φ, construct a CTP-Dep instance (G_Φ, c, s, t, BN) as follows (see Figure 3.1). G_Φ consists of a *variables section*, and an *exam section*. Vertices in the variables section have labels starting with v or o, and vertices of the exam section begin with r. An always unblocked edge (s, t), called the *default edge*, has a cost of $h > 0$ defined below (in Claim 3.2.2). All other edges, unless mentioned otherwise, are zero-cost edges known to be unblocked. In some cases, the only role such edges have in the proof is to simplify the notation or the physical layout of the figure, such as the edges (s, v_1) and (v'_n, r_0).

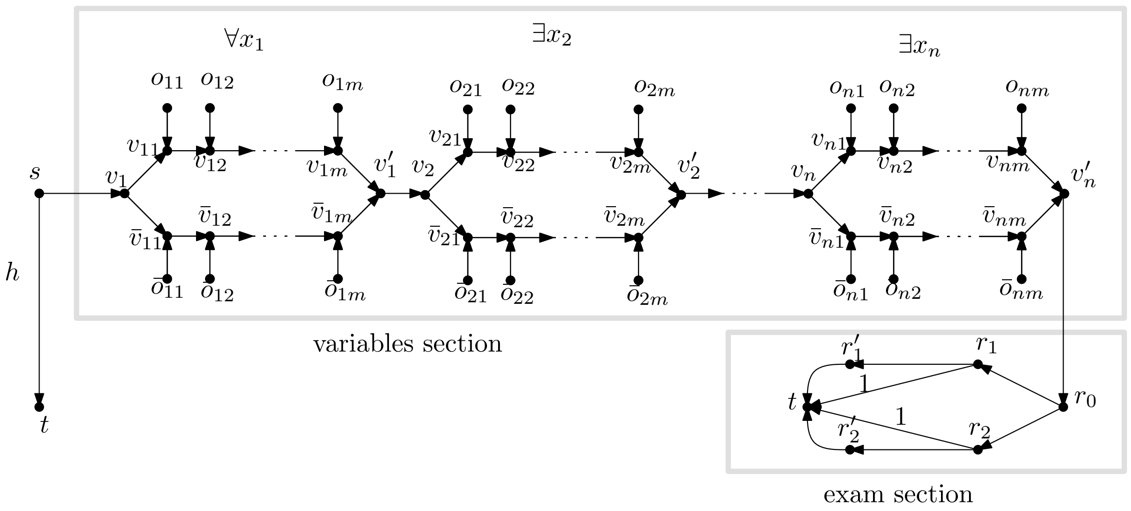

$\forall x_1$
$\exists x_2$
$\exists x_n$
o_{11} o_{12} o_{1m} o_{21} o_{22} o_{2m} o_{n1} o_{n2} o_{nm}
v_{11} v_{12} v_{1m} v'_1 v_2 v_{21} v_{22} v_{2m} v'_2 v_n v_{n1} v_{n2} v_{nm} v'_n
s v_1 h t
$\bar{v}_{11}$ $\bar{v}_{12}$ $\bar{v}_{1m}$ $\bar{v}_{21}$ $\bar{v}_{22}$ $\bar{v}_{2m}$ $\bar{v}_{n1}$ $\bar{v}_{n2}$ $\bar{v}_{nm}$
$\bar{o}_{11}$ $\bar{o}_{12}$ $\bar{o}_{1m}$ $\bar{o}_{21}$ $\bar{o}_{22}$ $\bar{o}_{2m}$ $\bar{o}_{n1}$ $\bar{o}_{n2}$ $\bar{o}_{nm}$
variables section
r'_1 r_1 t 1 1 r_0 r'_2 r_2
exam section

Figure 3.1: Reduction from QBF to CTP-Dep. Note that vertex t appears twice in order to simplify the physical layout.

The variables section contains a subsection X_i for every variable x_i, which begins at v_i and ends at v'_i. For every $i < n$, X_i is connected to X_{i+1} through an edge (v'_i, v_{i+1}).

Every X_i contains a *true-path* $(v_i, v_{i1}, \cdots, v_{im}, v'_i)$, and a *false-path* $(v_i, \bar{v}_{i1}, \cdots, \bar{v}_{im}, v'_i)$.

If x_i is a universal variable (resp. existential variable), the edges (v_i, v_{i1}), and $(v_i, \bar{v}_{i1})$ are called *universal edges* (resp. *existential edges*). While the existential edges are always unblocked, we set the universal edges to have blocking probability 1/2 and to be mutually exclusive: for each universal variable x_i, exactly one of (v_i, v_{i1}), and $(v_i, \bar{v}_{i1})$ is blocked.

In addition, for every $1 \leq i \leq n$ and $1 \leq l \leq m$, there are edges (o_{il}, v_{il}) and $(\bar{o}_{il}, \bar{v}_{il})$ called *observation edges*. These edges are only meant to be observed, as their source vertices are unreachable. Every observation edge is blocked with probability 1/2, and the dependency of the observation edges is defined according to appearance of variables in the clauses of Φ, as follows: an observation edge (o_{il}, v_{il}) (resp. $(\bar{o}_{il}, \bar{v}_{il})$) is considered "in" a clause C_l if x_i appears unnegated (resp. negated) in clause C_l. All observation edges that are "in" the same clause C_l co-occur: they are either all blocked or all are unblocked (with probability 1/2, as stated above), independent of all other edges that are not "in" C_l.

The exam section consists of an *odd-path* (r_0, r_1, r'_1, t), and an *even-path* (r_0, r_2, r'_2, t). In addition construct edges (r_1, t) and (r_2, t) with cost 1. The edges (r_1, r'_1) and (r_2, r'_2) are called *choice edges*. The edge (r_1, r'_1) (resp. (r_2, r'_2)) is unblocked if and only if the observation edges are unblocked for an *odd* (resp. *even*) number of clauses. Hence exactly one of the choice edges is blocked. If at least one observation edge in each clause is observed, the status of the choice edges can be determined with certainty. Otherwise the posterior blocking probability of each choice edge remains 1/2. A description of the layout of the related bayesian network BN appears in Appendix B.1. In order to prove the theorem, it is sufficient to prove the following claim:

Claim 3.2.2 *If Φ is true then there is an optimal policy with an expected cost 0, and the optimal first action is to traverse (s, v_1). If Φ is false, then for every $0 < h < 2^{-\frac{n}{2}-1}$, the optimal policy is to traverse (s, t) with a cost of h.*

Proof: Suppose first that Φ is true. Then there is a policy for assigning values to every existential variable x_i, each given every assignment to the universal variables enclosing x_i, such that φ is true. Following this policy for each existential variable x_i, i.e., traversing edge (v_i, v_{i1}) if x_i should be *true*, and $(v_i, \bar{v}_{i1})$ otherwise, leads (by construction) to following a path such that at least one observation edge is seen in every clause. Hence, the "exam" is passed (i.e., the zero-cost unblocked path in the exam section is chosen) with certainty.

Next, suppose Φ is *false*. Then there exists an "adversary" policy of assigning the universal variables for every assignment of the enclosing existential variables, in which eventually, some clause C_l is *false*. (as above, such an adversary policy may only depend on the values of existential variables enclosing the current universal variable). In that case no edge "in" clause C_l is observed. Since every assignment of the universal variables occurs with probability $2^{-\frac{n}{2}}$ (assuming w.l.o.g. that n is even), in these cases the exam is "flunked" (picking the path where only the expensive edge is unblocked) with probability 1/2, and thus the total expected cost of starting with (s, v_1) is at least $2^{-\frac{n}{2}-1}$. Hence, with $0 < h < 2^{-\frac{n}{2}-1}$, the optimal policy is to traverse (s, t) if and only if Φ *false*.

□

Observe that the proof of Theorem 3.2.1 also shows the following:

Corollary 3.2.3 *It is PSPACE-hard to determine the expected cost of the optimal policy in the CTP-Dep.*

3.3 Complexity of the CTP

Having shown that the CTP-Dep is PSPACE-hard, we extend the proof to the "standard" stochastic independent undirected edges CTP.

Theorem 3.3.1 *The CTP decision problem is PSPACE-complete.*

In order to prove Theorem 3.3.1, we use the same general outline of the reduction from QBF as in the proof of Theorem 3.2.1. However, in the CTP-Dep, dependencies and directed edges restrict the available choices, thereby simplifying the proof. Here we introduce special gadgets that limit choice de facto, and show that any deviation from these limitations is necessarily sub-optimal. Policies that obey these limitations are called *reasonable policies.* Each such gadget g has an *entry* terminal $Entry(g)$, and an *exit* terminal $Exit(g)$; an attempt to traverse g from $Entry(g)$ to $Exit(g)$ is henceforth called to *cross* g. The gadgets operate by allowing a potential *shortcut* to the target t; crossing these gadgets may either end up at t, or at $Exit(g)$, with some probability $q(g)$. The unblocked edges that allow direct access to t are called *shortcut* edges. The following invariant follows from the construction of the CTP graph in Section 3.3.3, and is used throughout the proof of Theorem 3.3.1.

Invariant 3.3.2 *Every gadget g is attached to any other graph component such that any partially specified policy executed at $Entry(g)$, in which g is not crossed, has an expected cost of at least* 1.

We introduce the gadgets in Sections 3.3.1 and 3.3.2, and the CTP graph construction in Section 3.3.3. The actual proof of Theorem 3.3.1 is in Section 3.3.4. In the description of the gadgets and the CTP graph, we sometimes add zero-cost always traversable edges. These edges, which appear unlabeled in Figures 3.2, 3.3, and 3.4, were added solely in order to simplify the physical layout as a figure; any u, v connected by such an edge can be considered to be the same vertex.

3.3.1 Baiting gadgets

A *baiting gadget* $g = BG(u, v)$ with a parameter $l > 1$ is a three-terminal weighted graph

(see Figure 3.2): an entry terminal $u = Entry(g)$, an exit terminal $v = Exit(g)$, and a shortcut terminal which is always t. The latter terminal is henceforth omitted in external reference to g, for conciseness.

The baiting gadget consists of $N + 1$ uniform sections of an undirected always unblocked path $(u, v_1, \cdots, v_N, v)$ with total cost l. Each intermediate vertex has a zero-cost shortcut to t with a blocking probability $1/2$. In addition, there is a shortcut edge with cost l from the terminals u, v to t. Set

$N = 2^{\lceil \log_2(4l) \rceil} - 1$. Then the size of g is $\Theta(l)$.

As the formal description of a policy is cumbersome, we informally describe the following policy as a conditional sequence of actions, with conditions being previous locations, actions, and observations.

Let π be the following partially specified policy: *when at u for the first time, proceed along the path $(u, v_1, \cdots, v_N, v)$ to v, taking the zero-cost shortcut to t whenever possible, but never backtracking to u. From v continue with any optimal policy.* This description of π has an obvious formal interpretation, which we write out as an example in Appendix B.2.

When at u for the first time, the expected cost of reaching t by executing π is less than 1, even if we need to take the shortcut edge (v, t) (proved in Appendix B.2). As the shortcut edge (v, t) costs l, the expected cost of any optimal policy once at v is no more than l. After reaching v, all the zero-cost shortcut edges are blocked; therefore g is not retraced by any reasonable policy. A similar argument holds for retracing to u from other locations along the path $(u, v_1, \cdots, v_N, v)$. Hence we have:

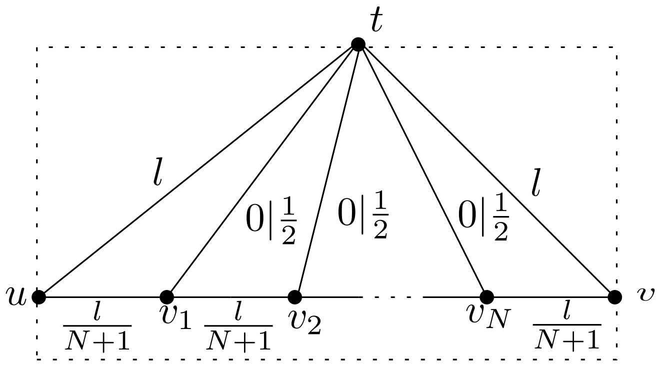


Figure 3.2: A baiting gadget $BG(u, v)$ with a parameter $l > 1$. Edge label $c \mid p$ denotes cost $\mid$ blocking probability. The optimal policy at u is to cross the path $(u, v_1, \cdots, v_N, v)$, taking a shortcut edge to t whenever such an edge is found unblocked. After reaching v, retracing to u in g costs at least l.

Claim 3.3.3 *When at u for the first time, under Invariant 3.3.2, π is optimal for a baiting gadget $g = BG(u, v)$ with a parameter $l > 1$. After reaching v, it is suboptimal to backtrack to u in g.*

Note that g is actually symmetric w.r.t. u, v. However, since by construction of the CTP graph, every reasonable policy always reaches one designated terminal u first, we treat g externally as if it were directional. A precise derivation of the parameters of baiting gadgets appears in Appendix B.2.

3.3.2 Observation gadgets

An *observation gadget* $g = OG(u, v, o)$, is a four-terminal weighted graph (see Figure 3.3): an entry terminal $u = Entry(g)$, an exit terminal $v = Exit(g)$, an observation terminal o, and a shortcut terminal (again omitted in external references) that is always t. The observation gadget begins with a baiting gadget $BG_1 = BG(u, v_1)$ with a parameter $l = L$ (the global value L is a problem-dependent value defined below for all the observation gadgets), which is connected to the "observation loop" beginning with a baiting gadget $BG_2 = BG(v_1, v_2)$ with a parameter $l = 3L/2$, a zero-cost edge (v_2, v_3) with blocking probability 3/4, and a cost $5L/8$ unblocked edge (v_3, o). A cost $3L/2$ unblocked shortcut edge (v_2, t) exists as a part of the baiting gadget BG_2. The observation loop is closed by a cost $5L/8$ unblocked edge (o, v_4) and a zero-cost edge (v_4, v_1) with blocking probability 3/4. From v_1, a cost 1 unblocked edge (v_1, v_1') followed by a baiting gadget $BG_3 = BG(v_1', u)$ with a parameter $l = L$ completes the gadget. Note that as every baiting gadget is of the size of $\Theta(L)$, we have that the size of g is $\Theta(L)$.

We next define the path component to which the observation terminal o that can be connected, and the path edges incident on o that can be observed. The *exam section path* is a path $(r_2, r_3, r_4, r_5, r_1', r_2')$ (o is identified with r_5) with the following properties: the edges (r_2, r_3), (r_1', r_2'), and (r_4, r_5) have zero cost and blocking probability p_1, where $p_1 > 1 - 2/(3L+1)$. (r_2, r_3) and (r_1', r_2') are called *guard edges*, (r_4, r_5) is called an *observation edge*. The edges (r_3, r_4) and (r_5, r_1') are always traversable edges with cost 1. The notations of the exam section path are chosen to match the description of the CTP graph construction in Section 3.3.3.

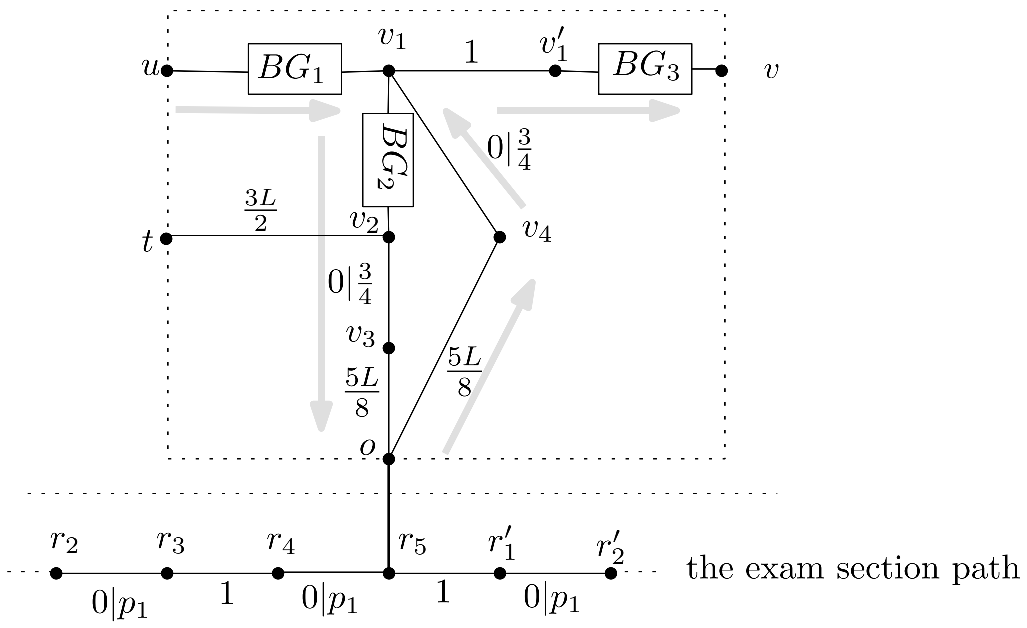


Figure 3.3: An observation gadget $OG(u, v, o)$. Light gray arrows indicate general traversal direction of the optimal policy π. BG_1 and BG_3 are baiting gadgets with a parameter $l = L$. BG_2 is a baiting gadget with a parameter $l = 3L/2$.

Invariant 3.3.4 *The observation terminal o is either not directly connected to the rest of the graph, or connected through the exam section path $(r_2, r_3, r_4, r_5, r_1', r_2')$, in which case o is identified with r_5. In addition, o is allowed to coincide only with observation terminals of other observation gadgets.*

We see in Section 3.3.3 that Invariant 3.3.4 follows from the construction of the CTP graph. Let π_g be the following partially specified policy for g: *when at u, cross BG_1. Then (observing (v_1, v_4)), cross BG_2. If either (v_2, v_3) or (v_1, v_4) is found blocked, reach t by traversing the shortcut edge (v_2, t), which costs $3L/2$. However, if both (v_2, v_3) and (v_1, v_4) are unblocked, traverse the path $(v_2, v_3, o, v_4, v_1, v_1')$ (observing any edges incident on o such as the observation edge (r_4, r_5)), and cross BG_3. Then from v continue with any optimal policy.*

Claim 3.3.5 *Assume $L > 8$. Then, when at u for the first time, under*

Invariants 3.3.2 and 3.3.4, π_g is an optimal policy for an observation gadget $g = OG(u, v, o)$.

Proof outline. First observe that following π_g, Invariant 3.3.2 holds for every baiting gadget in g. Therefore properties of the baiting gadgets ensure that g is traversed in the correct order. Next, the guard edges (r_2, r_3) and (r'_1, r'_2) ensure that it is suboptimal to "escape" from o by traversing edges in the exam section path. The uncertain edges (v_4, v_1) and (v_2, v_3) ensure that it is suboptimal to enter a previously uncrossed observation gadget from o. Likewise for a previously crossed observation gadget $\widetilde{g}$: entering $\widetilde{g}$ through o is suboptimal because all the baiting gadgets in $\widetilde{g}$ have been crossed and observed to contain no unblocked zero-cost shortcuts.

A detailed derivation of the properties of observation gadgets appears in Appendix B.3.

3.3.3 CTP graph construction

Having shown the properties of the baiting and observation gadgets, we are ready to construct the CTP graph: For a QBF Φ with n variables and m clauses, we construct G_Φ in the same general outline as the construction of the CTP-Dep graph (see Section 3.2) with the following changes (see Figure 3.4). The exam section is a path of $5(m+1)$ vertices $\{r^i_j \mid 1 \leq i \leq m+1, 1 \leq j \leq 5\}$, with an additional vertex r_0, as follows. For every $0 < i \leq m+1$, (r^i_1, r^i_2), (r^i_2, r^i_3), and (r^i_4, r^i_5) have zero cost and blocking probability p_1, except from (r^{m+1}_4, r^{m+1}_5), which has zero cost and is always traversable. r^{m+1}_5 is identical to t. (r^i_1, r^i_2), and (r^i_2, r^i_3) are called *guard edges.* The edge (r^i_4, r^i_5) is called a *clause edge*, and is denoted by e_i. The edges (r^i_3, r^i_4), and (r^i_5, r^{i+1}_1) are always traversable cost 1 edges. In addition, there is an always traversable cost 1 edge (r_0, r^1_1), as well as an always traversable cost L shortcut edge (r_0, t). In order to guarantee correct operation of the observation gadgets, we disallow reasonable policies to traverse exam edges too early while crossing the variable section. This is done by visiting the initially uncertain guard edges only later via a section called the *guards section*, which consists of

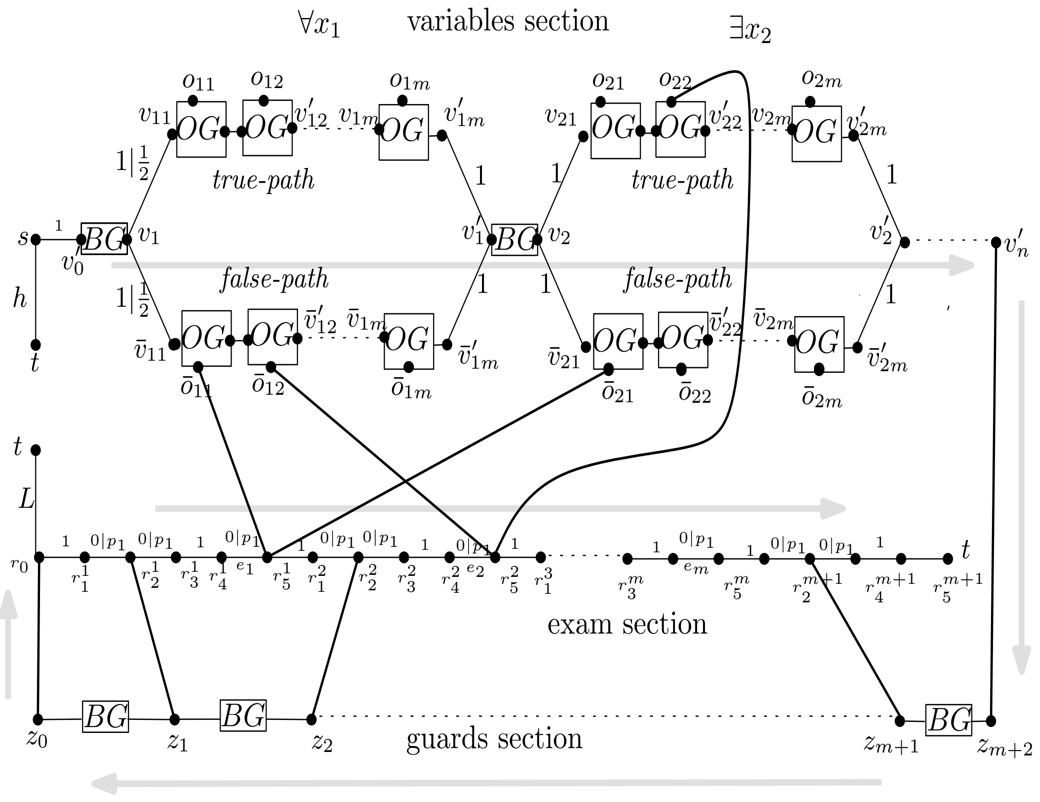


Figure 3.4: The CTP graph construction for $\Phi = \forall x_1 \exists x_2 \cdots (\bar{x}_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \cdots$. BG - a baiting gadget. OG - an observation gadget. Light gray arrows indicate the general traversal direction of the optimal policy when Φ is *true*.

a sequence of $m+2$ baiting gadgets $BG(z_i, z_{i-1})$, $0 < i \leq m+2$, with a parameter $l = L$ that visits r_2^i from every z_i (except from z_{m+2}).

The variables section is constructed as for CTP-Dep, except that the directed edges (v_i', v_{i+1}) are replaced by baiting gadgets $BG(v_i', v_{i+1})$ with a parameter $l = L$. For each universal variable x_i the universal edges (v_i, v_{i1}), and $(v_i, \bar{v}_{i1})$ are cost 1 edges with blocking probability 1/2. For each existential variable x_i, the existential edges (v_i, v_{i1}) and $(v_i, \bar{v}_{i1})$ are always traversable edges with cost 1. Inside each *true-path*, every (v_{ij}, o_{ij}), $(v_{ij}, v_{i(j+1)})$ pair is replaced by an observation gadget $g = OG(v_{ij}, v_{ij}', o_{ij})$. $(v_{ij}', v_{i(j+1)})$ are always unblocked zero-cost edges added for clarity. The observation vertex o_{ij} is identified with the vertex r_5^j incident on the appropriate clause edge e_j in the exam section. That is, if x_i appears unnegated in clause j, then o_{i_j} of the *true-path* is identified with r_5^j in the exam section. Likewise respectively for all the edges in the *false-paths*. Note that Invariant 3.3.2 holds for all the baiting gadgets, and observation gadgets in G_Φ, and that Invariant 3.3.4 holds for all the observation gadgets in G_Φ.

For example, Figure 3.4 demonstrates the reduction for $\Phi = \forall x_1 \exists x_2 \cdots (\bar{x}_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \cdots$. The variable x_1 appears negated in clause 2, so in G_Φ the vertex $\bar{o}_{12}$ at the section X_1, and the vertex r_5^2 of the exam section are connected by an unlabeled edge, hence the clause edge $e_2 = (r_4^2, r_5^2)$ can be observed from the observation gadget $OG(\bar{v}_{12}, \bar{v}'_{12}, \bar{o}_{12})$ when traversing the *false* path of X_1. The connection of other observation gadgets can be explained similarly.

3.3.4 Proof of Theorem 3.3.1

Given a QBF Φ with n variables and m clauses, we construct G_Φ as in Section 3.3.3. Set $L = 8m+16$ and $p_1 = 1 - 2^{-\lceil \log_2(\frac{3L+1}{2}) \rceil}$. We show that it is optimal to traverse (s, v_0) if and only if Φ is *true*.

Unless stated otherwise, we henceforth consider only reasonable policies for G_Φ that do not begin with the default action of traversing (s, t). Due to properties of the gadgets (Claims 3.3.3, 3.3.5) we have that by following any reasonable policy for G_Φ, Invariants 3.3.2, and 3.3.4 hold for all baiting and observation gadgets. Therefore any reasonable policy π for G_Φ must follow the restrictions in Table 3.1, as any other action is suboptimal.

Table 3.1: Reasonable policy actions in π.

Location	Action
v'_i, for $i < n$	cross $BG(v'_i, v_{1+1})$
v_i, for $i \leq n$,	go to v_{i1} or $\bar{v}_{i1}$
v_{il}, for $i \leq n$,	cross $OG(v_{il}, v'_{il}, o_{il})$
$\bar{v}_{il}$ for $i \leq n$,	cross $OG(\bar{v}_{il}, \bar{v}'_{il}, \bar{o}_{il})$
z_i, for $0 < i \leq m+2$	cross $BG(z_i, z_{i-1})$
r_0	pass exam or take shortcut

Most of these restrictions are immediate consequences of executing optimal policies at the baiting and observation gadgets (see Appendix B.2 and Appendix B.3 for details). The following claim, proved in Appendix B.4, shows the actions of any reasonable policy for G_Φ at r_0.

Claim 3.3.6 *At r_0, any reasonable policy acts as follows:*

- *If all the edges in the exam section were observed to be unblocked, cross $(r_0, r_1^1, \cdots, r_4^{m+1}, t)$ until reaching t for a cost of $2(m+1)$.*
- *Otherwise, cross the cost L shortcut edge (r_0, t).*

Therefore, reasonable policies for G_Φ differ only in the choices made in the universal and existential edges, and in the choice at r_0, which is either to traverse the exam section if all clause edges were observed, or otherwise take the expensive shortcut (r_0, t).

Now recall that for every policy π for G_Φ we have

$$C(\pi) = \sum_{w \in W} p_w C(\pi, w) \tag{3.1}$$

where W is the set of all possible weathers for G_Φ (see Section 2.5). Partition W into *full-trip weathers* $W^f(\pi)$, in which r_0 is reached while executing π; and *shortcut weathers* $W^s(\pi)$ in which r_0 is not reached due to taking a shortcut edge to t before reaching r_0. Then:

$$C(\pi) = \sum_{w \in W^s(\pi)} p_w C(\pi, w) + \sum_{w \in W^f(\pi)} p_w C(\pi, w) \tag{3.2}$$

Let π^T be a policy for G_Φ such that in every subsection X_i of the variables section, whenever possible, the *true-path* is always chosen. Define:

$$D_{st} = \sum_{w \in W^s(\pi^T)} p_w C(\pi^T, w) \tag{3.3}$$

As all the *true-paths* and *false-paths* of all the variables section are symmetric in the number of observation gadgets and other edges, there is a bijection

$g_\pi : W^s(\pi) \to W^s(\pi^T)$ such that $p_w = p_{g_\pi(w)}$ and $C(\pi, w) = C(\pi^T, g_\pi(w))$ for every $w \in W^s(\pi)$. Hence we have:

$$\sum_{w \in W^s(\pi)} p_w C(\pi, w) = D_{st} \tag{3.4}$$

and therefore

$$C(\pi) = D_{st} + \sum_{w \in W^f(\pi)} p_w C(\pi, w) \tag{3.5}$$

Again, due to symmetry, and the properties of the baiting and observation gadgets (Claims 3.3.3, 3.3.5), the total cost from s to r_0 while executing π in any weather $w \in W^f(\pi)$ is independent of w. We denote this cost by D_{pt}, and can compute it simply by summing the cost of traversing a path from s to r_0 through the variables section and guards section, assuming that r_0 is reached. More precisely, see that in every weather $w \in W^f(\pi)$, every crossed baiting gadget has a cost of L, and every crossed observation gadget has a cost of $(19L+4)/4$. Then, as the number of crossed observation gadgets is mn, with an additional n baiting gadgets (v_i', v_{i+1}) need to be crossed, as well as $m+1$ baiting gadgets of the guards section, we have that

$$D_{pt} = 1 + \left(2 + \frac{(19L+4)m}{4}\right) n + (n+m+1)L \tag{3.6}$$

Now, according to Claim 3.3.6, the cost of reaching t from r_0 is either $2(m+1)$ (if the exam section is known to be completely unblocked), or $L > 2(m+1)$ (taking the shortcut (r_0, t), if some edges in the exam section are known to be blocked, or some such unknown edges remain). Hence for any full-trip weather w, $C(\pi, w)$ is either $D_{pt} + L$, or $D_{pt} + 2(m+1)$.

Let $P_{rt} = (1-p_1)^{3m+2}$ be the probability that all the edges in the exam section are unblocked. Let $P_\Phi^\pi \in [0,1]$ be the probability that **not** all the clause edges of the exam section were observed in a full-trip weather by following π (this probability depends on the formula Φ). Then, with probability $P_{rt}(1 - P_\Phi^\pi)$ all the edges of the exam section were observed and were found unblocked before reaching r_0. Denote by P_{r_0} the probability of reaching r_0 by executing π. Again, due to symmetry of the baiting and observation gadgets, P_{r_0} is independent of π. We get:

$$\sum_{w \in W^f(\pi)} p_w C(\pi, w) = P_{r_0}\Big(D_{pt} + P^{\pi}_{\Phi} L + (1 - P^{\pi}_{\Phi})\big(P_{rt} 2(m+1) + (1 - P_{rt})L\big)\Big) \tag{3.7}$$

And therefore

$$C(\pi) = D_{st} + P_{r_0}\Big(D_{pt} + P^{\pi}_{\Phi} L + (1 - P^{\pi}_{\Phi})\big(P_{rt} 2(m+1) + (1 - P_{rt})L\big)\Big) \tag{3.8}$$

If Φ is *true*, then, as in the proof of Theorem 3.2.1, there is a reasonable policy π which follows the variables assignments that satisfy Φ; thus every clause edge is observed and $P^{\pi}_{\Phi} = 0$. Define $B_0 = C(\pi)$ for such a policy π when Φ is *true*. Then

$$B_0 = D_{st} + P_{r_0}\big(D_{pt} + P_{rt} 2(m+1) + (1 - P_{rt})L\big) \tag{3.9}$$

If Φ is *false*, then, again as in the proof of Theorem 3.2.1, there is an "adversary" policy of assigning the universal variables for which at least one clause in Φ is *false*, for every assignment of the enclosing existential variables. For every universal variable x_i, the probability that exactly one universal edge is unblocked is $1/4$. Therefore, there is a probability of at least $(\frac{1}{4})^{\frac{n}{2}}$ for the "adverse case", where the only universal edge that is unblocked for each universal variable x_i, is consistent with the adversary policy. In this adverse case not all the clause edges are visited upon reaching r_0. Note that P_{r_0} already excludes events where both universal edges are blocked for some variable, thus r_0 is not reached. Therefore if Φ is *false*, then for every reasonable policy π, $P^{\pi}_{\Phi} > (\frac{1}{3})^{\frac{n}{2}}$. Hence define B_1 as follows.

$$B_1 = D_{st} + P_{r_0}\Big(D_{pt} + 3^{-\frac{n}{2}} L + \big(1 - 3^{-\frac{n}{2}}\big)\big(P_{rt} 2(m+1) + (1 - P_{rt})L\big)\Big) \tag{3.10}$$

Then $B_1 > B_0$, and if Φ is *false*, then $C(\pi) \geq B_1$. Now let

$$h = c((s,t)) = B_0 + 2^{-n} m P_{r_0} \tag{3.11}$$

Then $B_1 > h > B_0$. Thus the optimal action at s is to traverse (s,t) if and only if Φ is *false*.

It remains to show that G_Φ is constructed in a time polynomial in the size of the input. As the size of every baiting and observation gadget is $\Theta(L)$, the CTP graph G_Φ contains a polynomial number of vertices and edges. By construction, all the probabilities of the edges can be described as a division of two polynomials. Likewise for every edge cost, except for the default edge (s,t) and its cost h (see Equations (3.11), and (3.9)). To show that h can be computed efficiently observe that due to symmetry of all the *true-paths* and *false-paths* of all the variables section, D_{st}, D_{pt}, and P_{r_0} can all be computed efficiently by using simple algebraic operations. For example, D_{pt} can be computed by summing the costs of traversing a path from s to r_0 through the variables section and the guards section, which is the same for all weathers where r_0 is reached. Therefore G_Φ can be constructed in a polynomial time.

Thus we have that determining whether there exists an optimal policy starting with traversing the edge (s,t) is PSPACE-hard. As membership in PSPACE has been shown in [31] (but see [14] for a CTP-specific algorithm), Theorem 3.3.1 follows. $\square$

The following corollary is immediate from the proof of Theorem 3.3.1.

Corollary 3.3.7 *The following CTP decision problem called "CTP expected cost decision problem" is PSPACE-hard: Given an instance of the CTP, and $k > 0$, does there exist an optimal policy with an expected cost of at most k.*

Several corollaries follow due to the construction of G_Φ: By replacing all the edges with appropriately directed edges, we get:

Corollary 3.3.8 *The CTP decision problem with directed edges and the CTP expected cost decision problem as stated in Corollary 3.3.7 with directed edges remain PSPACE-complete.*

Finally, as every unknown edge in this construction of G_Φ has cost 0 and a probability that is a power of 2 of being unblocked (the universal edges, for example, can be split into a two-edge path), we can replace every unknown edge with a path of zero-cost, blocking probability 1/2 edges and get:

Corollary 3.3.9 *The CTP decision problem and the CTP expected cost decision problem as stated in Corollary 3.3.7, remain PSPACE-complete even if all the unknown edges have zero cost and blocking probability* $1/2$.

Chapter 4

Decomposing the CTP

The intractability of the CTP, as shown in Chapter 3, calls for an analysis of the obstacles in solving the CTP. It seems that a major obstacle lies in the ability of the agent to traverse between several regions of the CTP graph, without having to fully exploit any of them. See example in Figure 5.2, Chapter 5. Thus a general "divide and conquer" approach of the CTP seems to be hopeless. In this chapter and in the following chapter, we explore a variety of specific topologies and specific CTP variants, for which a "divide and conquer" approach can be used to yield polynomial time solutions.

We start this chapter by showing how changing the cost function, or the blocking probability function of a CTP instance I, affects the optimal cost for I. We then introduce a technique to partition a CTP instance into CTP sub-instances, and show how to use these techniques for special graph topologies.

4.1 Comparing different CTP instances

We show that the optimal cost is monotonically increasing in the cost and blocking probability functions. This work is a generalization of previous work by [46] in which the only changes discussed are in unblocked edges that become blocked.

We start by showing that the optimal cost is monotonically increasing in the cost function. The following lemma is easily proved from Equation (2.8), as $C(\pi) = \sum_{e \in E} \alpha_e w(e)$, where $\alpha_e \geq 0$ depends on the edge's blocking probability.

Lemma 4.1.1 *Let $I = (V, E, s, t, p, w)$, and $I' = (V, E, s, t, p, w')$ be CTP instances such that $w' \leq w$. Let π^*, π'^* be optimal policies for I, I' respectively. Then $C(\pi'^*) \leq C(\pi^*)$.*

Proof: We show that if $w'(e) < w(e)$ for an edge $e \in E$, and $w'(e_1) = w(e_1)$ for every $e_1 \neq e$, then $C(\pi'^*) \leq C(\pi^*)$.

Assume $C(\pi^*) = \alpha_e w(e) + \sum_{e' \neq e} \alpha_{e'} w(e')$. As both I and I' have the same CTP graph (V, E), then the belief states of B_I and $B_{I'}$ have the same variables-status representation. Therefore we can define the following bijection $f : B_{I'} \to B_I$. For every belief state $b \in B_I$, $Loc(b) = Loc(f(b))$, and $b|e' = f(b)|e'$ for every $e' \in E$. We then construct a policy π' for I' such that $\pi'(b) = \pi^*(f(b))$ for every $b \in B_{I'}$. Hence $C(\pi') = \alpha_e w'(e) + \sum_{e' \neq e} \alpha_{e'} w(e')$ where $\alpha_e \geq 0$. As $w'(e) < w(e)$, we have $C(\pi') \leq C(\pi^*)$, and since by definition $C(\pi'^*) \leq C(\pi')$, we have $C(\pi'^*) \leq C(\pi^*)$.

□

Next, we show that the optimal cost is monotonically increasing in the blocking probability. A *stochastic policy* is a policy χ in which the action $\chi(b)$ is a random variable at every belief state b. It is immediate from the Bellman equation [4] (and in particular from Equation 2.2) that for every stochastic policy χ for a CTP instance I, there is a deterministic policy π such that $C(\pi) \leq C(\chi)$. However, stochastic policies can still be used for proof techniques, as in the proof for the following Lemma 4.1.2. We provide a proof outline; the full proof appears in Appendix C.

Lemma 4.1.2 *Let $I = (V, E, s, t, p, w)$, and $I' = (V, E, s, t, p', w)$ be CTP instances such that $p' \leq p$. Let π^*, π'^* be optimal policies for I, I' respectively. Then $C(\pi'^*) \leq C(\pi^*)$.*

Proof outline. We show that if $p'(e) < p(e)$ for some $e \in E$, and $p'(e_1) = p(e_1)$ for every $e_1 \neq e$, then $C(\pi'^*) \leq C(\pi^*)$.

We say that e is *revealed* in a transition from a belief state b' to a belief state b in a policy π if $b = b'^{o}_{\pi(b')}$ for some observation o, and if the status of e is unknown in b' and is known in b.

We then construct a *stochastic policy* π'_1 for I' such that in some cases, when e is revealed to be unblocked in a belief state reached in π_1, then π'_1 acts as if e is still blocked with a certain probability. Next we show that $C(\pi'_1) = C(\pi^*)$. As π'^* is an optimal policy for I, then $C(\pi'^*) \leq C(\pi'_1)$, and therefore $C(\pi'^*) \leq C(\pi^*)$. □

The following theorem follows immediately from Lemma 4.1.1 and Lemma 4.1.2.

Theorem 4.1.3 *The cost of an optimal policy for a CTP instance is monotonically non-decreasing in the edge costs and the edge blocking probabilities.*

4.2 Partitions and constrained policies

The objective in the CTP is to find a policy that minimizes the expected cost of traveling from s to t. This objective can be generalized to find a policy that minimizes the expected traveling cost, plus other constraints that every policy has to meet. For example, a policy is forced to visit a certain vertex, or to traverse a certain edge (thus practically forcing a policy to have "landmarks"). A policy that has to meet additional constraints is called a *constrained policy*, and a con_1 policy for a specific constraint con_1. A CTP variant in which the objective is to find a constrained policy with a minimized expected cost among all constrained policies (with the same constraints), is called *constrained-CTP*, and con_1-CTP for a specific constraint con_1. Finding an optimal (constrained) policy for constrained-CTP can yield a solution for CTP instances in which this constrained policy happens to be optimal among all policies.

4.2.1 Constrained policies

We say a graph G is *st-blocked* in a belief state b if all the paths from a vertex s to a vertex t in G are known to be blocked in b. We denote the probability that G is st-blocked in b_0 by $P_{G,s,t}$. The st-reliability problem, finding $P_{G,s,t}$, is known to be #P-hard [41, 31]. When s, t are obvious from the context we simply say that G is blocked, and denote $P_{G,s,t}$ by P_G.

Throughout this section $I = (V, E, s, t, p, w)$ is a CTP instance with a CTP graph $G = (V, E)$, and $G' = (V', E')$ is a subgraph of G with $s', t' \in V'$. The CTP instance $I' = (V', E', s', t', p \restriction E', w \restriction E')$ is called a sub-instance of I. We sometimes denote I' by the tuple (G', s', t').

Throughout this work we assume that G' is connected to the rest of the graph through only s' and t'. We also assume that the edges of $E_{t'} \cap (E \backslash E')$ are known to be unblocked. For example, Figure 4.1 depicts a CTP instance M, with a sub-instance M' over a sub-graph G'. The only two unknown edges in M are (u_1, t') and (u_3, t').

Note that unlike the CTP instance I, there is no guarantee that the CTP sub-instance I' of I has an always unblocked path from s' to t'. However, if G' is found $s't'$-blocked in a belief state b, then the agent must retrace to s' and "‘departure"’ G' through s' (see below for the exact definition of departure a sub-graph).

Therefore we virtually add, for computational purposes alone, an always unblocked edge $e = (s', t')$ with a very high cost $w(e)$. For every optimal policy, e is traversed if and only if all the paths in G' from s' to t' are found blocked. As e is traversed with probability $P_{G'}$, we subtract $P_{G'}w(e)$ when computing the expected cost of a policy for I'. See Figure 4.2 for the CTP sub-instance M' obtained from G' and M.

Recall that L is a function that assigns a belief state to every node in T_π, and assigns an action/observation to every arc in T_π (see Section 2.3.4). The following concepts are defined to be able to reason about properties of the policy π (by an "external" observation, rather than by the agent himself). An action $move(e)$ is said to be *inside* G' if $e \in E'$, and *outside* G' if $e \notin E'$. For an OR-node $z \in T_\pi$, which is not the root in T_π, and z'', the grandparent

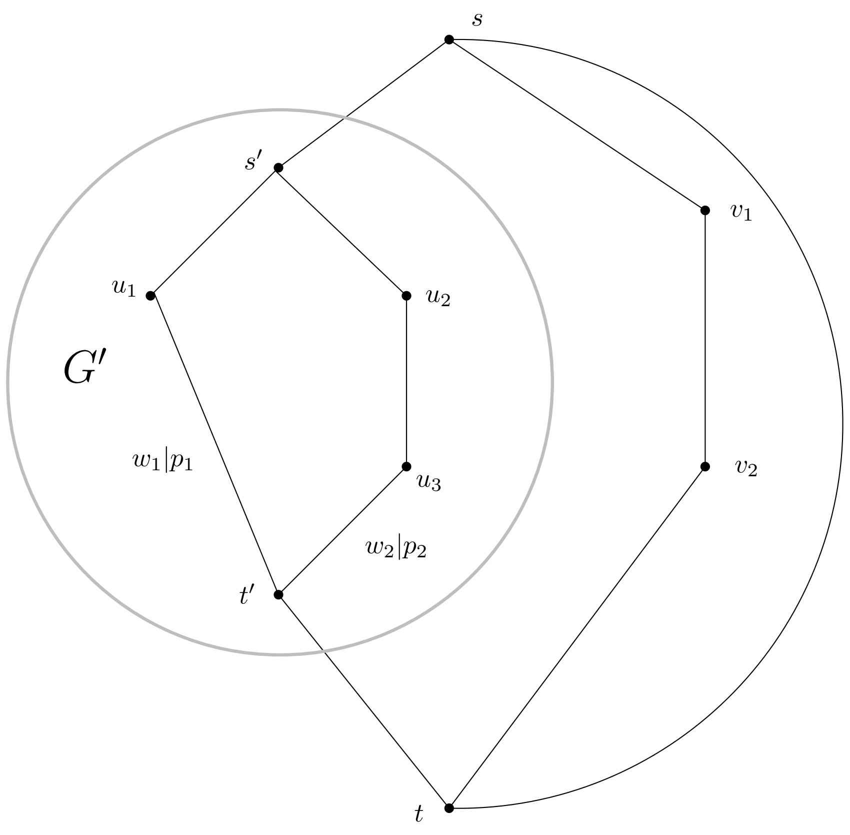


Figure 4.1: A CTP instance M, with a sub-graph G', circled in grey.

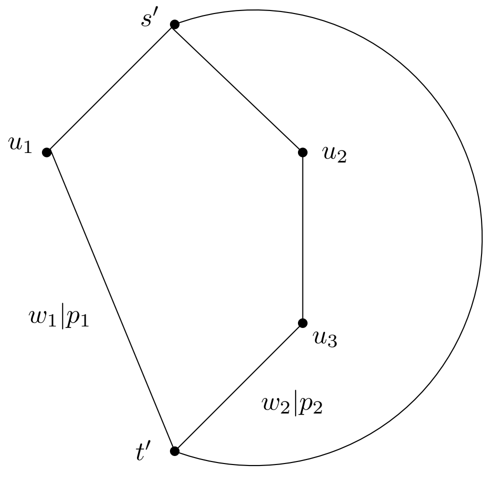


Figure 4.2: The CTP instance M'. Note that $M' = (G', s', t')$ is a sub-instance of M from Figure 4.1. The edge (s', t') is virtually added to define the optimal cost of M'.

of z, we define $PrevAction_{\pi}(z)$ to be $\pi(L(z''))$. $NextAction_{\pi}(z)$ is defined to be $\pi(L(z))$. If z is an AND-node in T_{π}, and z' is the parent of z, then $PrevAction_{\pi}(z) = \pi(L(z'))$. If z has only a single child z' in T_{π} (as a result of only a single observation received at $L(z)$), we define $NextAction_{\pi}(z)$ to be $\pi(L(z'))$.

We say that an OR-node $z \in T_{\pi}$, where $L(z) = b$, is an *entry point* (in T_{π}) of G', if $\pi(b)$ is inside G', and either $b = b_0$, or $PrevAction_{\pi}(z)$ is outside G'. We say that z is the *first entry* point (in π) of G', if z is an entry point of G', and there is no ancestor of z in T_{π} that is an entry point of G'. If $Loc(b) = v$ we say that z is the *first entry* point of G' (in T_{π}) through v. We define $Z^{in}(G', \pi)$ to be the set of OR-nodes in T_{π} that are the first entry points of G' through s'. Let $B^{in}(G', \pi) = \{L(z) \mid z \in Z^{in}(G', \pi)\}$.

For example, Figure 4.3 depicts a policy tree, for a policy π for M from Figure 4.1. The OR-node z is the first entry point of G' in π through s'. Therefore $L(z) \in B^{in}(G', \pi)$.

We say an AND-node z, with $L(z) = b$, is a *departure point* of G' (in T_{π}) , if $PrevAction_{\pi}(b)$ is inside G', a single observation is received at b, and $NextAction_{\pi}(b)$ is outside G'. z is the *first departure* point (in π) of G', if z is a departure point of G' and there is no ancestor of z in T_{π} that is a departure point. If $Loc(b) = v$ we say that z is the *first departure point* of G' (in T_{π}) through v.

We define $Z^{succ}(G', \pi)$ to be the set of AND-nodes that are a first departure point through t'. We define $Z^{fail}(G', \pi)$ to be the set of AND-nodes that are a first departure point through s', and such that G is $s't'$-blocked in $L(z)$ for every $z \in Z^{fail}(G', \pi)$. Let $Z^{out}(G', \pi) = Z^{succ}(G', \pi) \cup Z^{fail}(G', \pi)$. Let $B^{succ}(G', \pi) = \{L(z) \mid z \in Z^{succ}(G', \pi)\}$, and $B^{fail}(G', \pi) = \{L(z) \mid z \in Z^{fail}(G', \pi)\}$. Let $B^{out}(G', \pi) = B^{succ}(G', \pi) \cup B^{fail}(G', \pi)$.

For example, the AND-nodes z_1, z_2 in Figure 4.3, are the first departure points of G' through t', and therefore $L(z_1), L(z_2) \in B^{succ}(G', \pi)$. The AND-node z_3 is the first departure point of G' through s' and G' is $s't'$-blocked in $L(z_3)$; therefore $L(z_3) \in B^{fail}(G', \pi)$. $B^{out}(G', \pi) = \{L(z_1), L(z_2), L(z_3)\}$.

Claim 2.4.2 ensures that every belief state in T_{π} cannot be reached in π from two separate belief states; therefore $B^{in}(G', \pi)$, and $B^{out}(G', \pi)$ are well

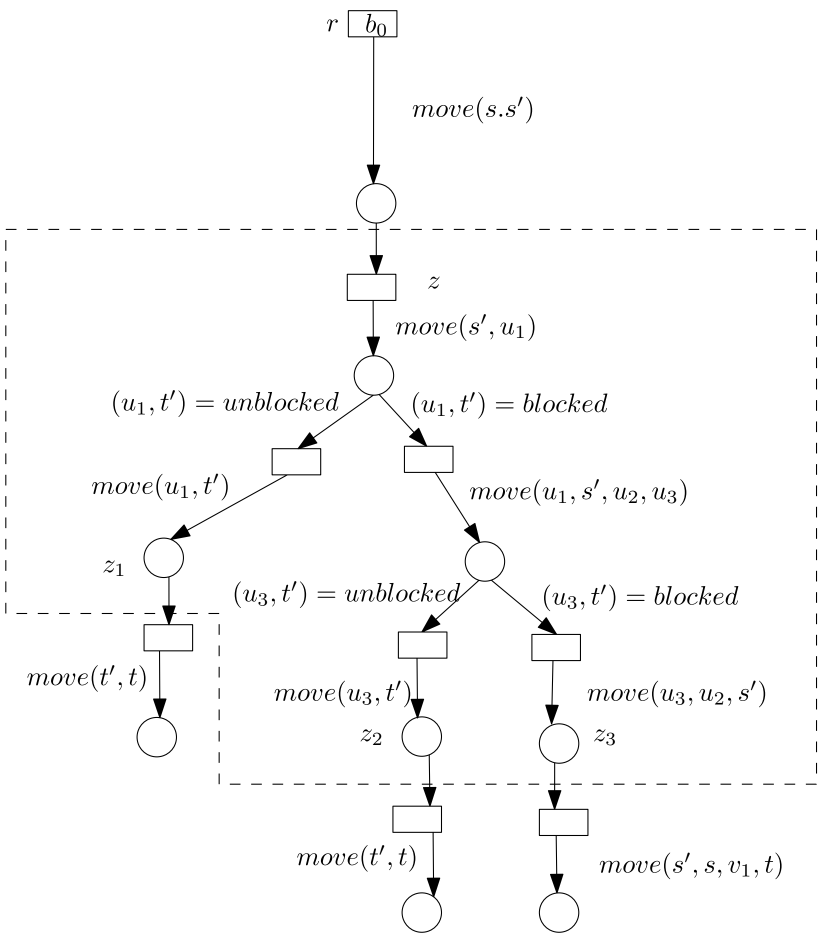


Figure 4.3: A policy π for M. The square nodes are OR-nodes, the round nodes are AND-nodes.

defined.

Recall that $B(b, \pi)$ is the set of belief states reachable in π from b. For $b \in B^{in}(G', \pi)$, let $B_b^{out}(G', \pi) = B^{out}(G', \pi) \cap B(b, \pi)$ be the belief states in $B^{out}(G', \pi)$ that are reachable from b in π. As the edges of $E_{t'} \cap (E \backslash E')$ are known to be unblocked, we have that if $b \in B^{in}(G', \pi)$, and $b_1, b_2 \in B_b^{out}(G', \pi)$, then $b_1|e = b_2|e$ for every $e \in E \backslash E'$.

We now define a committing policy. Informally π is I'-committing if once G' is first entered through s', the agent either departs G' through t' or finds G' to be $s't'$-blocked, and then departs through s'. In both cases, once G' is departed, no edge of G' is traversed again. The formal definition is as follows.

Definition 4.2.1 *A policy π is I'-committing (or (G', s', t')-committing) if the only entry points of G' in T_π are of $Z^{in}(G', \pi)$, and the only departure points z of G' in T_π are of $Z^{out}(G', \pi)$.*

When s', t' are obvious from the context we say π is G'-committing. The policy π depicted in Figure 4.3 is an M'-committing policy.

In Section 4.2.2 we see that by dividing a CTP instance I into specific sub-instances, and considering policies that are sub-graph committing, we can use a "'divide and conquer"' approach on the sub-instances of I. Thus on specific CTP instances, finding the optimal solution for every sub-instance yields an optimal solution for the general instance. For example, in a disjoint path topology (see Section 2.5), in which every two paths from s to t are vertex-disjoint, an optimal policy is a constrained policy that is (I_j, s, t)-committing for every path I_j.

Next, we establish a relation between I'-committing policies for I, and policies for I'. Let π be a policy for I, a belief state $b \in B_I$, and a collection of belief states $B \subseteq B(b, \pi)$. We define $trunc(\pi_b, B)$ to be the partial policy truncated from π_b by removing the actions in every $b' \in B$. That is, $T_{trunc(\pi_b,B)}$ is obtained from T_{π_b} by removing every out-going arc from every vertex v with $L(v) \in B$. For example, the tree in the dashed square, from Figure 4.3, is $T_{trunc(\pi_b,B)}$ for $b = L(z)$, and $B = \{L(z_1), L(z_2), L(z_3)\}$. Note as every such vertex v, with $L(v) \in B$, becomes a leaf in $T_{trunc(\pi_b,B)}$, then $C(trunc(\pi_b, B))$, the cost of $\pi_{b,B}$, can be computed as in Equation 2.8.

We say a belief state b is *between* B_1 and B_2 if there is a belief state $b_1 \in B_1$, such that b is reachable from b_1, and there is a maximal cut S in T_{π_b} such that $L(z) \in B_2$ for every $z \in S$. Informally, let π be an I'-committing policy for I. We say π *simulates* a policy π' for I' if for every belief state between $B^{in}(G', \pi)$ and $B^{out}(G', \pi)$, there is a corresponding belief state $b' \in B_{I'}$ with the same variables-status representation over the edges of E', such that $\pi(b) = \pi'(b')$. The policy π' is called the *contraction* of π to I', and $C(trunc(\pi_b, B^{out}_b(G', \pi))) = C(\pi')$ [1]. The formal definitions of policy simulation and contraction of a policy are delicate, and therefore

[1]We ignore the default edge of (s', t'), see Appendix D.1 for details.

appear in Appendix D.1. For example, the policy π' for M' on Figure 4.4 is a contraction of the policy π (see Figure 4.3) to M' (see Figure 4.2).

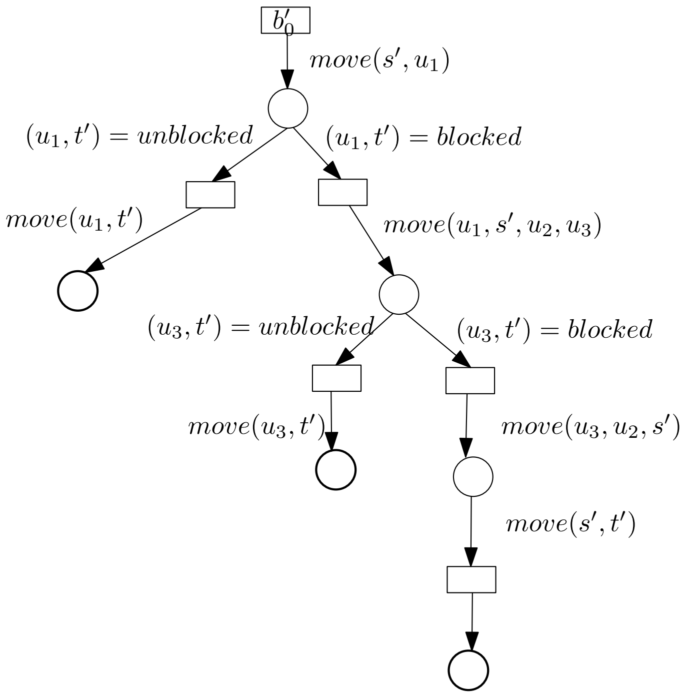


Figure 4.4: A policy π' for M', which is a contraction of π to M'.

Now suppose that π is an optimal I'-committing policy; that is, π is optimal among all I'-committing policies for I. One may ask if the contraction of π to I' is an optimal policy for I'. Indeed the answer is positive, as shown in the following lemma.

Lemma 4.2.2 *For every optimal I'-committing policy π for I, the contraction of π to I' is an optimal policy for I'.*

Proof: Let π be an optimal I'-committing policy for I, and let $b \in B^{in}(G', \pi)$. By an argument identical to Claim 2.3.1, we can assume that π_b is an optimal I'-committing policy for I_b. Let π' be the contraction of π_b to I'. Assume in contradiction that π' is not optimal. Then there is a policy χ' for I' such that $C(\chi') < C(\pi')$. Let χ_b be a policy for I_b that simulates χ'.

First note that for every $b_1 \in B_b^{fail}(G', \pi_b)$, and $d_1 \in B_b^{fail}(G', \chi_b)$, we have $Loc(b_1) = Loc(d_1)$, and $b_1|e = d_1|e$ for every $e \in E \backslash E'$. This due to the fact that the edges of $E_t \cap (E \backslash E')$ are known to be unblocked. Let E'-*exclusive* be the following constraint "the edges of E' cannot be traversed". As π_b is an optimal I'-committing policy, then π_{b_1} is an E'-exclusive policy. Again, by the same argument as in Claim 2.3.1, π_{b_1} can be assumed to be optimal among all E'-exclusive policies for I_{b_1}.

Therefore, there is an E'-exclusive policy ν_{d_1} for I_{d_1}, such that $C(\pi_{b_1}) = C(\nu_{d_1})$ (see Lemma D.2.1 in Appendix D for details). Following the same argument, for every $b_2 \in B_b^{succ}(G', \pi_b)$, and $d_2 \in B_b^{succ}(G', \chi_b)$, there is an E'-exclusive policy ν_{d_2} for I_{d_2}, such that $C(\pi_{b_2}) = C(\nu_{d_2})$.

Now construct a policy ν_b for I_b, from the policy χ_b, as follows. For every AND-node $z \in T_{\chi_b}$ where $L(z) = d_1$, and $d_1 \in B_b^{fail}(G', \chi_b)$, extract $T_{\chi_b}(z)$, and attach $T_{\nu_{d_1}}(z)$ instead. Likewise, for every AND-node $z \in T_{\chi_b}$ where $L(z) = d_2$, and $d_2 \in B_b^{succ}(G', \chi_b)$, extract $T_{\chi_b}(z)$, and attach $T_{\nu_{d_2}}(z)$ instead. Then ν_b is I'-committing, and we have

$$C(\nu_b) = C(\chi') + P_{G'}C(\pi_{b_1}) + (1 - P_{G'})C(\pi_{b_2})$$

Note that

$$C(\pi_b) = C(\pi') + P_{G'}C(\pi_{b_1}) + (1 - P_{G'})C(\pi_{b_2})$$

Therefore, as $C(\chi') < C(\pi')$, we have that $C(\nu_b) < C(\pi_b)$, contradicting the optimality of π_b as an optimal I'-committing policy for I_b.

$\square$

4.2.2 Decomposing CTP instances

Let $I = (G, s, t, p, w)$ be a CTP instance with $I' = (G', s', t)$, a sub-instance of I where $G' = (V', E')$. Note that the goal vertex of I and I' is the same. Let π^* be an optimal I'-committing policy for I, and let π'^* be an optimal policy for I'. By Lemma 4.2.2 we can assume that π'^* is the contraction of π^* to I'. Therefore, if $b \in B^{in}(G', \pi^*)$, then $trunc(\pi_b^*, B_b^{out}(G, \pi^*))$ can be considered as a single macro-action, denoted by $TRY(I')$, and we say that I' is *tried* in π^*. When I' is obvious from the context, we denote $TRY(I')$

by $TRY(G')$, and say that G' is tried in π^*.

The results of $TRY(I')$ are either that the agent is at t, or that the agent is at s' and G' is found to be $s't$-blocked. Note that $E[TRY(I')] = C(\pi'^*)$. Let $Q_{G'} = 1 - P_{G'}$ be the probability that G' is not blocked. Let $D_{G'} = \frac{E[TRY(G')]}{Q_{G'}}$ (for now we assume that $P_{G'} < 1$, see remark 4.2.4 below). When G' is obvious from the context we sometimes denote $E[TRY(G')]$ by $C(G')$. The parameter $D_{G'}$, called the *factored cost* of G', is a property of the subgraph (G', s', t) and is used for comparisons between different subgraphs. Note that these definitions are a generalization of the $TRY(i)$ and D_i obtained in disjoint path graphs (see Section 2.5).

Let $I_1 = (G_1, s', t)$ and $I_2 = (G_2, s', t)$ be sub-instances of a CTP instance $I = (G, s, t, p, w)$ such that $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Assume that $V_1 \cap V_2 = \{s', t\}$. Then by traversing G_1 no edges of G_2 are revealed, and by traversing G_2 no edges of G_1 are revealed. Let π be a policy for I that is both G_1-committing and G_2-committing. Then $TRY(G_1)$ is performed in every belief state in $B^{in}(G_1, \pi)$, and $TRY(G_2)$ is performed in every belief state in $B^{in}(G_2, \pi)$. We say that G_2 *succeeds* G_1 in π if $B^{fail}(G_1, \pi) = B^{in}(G_2, \pi)$ [2]. An alternative policy for I can be a policy π', obtained from π, in which G_1 succeeds G_2. We then say π' is obtained from π by *switching* G_1 and G_2. Note that this switching is possible as the entry and departure points of G_1 and G_2 are the same. We denote the constraint: " π is (G_1, G_2)-committing and G_2 is successor to G_1" by G_2*-succ-*G_1, and the constraint: "π is (G_1, G_2)-committing and G_1 is successor to G_2" by G_1*-succ-*G_2.

Lemma 4.2.3 *Let π be a G_2-succ-G_1 optimal policy for I, and let π' be a G_1-succ-G_2 optimal policy for I, such that π' is obtained from π by switching G_1 and G_2. Then $C(\pi) < C(\pi')$ iff $D_{G_1} < D_{G_2}$.*

Proof: As π' is obtained from π by switching G_1 and G_2, we have that $B^{in}(G_1, \pi) = B^{in}(G_2, \pi')$. Let $b \in B^{in}(G_1, \pi)$, and let $v \in T_\pi$, and $v' \in T_{\pi'}$ be such that $L(v) = L(v') = b$. We show that $V^\pi(v) < V^{\pi'}(v')$ iff $D_{G_1} < D_{G_2}$. Then proof follows by backward induction as in Claim 2.3.1.

[2]We can make this comparison as only a single observation is received in every belief state in $B^{fail}(G_1, \pi)$.

Note that for every $b_1 \in B_b^{fail}(G_2, \pi)$ and $b_2 \in B_b^{fail}(G_1, \pi')$, we have $Loc(b_1) = Loc(b_2)$, and $b_1|e = b_2|e$ for every $e \in E \backslash (E_1 \cup E_2)$. Let (E_1, E_2)-*exclusive* be the following constraint: "the edges of $E_1 \cup E_2$ cannot be traversed". As π is G_2-succ-G_1 optimal, and as π' is G_1-succ-G_2 optimal, we have that π_{b_1} is (E_1, E_2)-exclusive optimal for I_{b_1}, and π'_{b_2} is (E_1, E_2)-exclusive optimal for I_{b_2}. Now from Lemma D.2.1 in Appendix D, we have that $C(\pi_{b_1}) = C(\pi_{b_2})$. Denote $C(\pi_{b_1})$ by W. Then

$$V^{\pi}(v) = C(G_1) + P_{G_1}(C(G_2) + P_{G_2}W) \tag{4.1}$$

and

$$V^{\pi'}(v') = C(G_2) + P_{G_2}(C(G_1) + P_{G_1}W) \tag{4.2}$$

which implies
$V^{\pi}(v) < V^{\pi'}(v')$ iff $D_{G_1} < D_{G_2}$
as required.

□

Remark 4.2.4 *Note that if w.l.o.g. $P_{G_2} = 1$ then it follows straight from Equations 4.1 and 4.2 that $V^{\pi}(v') \leq V^{\pi'}(v)$. Therefore when comparing a policy with a switched policy, we assume throughout this work that $P_G < 1$ for every graph G.*

We now generalize the concept of "switching policies", and define a partition of a CTP instances to several sub-instances.

Definition 4.2.5 *A CTP instance $I = (V, E, s, t, p, w)$ (where $G = (V, E)$) is a partition of CTP-instances $((G_1, s', t), \cdots (G_k, s', t))$ of M, where $G_i = (V_i, E_i)$ if:*

- $\bigcup_{i<k} V_i = V$ *and* $\bigcup_{i \leq k} E_i = E$.
- $V_i \cap V_j = \{s', t\}$

I is called a $\{G_1, \cdots, G_k\}$-CTP partition .

Let π be an optimal policy for a $\{G_1, \cdots, G_k\}$-CTP partition I, and assume that π is G_i-committing for every $i \leq k$. By Lemma 4.2.2, we may assume that every contraction of π to G_i is an optimal policy for G_i. Therefore, as every edge in E belongs to some E_i, the policy π can be described as a permutation of $TRY(G_i)$ macro actions over $\{1 \cdots k\}$ as follows. At step i, unless t is reached, perform $TRY(G_i)$. Assuming w.l.o.g that the permutation order in π is $\{1, \cdots k\}$, we have that the cost of $C(\pi)$ is

$$C(\pi) = \sum_{i<k} (\prod_{l<i} P_{G_i}) C(\pi_{G_i}) \tag{4.3}$$

Then the following corollary is immediate from Lemma 4.2.3.

Corollary 4.2.6 *The optimal order of π is a non-decreasing order of the factored costs of the G_i.*

Using Corollary 4.2.6 we present a divide and conquer framework called the *partition framework* for finding optimal policies for a CTP instance I:

- Find a partition of I to CTP sub-instances $\{G_i \mid i \leq k\}$ for some k.
- Find D_{G_i} for every $i \leq k$.
- Show that a $\{G_i \mid i < k\}$-committing policy for I is optimal among all policies for M.

The difficulty of course is to find the "right" partition, assuming there is one, as every step in this framework can be intractable. To find D_{G_i}, one has to find the optimal cost for G_i, which is a PSPACE-complete problem (see Chapter 3). Finding Q_{G_i} is a #P-complete problem [41]. Also note that finding the optimal cost, is different from finding the actual optimal policy (or the first move in such policy): sometimes one is easy to solve while the other is hard. However, we suspect that by using this framework we can find policies with a better approximation to the optimal cost. For example, we can decompose a CTP instance into several sub-instances, where in each sub-instance a different heuristic is implemented in order to approximate

D_{G_i}, and thus provide a more accurate solution. In Chapter 5 we give an example of where this framework can be implemented and yield an optimal solution for various CTP instances. Note that it is generally not true that every optimal policy can be subgraph-partitioned, as the observations that the agent receives during traversal in a certain subgraph can affect the agent's decision making after a subgraph is departed.

Remark 4.2.7 *Recall that a vertex $v \in V$ is explored in a belief state b if the status of all its incident edges is known in b. Let $U(G) \subseteq V$ be the set of all vertices in V that are explored in b_0. By re-defining commitment of policy, Corollary 4.2.6 still holds for subgraphs with mutual vertices in $U(G)$. Therefore we can generalize the partition framework to such subgraphs. See Appendix D.3 for the exact details.*

Chapter 5

The CTP on Trees

CTP-Tree, defined below, is a CTP in which all the vertices in the CTP graph, apart from t, form a tree with s being the root. CTP-Tree is a generalization of CTP-DISJ (Section 2.5). As CTP-DISJ has a polynomial time solution, while CTP on a general graph is PSPACE-complete, the analysis of CTP-Tree is a natural research direction. We can only conjecture that CTP-Tree is intractable. However, as the st-reliability problem has a polynomial time solution on trees (well known, but see Lemma E.0.3 in Appendix E for the proof), we provide by using the partition framework (Section 4.2.2), several special variants for CTP-Tree for which there is a polynomial time solution.

A *free edge* is a zero cost edge known to be unblocked. *CTP-Tree* is a CTP instance $T = (V, E, s, t, p, w)$ such that the graph $(V \backslash \{t\}, E \backslash E_t)$ is a tree with a root s. The edges of E_t are free edges, called *terminal free edges*, which connect t with every leaf in $(V \backslash \{t\}, E \backslash E_t)$; see Figure 5.1.

As the objective in CTP is to find a strategy that minimizes the expected travel cost from s to t, we may assume that once a leaf l in $(V \backslash \{t\}, E \backslash E_t)$ is reached, the edge (l, t) is traversed. Therefore throughout this work we consider a CTP-Tree T as if it were a tree with a root s. The objective is then to find a policy that minimizes the expected travel cost from s to a leaf l in V [1].

[1]We can assume that the default edge is an edge (s, t') in which (t', t) is a terminal free edge.

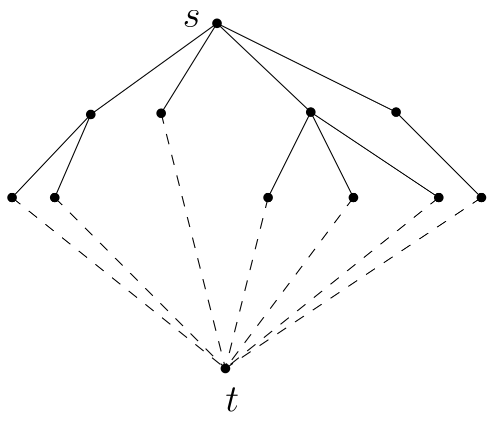


Figure 5.1: CTP Trees. Dashed edges are terminal free edges.

The problem of whether CTP-Tree admits a polynomial time solution is still open. The main difficulty lies in the fact that an optimal policy for a subtree does not necessarily yield an optimal policy for the entire tree. Therefore, unlike many solutions to problems with a tree layout, a dynamic programming method seems unlikely to work; see Figure 5.2 and Figure 5.3 for examples. In what follows we provide several approaches and variants for which a polynomial time solution can be found. For clarification, the policy tree of a policy π is denoted by $\mathcal{T}_\pi$ throughout this chapter.

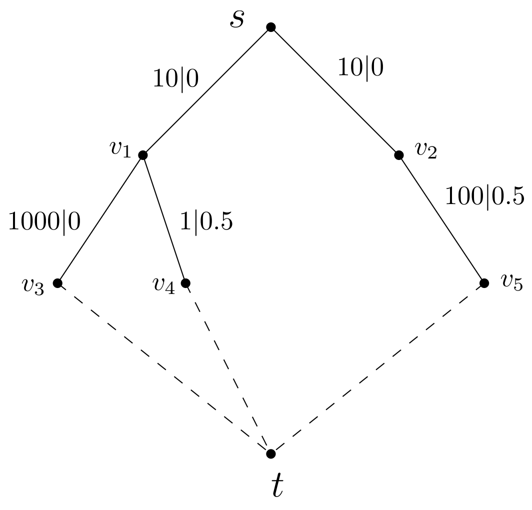


Figure 5.2: CTP Trees. The optimal policy (with a cost of 300.5) is to traverse (s, v_1); if (v_1, v_4) is blocked, traverse (s, v_1) and (s, v_2) to v_2. If (v_2, v_5) is blocked, retrace to v_1 and reach t through (v_1, v_3).

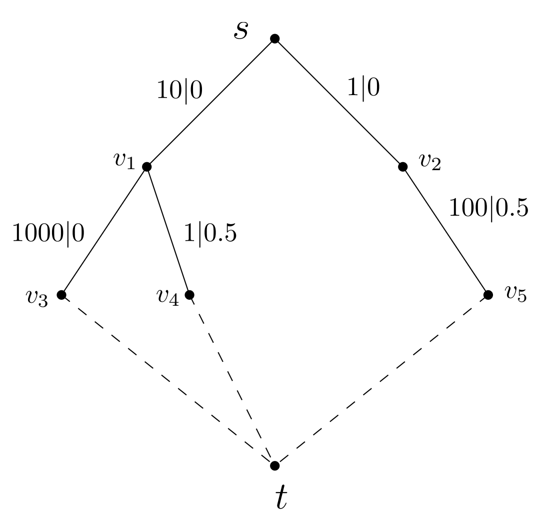


Figure 5.3: CTP Trees. The optimal policy (with a cost of 290.5) is to traverse (s, v_2). Then, regardless of whether (v_2, v_5) is blocked or unblocked, traverse (s, v_2) and (s, v_1) to v_1. If (v_1, v_4) is blocked, and (v_2, v_5) was found unblocked, retrace to v_2 and reach t through (v_2, v_5). (v_1, v_4) is unblocked then reach t through v_4

.

5.1 Trees with no exploration vertices

Recall that $T(v)$ is the subtree of a tree T with a root $v \in T$. $T^{Par}(v)$, for $v \neq s$, is the subtree gained from $T(v)$ with $Parent(v)$ as an additional vertex, and $(Parent(v), v)$ as an additional edge; see Figure 5.4. The probability that a subtree T with a root v is vt-blocked is denoted by P_T. For a constrained CTP-Tree T, a vertex $v \neq s$ is called a *committing-vertex*, if the only policies that are considered as a solution are those that are $(T^{Par}(v), v, t)$-committing (then the objective is to find an optimal $(T^{Par}(v), v, t)$-committing policy). If v is a committing vertex, and the agent traverses the edge $(Parent(v), v)$, then the agent retraces to $Parent(v)$ if and only if $T(v)$ is found to be vt-blocked. Thus using methods acquired from Section 4.2, an optimal solution for $T^{Par}(v)$ can be used to find an optimal solution for CTP with a committing vertex v. A vertex $v \neq s$ in T that is not committing is called an *exploration vertex* . A CTP-Tree T is called k-Exp-CTP-Tree, if there are at most k exploration vertices in T. 0-Exp-CTP-Tree is a CTP-Tree in which all the vertices, apart from s, are committing.

Theorem 5.1.1 *0-Exp-CTP-Tree admits a polynomial time solution.*

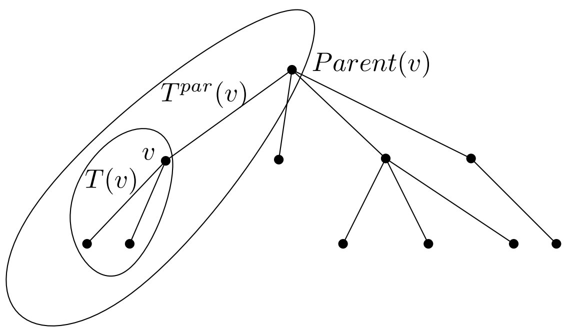


Figure 5.4: CTP Trees. $T^{Par}(v)$ has an additional vertex $Parent(v)$, and an edge $(Parent(v), v)$.

To prove Theorem 5.1.1, we present a recursive polynomial time algorithm, based on "sorted DFS", called *NoExpTreeSolver* (see Algorithm 1), which provides a polynomial time solution for 0-Exp-CTP-Trees.

Given a vertex $v \in V$ of a 0-Exp-CTP-Tree T, **NoExpTreeSolver**(v) returns the optimal cost for $T(v)$, denoted by $C(T(v))$, and the first action in a policy that achieves this optimum. Therefore **NoExpTreeSolver**(s) returns the optimal cost for T and a first action in an optimal policy for T. For $v \in T$, recall that v and all the children of v are committing. Therefore by using the partition framework described in Section 4.2.2, an optimal policy for $T(v)$ can be represented as a permutation over the macro-actions $TRY(T^{Par}(v_i))$ for the children v_i of v. The value $C'(u)$, where u is a child of v, is $E[TRY(T^{Par}(u))]$, which is computed to be:

$$E[TRY(T^{Par}(u))] = (1 - p((v, u)))\big(w((v, u)) + C(T(u)) + P_{T(u)}w((v, u))\big)$$

As $T(u)$ is a tree, $P_{T(u)}$ can be recursively found in polynomial time, see Lemma E.0.3 in Appendix E. Note that *BestCost* in Algorithm 1, is computed as in Equation 4.3 . Therefore the correctness and optimality of

NoExpTreeSolver is immediate from Corollary 4.2.6; *BestCost* is indeed the optimal cost for $T(v)$, and *BestAction* is a first move in a policy that achieves this optimum.

Algorithm 1: NoExpTreeSolver(v)

```
BestAction = NULL, BestCost = INF;
if  $v = t$ then
    return $(NULL, 0)$  /*target reached, cost 0, no action*/
$N_v \leftarrow$ the children of $v$;
foreach vertex $u \in N_v$ do
    $C'(u) \leftarrow (1 - p((v,u)))\big((1 + P_{T(u)})w((v,u)) + NoExpTreeSolver(u).BestCost\big)$;
    $D'(u) \leftarrow \frac{C'(u)}{1-P_{T^{Par}(u)}}$;
sort $N_v$ to an array $\{z_1, \cdots, z_l\}$ in a non-decreasing order of $D'(z_i)$;
$BestCost \leftarrow \sum_{i<l} \prod_{x<i} P_{T^{Par}(z_x)} D'(z_i)$;
$BestAction \leftarrow move(v, z_1)$;
return (BestCost, BestAction)
```

Note that for every vertex in T, **NoExpTreeSolver** is recursively called exactly once. As the calculation of $P_{T^{Par}(u)}$ for every vertex u can be done in $O(n)$, and as the children of every vertex v are sorted, we have that **NoExpTreeSolver** admits a run time of $O(n^2 log(n))$ [2].

5.2 Polynomial time solution for 1-Exp-CTP-Tree

We next discuss 1-Exp-CTP-Trees, which contain only a single designated exploration vertex. We provide a polynomial time solution for instances of 1-Exp-CTP-Tree in which the adjacent edges of the (single) exploration vertex are unblocked. This result can be easily extended for instances of 1-Exp-CTP-Tree in which only the outgoing edges of the exploration vertex are unblocked.

[2]In fact, the run time can be reduced to $O(nlogn)$ with a more careful calculation.

Let $T = (V, E, s, t, p, w)$ be a 1-Exp-CTP-Tree instance, and let $v^1 \in V$ be the exploration vertex in V. Assume that the outgoing edges of v are all unblocked. As $v^1 \neq s$ then v^1 has a parent $Parent(v^1)$, which is denoted in this section by v^0 (see Figure 5.5). Next, note that as every descendant u of v^1 (where $u \neq v^1$) is a committing vertex, the optimal cost of $T^{Par}(u)$ can be found in polynomial time by using *NoExpTreeSolver* (see Section 5.1 for details). Likewise, if u is a descendant of a sibling of v^1, then $T^{Par}(u)$ can be found in polynomial time by using *NoExpTreeSolver* as well. Thus, if u is a child of v^1 or a sibling of v^1, and the agent is located in u, then $Parent(u)$ is retraced if and only if $T(u)$ is found blocked. We say that a child u of v^0 or v^1 is *chosen* by the agent, if the next action performed by the agent is $TRY(T^{Par}(u))$. Table 5.1 shows the possible actions in every reasonable policy π when the agent is located anywhere in $T(v^0)$.

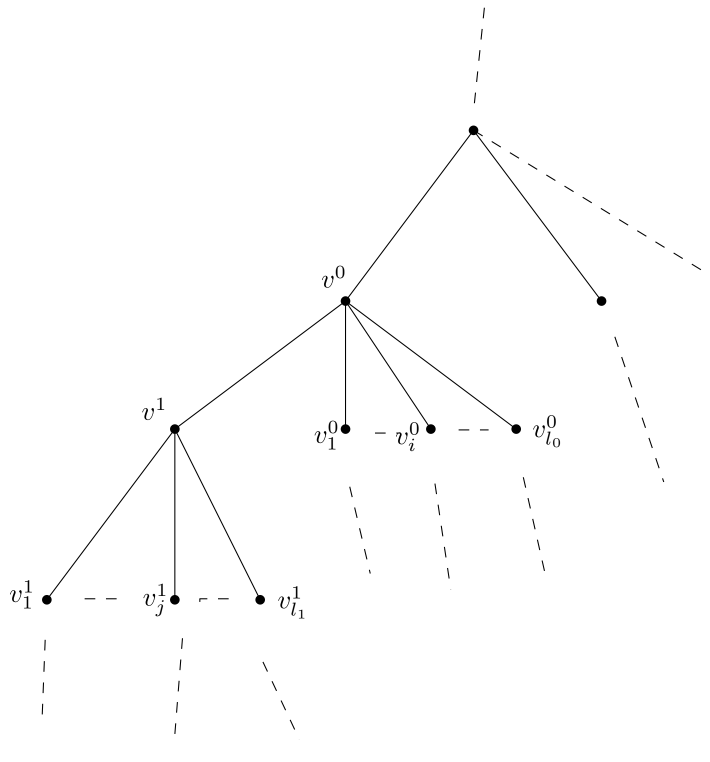


Figure 5.5: 1-Exp-CTP-Tree. v^0 is the parent of v^1. The children of v^1 are denoted by $\{v_1^1 \cdots v_{l_1}^1\}$. The siblings of v^1 are denoted by $\{v_1^0 \cdots v_{l_0}^0\}$.

As crossing (v^0, v^1) back and forth is clearly not optimal, it remains to

Table 5.1: Reasonable policy actions in π

Location	Action
v^0	cross (v^0, v^1) or choose the next child of v^0, see Rule 5.2.1
v^1	cross (v^0, v^1) or choose the next child of v^1 , see Rule 5.2.2
u: a child of v^1,	perform $TRY(T(u))$; if blocked, retrace to v^1
u: a sibling of v^1,	perform $TRY(T(u))$; if blocked, retrace to v^0
v^0 and $T(v^0)$ is blocked	cross $(Parent(v^0), v^0)$

see how the next sibling of v^1 is chosen at v^0, in case (v^0, v^1) is not crossed, and how the next child of v^1 is chosen at v^1, in case (v^0, v^1) is not crossed.

Let π be an optimal policy for T, and let $z^0 \in \mathcal{T}_\pi$ be a node in which $(T(v^0), v^0, t)$ is first entered. Let $L(z^0) = b^0$. As every reasonable policy must meet the restriction in Table 5.1, we may assume that the CTP sub-instance of T with the initial state b^0, can be described as a $\{G_i^0 | i \leq k_0\}$-partition, for $k_0 > 0$, as follows. For every $i < k_0$, either $G_i^0 = T^{Par}(u)$, where u is a sibling of v^1, or G_i^0 is a subgraph of $T^{par}(v^1)$ (which fully contains one or more subtrees of children of v^1). The last subgraph in the partition, $G_{k_0}^0$, is distinct from $T(v^0)$ and is entered through $(Parent(v^0), v^0)$ when $T(v^0)$ is found blocked. In this case, v^0 is not retraced again; therefore t is reached in $G_{k_0}^0$.

Denote the set of vertices of graph G_i^0, for $i \leq k_0$, by $V^{G_i^0}$. Then, as the outgoing edges of v^1 are unblocked, the vertices of $V^{G_i^0} \cap V^{G_j^0}$ (apart from v^0) are exposed vertices. [3], Therefore, following the partition framework in Section 4.2.2, and Remark 4.2.7, for subgraphs with mutual exposed vertices, π_{b^0} can be described as a permutation of macro actions $TRY(G_i^0)$ of $i < k_0$. Then by Corollary 4.2.6, we have the following rule.

Rule 5.2.1 *Let u_1, u_2 be siblings of v^1. Then at v^0, if $T(v^0)$ is not blocked, it is optimal to choose u_1 before u_2 if and only if $T^{Par}(u_1)$ is unblocked, and $D(T^{Par}(u_1)) \leq D(T^{Par}(u_2))$.*

Rule 5.2.1 states the next sibling of v^1 that is to be chosen in every reasonable policy when the location of the agent is v^0 (in case (v_0, v_1) is not

[3]The edge (v^0, v^1) can be assumed to be unblocked as well, as otherwise the entire problem becomes trivial.

traversed of course).

We assume w.l.o.g. that the G_i^0 are ordered such that if u_1, u_2 are siblings of v^1, $G_i^0 = T^{Par}(u_1)$, $G_j^0 = T^{Par}(u_1)$, and $D(T^{Par}(u_1)) \leq D(T^{Par}(u_2))$, then $i < j$.

We now repeat the same argument, but for v^1, with $z^1 \in \mathcal{T}_\pi$ being a node in which $(T(v^1), v^1, t)$ is first entered . Let $L(z^1) = b^1$. As before, we may assume that the CTP sub-instance of T with the initial state b^1 is a $\{G_i^1 | i \leq k_1\}$-partition as follows. For every $i < k_1$, either $G_i^1 = T^{Par}(u)$, where u is a child of v^1, or G_i^1 is a subgraph of $T^{par}(v^0)$, which contains (v^0, v^1) and is distinct from $T(v^1)$ (G_i^1 fully contains several subtrees of siblings of v^1). Again, the last subgraph, $G_{k_1}^1$, is distinct from $T(v^1)$, and is entered when $T(v^1)$ is found blocked. Following the partition framework, we have that π_{b^1} is also a permutation of macro actions $TRY(G_i^1)$ of $i < k_1$, and by Corollary 4.2.6 and Remark 4.2.7, we have the following rule:

Rule 5.2.2 *Let u_1, u_2 be children of v^1. Then at v^1, if $T(v^1)$ is not blocked, it is optimal to choose u_1 before u_2 if and only if $T^{Par}(u_1)$ is unblocked, and $D(T^{Par}(u_1)) \leq D(T^{Par}(u_2))$.*

Rule 5.2.2 states the next child of v^1 that is to be chosen in every reasonable policy when the location of the agent is v^1 (again, in case (v_0, v_1) is not traversed).

We again assume w.l.o.g. that the G_i^1 are ordered such that if u_1, u_2 are children of v^1, $G_i^1 = T^{Par}(u_1)$, $G_j^1 = T^{Par}(u_1)$, and $D(T^{Par}(u_1)) \leq D(T^{Par}(u_2))$, then $i < j$.

Using Table 5.1, and Rules 5.2.1 and 5.2.2, we provide a dynamic programming algorithm that runs in polynomial time, and computes the optimal cost of $T(v^0)$ and the first move in a policy that achieves this optimum. Denote the siblings of v^0 by $\{v_1^0, \cdots, v_{l_0}^0\}$ such that $D(T^{Par}(v_i^0)) \leq D(T^{Par}(v_{i+1}^0))$, and the children of v^1 by $\{v_1^1, \cdots, v_{l_1}^1\}$ such that $D(T^{Par}(v_i^1)) \leq D(T^{Par}(v_{i+1}^1))$.

For $k \in \{0, 1\}$, note that as every vertex in $T(v_i^k)$ is committing, then $D(T^{Par}(v_i^k))$ can be found in polynomial time by using **NoExpTreeSolver** to calculate $C(T^{Par}(v_i^k))$ (finding $P_{T^{Par}(v_i^k)}$ is easy; see Lemma E.0.3 in Appendix E).

Next, we construct a dynamic programming table H of size $2 \times (l_0 + 1) \times (l_1 + 1)$. The cell $H(0,i,j)$ holds the optimal cost for $(T^{Par}(v^0), v^0, t)$ at every belief state b in which (v^0, v^1) is unblocked, $Loc(b) = v^0$, the only not known to be blocked subtrees of v^0 (apart from $T^{Par}(v_1)$) are those of $\{T^{Par}(v_i^0) \cdots T^{Par}(v_{l_0}^0)\}$, and the only not known to be blocked subtrees of v^1 are those of $\{T^{Par}(v_j^1), \cdots T^{Par}(v_{l_1}^1)\}$. An additional variable $BestAction(0,i,j)$ holds the first action in an optimal policy for I_b.
The cell $H(1,i,j)$ holds the optimal cost for $(T^{Par}(v^0), v^0, t)$ at every belief state b in which (v^0, v^1) is unblocked, $Loc(b) = v^1$, the only not known to be blocked subtrees of v^0 (apart from $T^{Par}(v_1)$) are those of $\{T^{Par}(v_i^0) \cdots T^{Par}(v_{l_0}^0)\}$ and the only not known to be blocked subtrees of v^1 are those of $\{T^{Par}(v_j^1), \cdots T^{Par}(v_{l_1}^1)\}$. $BestAction(1,i,j)$ holds the first action in an optimal policy for I_b. For $i = l_0 + 1$, the cells $H(0,i,j)$ and $H(1,i,j)$ hold the optimal cost when all the subtrees of v^0, apart from $T(v^1)$, are known to be blocked. Likewise, for $j = l_1 + 1$, the cells $H(0,i,j)$ and $H(1,i,j)$ hold the optimal cost when all the subtrees of v^1 are known to be blocked. See example in Figure 5.6.

Then H is computed as follows.

- For every $k \in \{0,1\}$ and $j \leq l_1 + 1$, $H(k, l_0 + 1, j)$ is computed using **NoExpTreeSolver**.

- For every $k \in \{0,1\}$ and $i \leq l_0 + 1$, $H(k, i, l_1 + 1)$ is computed using **NoExpTreeSolver**.

- For $k = 0$, $i \leq l_0$ and $j \leq l_1$ we have

$$\begin{aligned} H(0,i,j) = min\Big\{ & C(T^{Par}(v_i^0)) + P_{T^{Par}(v_i^0)} H(0, i+1, j), \\ & w((v^0, v^1)) + C(T^{Par}(v_j^1)) + P_{T^{Par}(v_j^1)} H(1, i, j+1) \Big\} \end{aligned} \tag{5.1}$$

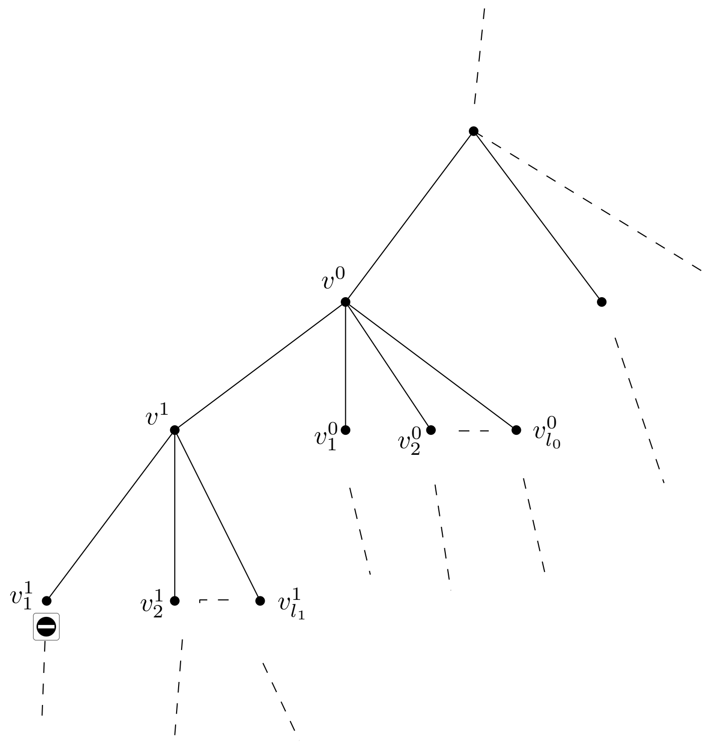


Figure 5.6: 1-Exp-CTP-Tree, in a belief state b where $T^{Par}(v_1^1)$ is known to be blocked. Assume $Loc(b) = v^0$. Then the optimal cost for $(T^{Par}(v^0), v^0, t)$ at b is computed in $H(0, 1, 2)$.

- For $k = 1$, $i \leq l_0$ and $j \leq l_1$ we have

$$\begin{aligned} H(1,i,j) = min\Big\{ & C(T^{Par}(v_j^1)) + P_{T^{Par}(v_j^1)} H(1,i,j+1), \\ & w((v^0,v^1)) + C(T^{Par}(v_i^0)) + P_{T^{Par}(v_i^0)} H(0,i+1,j) \Big\} \end{aligned} \quad (5.2)$$

Lemma 5.2.3 $H(0, 1, 1)$ *is the optimal cost for* $T(v^0)$.

Proof. By backward induction on i, j.

- When $T^{Par}(v_1^0) \cdots T^{Par}(v_{l_0}^0)$ are all blocked, the vertex v^1 is de facto a committing vertex. Therefore, for every $k \in \{0, 1\}$ and $j \leq l_1 + 1$, $H(k, l_0 + 1, j)$ is computed using **NoExpTreeSolver**.

- When $T^{Par}(v_1^1) \cdots T^{Par}(v_{l_1}^1)$ are all blocked then $T(v^1)$ is blocked. Therefore, for every $k \in \{0,1\}$, and $i \leq l_0 + 1$, $H(k, i, l_1 + 1)$ is computed using **NoExpTreeSolver** as well.

  Assume that $H(k, i, j+1)$ and $H(k, i+1, j)$ hold the optimal cost.

- In order to compute $H(0, i, j)$ for $i \leq m$ and $j \leq l$, there are two reasonable policies that can be considered. $BestAction(0, i, j)$ is the first action in a policy that achieves this optimum:

  1) Perform $TRY(T^{Par(v_i^0)})$. With probability $P_{T^{Par}(v_i^0)}$, the tree $T^{Par}(v_i^0)$ is found blocked, a belief state b with $Loc(b) = v^0$ is reached, in which by the induction assumption the optimal cost is $H(0, i+1, j)$. This is for a total cost of

  $$C(T^{Par}(v_i^0)) + P_{T^{Par}(v_i^0)} H(0, i+1, j)$$

  2) Cross (v^0, v^1), and then perform $TRY(T^{Par}(v_j^1))$. Then with probability $P_{T^{Par}(v_j^1)}$, the subtree $T^{Par}(v_j^1)$ is blocked, a belief state b, with $Loc(b) = v^1$ is reached, in which by the induction assumption the optimal cost is $H(1, i, j+1)$. This is for a total cost of

  $$w((v^0, v^1)) + C(T^{Par}(v_j^1)) + P_{T^{Par}(v_j^1)} H(1, i, j+1)$$

- Similarly, in order to compute $H(1, i, j)$ for $i \leq l_0$ and $j \leq l_1$, there are two reasonable policies that can be considered, and the policy with the minimum cost is computed in $H(1, i, j)$. $BestAction(1, i, j)$ is the first action in a policy that achieves this optimum.

  1) Perform $TRY(T^{Par}(v_j^1))$. With probability $P_{T^{Par}(v_j^1)}$, the subtree $T^{Par}(v_j^1)$ is found blocked, a belief state b, with $Loc(b) = v^1$ is reached, in which by the induction assumption the optimal cost is $H(1, i, j+1)$. This is for a total cost of

  $$C(T^{Par}(v_j^1)) + P_{T^{Par}(v_1^1)} H(1, i, j+1)$$

2) Cross (v^0, v^1), and perform $TRY(T^{Par}(v_i^0))$. With probability $P_{T^{Par}(v_i^0)}$, the subtree $T^{Par}(v_i^0)$ is blocked, a belief state b, with $Loc(b) = v^0$ is reached, in which by the induction assumption the optimal cost is $H(0, i+1, j)$. This is for a total cost of

$$w((v^0, v^1)) + C(T^{Par}(v_i^0)) + P_{T^{Par}(v_i^0)} H(0, i+1, j)$$

□

We can now prove the following theorem.

Theorem 5.2.4 *1-Exp-CTP-Tree admits a polynomial time solution when the edges adjacent to the exploration vertex are known to be unblocked.*

Algorithm 2: OneExpTreeSolver(v)

```
BestAction = NULL, BestCost = INF;
if $v = t$ then
    return $(NULL, 0)$  /*target reached, cost 0, no action*/
if $v$ is a parent of an exploration vertex then
    $BestCost \leftarrow \frac{H(0,1,1)}{1-P_{T^{Par}(v)}}$;
    $BestAction \leftarrow BestAction(0, 1, 1)$;
    return (BestCost, BestAction)
else
    $N_v \leftarrow$ the children of $v$;
    foreach vertex $u \in N_v$ do
        $C'(u) \leftarrow$
        $(1 - p((v,u)))\Big((1 + P_{T(u)})w((v,u)) + OneExpTreeSolver(u)\Big)$;
        $D'(u) \leftarrow \frac{C'(u)}{1-P_{T^{Par}(u)}}$;
    sort $N_v$ to an array $\{z_1, \cdots, z_l\}$ in non-decreasing order of $D'(z_i)$;
    $BestCost \leftarrow \sum_{i<l} \prod_{x<i} P_{T^{Par}(z_x)} D'(z_i)$;
    $BestAction \leftarrow move(v, z_1)$;
    return (BestCost, BestAction)
```

To prove Theorem 5.2.4, we provide an algorithm called **OneExpTree-Solver** (see Algorithm 2), which is similar to **NoExpTreeSolver**. **One-**

ExpTreeSolver(s) computes the optimal cost of a 1-Exp-CTP-Tree T in polynomial time, and returns the first action in a policy that achieves this optimum. The correctness of **OneExpTreeSolver** is a direct result of Lemma 5.2.3, and Theorem 5.1.1.

Note that by slightly modifying Algorithm 2 to cover cases where the edge $(Parent(v), v)$ is found blocked, one can easily extend Theorem 5.2.4 to 1-Exp-CTP-Trees where the edge $(Parent(v), v)$ is unknown.

As **NoExpTreeSolver** takes $O(n^2 logn)$, computing the table H can be done in $O(n^4 logn)$. As **OneExpTreeSolver** is called recursively once for every vertex $v \in T$, in which either the table H is filled (only once), or **NoExpTreeSolver** is repeated, the total run time of **OneExpTreeSolver** is $O(n^4 logn)$.

5.3 Optimal policy for EFC-CTP-Tree

Recall that a balanced tree is a tree in which every two vertices of the same depth have the same height. Denote the factored cost of a tree T by $D(T)$. *Equal Factored Cost CTP-Tree* (*EFC-CTP-Tree*) is a CTP-Tree T in which T is balanced, and $D(T^{Par}(v)) = D(T^{Par}(v'))$ for every vertices v, v' in the same depth. We prove that there is an optimal policy in EFC-CTP-Tree that is committing for every subtree; therefore every instance is an instance of 0-Exp-CTP-Tree to which a polynomial time appears in Section 5.1 [4]. We first show that a special case of EFC-CTP-Tree, called Identical-CTP-Tree, admits a polynomial time solution. Next, using similar methods, we show that EFC-CTP-Tree admits a polynomial time solution as well. Although the latter results subsume the tractability of Identical-CTP-tree, proving the former simplifies the proof.

[4]In fact, in this specific case, every committing policy turns out to be an optimal policy.

5.3.1 Polynomial time solution for Identical-CTP-Tree

Identical-CTP-Tree is a CTP-Tree T in which $T^{Par}(v)$ and $T^{Par}(v')$ are identical for every two vertices v, v' of the same depth [5]. Note that every Identical-CTP-Tree is EFC-CTP-Tree. By Lemma E.0.5 (see Appendix E), we have that $D(T^{Par}(v)) = D(T^{Par}(v'))$ even if $(Parent(v), v)$ is known to be unblocked (and $(Parent(v'), v')$ remains unknown) . In this section we assume that the only zero cost edges are the terminal free edges in T.

Claim 5.3.1 *Let T be an Identical-CTP-Tree. Then there is an optimal policy which is $T^{Par}(v)$-committing for every vertex v.*

Proof: By induction on the height of the vertices in T. If v is a leaf (of height 0) in T, then it is optimal to traverse the terminal free edge (v, t) for zero cost. Assume that there is an optimal policy which is $T^{Par}(v)$-committing for every vertex v of height $h - 1$.

Let π be an optimal policy for T. Let $z \in \mathcal{T}_\pi$ be a node where $L(z) = b$, in which for a vertex v of height h, z is a first departure point of $T(v)$ through v. We show that $T(v)$ must be blocked in b. Note that by Claim 2.3.1, π_b is optimal for the sub-instance of T with an initial belief state b.

Assume in contradiction that $T(v)$ is not blocked. Then π_b can be informally described as follows. Traverse a subgraph T' of T distinct from $T(v)$; if v is retraced, perform $TRY(T^{Par}(v'))$ for a child v' of v; if $T^{Par}(v')$ is blocked, retrace to v and continue with an optimal policy; see Figure 5.7.

Note that this traversal of T' in π_b can be described as a macro-action, which we denote by $Travel(T')$. The result of $Travel(T')$ is either reaching t or retracing to v with a probability $P^{T'}$. Denote by $B^{fail}(T')$ the set of belief states, with location v, reached by performing $Travel(T')$ in b. Note that T' need not be blocked in a belief state of $B^{fail}(T')$. Then, as $T^{Par}(u)$ and $T^{Par}(u')$ are identical for every u, u' children of v, we can make the following assumption.

Assumption 5.3.2 *There is a child u of v such that $T^{Par}(u)$ is not known to be blocked, and such that $\pi_b(b') = TRY(T^{Par}(u))$ for every $b' \in B^{fail}(T')$.*

[5] The default edge (s, t) is not considered a part of T.

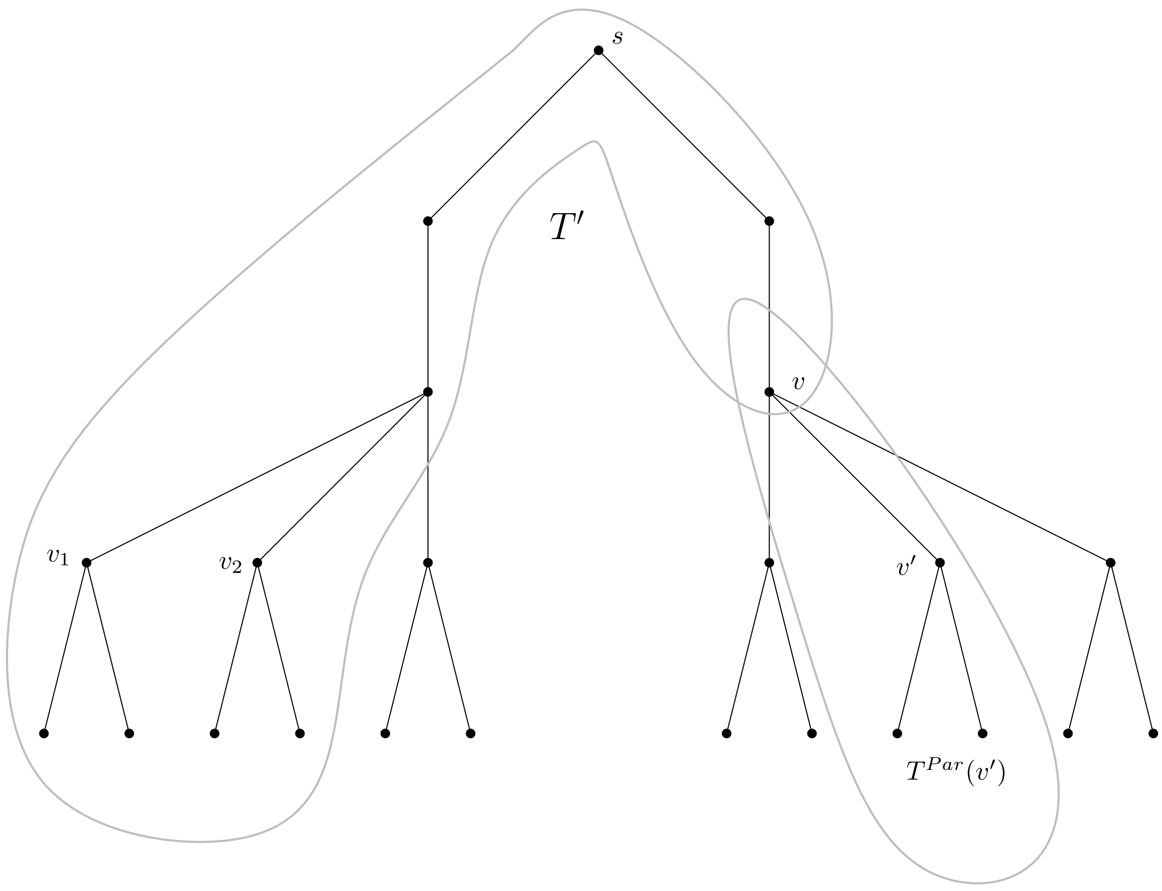


Figure 5.7: Identical-CTP-Tree. The vertex v is of height h. The vertices v', v_1, v_2 are of height $h-1$. The gray lines mark the subgraphs T' of T, for which $Travel(T')$ can be performed from v, and $T^{Par}(v')$ for which $TRY(T^{Par}(v'))$ can be performed from v as well.

Denote the expected cost of $Travel(T')$ by $C^{T'}$. Let $D^{T'} = \frac{C^{T'}}{1-P^{T'}}$. As T' and $T^{Par}(v)$ are distinct, we can define, as in Section 4.2.2, a policy π'_b for T_b, obtained from π_b, in which $Travel(T')$ and $TRY(T^{Par}(v'))$ are switched. Then by the same argument as in Lemma 4.2.3, we have that $C(\pi'_b) < C(\pi_b)$ iff $D^{Par}(v') < D^{T'}$ (see Remark 4.2.4 for the case when $P^{T'} = 1$). We then show that $D^{Par}(v') < D^{T'}$ (see Lemma E.0.4 in Appendix E for details); therefore $C(\pi'_b) < C(\pi_b)$ contradicting π_b being optimal.

□

With Claim 5.3.1 being proved, the following theorem is immediate:

Theorem 5.3.3 *Identical-CTP-Tree admit a polynomial time solution.*

Proof: Let T be an Identical-CTP-Tree. Then by Claim 5.3.1, there is a vertex-committing policy (that is, a policy which is committing for every vertex) π that is an optimal policy for T. Therefore we can consider T as a 0-Exp-CTP-Tree, and run **NoExpTreeSolver** to find an optimal vertex-committing policy π^* on T. As both π and π^* are vertex-committing, then $C(\pi^*) \leq C(\pi)$. However, as π is optimal, then $C(\pi) \leq C(\pi^*)$. Therefore **NoExpTreeSolver** returns an optimal policy for T, and a first move in a policy that achieves this optimum.

□

In fact, as **NoExpTreeSolver** sorts the children of every vertex by the factored cost, which in this case is equal, we get that *every* committing policy for Identical-CTP-Trees, and for EFC-CTP-Tree, is optimal.

5.3.2 Polynomial time solution for EFC-CTP-Trees

Proving that Identical-CTP-Tree admit a polynomial time solution, we next move to the more generalized EFC-CTP-Tree.

Theorem 5.3.4 *EFC-CTP-Tree admit a polynomial time solution.*

As in Section 5.3.1, the following theorem is immediate from the following claim, which is a generalization of Claim 5.3.1.

Claim 5.3.5 *Let T be an EFC-CTP-Tree. Then there is an optimal policy that is $T^{Par}(v)$-committing for every vertex v.*

Proof: The proof is very similar to the proof of Claim 5.3.1. The only difference is that in EFC-CTP-Trees, Assumption 5.3.2 is not obvious at all, since if u_1, u_2 are children of v, then $T^{Par}(u_1)$ and $T^{Par}(u_2)$ are not necessarily identical, and therefore the choice of which next child to "try" can be dependent on the status of edges that are revealed in $Travel(T')$.

Therefore we do as follows. Recall that $L(z) = b$, where $z \in \mathcal{T}_\pi$ is a first departure point for $T(v)$ and v is a vertex of height h. Assume in contradiction that $T(v)$ is not blocked. Then the possible actions in b are divided into two types: The first type, called *try-child* actions, is $TRY(T^{Par}(u))$ for a child u of v. The second type, called *try-graph* actions, is to traverse a subgraph of T, distinct from $T(v)$, in which the outcomes are either reaching t or retracing to v. We denote these actions by $Travel(T')$, and denote the set of belief states reached after $Travel(T')$ is performed in a belief state b, in which v is reached (with probability $P^{T'}$), by $B^{fail}(T')$.

Therefore we can describe π_b as a stochastic (finite) sequence of subgraphs $\{G(d_i)|i < k\}$, for some $k > 0$ (the length of the sequence is stochastic as well). We have $d_0 = b$ and $d_i \in B^{fail}(G(d_{i-1}))$. For every $i \geq 0$, $\pi(d_i)$ is either a try-child macro-action, and then $G(d_i)$ is a subgraph of T, distinct from $T(v)$, or $\pi(d_i)$ is a try-child macro-action, and then $G(d_i) = T^{Par}(u)$ for a child u of v.

As T is assumed to have a default edge (s, t), then in every stochastic sequence of graphs the last subgraph is a subgraph of T, distinct from $T(v)$. As in any traversal of a subgraph, at least one unknown edge is being observed, there is a belief state d reachable from $B(b, \pi)$, such that the stochastic sequence of graphs $\{G(d_i)|1 < i < k\}$ starting from d (then $d_1 = d$) looks as follows.

- $G(d_1)$ is a subgraph of T, distinct from $T(v)$.
- For every $d_2 \in B^{fail}(G(d_1))$, the policy π_{d_2} is a permutation of try-child macro-actions. Once $T(v)$ is found blocked, a (final) try-graph macro-action is performed.

From Claim 2.3.1, we have that π_{d_1} is optimal for $\mathcal{T}_{d_1}$, and π_{d_2} is optimal for $\mathcal{T}_{d_2}$. Therefore π_{d_2} is a sequence of macro-actions of distinct graphs, and we can follow the partition framework (see Section 4.2.2). Hence by Corollary 4.2.6 and as the $D(T^{Par}(u_i))$ are the same for the children u_i of v, we can assume that there is a designated child u of v such that in every $d_2 \in B^{fail}(G(d_1))$, $\pi(d_2) = TRY(T^{Par}(u))$. Therefore Assumption 5.3.2 is verified.

We proceed as in Claim 5.3.1, and construct a policy π'_{d_1} from π_{d_1}, in which $TRY(T^{Par}(u))$ is switched with $Travel(T')$, and continue exactly as in the proof of Claim 5.3.1 to show that $C(\pi'_{d_1}) < C(\pi_{d_1})$ contradicting the optimally of π_{d_1}.

□

5.3.3 Factored-cost conjecture

We explore a relation between the factored-cost of a sub-tree, the cost of the higher layers, and the approximation ratio of a CTP instance. Figure 5.8 depicts two layers of a CTP-Tree T.

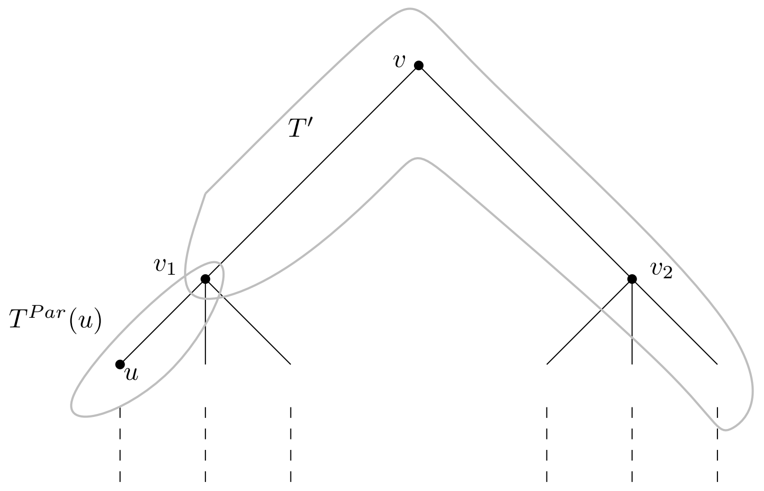


Figure 5.8: Two layers of a CTP-Tree. The grey lines indicate the macro-actions $TRY(T^{Par}(u))$ and $Travel(T')$.

Assume that the agent is at v_1, and the agent can either try a subtree $T^{Par}(u)$ for a child u of v_1 or cross (v, v_1) for a cost $w((v, v_1))$, and perform a $Travel(T')$ macro action, in which T', a subtree of T disjoint from $T(v_1)$, is traversed for an expected cost C'. The vertex v_1 is retraced with some certain probability $P' < 1$. Following the analysis in Sections 5.3.1 and 5.3.2, it is optimal for the agent to traverse a subtree $T^{Par}(u)$ if and only if

$$D(T^{Par}(u)) < \frac{w((v, v_1)) + C + P'w((v, v_1))}{1 - P'} < w((v, v_1))$$

Based on this analysis, we can make the following conjecture, to which we perform experiments in Section 5.4.

Conjecture 5.3.6 *Let N_v be the children of vertex v in a CTP-Tree T. Then the optimal cost for T is the optimal committing cost if for every vertex $v \neq s$ we have*

$$\max_{u \in N_v}\{D(T^{par}(u))\} < w((Parent(v), v))$$

.

5.4 Experimental Results

The results in Section 5.3 leave a gap that can be examined empirically w.r.t the relation between optimal committing policies and optimal policies. The objective in our experiments is twofold. In the first experiment, called **Likelihood of commitment**, we compare optimal committing policies w.r.t. optimal policies. We show an example where the cost of an optimal committing policy is exponentially worse than the cost of an optimal policy. However, we believe that for a uniformly sampled CTP tree, an optimal committing policy is indeed optimal among all policies.

In our second experiment, called **Comparing factored-cost**, we compare the factored-cost of the trees w.r.t the optimal committing policies. We believe that the gap between the factored-cost of the subtrees is proportional to the inability of an optimal committing policy to approximate an optimal policy.

Recall that the cost of an optimal policy is called the optimal cost. The cost of an optimal committing policy is called the *optimal committing cost.* We first show the following example in which the optimal committing cost is exponentially greater (in the size of the problem) than the optimal cost.

Example 5.4.1 *Figure 5.9 is constructed with low cost unblocked edges (s, v_i) to allow the agent to "'visit"' subtrees. In addition, there are zero cost edges*

(v_i, u_i) with a blocking probability of $1/2$ to "'lure"' the agent into the subtrees, and very expensive always unblocked edges to encourage the agent to test other subtrees. In a committing policy, once w.l.o.g. (s, v_1) is traversed, the agent is bound to cross the expensive edge (v_1, w_1) after revealing the cheap adjacent edge (v_1, u_1) to be blocked. Therefore the optimal committing cost is $W/2+1$. However, in an optimal policy, the low cost edges (s, v_i) enable the agent to try another subtrees $T^{Par}(v_i)$, and cross an expensive edge only after all the cheap edges (v_i, u_i) in the graph are found to be blocked. Therefore the optimal cost is no more than $4 + (1/2)^{n-1}W(n-1)$.

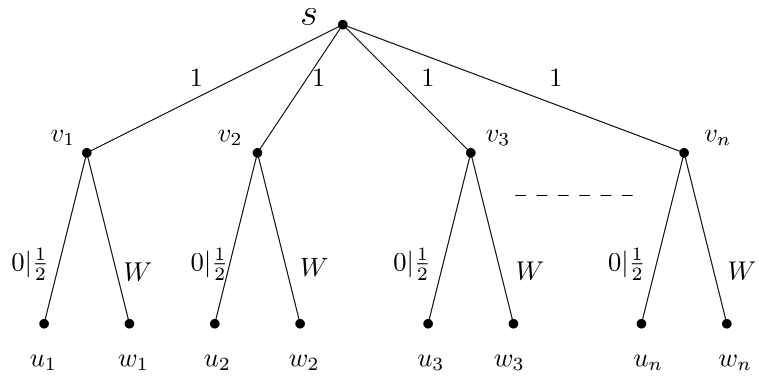


Figure 5.9: CTP-Tree. The optimal committing policy has a cost of $W/2+1$, while the optimal policy has a cost of no more than $4 + (1/2)^{n-1}W(n-1)$.

.

We conduct our experiments on a balanced tree as shown in Figure 5.10. The edges (s, r_i) are called the *first layer*, the edges (r_i, v_i) are called the *second layer*. The edges (v_i, u_i) are called the *cheap-edges*, the edges (v_i, w_i) are called the *expensive-edges*. Unless mentioned otherwise, all edge costs and blocking probabilities are uniformly distributed.

Next, we constructed specific CTP models based on the CTP graph layout from Figure 5.10, see Table 5.2. In model C all the edges are uniformly sampled from the same cost and blocking probability intervals. Models D,E,F,$E1$,$E2$,$F1$, and $F2$ were constructed by having first and second layers with low cost and small blocking probability. Models $I1$, $I2$, and $I3$ follow Example 5.4.1. Specifically, these models have cheap first and second layers, and in addition a low cost, cheap edge to "'lure"' the agent into the

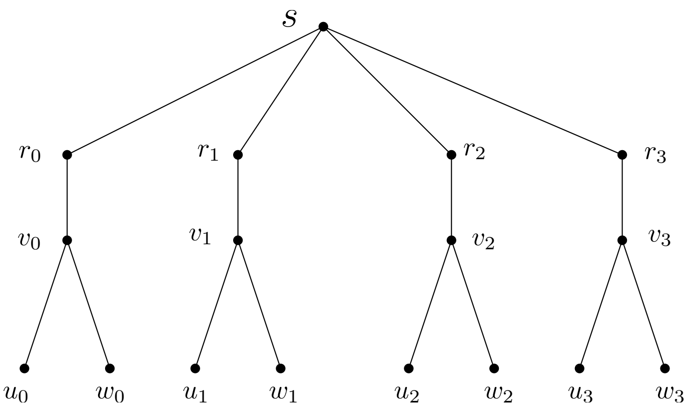


Figure 5.10: CTP-Tree. The edges (s, r_i) are called the first layer, the edges (r_i, v_i) are called the second layer. The third layer is composed of the so called "'cheap-edges"' (v_i, u_i), and the so called "'expensive-edges"' (v_i, w_i).

subtree, and an additional high cost, expensive edge, to "encourage"' the agent to visit other subtrees. We have taken 1000 samples from each CTP Model.

Table 5.2: The CTP models for the CTP graph layout from Figure 5.10. Cost range appears as (min cost, max cost). Blocking probability range appears as (min blocking probability,max blocking probability). First layer edges are always unblocked.

CTP model	1st layer	2nd layer		cheap-edge		expensive-edge	
	cost	cost	prob.	cost	prob.	cost	prob.
C	$(0, 500)$	$(0, 500)$	$(0.1, 0.9)$	$(0, 500)$	$(0.1, 0.9)$	$(0, 500)$	$(0.1, 0.9)$
D	$(0, 5)$	$(0, 5)$	$(0.1, 0.9)$	$(0, 5)$	$(0.6, 0.95)$	$(500, 1000)$	$(0.1, 0.9)$
E	$(0, 10)$	$(0, 500)$	$(0.1, 0.9)$	$(0, 500)$	$(0.1, 0.9)$	$(0, 500)$	$(0.1, 0.9)$
F	$(0, 5)$	$(0, 5)$	$(0.1, 0.9)$	$(0, 500)$	$(0.1, 0.9)$	$(0, 500)$	$(0.1, 0.9)$
$E1$	$(5, 35)$	$(20, 60)$	$(0.65, 0.95)$	$(50, 150)$	$(0.4, 0.8)$	$(50, 150)$	$(0.4, 0.8)$
$E2$	$(5, 35)$	40	0.8	100	0.6	100	0.6
$F1$	$(0, 10)$	$(0, 10)$	$(0.65, 0.95)$	$(50, 150)$	$(0.4, 0.8)$	$(50, 150)$	$(0.4, 0.8)$
$F2$	$(0, 10)$	5	0.8	100	0.6	100	0.6
$I1$	$(0, 10)$	$(0, 10)$	$(0, 0.2)$	$(0, 500)$	$(0.2, 0.8)$	$(500, 1000)$	$(0.4, 0.8)$
$I2$	$(0, 10)$	$(0, 10)$	$(0, 0.2)$	$(0, 500)$	$(0.2, 0.8)$	$(1000, 2000)$	$(0.4, 0.8)$
$I3$	$(0, 10)$	$(0, 10)$	$(0, 0.2)$	$(0, 1500)$	$(0.2, 0.8)$	$(100, 2000)$	$(0.4, 0.8)$

Likelihood of commitment We explore the likelihood that an optimal committing policy is indeed optimal. Our results appear in Table 5.3. The *committing ratio* is the ratio of samples in which the optimal committing cost is the optimal cost. The *approximation ratio* is the ratio between the optimal committing cost and the optimal cost. Our results show that when all the edge weights and blocking probabilities are uniformly distributed over $(0, 500)$ and $(0.1, 0.9)$, respectively(model C), then the committing ratio is 0.999. The approximation ratio among the non-committing samples is 1.014. In fact our results show a maximum approximation ratio of 1.015, even in models with a low committing ratio (Models D,E,F, and $F1$). Furthermore, our experiments show that although the committing ratio was extremely low (0 on models I_1, I_2, and 0.02 on I_3), the maximal approximation ratio was 1.299 (model $I3$).

Another part of our experiments was to separate Identical-CTP-Trees from the more general EFC-CTP-Trees. However, our experiments showed no significant difference in the committing ratio and the approximation ratio.

To test Conjecture 5.3.6, we have sample model D in a search for a counter-example. Such counter-example would be a sample in which the optimal policy is non-committing, yet there is a vertex u, such that the factored-cost of $T^{par}(u)$ is bigger than $w(Parent(u), u)$. After running over 5000 samples, no such counter-example was found.

Table 5.3: Results for the CTP-model samples from Table 5.2.

CTP model	committing ratio	approximation ratio
C	0.999	1.014
D	0	1.015
E	0.984	1.001
F	0.109	1.004
$E1$	1	1
$E2$	1	1
$F1$	0.989	1.00002
$F2$	1	1
$I1$	0	1.134
$I2$	0	1.299
$I3$	0.02	1.2904

Comparing factored-cost We tested the factored-cost of the trees w.r.t. the approximation ratio. To do that we define the *factored-cost gap* as follows: Denote the vertices in depth i by $V(i)$, and denote the factored-cost of every $v \in V(i)$ by $D(v, i)$. Let

$$gap(i) = \max_{v \in V(i)} \{D(v, i)\} - \min_{v \in V(i)} \{D(v, i)\}$$

We define the factored-cost gap of a CTP-Tree instance to be $\max_i\{gap(i)\}$. Note that a CTP-Tree with a factored-cost gap of 0 is an EFC-CTP-Tree discussed in Section 5.3. Table 5.4 shows the approximation ratio and the average factored-cost gap for every model with a low committing ratio. The scatter of the samples for every model appears in Figure **??** and **??**. Although we did not find a significant relation between the factored-cost gap and the committing ratio, it can be clearly seen that the more the factored-cost gap grows, the more samples have a higher approximation ratio.

Table 5.4: Results for comparing approximation ratio with factored-cost gap.

CTP model	committing ratio	approximation ratio	average factored-cost gap
D	0	1.015	895.247
F	0.109	1.004	387.066
$I1$	0	1.134	801.636
$I2$	0	1.299	1693.1506
$I3$	0.02	1.2904	1504.811

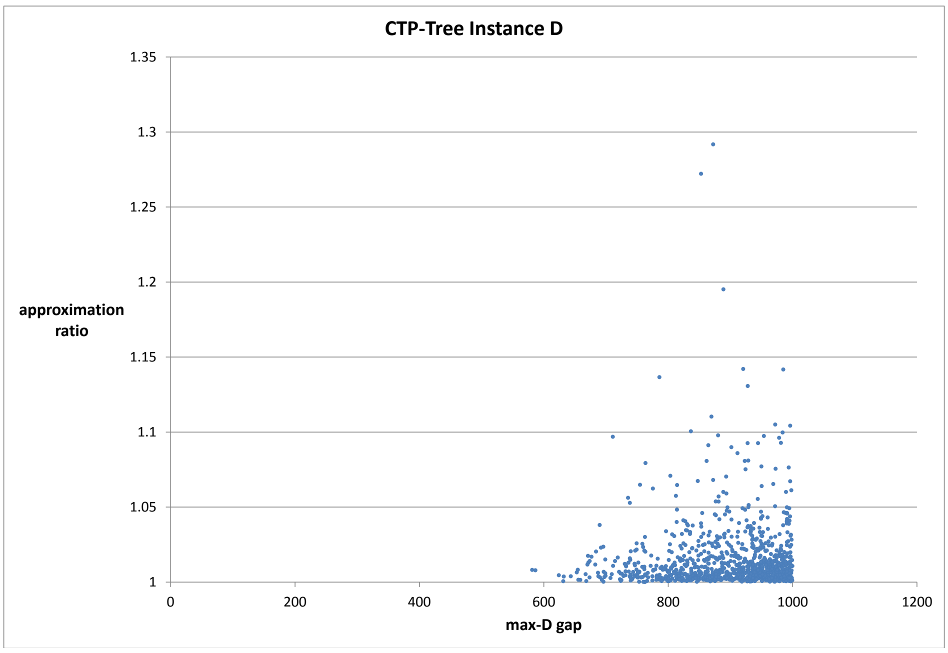


Figure 5.11: Scatter graph for CTP-Models D.

.

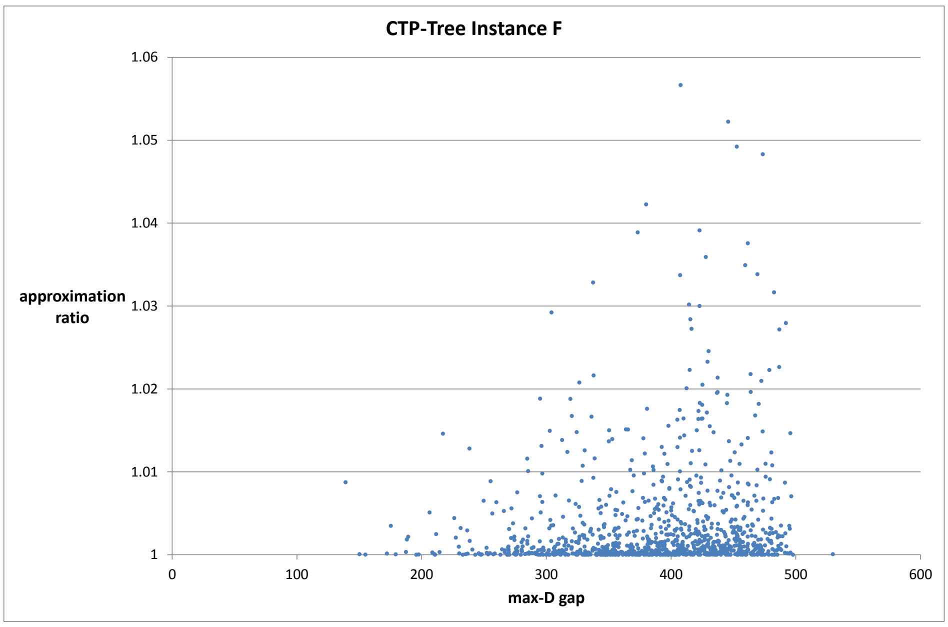


Figure 5.12: Scatter graph for CTP-Models F.

.

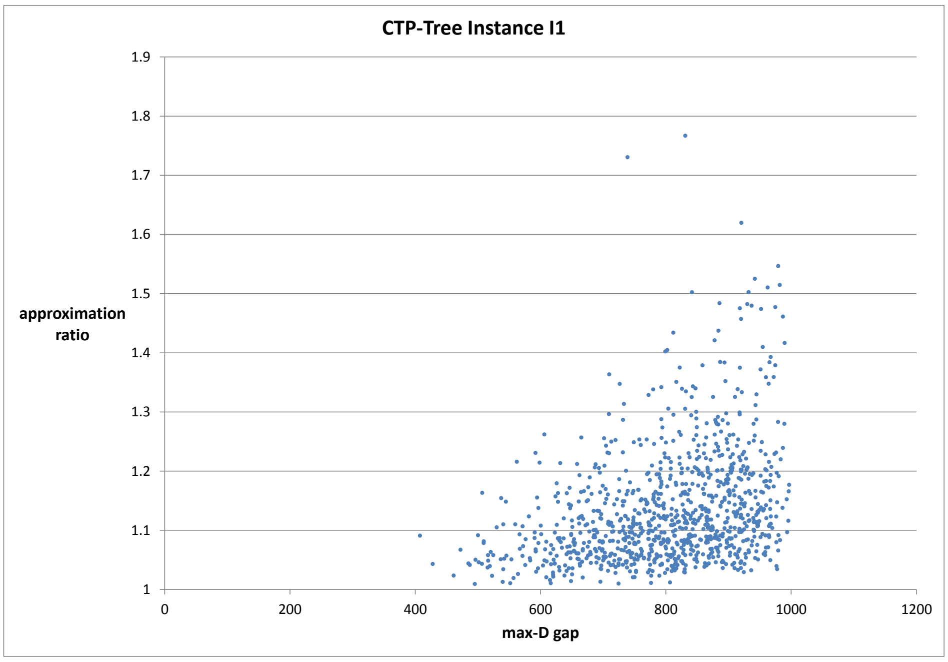


Figure 5.13: Scatter graph for CTP-Models $I1$.

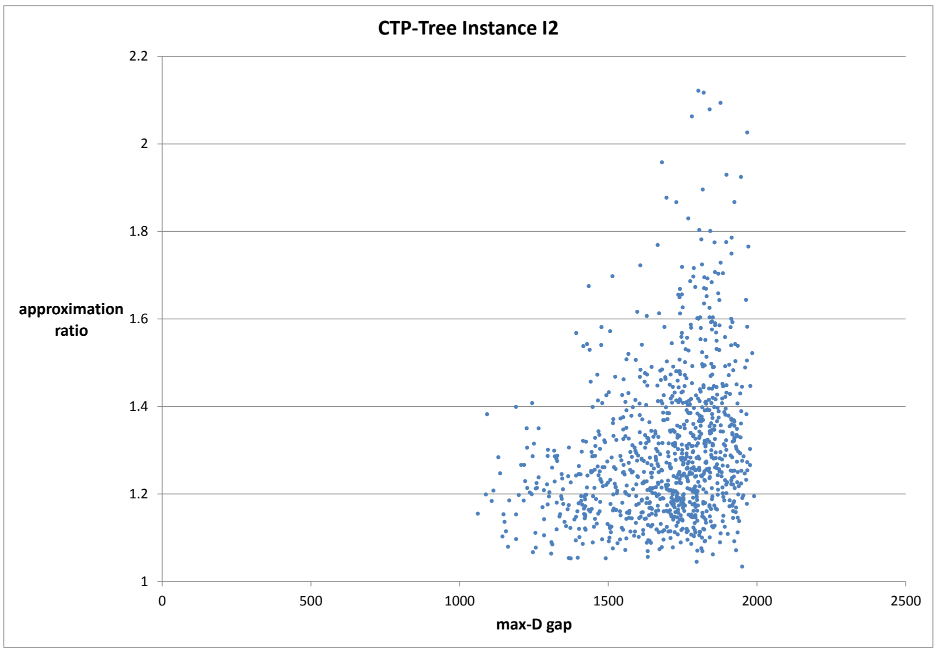


Figure 5.14: Scatter graph for CTP-Models $I2$.

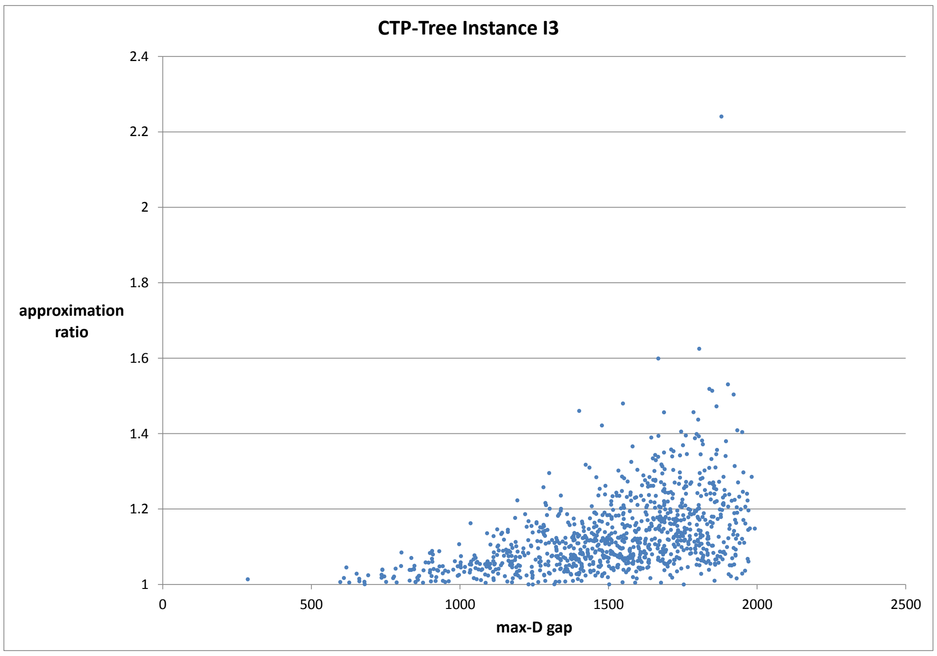


Figure 5.15: Scatter graph for CTP-Models $I3$.

Chapter 6

Repeated-CTP

6.1 Repeated CTP in disjoint-path graphs

In this chapter, we generalize CTP to a multi-agent variant where n agents operate in the given graph. Note that there are many possible communication and knowledge-sharing paradigms as well as different agent types for multi-agent systems. Here we assume that the agents are fully cooperative and aim to minimize their total travel cost. In addition, we assume a communication paradigm of *full knowledge sharing*. That is, any new information discovered by an agent (e.g., whether an edge is blocked or traversable) is immediately made known (broadcast) to all other agents. This assumption is equivalent to having a centralized control of all agents. Specifically, we introduce the *Repeated task* multi-agent CTP, called Repeated-CTP, and denoted by CTP-REP(n) for short, in which n agents need to travel from the start state to the goal state. However, there is only one *active agent* at each point in time. All other agents are inactive until the currently active agent reaches t. An agent that reaches t becomes inactive again (is "out of the game"), and can make no additional actions or observations. The goal is a natural extension of single-agent CTP: all n agents begin at s and must reach t. We need to find a policy for the agents that minimizes the expected total travel cost of reaching this goal. The content of this chapter was published in [7]. A journal version was recently submitted as well.

6.2 Repeated CTP in disjoint-path graphs

We extend the results on single-agent disjoint graphs CTP from Section 2.5, to the case of repeated CTP with n agents, CTP-DISJ-REP(n). We show that in CTP-DISJ-REP(n) there exists an optimal policy that is committing for the first (leading) agent, and such that the rest of the agents (the following agents) follow the last path successfully traversed by the leading agent. This optimal policy, like the single-agent case, can be computed efficiently by sorting the paths according to a simple measure, which needs to be adjusted in order to account for the traversal costs of the $n-1$ following agents. Though apparently simple, proving optimality of such a policy is non-trivial. The notations in this section are based on the notations of disjoint-path graphs, see Section 2.5.

Let M be an instance of CTP-DISJ-REP(n) with k paths. Note that any *reasonable* policy in M can be represented using only TRY and INV macro actions as follows.

Let $TRY(l,i)$ be the action in which agent A_l tries path I_i. Let $INV(l,i,j)$ be the action in which agent A_l performs $INV(i,j)$. When the agent is obvious from the context, we shorten the notation to $TRY(i)$ and $INV(i,j)$ respectively.

Therefore, given a policy π for M, we consider the policy tree T_π as consists on the macro actions TRY and INV, and therefore contains only two different types of action-arcs: TRY-arc, for the macro action TRY, and INV-arcs for the macro action INV.

A policy that contains only TRY actions for an agent A_i is *committing for* A_i. Likewise, a policy is *committing* (for a set of agents) if it consists of only TRY actions for all these agents. Note that for a single agent, a committing policy is also committing in the sense of Section 4.2. It is non-trivial to show that in repeated CTP, TRY actions suffice for optimality – this requires definition of the constrained followers-committing policies, discussed next.

Let π be a committing policy for M, where whenever A_0 reaches t through path I_i, the agents $A_1, \cdots A_{n-1}$ traverse I_i as well. A policy π with this property is called a *followers-committing* policy, and the agents $A_1, \cdots A_{n-1}$

are said to *follow* A_0 in π. Note that this property allows us to define a multi-agent macro-action for a path I_i, which we denote by $TRY_n(i)$ and acts as follows. A_0 *tries* I_i. *If* I_i *is found unblocked,* A_0 *reaches* t *and* $A_1, \cdots A_{n-1}$ *traverse* I_i *as well; otherwise, if* I_i *is found blocked,* A_0 *retraces to* s *(other agents staying idle).* The results of $TRY_n(i)$ are that either a terminal belief state is reached by having all the agents in t (after traversing I_i), or a belief state is reached in which all the agents $A_0, \cdots A_{n-1}$ are in s and I_i is known to be blocked.

Recall that $Q_i = 1 - P_i$ is the probability that path I_i is unblocked. Denoting the expected cost of $TRY_n(i)$ by $E[TRY_n(i)]$, we have:

$$E[TRY_n(i)] = nQ_iW_i + E[BC(i)] \tag{6.1}$$

Let π^*_M be the followers-committing policy where A_0 executes the committing policy of trying the paths by increasing order of $\frac{E[TRY_n(i)]}{Q_i}$, and $A_1, \cdots, A_{n-1}$ follow A_0 [1].

Theorem 6.2.1 *π^*_M is an optimal policy for M.*

As the proof is non-trivial, we first present a proof outline, followed by an example. The complete proof is in Section 6.4.

Proof outline: We first show that π^*_M is optimal among all followers-committing policies for M. Then we continue by induction on the number of agents, n. For $n = 1$, M is also an instance of CTP-DISJ. Hence, by Theorem 2.5.3, π^*_M is optimal (in fact, with some adjustments, the proof of Theorem 6.2.1 serves as an alternative proof of Theorem 2.5.3 as well; see Remark 6.4.5 in Section 6.4).

We now assume inductively that for every instance M' of CTP-DISJ-REP$(n-1)$, the followers-committing policy $\pi^*_{M'}$ is optimal, and show that π^*_M is optimal for M.

Recall that an INV-arc is an action arc in T_π is which the action is INV. Note that $T_{\pi^*_M}$ contains no INV-arcs.

[1]For $n = 1$, $\frac{E[TRY_n(i)]}{Q_i}$ is the factored cost for (I_i, s, t)

Let π be an optimal policy for M, with a minimal number of INV-arcs in T_π. If π is followers-committing we are done. Hence we may assume that π is not a followers-committing policy. Then there are two cases:

(1:) π **is committing.** Then T_π does not contain INV-arcs. Assume w.l.o.g. that A_0 tries the paths in π in the order of $\{I_0, I_1, \cdots I_{k-1}\}$. By the induction assumption we may assume that A_1 executes a followers-committing policy, hence $A_2, \cdots A_{n-1}$ follow A_1 in π. As π is not a followers-committing policy, we may assume that A_1 does not follow A_0. Then we can show that there is a path I_m such that

$$\frac{E[TRY_n(m+1)]}{Q_{m+1}} < \frac{E[TRY_n(m)]}{Q_m}$$

We can then define a policy π' that is the same as π, except that I_{m+1} is tried right before I_m, such that $C(\pi') < C(\pi)$, contradicting the optimality of π.

(2:) π **is not committing**. We can then show that T_π contains a subtree, T, with only one INV-arc, and define a policy π', which is obtained from π by replacing T with another tree, T', which has no INV-arcs at all. We then show that $C(\pi') \leq C(\pi)$, contradicting the minimal number of INV-arcs in T_π among the optimal policies of M. $\square$

Example: Consider Figure 6.1. We have $\frac{E[TRY_1(0)]}{Q_0} = 39.5$, and $\frac{E[TRY_1(1)]}{Q_1} = 2.6$. Hence by Theorem 6.2.1, the optimal single agent policy is committing to try path I_1 before I_0. However, $\frac{E[TRY_{38}(0)]}{Q_0} = 95$ and $\frac{E[TRY_{38}(1)]}{Q_1} = 95.1$, hence for $n \geq 38$ agents, the optimal policy is for the first agent to try path I_0 before I_1, and for the other agents to follow the first agent's path to t.

6.3 Interleaved-action CTP in disjoint-path graphs

We briefly consider interleaved action CTP in disjoint-path graphs (CTP-DISJ-MULTI(n)), in which agents can start moving before the first active

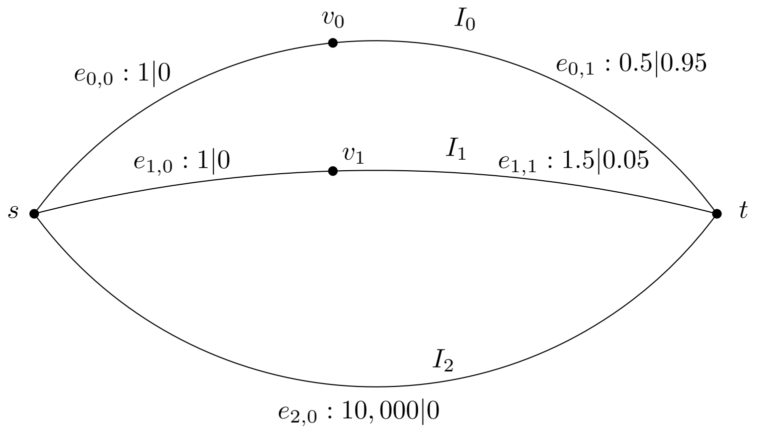


Figure 6.1: An simple example for CTP-DISJ-REP(n). For $n < 38$ the optimal first macro-action for the first agent is $TRY(I_1)$. For $n \geq 38$, the optimal first macro-action for the first agent is $TRY(I_0)$.

agent has reached t. In this variant, it is by no means clear that the optimal policy can be described using only TRY and INV macro actions. In fact, it is easy to see that for more general graphs, the optimal policy requires interleaved actions. For example, see Figure 6.2, which is constructed from Figure 2.6 by adding a (certainly traversable) path that costs 100 from v_1 to t. The optimal 2-agent policy is to send the first agent to v_1, and if $e_{1,1}$ is blocked, send the second agent to v_0 to check $e_{0,1}$, while the first agent waits at v_1.

Since for disjoint paths this type of scenario cannot occur, we are led to suspect that there is no advantage to having more than one active agent at a time in this topology. This instance was empirically checked in [7] by generating the optimal policies (using value iteration) with and without interleaved actions for small randomly generated problem instances. From hundreds of such non-disjoint-path instances, more than 10% of the cases required interleaved actions to achieve the optimal policy. Conversely, in all of over a thousand such disjoint-path graph instances, the optimal policy for CTP-DISJ-REP(n) was also optimal for CTP–DISJ-MULTI(n). Hence we state the following:

Conjecture: Every optimal policy for CTP-DISJ-REP(n) is also optimal

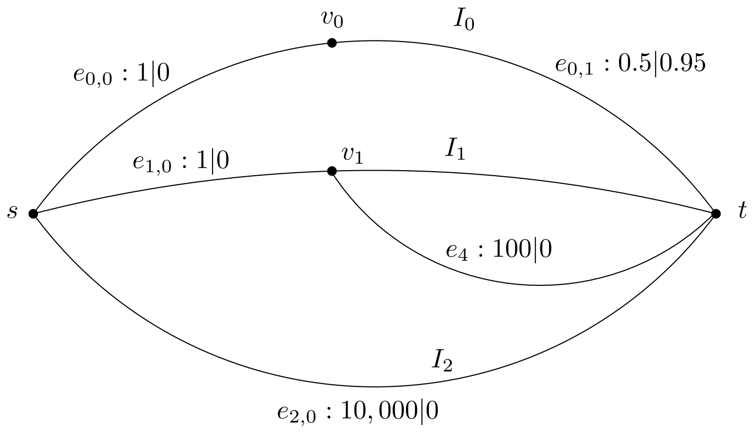


Figure 6.2: A CTP-DISJ-MULTI(n) example in which the optimal policy requires interleaved actions. Edge label $w|p$ denotes edges cost w, blocking probability p.

for CTP-DISJ-MULTI(n).

6.4 Complete proof for Theorem 6.2.1

We begin with some definitions and notations. If an agent traverses a path I_j and reaches t, we say the agent has *successfully traversed* I_j. As every belief state is defined by the probability function p over the edges (with $b(e) = p(e)$ for every $e \in E$), we can denote the expected cost of the macro action $TRY_n(i)$ in a belief state b, by $E_b[TRY_n(i)]$. Furthermore, as the status of the edges of a path I_j is belief state dependent, we denote the probability that I_j is unblocked in belief state b by $Q_{j(b)}$ (thus $Q_{j(b_0)} = Q_j$), and define

$$D_i^n(b) = \frac{E_b[TRY_n(i)]}{Q_{i(b)}} \tag{6.2}$$

For conciseness we denote $D_i^l(b_0)$ by D_i^l and in particular denote D_i^1 by D_i. Note that

$$D_i^n = \frac{E[BC(i)]}{Q_i} + nW_i \tag{6.3}$$

hence $D_i \geq W_i$, and for every $l \leq n$ we have:

$$D_i^n = D_i^l + (n-l)W_i \tag{6.4}$$

Finally, recall that an action-arc in T_π that represents an INV action is called an *INV-arc*. Note that $T_{\pi^*_M}$ contains no INV-arcs.

In order to show that π^*_M is optimal, we first show that π^*_M is optimal among all followers-committing policies.

Lemma 6.4.1 *Let π be a followers-committing policy for M. Then $C(\pi^*_M) \leq C(\pi)$.*

Proof: Every followers-committing policy π for an instance M of CTP-DISJ-REP(n) can be re-cast as an equivalent CTP-DISJ problem instance M'. This is done as follows. M' extends M by adding, at the end of each path I_i, an additional traversable edge e_{i,r_i} incident on t and bearing a cost of $(n-1)W_i$. In a followers-committing policy for M, all agents follow the first agent, and all incur a cost of W_i. Thus there is a bijection F from followers-committing policies in M, to committing policies in M', that preserves expected costs, therefore $C(\pi) = C(F(\pi))$. Now suppose that π is a followers-committing policy for M. By Theorem 2.5.3 (but see Remark 6.4.5), $F(\pi^*_M)$ is optimal for M', therefore $C(F(\pi^*_M)) \leq C(F(\pi))$, which entails $C(\pi^*_M) \leq C(\pi)$. □

Next, we prove that π^*_M is optimal among *all* policies for M. The proof goes by induction on n, the number of agents in M. For $n = 1$, M is an instance of CTP-DISJ-REP(1) that is also an instance of CTP-DISJ. Hence, by Theorem 2.5.3, π^*_M is optimal (in fact, with some adjustments, the proof of Theorem 6.2.1 serves as an alternative proof of Theorem 2.5.3 as well; see Remark 6.4.5 at the end of this proof). We now assume inductively that $\pi^*_{M'}$ is an optimal policy for every instance M' of CTP-DISJ-REP($n-1$), and show that π^*_M is optimal for M.

Let π be a policy for M that is *not* followers-committing. Assume in contradiction that π is an optimal policy in which the number of INV-arcs in T_π is minimal. We show that there is a policy π' that leads to a contradiction in the following way: either $C(\pi') < C(\pi)$, contradicting π being optimal, or otherwise $C(\pi') = C(\pi)$ and the number of INV-arcs in $T_{\pi'}$ is smaller than those of T_π, contradicting the minimality of the number of INV-arcs in T_π.

We then have two cases to consider:

(Case 1:) π is committing. Then T_π contains no INV-arcs. Assume w.l.o.g. that A_0 tries the paths in π in the order of $\langle I_0, I_1, \cdots, I_{k-1}\rangle$. By the induction assumption we assume that π is followers committing for A_1, meaning that in π the only actions for A_1 are TRY actions, and $A_2, \cdots, A_{n-1}$ follow A_1. Therefore, as we show below in Remark 6.4.2, we may assume that after A_0 has reached t through a path I_j, there is a unique policy for $A_1 \cdots A_n$, called π_j, which is independent of the edges which were found blocked in the "previous" paths I_l for $l < j$. Then, every committing policy π can be described as follows (see Figure 6.3 for an example). A_0 tries the paths in the order of $\langle I_0, I_1, \cdots, I_{k-1}\rangle$; once A_0 has successfully traversed a path I_j, then in π_j, the agent A_1 tries the paths in $\{I_j, \cdots, I_{k-1}\}$ in a certain permutation (note that I_j is known to be unblocked). These permutations fully describe the policy since $A_2, \cdots, A_{n-1}$ follow A_1. Note that the policies (permutations) π_j are truncated as $TRY(1, j)$ always succeeds in π_j and no additional paths are tried.

Remark 6.4.2 *Formally, let B_j be the collection of all possible belief states that label a node in T_π with an incoming $TRY(0, j)$-arc, and in which A_0 is in t. Note that in these belief states, I_j is known to be unblocked hence $D_j^{n-1}(b) = (n-1)W_j$ for every $b \in B_j$; I_l is known to be blocked for every $l < j$ hence $D_l^{n-1}(b) = \infty$. Finally, as the I_l are not yet traversed for every $l > j$, we have $D_l^{n-1}(b) = D_l^{n-1}$. Therefore we can define π_j to be the partial policy of π starting at every such belief state $b \in B_j$.*

Note that if A_1 always follows A_0 in π, then π is a followers-committing policy and we are done. Therefore we assume there is an optimal policy, in which there is a path successfully traversed by A_0 and is not first tried by A_1. We then intend to show, as facilitated by the following lemma, that there is a path I_m for which $D_{m+1}^n < D_m^n$.

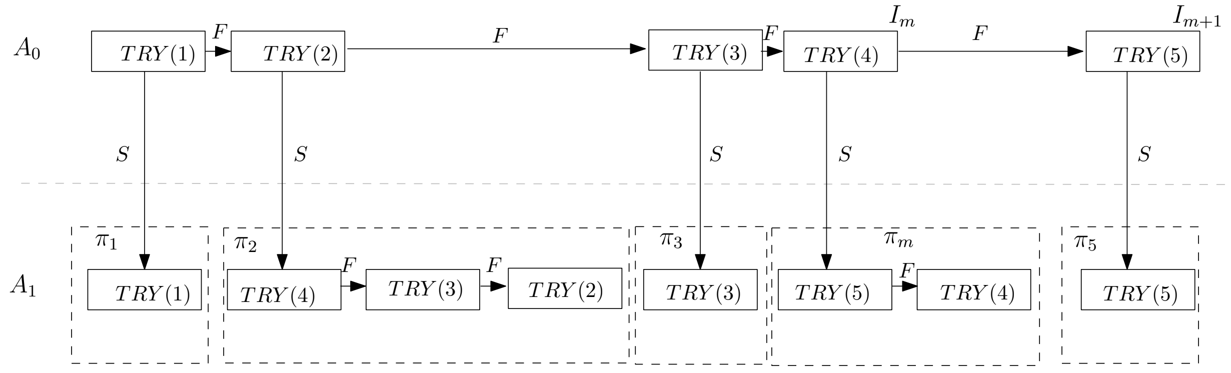


Figure 6.3: An illustration of a committing policy. The boxes indicate the actions taken in every belief state. The directed arrows indicate the outcome of such actions where S indicates that the agent A_0 has reached t through a certain path, while F indicates A_0 has found that path blocked and retraced to s.

Lemma 6.4.3 *If $D_l^n < D_j^n$ then $(n-1)W_l \leq D_j^{n-1}$ for every $n \geq 2$.*

Proof: Assume in contradiction that $(n-1)W_l > D_j^{n-1}$. Note that $D_j^{n-1} = D_j + (n-2)W_j$, and $D_j \geq W_j$, implying $(n-1)W_l > (n-1)W_j$, hence $W_l > W_j$. Then we have $D_l \geq W_l$, implying $D_l > W_j$, and as $(n-1)W_l > D_j^{n-1}$, we have $D_l + (n-1)W_l > D_j^{n-1} + W_j$, therefore $D_l^n > D_j^n$, a contradiction. □

Now assume that $D_i^n < D_{i+1}^n$ for every path I_i. As for every path I_j, every $b_j \in B_j$ and every $l > j$, we have $D_j^{n-1}(b_j) = (n-1)W_j$, and $D_l^{n-1}(b_j) = D_l^{n-1}$, we have by, Lemma 6.4.3, that $\pi(b_j) = TRY(1,j)$ which means A_1 follows A_0. Hence π is a followers-committing policy in contradiction to our assumption. Therefore, as we assumed that all the D_i are different, we have $D_i > D_{i+1}$ for some i.

Let I_m be the last path such that $D_m^n > D_{m+1}^n$ (in the example in Figure 6.3, the path I_m is I_4). Let π' be the policy obtained from π by switching $TRY(0,m)$ and $TRY(0,m+1)$. We show that $C(\pi') < C(\pi)$.

To write down the expected cost $C(\pi)$ and $C(\pi')$, we need the following observation. Note that π can be represented as a sequence of conditional operations for the paths, such that in each operation, A_0 tries a path I_l, and either I_l is traversed successfully and A_1 executes π_l, or I_l is found blocked and A_0 returns to s. Therefore the expected cost of π is

$$C(\pi) = \sum_{l<k} (\prod_{c<l} P_c) Q_l (D_l + C(\pi_l)) \tag{6.5}$$

and explicitly we have

$$\begin{aligned} C(\pi) = \sum_{l<m} (\prod_{c<l} P_c) Q_l (D_l + C(\pi_l)) + \\ (\prod_{c<m} P_c) Q_m (D_m + C(\pi_m)) \\ + (\prod_{c<m} P_c) P_m Q_{m+1} (D_{m+1} + C(\pi_{m+1})) \\ + \sum_{m+1<l<k} (\prod_{c<l} P_c) Q_l (D_l + C(\pi_l)) \end{aligned}$$

and

$$\begin{aligned} C(\pi') = \sum_{l<m} (\prod_{c<l} P_c) Q_l (D_l + C(\pi'_l)) + \\ (\prod_{c<m} P_c) Q_{m+1} (D_{m+1} + C(\pi'_{m+1})) \\ + (\prod_{c<m} P_c) P_{m+1} Q_m (D_m + C(\pi'_m)) \\ + \sum_{m+1<l<k} (\prod_{c<l} P_c) Q_l (D_l + C(\pi'_l)) \end{aligned}$$

Note that $C(\pi_l) = C(\pi'_l)$ for every $l < m$, and every $l > m + 1$. We show that $C(\pi_{m+1}) = C(\pi'_{m+1})$. As $m < m + 1$, and for every $l > j > m$, we have$D^n_l > D^n_j$, it is implied from Lemma 6.4.3 that
$C(\pi_{m+1}) = (n-1)W_{m+1}$. On the other hand, as $D^n_{m+1} < D^n_m$, then, again by Lemma 6.4.3, $(n-1)W_{m+1} \leq D^{n-1}_m$. Hence if A_0 successfully traverses I_{m+1} in π', we may assume that A_1 follows A_0, so $C(\pi'_{m+1}) = (n-1)W_{m+1}$ as well. Then, as $D^n_{m+1} = D_{m+1} + (n-1)W_{m+1}$, we get the following property:

Property 1

$C(\pi') < C(\pi)$ if and only if

$$Q_{m+1}(D^n_{m+1} - D_m) < C(\pi_m) - P_{m+1}C(\pi'_m)$$

Let I_y be the first path that A_1 tries in π_m (for example in Figure 6.3, I_y is I_5). Obviously, the last path that A_1 tries in π_m is the unblocked path I_m. Note that once A_0 successfully traverses I_m in π', A_1 tries the remaining paths in the same order as in π_m, skipping I_{m+1}, which is already known to be blocked. We then have two cases to consider of whether I_{m+1} precedes I_m in π_m. For each such case we show that $C(\pi') < C(\pi)$.

For example, in Figure 6.3, I_m is I_4, and I_{m+1} is I_5. The first action in π_m is $TRY(5)$. Hence A_1 tries I_5 in π_m, before traversing I_4, which was already successfully traversed by A_0 (hence known to be unblocked).

To handle the two cases we need the following technical lemma.

Lemma 6.4.4 $D^n_{m+1} < D_m + D^{n-1}_y$

proof: First observe that $W_y < D_m$. Note that in π_m, I_y is tried before I_m, implying $D^{n-1}_y < (n-1)W_m$. Therefore, as $D^{n-1}_y = D_y + (n-2)W_y$ and as $W_y \leq D_y$, we have $(n-1)W_y < (n-1)W_m$ which entails $W_y < W_m \leq D_m$.

Now, $m + 1 \leq y$ implies $D^n_{m+1} < D^n_y$. Then since $D^n_y = W_y + D^{n-1}_y$ and $W_y < D_m$, Lemma 6.4.4 follows. $\Box$

Case 1.1: I_m precedes I_{m+1} in π_m.

Therefore in π'_m, A_1 tries exactly the same paths and in the same order as in π_m, which implies $C(\pi'_m) = C(\pi_m)$.

In addition, as the first path that A_1 tries in π_m is I_y, we have that $D^{n-1}_y \leq C(\pi_m)$. Then from Lemma 6.4.4 we have $D^n_{m+1} < D_m + C(\pi_m)$. Therefore, as $C(\pi'_m) = C(\pi_m)$ and as $Q_{m+1} = 1 - P_{m+1}$, we have from Property 1 that $C(\pi') < C(\pi)$.

Case 1.2: I_{m+1} precedes I_m in π_m. Denote the order on permutation π_m by $<_m$. Then we have

$$C(\pi_m) = \sum_{l <_m m+1} (\prod_{h <' l} P_h) Q_l D_l^{n-1} \\ + (\prod_{h <_m m+1} P_h) Q_{m+1} D_{m+1}^{n-1} \\ + \sum_{m+1 <_m l <_m m} (\prod_{h <' l} P_h) Q_l D_l^{n-1} + (\prod_{h <_m m} P_h)(n-1) W_m \quad (6.6)$$

As I_{m+1} is already known to be blocked, whenever executing π'_m, we have

$$C(\pi'_m) = \sum_{l <_m m+1} (\prod_{h <_m l} P_h) Q_l D_l^{n-1} \\ + \frac{1}{P_{m+1}} \sum_{m+1 <_m l <^m m} (\prod_{h <_m l} P_h) Q_l D_l^{n-1} + \\ \frac{1}{P_{m+1}} (\prod_{h <_m m} P_h)(n-1) W_m \quad (6.7)$$

Hence

$$C(\pi_m) - P_{m+1} C(\pi'_m) = \\ (\prod_{h <_m m+1} P_h) Q_{m+1} D_{m+1}^{n-1} \\ + Q_{m+1} \sum_{l <_m m+1} (\prod_{h <_m l} P_h) Q_l D_l^{n-1} \geq \\ (\prod_{h <_m m+1} P_h) + \sum_{l <_m m+1} (\prod_{h <_m l} P_h) Q_l = \\ Q_{m+1} D_y^{n-1} \quad (6.8)$$

where Equation (6.8) occurs because I_y is the first path that A_1 tries in both π_m and π'_m; therefore $D_l^{n-1} > D_y^{n-1}$ for every $l >^m y$. Equation 6.7 is due to the fact that

$$\sum_{l <_m m+1} (\prod_{h <_m l} P_h) Q_l + \prod_{h <_m m+1} P_h = 1$$

Then, from Lemma 6.4.4, we get $D^n_{m+1} - D_m < D^{n-1}_y$, and applying Property 1, we get $C(\pi') < C(\pi)$.

(Case 2:) π **is not committing**. Then T_π contains INV-arcs. We find another policy π' such that $C(\pi') \leq C(\pi)$ and $T_{\pi'}$ contains a smaller number of INV-arcs than T_π, thus contradicting the minimality of the number of INV-arcs in T_π among optimal policies.

By the induction assumption, π is a followers-committing policy for $A_1, \cdots A_{n-1}$; therefore the only INV actions in π are for A_0 (see Figure 6.4(a) for an example of such a non committing policy). The last action before A_0 reaches t must be a TRY action. Hence T_π contains a subtree with a root with a label b, called T_{π_b}, such that $\pi(b) = INV(0, i, j)$ for some $i < k$, $j < k_i$, and $\pi(b)$ is the only INV-arc in T_{π_b}. Assume without loss of generality that no paths are known to be blocked in b, and that the paths are ordered by the non-decreasing order of the $D^n_i(b)$.

Let π'_b be the optimal followers-committing policy starting at b (meaning the optimal followers-committing policy for M_b). Note that $T_{\pi'_b}$ contains no INV-arcs at all.

We first show that $C(\pi'_b) \leq C(\pi_b)$. Next we define π' to be the policy for M obtained from π by replacing π_b with π'_b. As the number of INV-arcs in T'_π is smaller than in T_π, and as $C(\pi'_b) \leq C(\pi_b)$, we have that $C(\pi') \leq C(\pi)$, and $T_{\pi'}$ contains a smaller number of INV-arcs than T_π, violating our assumption of a minimal number of INV-arcs in T_π.

To see that $C(\pi'_b) \leq C(\pi_b)$, first note that

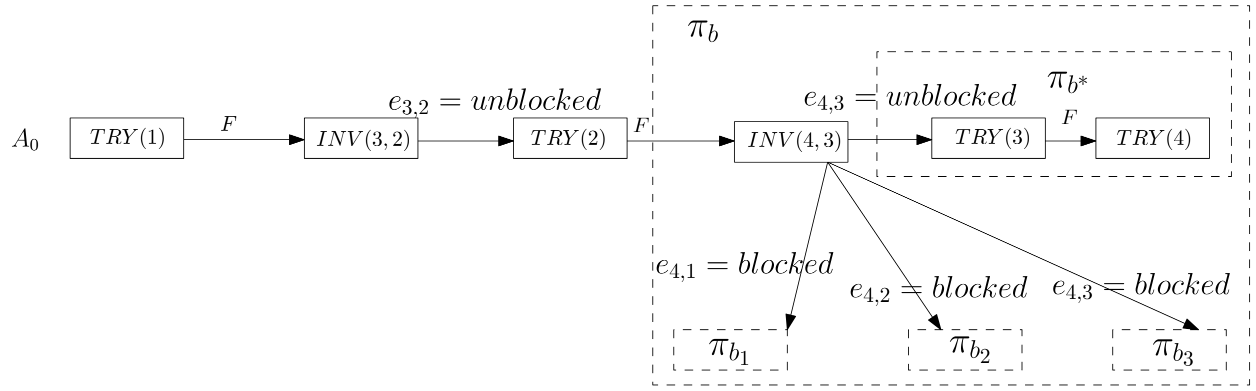
$$C(\pi'_b) = \sum_{0 \leq l < k} (\prod_{x<l} P_x) Q_l D^n_l(b) \tag{6.9}$$

(a product over a zero term is defined to be 1).

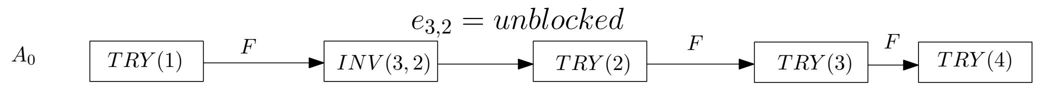
To develop $C(\pi_b)$, we make the following observation. Recall that the outcome of all $INV(0, i, j)$ actions is that A_0 is at s. Let b_l, for $l \leq j$, be the belief state reached as the outcome of the action $INV(0, i, j)$ performed at b, in which the edge $e_{i,l}$ is found blocked. Let b^* be the belief state reached as the outcome of $INV(0,i,j)$ executed at b in which $e_{i,j}$ is found unblocked.

Using this notation, π_b can be described as follows: *execute* $INV(0,i,j)$*; if* $e_{i,l}$ *is found blocked for some* $l \leq j$*, execute* π_{b_l}*; otherwise execute* π_{b^*}*.*

For example, Figure 6.4(a) describes a policy for A_0 in a CTP-REP instance with 4 paths in which the last INV-action is $INV(I_4,3)$. If $e_{4,j}$, for $j \leq 3$, is found blocked, the followers-committing policy π_{b_j} is executed. If $e_{4,3}$ is found unblocked, the followers-committing policy π_{b^*} is executed. The alternative policy in which π_b is replaced with a followers-committing policy is seen in Figure 6.4(b).



(a) An illustration of a non-committing policy. F indicates agent A_0 has found a certain path blocked and retraced to s. The rest of the agents are assumed to follow A_0.



(b) An illustration of an alternative policy for the policy in Fig, 6.4(a) in which π_b is replaced by a followers-committing policy. The rest of the agents are assumed to follow A_0

Figure 6.4:

For an edge e_j, let $Q_i^j := \prod_{x \leq j} q(e_{i,x})$ be the probability that the edges $\{e_{i,0}, \cdots .e_{i,j}\}$ are unblocked. As neither the $T_{\pi_{b_l}}$ nor $T_{\pi_{b^*}}$ contains an INV-arc (thus the π_{b_l} and π_{b^*} are committing policies), we may assume without loss of generality, by Case 1 (in which we have shown that there is an optimal followers-committing policy among all committing policies) that the π_{b_l} and π_{b^*} are followers-committing policies.

Note that $D_l^n(b_h) = D_l^n(b_j)$ for every $h \leq j$, and $D_l^n(b_j) = D_l^n(b^*)$ for every $l \neq i$. Therefore, as the same paths are blocked in both π_{b_h} and π_{b_j}, we have that $C(\pi_{b_h}) = C(\pi_{b_j})$ for every $h \leq j$. Therefore:

$$C(\pi_b) = Q_i^j(2W_{i,j} + C(\pi_{b^*})) + \sum_{x \leq j} E[BC_x(i)] + (1 - Q_i^j)C(\pi_{b_j}) \tag{6.10}$$

The expected cost of π_{b_j} is easily shown to be:

$$C(\pi_{b_j}) = \sum_{0 \leq l < i} (\prod_{x<l} P_x) Q_l D_l^n(b) + \frac{1}{P_i} \sum_{i<l<k} (\prod_{x<l} P_x) Q_l D_l^n(b) \tag{6.11}$$

The development of $C(\pi_{b^*})$ is a bit more complicated. This is because in b^*, some of the edges of path I_i are known to be unblocked; hence, in π_{b^*}, the path I_i may be tried sooner than in π_{b_j}.

Therefore we proceed as follows. For $x < k_i$ let $E[BC_x(i)]$ be the expected backtracking cost given that $e_{i,y}$ is unblocked for every $y < x$, and given that $e_{i,x}$ is blocked. Recall that $q(e)$ is the probability that edge e is unblocked, giving:

$$D_i^n(b^*) = \frac{\sum_{j<x<k_i} E[BC_x(i)]}{\prod_{j<x<k_i} q(e_{i,x})} + nW_i$$

hence

$$D_i^n(b^*) = D_i^n(b) - \frac{\sum_{x \leq j} E[BC_x(i)]}{Q_i} \tag{6.12}$$

Therefore $D_i^n(b^*) \leq D_i^n(b)$.

Now let I_m be the first path such that $D_i^n(b^*) \leq D_m^n(b) \leq D_i^n(b)$. Then we have:

$$C(\pi_{b^*}) =$$

$$\sum_{0\le l<m} (\prod_{x<l} P_x) Q_l D_l^n(b) + (\prod_{x<m} P_x) \frac{Q_i}{Q_i^j} D_i^n(b) +$$
$$(1 - \frac{Q_i}{Q_i^j}) \sum_{m\le l<i} (\prod_{x<l} P_x) Q_l D_l^n(b) +$$
$$(\frac{1}{P_i})(1 - \frac{Q_i}{Q_i^j}) \sum_{i<l<k} (\prod_{x<l} P_x) Q_l D_l^n(b) \quad (6.13)$$

Therefore

$$C(\pi_b) - C(\pi'_b) =$$

$$Q_i^J 2W_{<i,j>} + \sum_{x\le j} E[BC_x(i)] +$$
$$Q_i D_i^n(b)(\prod_{x<m} P_x - \prod_{x<i} P_x) -$$
$$Q_i \sum_{m\le l<i} (\prod_{x<l} P_x) Q_l D_l^n(b) \quad (6.14)$$

As $D_l^n(b) \le D_i^n(b)$ for every $l \le i$, and as

$$\sum_{m\le l<i} (\prod_{x<l} P_x) Q_l = \prod_{x<m} P_x - \prod_{x<i} P_x$$

we have

$$C(\pi_b) - C(\pi'_b) \ge Q_i^J 2W_{<i,j>} + \sum_{x\le j} E[BC_x(i)]$$

and as obviously,

$$Q_i^J 2W_{<i,j>} + \sum_{x\le j} E[BC_x(i)] \ge 0$$

we have $C(\pi'_b) \le C(\pi_b)$, as required. □

Remark 6.4.5 *Note that the proof of Theorem 6.2.1 serves as a proof for Theorem 2.5.3 of a single agent CTP (i.e., for $n = 1$). As in Case 2, the number of agents does not play a role, we only need to adjust Case 1 to do the proper adjustments for $n = 1$. Recall that for $n = 1$, there are no agents to follow A_0. Therefore, in Case 1, the cost of the policy for the following agents should be set to 0, and the value of all the $D_l^{n-1}(b)$ should be set to 0 as well.*

Chapter 7

Conclusion

7.1 Related Work

In 1989, Papadimitriou & Yannakakis [31] introduced the Canadian Traveler Problem and showed that an online version of the CTP, in which the ratio between the optimal solution and the shortest path is bounded, is PSPACE-complete. Membership in PSPACE and #-P hardness was shown for the stochastic version. The proof that the CTP is PSPACE-complete (Chapter 3, and also [14]) closes this complexity gap. One of the techniques we use in that proof, the use of CTP with dependency (CTP-Dep), was explored in an M.Sc. Thesis of Doron Zarchi [46] in which heuristic solutions were considered for CTP-Dep. An alternative preliminary and independent proof of the result in Chapter 3 appears in a graduate thesis of Cenny Wenner [43], as well as the complexity of policy representation, which we did not pursue in this work.

In 1991 Bar-Noy and Scheiber [3] introduced several variants of the CTP. In one of them, called Recoverable CTP, a blocked edge does not remain blocked forever, but each vertex v has a *recovery time* $l(v) \in [0, \infty)$ for the edges adjacent to v, such that after an edge is revealed to be blocked, the agent can wait $l(v)$ time and afterward the edge becomes unblocked. Bar-Noy and Scheiber showed a polynomial time strategy that minimizes the expected travel time in cases where the recovery time of a vertex v is short - relative to the travel time of the edges adjacent to v.

Another variant first explored in [3] is the so-called $k-CTP$, and in which an upper bound of k blocked edges is given as a part of the problem. Bar-Noy and Scheiber presented a travel strategy, polynomial for any constant k, which finds the shortest worst-case travel time. They also showed that in cases where k is not constant, finding such a strategy is PSPACE-complete. $k - CTP$ was further explored [44, 45].

In 2008, Nikolova & Karger [30] explored another variant of CTP with the costs of the edges coming from a general known probability distribution . Nikolova & Karger showed that if the value of the edges incident to a vertex v is re-sampled every time we reach v, there is a natural MDP that solves this problem in polynomial time. The problem is also easy to solve in cases where the graph under discussion is directed and acyclic (DAG). Then, assuming that all the edges have independent and identical distribution, Nikolova & Karger showed an optimal policy for disjoint-path graphs. With a further limitation to (0,1) Bernoulli edges, they also showed optimal policy for binary trees. Note that this variant of trees is different from the CTP-Tree presented in Chapter 5.

In 2010, Eyerich, Keller, and Helmert [12] introduced and compared several heuristics and algorithms for the CTP. Apart from introducing observations and notations that are used throughout this work, one of their contributions was to consider a state-of-the-art approach called UCT [22], as a suggested sampling scheme for a CTP solver. This work was later followed by Bnaya et al. [7], who generalized the UCT algorithm for Repeated-CTP, and in which the results in Chapter 6 were published. The "optimistic" policy, suggested in [7] is based on the "free-space assumption" introduced in [23]. Other works on repeated, multi-agent navigation problems appear in [45, 29, 13].

In 2009, Bnaya, Felner and Shimony [6] showed that the CTP has a polynomial time solution on disjoint path graphs. This result, which this work *lies heavily* on, is given in Section 2.5 and in Chapter 6. Apart from that, a variant of the CTP with Remote Sensing (Sensing-CTP) was introduced in [6]. In Sensing-CTP, the agent can reveal the status of a remote edge for a non-negative cost. Bnaya, Felner, and Shimony [6] compared various algorithms which solve Sensing-CTP.

Another variant of Sensing-CTP called First-Sensing-CTP was discussed in the M.Sc. Thesis of Olga Maksin [27]. In First-Sensing-CTP, the agent can only traverse a path known to be unblocked. Therefore, revealing the status of the edges must be done through sensing actions, prior to any move action. A variant of First-Sensing-CTP in which all the edges have a move cost 0, is a special case of a problem known as the Sequential Testing Problem, in which a system is tested through its components, which are connected in a graph structure. Every component has a failure probability, and a testing cost. The objective is to find a policy that minimizes the expected testing cost of the system [39, 40, 5]. Sequential testing is widely explored, and appears under various names [20, 17].

Variants and related problems. An alternative definition of the CTP lies in Operation Research. Polychronopolous and Tsitsiklis (1996) [33] have defined the *Independent Stochastic Path Problem with Recourse (I-SSPPR)* in which there is a distribution over the costs of the edges. This problem is a special model of the R-SSPPR problem, defined in [33], in which a limited dependency over the cost of the edges is allowed. Dynamic programming algorithms were presented in [33] for both models, as well as complexity gaps.

An earlier version of the Stochastic Shortest Path Problem with Recourse (SSPPR) appears in [2]. The actions allowed in [2] are for the agent to follow a pre-chosen path from s to t, and choose a recourse if and only if this path has become blocked. A dynamic programming algorithm was presented in [2] that solves the SSPPR. Our result in Chapter 3 shows that this variant of the SSPPR is PSPACE-complete as well. In 2003, Provan [34] discussed various versions of the SSPPR, and suggested a polynomial time algorithm for cases in which every edge is re-sampled after every move the agent makes. Other works in which the problem is to find the expected shortest path in a stochastic network appear in [38, 9, 15, 25, 24, 42].

Generally, navigating in stochastic graphs, or stochastic networks, has received a lot of attention. For example, the *Online graph exploration* [28] where the agent must visit each vertex in an unknown graph, or [18], where certain paths can be quarried in an unknown network. Another problem is the so called "sabotage game" [21], where an adversary can block edges in a graph, after every move the agent makes. Another example for network routing problems with local or global reliability appears in [37]. Other properties of stochastic weighted graphs can be found in [10].

Finally, a recent variant is the Stochastic on Time Arrival (SOTA) problem, where a time budget is given for a stochastic network, and the objective is to find a strategy that maximizes the probability of arriving at the destination within the specific time frame. This variant was explored in [35].

7.2 Summary and future work

The Canadian Traveler Problem (CTP) formalizes a problem in navigation under uncertainty. The objective is to find a strategy to reach from a given source to a given destination where a road can be found to be blocked upon reaching that road. As the CTP is a problem in decision-making under uncertainty, we model the CTP as a Deterministic Partial Observable Markov Decision Process (Det-POMDP). The description of a Det-POMDP as an AND/OR tree is used for theoretical analysis.

Having shown that the CTP is PSPACE-hard (Chapter 3), several related questions on variants of CTP and CTP with restricted topologies arise. One issue of particular interest is the question of efficiently finding approximately optimal actions. The proof in Section 3.3 makes use of rather small gaps between expected values of two candidate actions, and thus leaves open the possibility of efficient approximation algorithms.

Studies of the *competitive analysis* of the CTP reveal rudimentary bounds on approximability. Denoting by k the number of uncertain edges in an instance, there exist for the undirected case polynomial-time algorithms achieving competitive ratios of $2k + 1$ [44]. As a consequence, the stochastic CTP can be approximated within a factor of $2k + 1$. With a slightly improved analysis, the same algorithm yields a $2n + 1$-approximation. In the directed case, existing results from competitive analysis only yield approximations of $2^{k+1}+1$ and $2^{n+1}+1$, respectively [45].

These approximation algorithms forgo entirely the stochastic nature of the problem and leave open considerable improvements. A preliminary work of ours (with Cenny Wenner) shows, by reduction from the $\{1,2\}$-TSP [11], that the CTP is hard to approximate up to a certain constant.

Another variant of CTP that we find of interest is CTP with "reset edges" (Reset-CTP). In Reset-CTP, the CTP graph is directed acyclic, apart from always unblocked directed edges, called *reset edges*, that connect every vertex to s. The motivation is a decision-making system in which retracing is impossible, except for the possibility to go back to the starting position (while keeping all the knowledge acquired so far) for a certain cost. A special variant called 0-Reset-CTP is when all the reset edges have zero cost. A preliminary work of ours (with Amit Benbassat) shows that 0-Reset-CTP is PSPACE-complete as well.

The partition framework technique, presented in Chapter 4, can serve to decompose a CTP-instance into several components, followed by implementing an independent heuristic on each component, thus gaining an overall improved heuristic. Such a decomposition can be obtained by considering certain edges to be blocked, at the cost of losing optimality.

The problem of whether CTP-Tree is NP-hard remains open. Solving 1-CTP-Tree where the outgoing edges of the exploration vertex are unknown, remains a challenge as well. With minor modification, the algorithm **1-Exp-CTP-Tree** presented in Chapter 5 can be implemented for 1-CTP-Tree with a constant j number of unknown outgoing edges of the exploration vertex v^1. This is done by constructing 2^j tables; one table for every possible observation of the outgoing edges of v^1. With another minor modification, **1-Exp-CTP-Tree** can be used to solve k-CTP-Tree, where every exploration vertex has a committing parent, committing children, and committing siblings. The problem of 2-CTP-Tree where both a vertex and its parents are exploration vertices is still a challenge.

A possible generalization of CTP-Tree is a CTP with a series-parallel graph structure (SP-CTP). In SP-CTP, unlike CTP-Tree, a vertex can be reached from s through more than one simple path; therefore we believe that SP-CTP contains more challenges than CTP-Tree. A possible constraint for SP-CTP is that the agent is not only committed to a certain subgraph, but must retrace his own steps, once that subgraph is found blocked. The theoretical and empirical analysis of this variant is a future work.

Repeated-CTP was found tractable on disjoint-path graphs (Chapter 6). We conjectured that Interleaved-CTP on disjoint-path graphs is tractable as well, and specifically, given a disjoint-path graph, every optimal for Repeated-CTP is also optimal for Interleaved-CTP. This conjecture was empirically checked in [27] in which, in over a thousand disjoint-path graph instances, no counter-example was found.

In short, the following presents the contributions of this work.

1. Proving that CTP is PSPACE-complete, thus solving a two decades old open problem (Chapter 3). This result appeared in [14].
2. Proving that the optimal cost for the CTP is monotonically non-decreasing in edges costs and blocking probabilities (Section 4.1).
3. Introducing constrained-CTP (Section 4.2).
4. Introducing the CTP partition framework, which provides a mechanism for CTP decomposition (Section 4.2.2).
5. Introducing CTP-Tree (Chapter 5).
6. Efficiently solving CTP-Tree with no exploration vertices (Section 5.1).
7. Efficiently solving CTP-Tree, which has a single exploration vertex, with known outgoing edges (Section 5.1).
8. Efficiently solving EFC-CTP-Tree (Section 5.3), empirically testing other CTP-Tree instances (Section 5.4).
9. Introducing Repeated-CTP and efficiently solving Repeated-CTP on disjoint path graphs (Chapter 6). This result appeared in [7]. A journal version was recently submitted as well.

Appendix A

Claim 2.3.1 *If π^* is an optimal policy for a Det-POMDP M, and $b \in B_M(b_0, \pi^*)$, then π^*_b is an optimal policy for M_b.*

Proof: Assume in contradiction that there is a policy π'_b for M_b such that $C(\pi'_b) < C(\pi^*_b)$. Recall that $B_{M_b} = B_M(b)$ is a subset of B_M. Let π' be the following policy for M: for every $d \in B_{M_b}$, let $\pi'(d) = \pi'_b(d)$, and for every $d \notin B_{M_b}$, let $\pi'(d) = \pi^*(d)$. For $v \in V$ where $L(v) = b$, let $H_v = \{v_0, \cdots v_l\}$ be the trunk of v. Then $v_0 = r$, and $v_l = v$. Denote $L(v_i)$ by b_i, for every $i < l$. We prove by backward induction on l, that $V^{\pi'}(v_i) < V^{\pi^*}(v_i)$ for every $i < l$. Then, as $C(\pi') = V^{\pi'}(r)$ and $C(\pi^*) = V^{\pi^*}(r)$, it follows that π^* is not optimal, contradicting the claim assumption.

For the basis of v_l we have that $L(v_l) = b$, and as $C(\pi'_b) = V^{\pi'}(v)$ and $C(\pi^*_v) = V^{\pi^*}(v)$, we have that $V^{\pi'}(v_l) < V^{\pi^*}(v_l)$.

We prove that $V^{\pi'}(v_i) < V^{\pi^*}(v_i)$ for $i < l$. First assume that v_i is an OR-node. As $b_i \notin B_{M_b}$ we have that $\pi'(b_i) = \pi^*(b_i)$ Therefore

$$V^{\pi'}(v_i) = c(v_i, v_{i+1}) + V^{\pi'}(v_{i+1}) \tag{A.1}$$

and

$$V^{\pi^*}(v_i) = c(v_i, v_{i+1}) + V^{\pi^*}(v_{i+1}) \tag{A.2}$$

By the induction assumption $V^{\pi'}(v_{i+1}) < V^{\pi^*}(v_{i+1})$, therefore $V^{\pi'}(v_i) < V^{\pi^*}(v_i)$.

Next, assume v_i is an AND-node. Denote the set of observation received in a state j, after an action a was performed at belief state b, by $Z(b, a, j)$. As $b_{i-1} \notin B_{M_b}$, we again have that $\pi'(b_{i-1}) = \pi^*(b_{i-1})$. Hence $Z(b_{i-1}, \pi'(b_{i-1}), j) = Z(b_{i-1}, \pi^*(b_{i-1}), j)$ for every state $j \in sup(b_i)$. Therefore we have by Equation (2.8)

$$V^{\pi'}(v_i) = \sum_{(v_i,u)\in E} (p((v_i, u))V^{\pi'}(u)) \tag{A.3}$$

and

$$V^{\pi^*}(v_i) = \sum_{(v_i,u)\in E} (p((v_i, u))V^{\pi^*}(u)) \tag{A.4}$$

where E is the set of outgoing observation-arcs from v_i.

Note that $(v_i, v_{i+1}) \in E$, and for every node u where $(v_i, u) \in E$ and $L(u) \neq b_{i+1}$, we have $B_M(L(u)) \cap B_M(b) = \emptyset$. Hence for every belief state $d \in B_M(L(u))$, we have $\pi'(d) = \pi^*(d)$, which implies $V^{\pi'}(u) = V^{\pi^*}(u)$. By the induction assumption we have $V^{\pi'}(v_{i+1}) < V^{\pi^*}(v_{i+1})$. Therefore from Equation A.3 and Equation A.4 we have $V^{\pi'}(v_i) < V^{\pi^*}(v_i)$ as required.

□

Appendix B

B.1 Bayes Network construction

We describe the layout of the Bayes Network (Y, A, P) that is required for the proof of Theorem 3.2.1. The set Y of random variables is described as follows. The universal edges (v_i, v_{i1}), and $(\bar{v}_i, \bar{v}_{i1})$ are represented by the random variables f_i, $\bar{f}_i$ for every universal variable x_i. For every variable x_i and clause c_j, the observation edges (o_{ij}, v_{ij}), and $(\bar{o}_{ij}, \bar{v}_{ij})$ are represented by the random variables e_{ij}, and $\bar{e}_{ij}$ respectively. The chance edges (r_1, r'_1) and (r_2, r'_2) are represented by the random variables *odd*, and *even* respectively. All the random variables described so far have a range of $\{blocked, unblocked\}$. In addition for every $i \leq m$, we have random variables c_i, and d_i called clause-variables with a range of $\{0, 1\}$. The clause-variable c_i is 1 if and only if the observation edges that are related to the clause c_i are found blocked. The clause-variable d_i describes the XOR operation between the clause-variables d_{i-1}, and c_i.

The layout of the network along with the conditional probability tables appears in Figures B.1,B.2, and B.3. Figure B.1 describes the dependency between the universal edges. Figure B.2 describes an example of the dependency between the observation edges that represent three literals that appear in the same clause. Finally, Figure B.3 describes the network that allows the dependency of the chance edges. Note that BN is a polytree with a size linear in the size of the input formula Φ. Therefore probabilistic inference on BN can be done in a time linear to the size of Φ as well.

$$\begin{tikzcd} f_i \arrow[d] \\ \bar{f}_i \end{tikzcd}$$

f_i	$P(f_i)$
Blocked	0.5
Unblocked	0.5

f_i	$\bar{f}_i$	$P(\bar{f}_i \mid f_i)$
Blocked	*Blocked*	0
Blocked	*Unblocked*	1
Unblocked	*Blocked*	1
Unblocked	*Unblocked*	0

Figure B.1: The segment of the Bayes network construction that describes the dependency between the universal edges.

e_{13}	$P(e_{13})$
Blocked	0.5
Unblocked	0.5

e_{13}	$\bar{e}_{23}$	$P(\bar{e}_{23} \mid e_{13})$
Blocked	*Blocked*	0
Blocked	*Unblocked*	1
Unblocked	*Blocked*	1
Unblocked	*Unblocked*	0

$$\begin{tikzcd} e_{13} \arrow[d] \\ \bar{e}_{23} \arrow[d] \\ e_{43} \arrow[d] \\ c_3 \end{tikzcd}$$

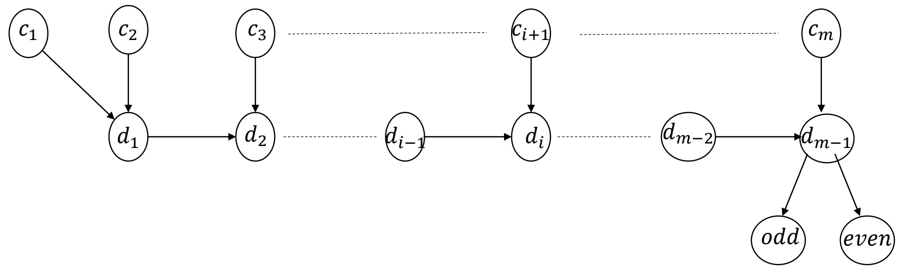
$\bar{e}_{23}$	e_{43}	$P(e_{43} \mid \bar{e}_{23})$
Blocked	*Blocked*	0
Blocked	*Unblocked*	1
Unblocked	*Blocked*	1
Unblocked	*Unblocked*	0

c_3	e_{43}	$P(c_3 \mid e_{43})$
0	*Blocked*	0
0	*Unblocked*	1
1	*Blocked*	1
1	*Unblocked*	0

Figure B.2: An example of the Bayes network construction that describes the dependency between the observation edges. The clause in description is $c_3 = (x_1 \vee \neg x_2 \vee x_4)$.

B.2 Baiting gadgets

Let $g = BG(u, v)$ be a baiting gadget with a parameter $l > 1$, defined in Section 3.3.1 (see Figure 3.2, appears below as well). Recall that π (as defined in Section 3.3.1) is the following policy for g: *when at u for the first time, proceed along the path $(u, v_1, \cdots, v_N, v)$ to v, taking the zero-cost shortcut to t whenever possible, but never backtracking to u. From v continue with any optimal policy.*



c_1	c_2	d_1	$P(d_1\|c_1,c_2)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

d_{i-1}	c_{i+1}	d_i	$P(d_i\|d_{i-1},c_{i+1})$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

d_{m-1}	$even$	$P(even\|d_{m-1})$
0	Blocked	0
0	Unblocked	1
1	Blocked	1
1	Unblocked	0

d_{m-1}	odd	$P(odd\|d_{m-1})$
0	Blocked	1
0	Unblocked	0
1	Blocked	0
1	Unblocked	1

Figure B.3: The segment of the Bayes network construction that describes the dependency between the chance edges. The d_i's variables describe a XOR operation between d_{i-1}, and c_i.

In formal terms, the above description of π should be interpreted as follows. A partially specified policy is *relevant* only to belief states for which it specifies an action. π is relevant to any belief state b, where $Loc(b) = u$ and the state of all the zero-cost shortcut edges (v_i, t) in the gadget are unknown. π is also relevant to belief states reachable from b by acting according to π. In any belief state consistent with such b, traverse the edge (u, v_1). Likewise, at any belief state where we are at vertex v_i (for $1 \leq i \leq N$), if the zero-cost shortcut edge (v_i, t) is traversable, then traverse it. Otherwise traverse the edge (v_i, v_{i+1}) (except when $i = N$, in which case traverse (v_N, v)). At any belief state where we are at v, perform an action that is the first action in *some* optimal policy. Note that when acting according to π, it is indeed only possible for the zero-cost shortcut edges in the baiting gadget to be unobserved if this is the first time we are at u.

Apart from π, other policies at u that are not clearly suboptimal are:

- *Choose not to traverse* (u, v_1).
- The following type of policies denoted by π_j, for $j \leq N$: *execute* π *until reaching* v_j*; if* (v_j, t) *is unblocked, reach the destination through* (v_j, t)*; otherwise, retreat to* u *and execute an optimal policy with an*

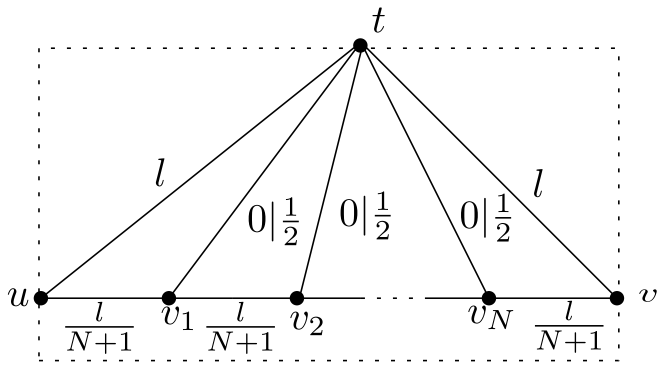


Figure 3.2: A baiting gadget $BG(u, v)$ with a parameter $l > 1$. Edge label $c \mid p$ denotes cost | blocking probability. The optimal policy at u is to cross the path $(u, v_1, \cdots, v_N, v)$, taking a shortcut edge to t whenever such an edge is found unblocked. After reaching v, retracing to u in g costs at least l.

expected cost of $M_j \geq 0$.

Finally, we set $N = 2^{\lceil \log_2(4l) \rceil} - 1$, implying $N + 1 \geq 4l$.

Claim 3.3.3 *When at u for the first time, under Invariant 3.3.2, π is optimal for a baiting gadget $g = BG(u, v)$ with a parameter $l > 1$. After reaching v, it is suboptimal to backtrack to u in g.*

Proof: Denote by $K \geq 0$ the expected cost of every optimal policy executed once v is reached. As there is a cost l shortcut edge (v, t), it is clear that $K \leq l$. Therefore, as retracing g from v to u costs l, it is always suboptimal to retrace g once v is reached. We first show that $C(\pi) < 1$, hence under Invariant 3.3.2, choosing not to traverse (u, v_1) is suboptimal.

Note that for every $i \leq N$, the probability that (v_i, v_{i+1}) is traversed in π is $(\frac{1}{2})^i$. Hence we have

$$C(\pi) = \frac{l}{N+1} \sum_{i=0}^{N} (\frac{1}{2})^i + (\frac{1}{2})^N K \tag{B.1}$$

Thus

$$C(\pi) = \frac{2l}{N+1}(1 - 2^{-(N+1)}) + 2^{-N} K \tag{B.2}$$

Then, as $K \leq l$, $N + 1 \geq 4l$, and $l > 1$, we have that

$$C(\pi) < \frac{2l}{4l} + 2^{-N}l < \frac{3}{4} < 1. \tag{B.3}$$

as required.

Finally we show that for every $j \leq N$, $C(\pi) < C(\pi_j)$, hence the policy π_j is suboptimal. We have that:

$$C(\pi_j) = \frac{l}{N+1}\sum_{i=0}^{j-1}(\frac{1}{2})^i + (\frac{1}{2})^j\left(\frac{jl}{N+1} + M_j\right) \tag{B.4}$$

Thus:

$$C(\pi_j) = \frac{2l}{N+1}(1-2^{-j}) + \frac{2^{-j}jl}{N+1} + 2^{-j}M_j \tag{B.5}$$

Then in order to prove that $C(\pi) < C(\pi_j)$, we need to prove, from (B.2), and (B.5), that

$$\frac{2l}{N+1}(1-2^{-(N+1)}) + 2^{-N}K < \frac{2l}{N+1}(1-2^{-j}) + \frac{2^{-j}jl}{N+1} + 2^{-j}M_j \tag{B.6}$$

From Invariant 3.3.2, we have that $1 \leq M_j$. Then, as $K \leq l$, it is sufficient to show that for every $0 < j \leq N$:

$$\frac{2l}{N+1}(1-2^{-(N+1)}) + 2^{-N}l < \frac{2l}{N+1}(1-2^{-j}) + \frac{2^{-j}jl}{N+1} + 2^{-j} \tag{B.7}$$

For this we need to prove

$$\frac{2l}{N+1}(2^{-j} - 2^{-(N+1)} - 2^{-j-1}j) + 2^{-N}l < 2^{-j}$$

By multiplying both sides by 2^j, we need to show that for every $0 < j \leq N$,

$$\frac{2l}{N+1}\left(1 - 2^{j-N-1} - \frac{j}{2}\right) + 2^{j-N}l < 1$$

For this, it is sufficient to show that for every $0 < j \leq N$,

$$\frac{2l}{N+1}\left(1-\frac{j}{2}\right)+2^{j-N}l<1 \tag{B.8}$$

Finally, inequality (B.8) follows since the function:

$$f(x)=\frac{2l}{N+1}\left(1-\frac{x}{2}\right)+2^{x-N}l$$

over the reals, has only one extremum, and as $N+1\geq 4l$ and $l>1$, we have that $f(0)<1$, $f(N)<1$ and

$$\lim_{x\to\infty} f(x)=\lim_{x\to-\infty} f(x)=\infty$$

□

B.3 Observation gadgets

Let $g=OG(u,v,o)$ be an observation gadget as defined in Section 3.3.2, and seen in Figure 3.3, appears below as well). Recall that π_g is the following partially specified policy for $OG(u,v,o)$: *At u, cross BG_1 (observe (v_1,v_4)). Then cross BG_2. If either (v_1,v_4) or (v_2,v_3) is found blocked, reach t by traversing the shortcut edge (v_2,t) with cost $3L/2$. However, if both (v_1,v_4), and (v_2,v_3) are unblocked, traverse (v_2,v_3,o,v_4,v_1,v_1') (at o, observe any edges incident on o), and cross BG_3. From v continue with any optimal policy.*

Claim 3.3.5 *Assume $L>8$. Then, when at u for the first time, under Invariants 3.3.2 and 3.3.4, π_g is an optimal policy for an observation gadget $g=OG(u,v,o)$.*

Proof: First note that by following π_g, Invariants 3.3.2 holds for every baiting gadget in g. At u, as BG_1 is a baiting gadget, then by Claim 3.3.3, it is optimal to cross BG_1. When first arriving at v_1, after BG_1 is crossed, (v_1,v_4) is observed. As (o,v_4) has a cost of $5L/8>1$, and (v_1,v_1') has a cost

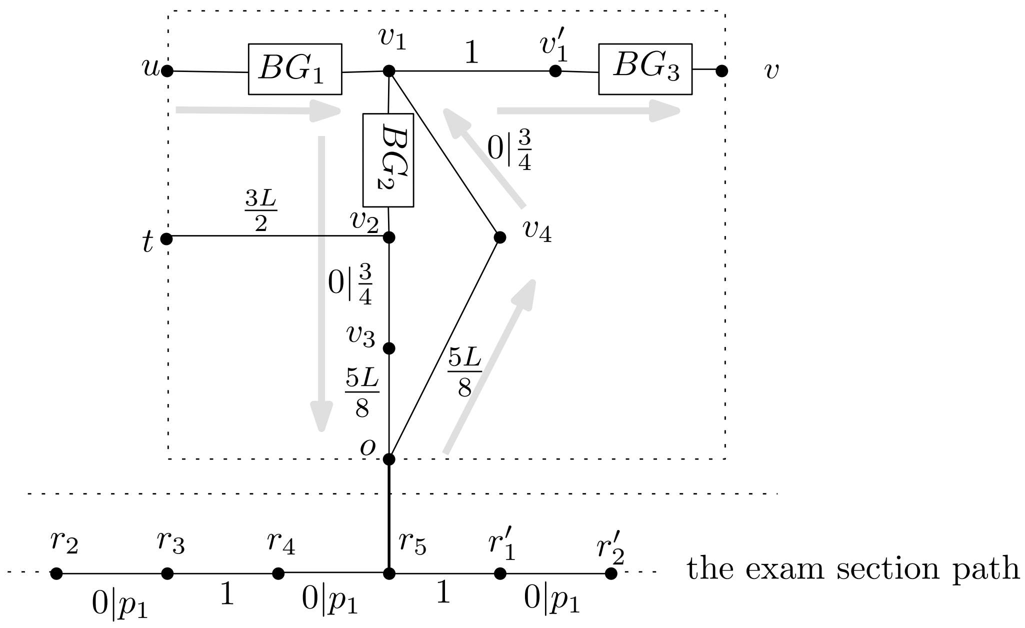


Figure 3.3: An observation gadget $OG(u, v, o)$. Light gray arrows indicate general traversal direction of the optimal policy π. BG_1 and BG_3 are baiting gadgets with a parameter $l = L$. BG_2 is a baiting gadget with a parameter $l = 3L/2$.

of 1, then by Claim 3.3.3, it is optimal to cross BG_2. Once at v_2, if (v_2, v_3) is blocked, it is optimal to take the shortcut (v_2, t) for a cost of $3L/2$.

It remains to show that if (v_2, v_3) is unblocked, the optimal policy at v_2 is:

1. If (v_1, v_4) is unblocked, traverse $(v_2, v_3, o, v_4, v_1, v_1')$, cross BG_3, and from v continue with any optimal policy.

2. Otherwise, traverse the shortcut (v_2, t) for a cost of $3L/2$.

Case 1: (v_1, v_4) is unblocked.

First note that arriving at v_1 a second time through (v_4, v_1), BG_1 and BG_2 are known not to have any unblocked shortcut edges, thus crossing them costs at least $L > 8$. Hence by Claim 3.3.3, the optimal policy when arriving at v_1 a second time is to traverse (v_1, v_1'), cross the baiting gadget BG_3, and from v continue with any optimal policy for a total expected cost of less than 2.

Now, traversing $(v_2, v_3, o, v_4, v_1, v_1')$, crossing BG_3, and continuing with an optimal policy, bears an expected cost of at most $2(5L/8) + 2$, while traversing (v_2, t) costs $3L/2$. Hence, as $2(5L/8) + 2 < 3L/2$, it is optimal at v_2 to traverse (v_2, v_3, o) to the vertex o.

We now inspect the possible partially specified policies at o:

Case 1.a: Traverse (o, v_4, v_1, v_1'), cross BG_3, and from v continue with any optimal policy, for an expected cost of at most $5L/8 + 2$. We denote this partially specified policy by π'.

Case 1.b: Traverse edges of another observation gadget $\widetilde{g}$, if there exists such $\widetilde{g}$ incident on o, and continue with any optimal policy. Suppose that $\widetilde{g}$ has not already been traversed (label the vertices of $\widetilde{g}$ as $\widetilde{v_i}$). Then traversing either $(o, \widetilde{v_3})$ and trying to traverse $(\widetilde{v_3}, \widetilde{v_2})$, or traversing $(o, \widetilde{v_4})$ and trying to traverse $(\widetilde{v_4}, \widetilde{v_1})$, results in an expected cost of at least $5L/8 + 3(5L/8)/4$. Hence, as $5L/8 + 2 < 5L/8 + 3(5L/8)/4$, we have that executing π' is cheaper than traversing any edges of $\widetilde{g}$.

Next suppose that $\widetilde{g}$ has already been traversed. Therefore we may assume that the policy $\pi_{\widetilde{g}}$ was executed in $Entry(\widetilde{g})$; thus the baiting gadgets of $\widetilde{g}$ are known not to contain any unblocked zero-cost shortcuts, hence crossing each such baiting gadget costs at least L. Then traversing $\widetilde{g}$ results in an expected cost of at least $5L/8 + L$, and as $5L/8 + 2 < 5L/8 + L$, we again have that executing π' is cheaper than traversing any edges of $\widetilde{g}$.

Case 1.c: Traverse the exam section path, if o is connected to this path. Recall that o is identified with r_5. First suppose that the observation edge (r_4, r_5) is blocked. At o, denote the following partially specified policy by π_1: *cross (r_5, r_1'); if (r_1', r_2') is unblocked, continue with any optimal policy, otherwise, return to o, and execute π', which can still be executed, for an expected cost of* $C(\pi') < 5L/8 + 2$. Then we have

$$C(\pi_1) \geq 1 + p_1(1 + C(\pi'))$$

and as $p_1 > 1 - 2/(3L+1)$, we have that

$$C(\pi') < 1 + p_1(1 + C(\pi')) \leq C(\pi_1)$$

Therefore executing π' is cheaper than executing π_1.

Now suppose that (r_4, r_5) is unblocked. Then we can either execute π_1 (or the symmetric case in which (r_2, r_3) is being inspected) with the same analysis, or we can extend π_1 with the following policy denoted by π_2:

Execute π_1; upon returning to o (after (r'_1, r'_2) is found blocked), cross (r_5, r_4) and (r_4, r_3); if (r_3, r_2) is unblocked, continue with any optimal policy; otherwise return to r_5, and execute π', which can still be executed for an expected cost of $C(\pi') < 5L/8 + 2$. Then we have that

$$C(\pi_2) \geq 1 + 2p_1 + p_1^2(1 + C(\pi'))$$

However, $p_1 > 1 - 2/(3L+1)$ entails $C(\pi') < C(\pi_2)$. Therefore executing π' is cheaper than executing π_2 as well. The policy in which (r_2, r_3) is the first edge among (r_2, r_3) and (r'_1, r'_2) to be inspected is symmetric to π_2. Hence we see that traversing any edges of the exam section path is suboptimal.

Case 2: (v_1, v_4) is blocked.

In this case the following partially specified policies can be executed at v_2:

Case 2.a: Take the shortcut edge (v_2, t) for a cost of $3L/2$. Denote this policy by π'.

Case 2.b: Traverse (v_2, v_3) and (v_3, o) for a cost of $5L/8$ and at o traverse edges of another observation gadget. As in Case 1.b we have that traversing edges of $\widetilde{g}$ results in an expected cost of at least $5L/8 + 3(5L/8)/4$. Then, as $3L/2 < 5L/8 + 5L/8 + 3(5L/8)/4$, we have that π' is cheaper than reaching o and traversing any edges of $\widetilde{g}$.

Case 2.c: Traverse (v_2, v_3) and (v_3, o), for cost of $5L/8$, and at o traverse the exam section path. We define π'_1 and π'_2 as π_1, π_2 respectively in Case 1.c, with the one difference that when back at r_5, the cost $5L/8$ edge (v_3, o) is traversed before executing π'.

Then we have that

$$C(\pi'_1) \geq 1 + p_1(1 + 5L/8 + C(\pi'))$$

and

$$C(\pi'_2) \geq 1 + 2p_1 + p_1^2(1 + 5L/8 + C(\pi'))$$

Recall that $C(\pi') = 3L/2$. However as $p_1 > 1 - 2/(3L + 1)$, we have that $C(\pi') < 5L/8 + C(\pi'_1)$ and $C(\pi') < 5L/8 + C(\pi'_2)$. Hence π' is cheaper than traversing (v_2, v_3, o) and then traversing any edges of the exam section path.

This concludes the proof.

□

B.4 Behavior of reasonable policies

Claim 3.3.6 *At r_0, any reasonable policy acts as follows:*

- *If all the edges in the exam section were observed to be unblocked, cross $(r_0, r_1^1, \cdots, r_4^{m+1}, t)$ until reaching t for a cost of $2(m + 1)$.*
- *Otherwise, cross the cost L shortcut edge (r_0, t).*

Proof: We first note that any deviation from the exam section results in a cost of at least L. Then note that unless all the edges in the exam section were observed to be unblocked, any partially specified policy executed at r_0 results in a cost of more than L; therefore it is cheaper to take the shortcut (r_0, t) for a cost of L.

Retracing $BG(z_q, z_0)$ clearly results in a cost of at least L. At every vertex r_i^l, $l \leq m+1$, , $i \leq 5$, any unblocked edge on the exam section path, incident on r_i^l, can be traversed. At r_2^l there is an additional option to cross either $BG(z_{l+1}, z_l)$ or $BG(z_l, z_{l-1})$, which hold no unblocked shortcut edges; hence crossing these results in a cost of at least L. If r_5^l is identified with an observation point of some observation gadget $\widetilde{g}$, there is an additional option to traverse edges of $\widetilde{g}$. However, by an argument identical to Case 1.b of the proof of Claim 3.3.5, traversing any edges of $\widetilde{g}$ results in a cost of at least L. Hence any deviation from the exam section path results in a cost of at least L.

Now, suppose that all the edges of the exam section are known to be unblocked. Then, as the exam section contains $2(m+1)$ always traversable cost 1 edges, and as $2(m+1) < L$, the optimal policy is to cross the exam section $(r_0, r_1^1, \cdots, t)$ for the cost of $2(m+1)$.

Otherwise, suppose there are edges in the exam section with unknown status. Let e be the first unknown clause edge in a sense that that every edge in the path from r_0 to e is known to be unblocked. Finding e blocked results in either retracing the exam section to r_0 and taking the cost L shortcut to t, or in deviating from the exam section for a cost of at least L. Hence, as (r_0, r_1) is an unblocked cost 1 edge, traversing the path from r_0 to e results in an expected cost of at least $1+p_1L$. And as $L > 1$ and $p_1 > 1-2/(3L+1)$, we have that $1+p_1L > L$. Therefore traversing the shortcut edge (r_0, t) for a cost L is cheaper. Obviously, the same argument holds when e is previously known to be blocked.

$\square$

Appendix C

Lemma 4.1.2 *Let $I = (V, E, s, t, p, w)$, and $I' = (V, E, s, t, p', w)$ be a CTP instance such that $p' \leq p$. Let π^*, π'^* be optimal policies for I, I' respectively. Then $C(\pi'^*) \leq C(\pi^*)$.*

Proof: We show that if $p'(e) < p(e)$ for some $e \in E$, and $p'(e_1) = p(e_1)$ for every $e_1 \neq e$, then $C(\pi'^*) \leq C(\pi^*)$.

As both I and I' have the same graph layout (V, E), the belief states of B_I and of $B_{I'}$ have the same variables-status representation. Therefore we can define the bijection $f : B_{I'} \to B_I$, as in Lemma 4.1.1, in which $Loc(b) = Loc(f(b))$, and $b|e' = f(b)|e'$, for every $b \in B_I$ and $e' \in E$. In addition let $g_e : B_I \to B_I$ be the following function. For every belief state $b \in B_I$:

- $Loc(g_e(b)) = Loc(b)$
- $g_e(b)|e' = b|e'$ for every $e' \neq e$
- $g_e(b)|e = b|e$ if $b|e = unknown$
- $g_e(b)|e = blocked$ if $b|e = unblocked$
- $g_e(b)|e = unblocked$ if $b|e = blocked$.

We say that e is *revealed* in a transition from a belief state b' to a belief state b in a policy π if $b = b'^{o}_{\pi(b')}$ for some observation o, and if the status of e is unknown in b' and is known in b. We say that e is revealed in a belief state b if there is a belief state b' such that e is revealed from b' to b in π.

Note that every two distinct belief states b_1, b_2, in which e is revealed, are separated.

Next, we construct a stochastic policy π'_1 for I' such that in some cases, when e is revealed unblocked in a belief state reached in π_1, then π'_1 acts as if e is still blocked. Then we show that $C(\pi'_1) = C(\pi^*)$. As π'^* is an optimal policy for I', then $C(\pi'^*) \leq C(\pi'_1)$, which concludes the proof.

We now construct π'_1. For every belief state $b \in I'$ in which e is revealed to be unblocked we do as follows. Let

$$x = \frac{p(e) - p'(e)}{1 - p'(e)}$$

With probability x, for every belief state b' reachable from b in π^*, let $\pi'_1(b') = \pi^*(g_e(f(b')))$; and with probability $1 - x$, for every b' reachable from b in π^* let $\pi'_1(b') = \pi^*(f(b'))$. Finally, let $\pi'_1(b') = \pi^*(f(b'))$ for all other belief states.

Now assume that e is revealed from belief state $b' \in B_I$ to a belief state b. As b' itself is not reachable from a belief state in which e is revealed, we have $\pi'_1(b') = \pi^*(f(b'))$. Let z, z' be OR-nodes in $T_{\pi^*}, T_{\pi'_1}$, respectively, such that $L_{T_{\pi'_1}}(z') = b$, and $L_{T_{\pi^*}}(z) = f(b)$. We show that $V^{\pi'_1}(z') = V^{\pi^*}(z)$. Note that if $b_1 \in B_I$ is separated from b, then for every nodes $u' \in T_{\pi'_1}$ with $L(u') = b_1$, and $u \in T_{\pi^*}$ with $L(u) = f(b_1)$, we have $V^{\pi'_1}(u') = V^{\pi^*}(u)$. Therefore, by using the same inductive method as in Claim 2.3.1, we have that $V^{\pi'_1}(k') = V^{\pi^*}(k)$ for every node k' ancestor of z', and node k ancestor of z, thus the proof is complete.

To see that $V^{\pi'_1}(z') = V^{\pi^*}(z)$, denote the action $\pi'_1(b')$ by a (where a is the action $Move(e_1)$ for some edge e_1). Denote the set of observations received when performing a in b' by O'. Denote the grandchild of z for which $L(z^o_a) = b^o_a$ by z^o_a. Then,

$$V^{\pi^*}(z) = w(e_1) + \sum_{o \in O'} (p(b'|a, o) V^{\pi'}(z^o_a))$$

Note that as e is revealed by b' in π, then in any observation $o \in O'$, e is revealed to be either blocked or unblocked in b'^o_a. Therefore we divide

O into O^1: the observations in which e is revealed blocked, and O'^2: the observations in which e is revealed to be unblocked. Note that there is a bijection $h : O'^1 \to O'^2$ such that for every edge $e' \neq e$, the status of e' in o and $h(o)$ is identical. Let $B(O'^1) = \{b'^o_a | o \in O^1\}$, and $B(O'^2) = \{b'^o_a | o \in O^2\}$. Hence for every $o \in O^1$, we have $g_e(b'^o_a) = b'^{h(o)}_a$.

Therefore for every $b \in B(O'^1)$, such that $b = b'^o_a$, and $p(b'|a, o) = q(o)p(e)$ for some probability $q(o)$, we have $p(g_e(b)|a, h(o)) = q(o)(1 - p(e))$. Then we can write

$$V^{\pi^*}(z) = w(e_1) + \sum_{o \in O'^1} (q(o)p(e)V^{\pi'}(z^o_a)) + \sum_{o \in O'^2} (q(o)(1 - p(e))V^{\pi'}(z^o_a))$$

To calculate $V^{\pi'_1}(z')$ we repeat the same analysis. As $\pi'_1(f(b')) = a$ as well, we have that after performing a in $f(b')$, the same set of observations O'^1 is received. Since $p(e_1) = p'(e_1)$ for every $e_1 \neq e$, then $p(f(b')|a, o) = q(o)p'(e)$ for every $o \in O'^1$, where $p(b'|a, o) = q(o)p(e)$. Therefore we have

$$V^{\pi'_1}(z') = w((u, v)) + x \sum_{o \in O'^1} (q(o)p'(e)V^{\pi'}(z'^o_a)) + (1-x) \sum_{o \in O'^2} (q(o)(1-p'(e))V^{\pi'}(z'^o_a))$$

However, as $xp'(e) = p(e)$, and $(1 - x)(1 - p'(e)) = 1 - p(e)$, we have $V^{\pi'_1}(z') = V^{\pi^*}(z)$ as required.

$\square$

Appendix D

D.1 Policy simulation

We give a formal definition of policy simulation as informally defined in Section 4.2.1.

Let M be a Det-POMTP with a set of actions A, and let M' be a Det-POMDP with a set of actions A'. Let $g : A \to A'$ and $h : B_M \to B_{M'}$ be two functions, and let $B_M^{in}, B_M^{out} \subseteq B_M$ be two sets of belief states in B_M. Let π be a policy for M. Then a policy π' for M' is an $(h, g, B_M^{in}, B_M^{out})$-simulation of a policy π for M, if we have $g(\pi(b)) = \pi'(h(b))$ for every belief state b between B_M^{in} and B_M^{out} (see Figure 3.3). Throughout this work, the set of actions of M and the set of actions of M' are the same, and g is the identity function (see Figure 3.3). When $h, g, B_M^{in}, B_M^{out}$ are obvious from the context, we say π' is a simulation of π, and π simulates π'.

Now, let $I = (V, E, s, t, p, w)$ be a CTP instance, and let $I' = (G', s', t')$, where $G' = (V', E')$, be a CTP sub-instance of I. For a belief state $b \in B_I$, such that $Loc(b) \in V'$, we denote by $b \restriction E'$, the belief state $b' \in B_{I'}$ in which $Loc(b) = Loc(b')$ and $b|e = b'|e$ for every edge $e \in E'$. Let h be a partial function from B_I to $B_{I'}$, defined in all belief states b with $Loc(b) \in V'$, such that $h(b) = b \restriction E'$. Note that h is onto. Let id be the identity function over the set of actions $\{move(e) | e \in E'\}$. Let $F_{I,I'}$ be a function from the set of I'-committing policies for I, to the set of policies for I', such that $F_{I,I'}(\pi)$ is the $\big(h, id, B^{in}(G', \pi), B^{out}(G', \pi)\big)$-simulation of π. Note that as I' is a sub-instance of I, then $F_{I,I'}$ is onto, and that if π' is a

$$\begin{tikzcd}
B_M \arrow[r, "h"] \arrow[d, "\pi"'] & B_{M'} \arrow[d, "\pi'"] \\
A \arrow[r, "g"'] & A'
\end{tikzcd}$$

Figure 3.3: Policy simulation where g is a general function from A to A'.

$$\begin{tikzcd}
B_M \arrow[rr, "h"] \arrow[dr, "\pi"'] & & B_{M'} \arrow[dl, "\pi'"] \\
& A &
\end{tikzcd}$$

Figure 3.3: Policy simulation where A and A' are identical and g is the identity function.

$\big(h, id, B^{in}(G', \pi), B^{out}(G', \pi)\big)$-simulation of π, then $C(trunc(\pi_b, B_b^{out}(G', \pi))) = C(\pi')$. π' is then called the *contraction* of π to I.

D.2 The sub-graph exclusive lemma

Let $G = (V, E)$ be a graph and let $G' = (V', E')$ be a subgraph of G'. Let E'-exclusive be the following constraint: "the edges of E' cannot be traversed". Let $M = (G, s, t, p, w)$ and $\tilde{M} = (G, s, t, \tilde{p}, w)$ be E'-exclusive CTP instances such that $p(e) = \tilde{p}(e)$. Then we have the following lemma .

Lemma D.2.1 *Let π^* be an optimal E'-exclusive policy for M, and let $\tilde{\pi^*}$ be an optimal E'-exclusive policy for $\tilde{M}$. Then $C(\pi^*) = C(\tilde{\pi^*})$.*

Proof: If the edges of E' cannot be traversed in M or in $\tilde{M}$, they might as well be blocked. Therefore if $\tilde{M} = (G, s, t, \bar{p}, w)$ is a CTP instance such that $\tilde{p}(e) = p(e')$ for every $e \in E \backslash E'$, and $\bar{p}(e) = 1$ for every edge $e \in E'$, then we have that $C(\pi^*) = C(\chi^*)$ where χ^* is an optimal policy for $\tilde{M}$. For the same argument we have that $C(\tilde{\pi^*}) = C(\chi^*)$. Therefore $C(\pi^*) = C(\tilde{\pi^*})$ as required.

□

D.3 Generalization of the partition framework

We generalize the partition framework to subgraphs with mutual vertices. Let $U(G) \subseteq V$ be the set of all vertices in V that are explored in b_0. Let $I' = (G', s', t')$ be a sub-instance of I where G' is connected to the rest of the graph only through s', t' and vertices of $U(G)$. In order to define I'-committing policy, we need to re-define the entry-point and the departure-point of G'.

An *explored edge* in a belief state b is an edge that is known in b.

A *meaningful tour* in G' can be either of the following

- a sequence of moves in G' along explored edges in b_0, which starts at s' and ends in t'.
- a conditional sequence of move actions, all in G', in which a status of an edge in E' is revealed with probability 1.

We say that an OR-node $z \in T_\pi$, where $L(z) = b$, is an *entry point* (in T_π) of G', if $\pi(b)$ is the first action in a meaningful tour in G', and either $b = b_0$ or $PrevAction_\pi(z)$ is outside G'. We say an AND-node z, with $L(z) = b$, is a *departure point* of G' (in T_π), if $PrevAction_\pi(b)$ is an action in a revealing tour in G', a single observation is received at b, and $NextAction_\pi(b)$ is outside G'.

Definition D.3.1 *A policy π is I'-committing (or (G', s', t')-committing) if the following hold:*

- *the only entry points of G' in T_π are of $Z^{in}(G', \pi)$.*
- *the only departure points z of G' in T_π are of $Z^{out}(G', \pi)$.*
- *any move actions in G' at any belief state reachable from $Z^{out}(G', \pi)$ are along explored edges.*

See that Lemma 4.2.2 still holds for this generalization.

Now let (G_1, s', t'), (G_2, s', t') be sub-graphs where each G_i is connected to the rest of the graph through s', t' and vertices of $U(G)$. Therefore we still have that when traversing G_1, no edges of G_2 are revealed; and when traversing G_2, no edges of G_1 are revealed. Hence Lemma 4.2.3 still holds as well.

Finally, we define the following generalization of a CTP partition.

Definition D.3.2 *A CTP instance $I = (V, E, s, t, p, w)$ (where $G = (V, E)$) is a partition of CTP-instances $((G_1, s', t), \cdots (G_k, s', t))$ of M, where $G_i = (V^i, E_i)$ if:*

- *$\bigcup_{i<k} V^i = V$ and $\bigcup_{i \leq k} E_i = E$.*
- *$V^i \cap V^j \subseteq (U(G) \cup \{s', t\})$*

I is called a $\{G_1, \cdots, G_k\}$-CTP partition .

Then Corollary 4.2.6 still holds for this generalization as well.

Appendix E

Lemma E.0.3 *Let T be a tree. Then P_T, the probability that T is blocked, can be recursively found in polynomial time.*

Proof: Let r be the root of T. If l is a leaf then $p(l,t)=0$. Let $u_1, \cdots u_l$ be the children of a vertex u then

$$P_{T(u)} = \prod_{i \leq l} P(T^{Par}(u_i)) \tag{E.1}$$

where for every $i \leq l$

$$P_{T^{Par}(u_i)} = p((u,u_i)) + (1 - p((u,u_i)))P_{T(u_i)} \tag{E.2}$$

Therefore P_T, which is the same as $P_{T(r)}$ can be found in polynomial time.

$\square$

Lemma E.0.4 $D^{Par}(v') < D^{T'}$

Proof: Let T^1 be a CTP-Tree obtained from T as follows. For every edge e that is not contained in a subtree $T^{Par}(u')$, where u' is of height $h-1$, the cost of e in T^1 is reduced to 0. As T and T^1 have the same graph layout, the belief states of B_T and B_{T^1} have the same variables-status representation. Therefore for every belief state $b \in B_T$, there is a unique belief state $b^1 \in B_{T^1}$ such that b^1 has the same status-variables representation as b (see Section 2.4 for exact definition of the status-variables representation). Therefore π_b

can be executed in B_{T^1} as well, and specifically $Travel(T'^1)$, for the analogue subgraph T'^1 of T^1, can be performed in b^1.

Let X be the random variable that describes the expected cost of $Travel(T'^1)$ performed in b^1 . Then $C^{T'^1} = E[X]$. Let $u_1^0, \cdots u_{z_0}^0$ be the vertices of T'^1 of height $h-1$, in which $(Parent(u_i^0), u_i^0)$ is known in b^1 to be unblocked, and let X_i^0 be the random variable that describes the cost of traversing $T^{Par}(u_i^0)$ (as a part of the macro action $Travel(T'^1)$). Likewise, let $u_1^1, \cdots u_{z_1}^1$ be the vertices of T'^1 of height $h-1$, in which $(Parent(u_i^1), u_i^1)$ is not known in b^1 to be unblocked , and let X_i^1 be the random variable that describes the cost of traversing $T^{Par}(u_i^1)$ (again, as a part of the macro action $Travel(T'^1)$). As the only edges of cost greater than 0 are in edges of the $T^{Par}(u_i^k)$, we have that $X = \sum_{i<z_0} X_i^0 + \sum_{i<z_1} X_i^1$, and therefore $E[X] = \sum_{i<z_0} E[X_i^0] + \sum_{i<z_1} E[X_i^1]$. For $k \in \{0,1\}$, denote by Q_i^k the probability that $T^{Par}(u_i^k)$ is entered. As u_i^k is of height $h-1$, then by the induction assumption, π_b is $T^{Par}(u_i^k)$-committing, and therefore $E[X_i^k] = Q_i^k C(T^{Par}(u_i^k))$. However, as for every $k \in \{0,1\}$, all the $T^{Par}(u_i^k)$ are identical, we have that $C(T^{Par}(u_i^k)) = C(T^{Par}(u_1^k))$, and $P_{T^{Par}(u_i^k)} = P_{T^{Par}(u_1^k)}$ for every $i \leq z_k$.

Therefore we have

$$C^{T'^1} = \sum_{i<z_0} Q_i^0 C(T^{Par}(u_1^0)) + \sum_{i<z_1} Q_i^1 C(T^{Par}(u_1^1))$$

Now, as the probability that t is reached through a leaf of $T^{Par}(u_i^k)$, assuming $T^{Par}(u^k)$ is entered, is $1 - P_{T^{Par}(u_1^k)}$, we have that

$$1 - P^{T'^1} = \sum_{i<z_0} Q_i^0 (1 - P_{T^{Par}(u_1^0)}) + \sum_{i<z_1} Q_i^1 (1 - P_{T^{Par}(u_1^1)})$$

and as $\frac{C(T^{Par}(u_1^0))}{1-P_{T^{Par}(u_1^0)}} = \frac{C(T^{Par}(u_1^1))}{1-P_{T^{Par}(u_1^1)}} = D^{Par}(v')$, we have that

$$D^{T'^1} = \frac{C^{T'^1}}{1 - P^{T'^1}} = \frac{\sum_{i<z_0} Q_i^0 C(T^{Par}(u_1^0)) + \sum_{i<z_1} Q_i^1 C(T^{Par}(u_1^1))}{\sum_{i<z_0} Q_i^0 (1 - P_{T^{Par}(u_1^0)}) + \sum_{i<z_1} Q_i^1 (1 - P_{T^{Par}(u_1^1)})} = D^{Par}(v') \tag{E.3}$$

Now, as in T we have $w((Parent(v), v)) > 0$, then we have $C^{T'^1} < C^{T'}$. As $P^{T'} = P^{T'^1}$, we have that $D^{T'^1} < D^{T'}$, meaning $D^{Par}(v') < D^{T'}$, as required.

□

Lemma E.0.5 *For a vertex $v \in T$ that is not the root, denote by $Q^0_{T^{Par}(v)}$ the probability that $T^{Par}(v)$ is unblocked given $(Parent(v), v)$ is unblocked. Denote by $C^0(T^{Par}(v))$ the expected cost of $TRY(T^{Par}(v))$ given $(Parent(v), v)$ is unblocked. Let $D^0(T^{Par}(v)) = \frac{C^0(T^{Par}(v))}{Q^0_{T^{Par}(v)}}$. Then*

$$D^0(T^{Par}(v)) = D(T^{Par}(v))$$

Proof: Follows straight from the definitions. As

$$C^0(T^{Par}(v)) = w((Parent(v), v)) + C(T(v)) + P_{T(v)}w((Parent(v), v))$$

and

$$Q^0_{T^{Par}(v)} = 1 - P_{T(v)}$$

and we have

$$C(T^{Par}(v)) = (1 - p((Parent(v), v)))C^0(T^{Par}(v))$$

and

$$P_{T^{Par}(v)} = (1 - p((Parent(v), v)))P_{T(v)}$$

□

Table of Notations

Symbol		Page
E_v	the edges incident on v	5
$Parent(v)$	the parent of v	5
$\preceq_T$	partial order on the tree T	6
$depth(v)$	depth of vertex v	6
$height(v)$	height of vertex v	6
$Depth(T)$	the depth of T	6
$Height(T)$	the height of T	6
$Rank(v)$	the rank of T	6
$T(v)$	the subtree of T with root v	6
$T^{Par}(v)$	$T(v)$ with $Parent(v)$, and $(Parent(v), v)$	6
$\restriction$	restriction of a function	6
$T(s, a, s')$	transition function	8
$R(s, a, s')$	reward function	8
$V^{\pi}(s)$	value of π at state s	10
$C(\pi)$	the cost of π	10
$O(s, a, o)$	observation function	10
b_0	the initial belief states	11
B_M	the set of belief states in M	11
$p(o\|a, b)$		11
b_a^o		11
b_a		12
$sup(b)$	the set of states i in which $b(i) > 0$	13
A_b	actions available at belief state b	13
$B_M(b)$	belief states reachable from b	13
$B_M(b, \pi)$	belief states reachable in π from b	13
V_{AND}	AND vertices	13
V_{OR}	OR vertices	13
E_{OR}	OR arcs	13
E_{AND}	AND arcs	14
L	a label function from nodes in T to belief states	14

$trunc(\pi_b, B)$	a partial truncated policy	52
$w(e)$	cost of edge e	6
$p(e)$	probability of e to be blocked	6
$q(e)$	probability of e to be unblocked	6
Y_e	edge status function	16
loc	vertices location function	16
$Loc(b)$	location at belief state b	17
$B\|e$	status of edge e in belief state b	18
$Nex(b)$	explored neighborhood of b	21
$\partial(b)$	the fringe of b	21
$Traverse(v)$	macro action $Traverse$	21
$W_{i,j}$	cost of path I_i until the j'th edge	23
$TRY(i)$	macro-action TRY for a path I_i	24
$INV(i,j)$	macro-action INV for a path I_i and an edge $e_{i,j}$	24
Q_i	the probability that I_i is unblocked	23
$BC(i)$	backtracking cost of I_i	24
$U(G)$	all vertices v with E_v unblocked	129
$PrevAction_\pi$	previous action in π	50
$NextAction_\pi$	next action in π	50
$Z^{in}(G', \pi)$		50
$Z^{succ}(G', \pi)$		50
$Z^{fail}(G', \pi)$		50
$Z^{out}(G', \pi)$		50
$B^{in}(G', \pi)$		50
$B^{succ}(G', \pi)$		50
$B^{fail}(G', \pi)$		50
$B^{out}(G', \pi)$		50
$B_b^{out}(G', \pi)$		51
$TRY(I')$	macro action TRY for a sub-instance I'	55
$Travel(T)$	macro action $Travel$	73
TRY_n	macro action TRY for n agents	87

Index

Bibliography

[1] Eric A.Hansen. Indefinite-Horizon POMDPs with Action-Based Termination. In *AAAI*, pages 1237–1242, 2007.

[2] Giovanni Andreatta and Luciano Romeo. Stochastic shortest paths with recourse. *Networks*, 18(3):193–204, 1988.

[3] Amotz Bar-Noy and Baruch Schieber. The Canadian Traveller Problem. In *SODA*, pages 261–270, 1991.

[4] Richard Ernest Bellman. *Dynamic Programming.* Dover Publications, Incorporated, 2003.

[5] Daniel Berend, Ronen Brafman, Shimon Cohen, Solomon E. Shimony, and Shira Zucker. Optimal ordering of independent tests with precedence constraints. *Discrete Applied Mathematics, to appear.*

[6] Z. Bnaya, A. Felner, and S.E. Shimony. Canadian Traveler Problem with Remote Sensing. *IJCAI*, pages 437–442, 2009.

[7] Zahy Bnaya, Ariel Felner, Dror Fried, Olga Maksin, and Solomon Eyal Shimony. Repeated-task canadian traveler problem. In *SOCS*, 2011.

[8] Blai Bonet. Deterministic pomdps revisited. In *UAI*, pages 59–66, 2009.

[9] Amy J. Briggs, Carrick Detweiler, Daniel Scharstein, and Alexander Vandenberg-Rodes. Expected shortest paths for landmark-based robot navigation. *I. J. Robotic Res.*, 23(7-8):717–728, 2004.

[10] Yuval Emek, Amos Korman, and Yuval Shavitt. Approximating the statistics of various properties in randomly weighted graphs. In *SODA*, pages 1455–1467, 2011.

[11] Lars Engebretsen and Marek Karpinski. TSP with bounded metrics. *J. Comput. Syst. Sci.*, 72(4):509–546, 2006.

[12] P. Eyerich, T. Keller, and M. Helmert. High-Quality Policies for the Canadian Traveler's Problem. In *In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 51–58, 2010.

[13] A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu. PhA*: Finding the shortest path with A* in unknown physical environments. *Journal of Artificial Intelligence Research*, 21:631–679, 2004.

[14] Dror Fried, Solomon Eyal Shimony, Amit Benbassat, and Cenny Wenner. Complexity of Canadian traveler problem variants. *Theor. Comput. Sci.*, 487:1–16, 2013.

[15] L. Fu, D. Sun, and L. R. Rilett. Heuristic shortest path algorithms for transportation applications: State of the art. *Computers & OR*, 33(11):3324–3343, 2006.

[16] M.R. Garey and D.S. Johnson. *Computers and Intractability, problem number LO11.* W.H. Freeman, 1979.

[17] Russell Greiner, Ryan Hayward, Magdalena Jankowska, and Michael Molloy. Finding optimal satisficing strategies for and-or trees. *Artif. Intell.*, 170(1):19–58, 2006.

[18] Ming Hua and Jian Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *EDBT*, pages 347–358, 2010.

[19] Masoumeh T. Izadi and Doina Precup. Using Rewards for Belief State Updates in Partially Observable Markov Decision Processes. In *ECML*, pages 593–600, 2005.

[20] Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. In *STOC*, pages 356–365, 2005.

[21] Dominik Klein, Frank G. Radmacher, and Wolfgang Thomas. Moving in a network under random failures: A complexity analysis. *Sci. Comput. Program.*, 77(7-8):940–954, 2012.

[22] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML*, pages 282–293, 2006.

[23] Sven Koenig, Craig Tovey, and Yuri Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147(12):253 – 279, 2003. Planning with Uncertainty and Incomplete Information.

[24] Maxim Likhachev and Anthony Stentz. PPCP: Efficient Probabilistic Planning with Clear Preferences in Partially-Known Environments. In *AAAI*, pages 860–867, 2006.

[25] Maxim Likhachev and Anthony Stentz. Probabilistic planning with clear preferences on missing information. *Artif. Intell.*, 173(5-6):696–721, 2009.

[26] M. Littman. *Algorithms for Sequnetial Decision Making.* PhD thesis, Brown University, 1996.

[27] Olga Maksin. Heuristics for sensing centered canadian traveler problem. *Master Thesis, Ben-Gurion University of the Negev*, 2011.

[28] Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. Online graph exploration: New results on old and new algorithms. In *ICALP (2)*, pages 478–489, 2011.

[29] Ram Meshulam, Ariel Felner, and Sarit Kraus. Utility-based multi-agent system for performing repeated navigation tasks. In *AAMAS*, pages 887–894, 2005.

[30] Evdokia Nikolova and David R Karger. Route Planning under Uncertainty: The Canadian Traveller Problem. *AAAI*, pages 969–974, 2008.

[31] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest Paths Without a Map. *Theor. Comput. Sci.*, 84(1):127–150, 1991.

[32] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

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

[34] J. Scott Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003.

[35] Samitha Samaranayake, Sebastien Blandin, and Alexandre M. Bayen. Speedup techniques for the stochastic on-time arrival problem. In *ATMOS*, pages 83–96, 2012.

[36] S.Russel and P.Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1994.

[37] Stamatis Stefanakos. Reliable routings in networks with generalized link failure events. *IEEE/ACM Trans. Netw.*, 16(6):1331–1339, 2008.

[38] Florent Teichteil-Königsbuch. Stochastic safest and shortest path problems. In *AAAI*, pages 744–749, 2012.

[39] Tonguç Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004.

[40] Tonguç Ünlüyurt. Testing systems of identical components. *J. Comb. Optim.*, 10(3):261–282, 2005.

[41] Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[42] S. Travis Waller and Athanasios K. Ziliaskopoulos. On the online shortest path problem with limited arc cost dependencies. *Networks*, 40(4):216–227, 2002.

[43] Cenny Wenner. Hardness results for the shortest path problem under partial observability. *Diploma Thesis, Lund University*, 2009.

[44] Stephan Westphal. A note on the k-Canadian Traveller Problem. *Inf. Process. Lett.*, 106(3):87–89, 2008.

[45] Yinfeng Xu, Maolin Hu, Bing Su, Binhai Zhu, and Zhijun Zhu. The canadian traveller problem and its competitive analysis. *J. Comb. Optim.*, 18(2):195–205, 2009.

[46] Doron Zarchy. Theoretical and experimental analysis of the canadian traveler problem. *Master Thesis, Ben-Gurion University of the Negev*, 2010.

במסלולים [6], גרף הבעיה הינו מושתת על עץ. אנו מציגים את *אילוץ החיוב לקודקוד*, בו הסוכן "מחויב" לחקור בשלמות תת-עץ ששורשו הוא קודקוד זה. באמצעות סכימת החלוקה אנו מציגים פתרון אופטימלי יעיל לבעיה בה הסוכן מחויב לכל קודקוד בגרף, וכן פתרון אופטימלי יעיל לבעיה בה הסוכן מחויב לכל קודקוד, מלבד לאחד שצלעותיו היוצאות הינן ידועות. בנוסף לכך אנו מציגים פתרון אופטימלי יעיל לווריאנט של CTP-Tree הקרוי *בעיית הנוסע הקנדי לעצים שווי-מחיר-משוקלל* (EFC-CTP-Tree), בו המחיר-המשוקלל לכל תתי העצים באותו גובה הינו שווה. לאחר מכן, אנו בודקים אמפירית באיזו מידה פתרון זה מהווה קירוב לפתרון אופטימלי לבעיית הנוסע הקנדי לעצים.

לבסוף, מאחר ובהרבה בעיות אמיתיות מדובר ביותר מסוכן אחד, אנו מציגים וריאנט של בעיית הנוסע הקנדי עם מספר נוסעים (Repeated-CTP). בווריאנט זה ישנם מספר סוכנים, וכל סוכן מתחיל לנוע רק לאחר שקודמו הגיע למטרה. לדוגמה, בעל צי של משאיות השולח מספר משאיות, אחת אחרי השניה, אל המטרה. בעל הצי מעוניין למקסם את המחיר כולו, במקום שכל משאית בנפרד תמקסם את המחיר עבור עצמה. בעבודה זו אנו נותנים פתרון אופטימלי יעיל לבעיה זו על גרפים זרים במסלולים (disjoint-path graphs). עבודה זו פורסמה ב [7].

מילות מפתח: בעיית הנוסע הקנדי, ניווט תחת אי-וודאות, בעיית מציאת המסלול האקראי הקצר ביותר- כולל חזרה.

תקציר

בעיות חיפוש הן חלק מקבוצת הבעיות הבסיסיות הנחקרות במדעי המחשב בכלל, ובבינה מלאכותית בפרט. עם זאת, פתרונות לרוב בעיות החיפוש הקלאסיות אינן ישימות בעולם האמיתי. נבחן לדוגמה את בעיית הניווט הבאה, הנגזרת מן העולם האמיתי. ייתכן כי מתכנן המסלול (Planner) יכיר את מפת האזור ואת עלות השימוש בכבישים, בין אם העלות מוגדרת כאורך הכביש או כזמן הנדרש לעבור בו ; ובכל זאת, בידי המתכנן מידע חלקי בלבד הנוגע למצב הכבישים הנוכחי, שכן אין באפשרותו לדעת אם כביש מסוים פתוח או חסום, עד אשר הגיע לנקודות ציון הסמוכות לכביש. מכאן נוכל להסיק כי אלגוריתמים שייעודם מציאת המסלול הקצר ביותר בגרף, ייתכן ויימצאו כבלתי-יעילים למציאת מסלול מועדף. הבעיה שהמתכנן יהיה מעוניין בפתרונה היא מציאת תוכנית מעבר מנקודת מקור לנקודת יעד, שתוחלת עלותה מינימלית.

בעיית הנוסע הקנדי (The Canadian Traveler Problem) היא מודל טבעי לבעיה זו : בהינתן גרף G וסוכן המוצב בקודקוד s, על הסוכן להגיע לנקודת היעד t באמצעות מעבר בחלק מקשתות הגרף. בידי הסוכן מידע מלא על מבנה הגרף ומחיר הקשתות, אולם ייתכן כי חלק מן הקשתות חסומות, כל אחת בהסתברות מסוימת נתונה מראש. הסוכן מגלה כי קשת חסומה רק כאשר הוא מגיע לקדקוד הסמוך לאותה הקשת. הבעיה בפניה אנו עומדים היא מציאת אסטרטגיה שתשמש את הסוכן עבור חישוב מסלול מ s ל t, במינימום תוחלת המחיר אותו יידרש לשלם.

מחקר זה מנתח היבטים תאורטיים של בעיית הנוסע הקנדי, וכן מנתח וריאנטים שונים שלה. מידול הבעיה כתהליך החלטה מארקובי הניתן לצפייה חלקית (Partially Observable Markov Decision Process) מקנה לנו את היכולת לאפיין ולנתח בצורה מדויקת אסטרטגיות שונות לפתרון הבעיה ועל-ידי כך לתכנן אסטרטגיות אופטימליות הפותרות את הבעיה.

בעיית הנוסע הקנדי הוגדרה לראשונה כבעיית משחק מול יריב [31]. בגרסה זו נמצאה הבעיה כ PSPACE-שלמה. הגרסה האקראית, אותה אנו סוקרים בעבודה זו, נמצאה על-ידי [31] כשייכת ל PSPACE, אך רק כ #P-קשה. בעבודה זו אנו סוגרים את פער הסיבוכיות שנוצר, ומראים שגם הגרסה האקראית הינה PSPACE-שלמה : תחילה עבור גרסה מסוימת של בעיית הנוסע הקנדי, המתירה תלות בין צלעות (CTP-Dep), ולאחר מכן, עבור הבעיה המקורית. תוצאה זו, שפורסמה ב [14], פותרת בעיה שהייתה פתוחה במשך כשני דורות.

גישה ידועה לפתרון בעיה מסוימת במדעי המחשב היא פירוק הבעיה לתתי-בעיות, ומציאת פתרון אופטימלי לבעיה כולה על-ידי מציאת פתרון אופטימלי לכל תת-בעיה. בעבודה זו אנו אכן מציעים פירוק לבעיית הנוסע הקנדי לתתי-בעיות שונות. מאחר ולגרף כללי, פתרון אופטימלי לתת בעיה של בעיית הנוסע הקנדי, לא בהכרח ייתן פתרון אופטימלי לבעיה כולה, אנו מציגים תנאים, או אילוצים מסוימים בהם נקבל את מבוקשנו. זאת על-ידי הגדרת וריאנט בו כל פתרון הינו "מחויב" לפתור תת-בעיה מסוימת על מנת לפתור את הבעיה הכללית. באמצעות שיטה זו והגדרת *המחיר-המשוקלל* של תת בעיה, אנו מציגים את *סכימת החלוקה* המציעה דרך (לא תמיד יעילה) לחלק את בעיית הנוסע הקנדי לתתי בעיות, כך שבאמצעות פתרון אופטימלי לכל תת-בעיה ניתן יהיה לפתור בצורה אופטימלית וביעילות את הבעיה בכללותה.

וריאנט נוסף של בעיית הנוסע הקנדי, אותו אנו מציגים בעבודה זו, הינו בעיית הנוסע הקנדי לעצים (CTP-Tree). בווריאנט זה, המהווה הכללה של בעיית הנוסע הקנדי לגרפים זרים-

<u>הצהרת תלמיד המחקר עם הגשת עבודת הדוקטור לשיפוט</u>

אני החתום מטה מצהיר/ה בזאת: (אנא סמן):

☑ חיברתי את חיבורי בעצמי, להוציא עזרת ההדרכה שקיבלתי מאת מנחה/ים.

☑ החומר המדעי הנכלל בעבודה זו הינו פרי מחקרי <u>מתקופת היותי תלמיד/ת מחקר</u>.

___ בעבודה נכלל חומר מחקרי שהוא פרי שיתוף עם אחרים, למעט עזרה טכנית הנהוגה בעבודה ניסיונית. לפי כך מצורפת בזאת הצהרה על תרומתי ותרומת שותפי למחקר, שאושרה על ידם ומוגשת בהסכמתם.

תאריך: 27.5.2014 שם התלמיד: דרור פריד חתימה: [illegible]

העבודה נעשתה בהדרכת

פרופסור אייל שמעוני

במחלקה למדעי מחשב
בפקולטה למדעי הטבע

נושאים תיאורטיים בבעיית הנוסע הקנדי המוכללת

מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

דרור פריד

הוגש לסינאט אוניברסיטת בן גוריון בנגב

אישור המנחה [illegible] 28/5/2014

אישור דיקן בית הספר ללימודי מחקר מתקדמים ע"ש קרייטמן ____________________

כ"א באב, תשע"ג 28.7.2013

באר שבע

<u>נושאים תיאורטיים בבעיית הנוסע הקנדי המוכללת</u>

מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

דרור פריד

הוגש לסינאט אוניברסיטת בן גוריון בנגב

כ"א באב, תשע"ג **28.7.2013**

באר שבע