5-2023

Efficient Routing for Disaster Scenarios in Uncertain Networks: A Computational Study of Adaptive Algorithms for the Stochastic Canadian Traveler Problem with Multiple Agents and Destinations

Neel Chanchad
University of Arkansas, Fayetteville

Efficient Routing for Disaster Scenarios in Uncertain Networks: A Computational Study of Adaptive Algorithms for the Stochastic Canadian Traveler Problem with Multiple Agents and Destinations

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering, with a concentration in Industrial Engineering

by

Neel Chanchad
Mumbai University
Bachelor of Technology in Mechanical Engineering, 2018
University of Arkansas
Master of Science in Industrial Engineering, 2023

May 2023
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Ashlea Bennett Milburn, Ph.D.
Dissertation Director

Haitao Liao, Ph.D.
Committee Member

Chase Rainwater, Ph.D.
Committee Member

Abstract

The primary objective of this research is to develop adaptive online algorithms for solving the Canadian Traveler Problem (CTP), which is a well-studied problem in the literature that has important applications in disaster scenarios. To this end, we propose two novel approaches, namely Maximum Likely Node (MLN) and Maximum Likely Path (MLP), to address the single-agent single-destination variant of the CTP. Our computational experiments demonstrate that the MLN and MLP algorithms together achieve new best-known solutions for 10,715 instances. In the context of disaster scenarios, the CTP can be extended to the multiple-agent multiple-destination variant, which we refer to as MAD-CTP. We propose two approaches, namely MAD-OMT and MAD-HOP, to solve this variant. We evaluate the performance of these algorithms on Delaunay and Euclidean graphs of varying sizes, ranging from 20 nodes with 49 edges to 500 nodes with 1500 edges. Our results demonstrate that MAD-HOP outperforms MAD-OMT by a considerable margin, achieving a replan time of under 9 seconds for all instances. Furthermore, we extend the existing state-of-the-art algorithm, UCT, which was previously shown by Eyerich et al. (2010) to be effective for solving the single-source single-destination variant of the CTP, to address the MAD-CTP problem. We compare the performance of UCT and MAD-HOP on a range of instances, and our results indicate that MAD-HOP offers better performance than UCT on most instances. In addition, UCT exhibited a very high replan time of around 10 minutes. The inferior results of UCT may be attributed to the number of rollouts used in the experiments but increasing the number of rollouts did not conclusively demonstrate whether UCT could outperform MAD-HOP. This may be due to the benefits obtained from using multiple agents, as MAD-HOP appears to benefit to a greater extent than UCT when information is shared among agents.

Acknowledgments

I am grateful to the divine powers that have bestowed upon me the strength and courage to pursue a doctorate degree. Furthermore, I extend my appreciation to my esteemed advisor, Dr. Milburn, whose unwavering support and guidance were instrumental in my academic success. Dr. Milburn has a unique ability to understand and empathize with her students' needs and aspirations, which has contributed to my personal and professional growth.

I also wish to express my gratitude to Dr. Liao, whose invaluable advice and mentorship aided me in making important decisions during the first year of my doctoral program. Dr. Liao's enthusiasm and dedication to teaching statistics have contributed to my understanding of the subject. In addition, I would like to thank Dr. Rainwater, a member of my committee, whose constructive feedback helped me in my research work.

I extend my gratitude to the diligent INEG staff members, Mrs. Carrie Pennington, Mrs. Ashley Reeves, Mrs. Tamara Ellenbecker, and Mrs. Sandy Sehon, for their unfailing assistance in the documentation process. My friend Madhur is like a brother to me and has been a constant pillar of support throughout my academic journey, and for that, I am truly grateful.

Lastly, I would like to acknowledge my dear friends, including Jakhon, Kushal, Nayan, Nirosh, Sushant, Yogesh bhai, Ashish bhai, Ravi, Shivani, Bhargavi, Sneha, Shiva, Shilpi, Raghu, Varma, Sudharashan, Daniel, Heiu, Susan, Amisha, Dhiraj, Tola, Siddesh, Jigesh, Kaushik, Aaqib, Niloy, Rahul, Abhijeet, Prashant, Nitin, Amogh, Vikas, and many others who have accompanied me on this academic journey.

Finally, I would like to express my gratitude to Suresh Dani and his team, who laid the foundation for my academic success by imparting knowledge and wisdom in the fields of mathematics, science, and chemistry during my high school years.

Dedication

I would like to dedicate my work to my family, including my mother Hansa Chanchad, father Valjibhai Chanchad, sister Mitali Adatiya, and brother-in-law Rahul Adatiya.

Contents

List of Tables

List of Figures

1. Introduction

1.1 Introduction

Each year disasters cause millions of dollars in restoration of damaged infrastructure (NCEI, 2023). Further, millions of people suffer every year during disasters without access to food, water, and other essential supplies and services (FEMA, 2020). In the immediate disaster response phase, there is an urgent need for critical supplies to be delivered to critical locations. These critical locations can include temporary shelters such as schools, hospitals, or public buildings (FEMA, 2020). The task of delivering these critical supplies from a central staging area to critical locations is challenging for a number of reasons, including that the transportation infrastructure has often sustained damages.

In a large scale domestic disaster response effort, these critical supplies are typically located at a Federal Staging Area (FSA), which is defined as a base located near an area impacted by a disaster, from which logistical support for the disaster response operation is provided. Truck carrying loads of critical supplies begin driving from the FSA, following a planned path to their destination. While moving along this path the trucks may discover a blocked road or roads in their path. When that happens, an alternate plan which requires rerouting the vehicle must be determined. Because time is critical in the immediate disaster response phase, it is important for the initially planned path to be reliable. However, in the immediate disaster response phase, information regarding where roads are blocked is not always available. Instead, the planner relies on the probability of blockages to develop a reliable path to the destination, both during initial planning at the FSA, and during any replanning required during the truck's movement towards their destination

The problem of finding the shortest path between a source and a destination given an uncertain road network is called a Canadian Traveler Problem (CTP). In CTP variants, the status of a road as either available or blocked is only revealed when an agent reaches one of its endpoints. All chapters in this dissertation focuses on the Stochastic CTP, in which edge blockage probabil-

ities are assumed to be available. Formally a graph $G = (N, E)$ with node set N and edge set E is known to the agent, along with the associated edge cost c_e and blockage probability p_e for each edge $e \in E$. A realization of an instance is called a weather, denoted w, which contains only those edges that are available to travel. The problem is for an agent to traverse G from source $v_s \in N$ to destination $v_d \in N$, using only those edges that are available in w. Weather w is not revealed to the agent initially but is dynamically made available as the agent observes road statuses while traversing the graph. In other words, a blocked edge ($e \in E \setminus w$) is revealed to the agent upon reaching a node incident to it. The objective of the agent is to minimize the cost of the path traveled to reach v_d from v_s.

Chapter 2 of this work is concerned with the computational aspects of the single source single destination single agent (SS-SD-SA) variant of the CTP. While the theoretical aspects of this problem have been studied in several papers, only two studies have explored the computational aspects. One such study proposes four policies, three of which are based on rollouts (Eyerich et al., 2010). The other proposes a rollout-based approach known as HOP-EF(l), designed primarily for the case of $l > 1$ agents traveling to the same destination (Alseth, 2020). This chapter proposes two new policies, Maximum Likely Node (MLN) and Maximum Likely Path (MLP), which are also based on rollouts, but employ a consensus function to identify effective agent movements across a range of scenarios. Our proposed algorithms are easy to implement and understand and provide superior results for a significant number of instances. These contributions add to the existing literature on computational approaches to the SS-SD-SA Stochastic CTP variant.

During a disaster, more than one critical location may need to be addressed, requiring essential life-sustaining resources. The Federal Emergency Management Agency (FEMA) uses three models for distributing essential supplies during disasters, namely the hub-and-spoke model, fixed location model, and cross-docking model, which can all be mathematically modeled as single source multiple destination multiple agents (SS-MD-MA) variants of the Stochastic CTP FEMA (2020). Therefore, chapter 3 focuses on this more practical variant, which we refer

to as MAD-CTP (Multiple Agent Destination CTP). This variant has only been studied once in the literature by Lita et al. (2001), with computational details and experiments that are challenging to reproduce and use for comparison. In this paper, we propose a framework for MAD-CTP and introduce two different approaches, MAD-OMT and MAD-HOP. Our proposed framework provides a starting point for future research on this variant. Additionally, we analyze the benefits of allowing communication between agents to improve the travel cost of the agents.

The UCT algorithm, an acronym for the upper confidence bounds applied to trees, has gained popularity as a leading algorithm for various applications in the literature. In the context of the SS-SD-SA CTP variant, Eyerich et al. (2010) research reveals that the rollout-based UCT policy outperforms other policies tested in a computational study. Accordingly, we expand the scope of the UCT policy to address the MAD-CTP problem and compare the resulting performance with MAD-HOP, which we proposed in chapter 3. Unlike MAD-HOP, which only replans when a blocked edge is detected along the agent's path, UCT replans at each unvisited node. This can cause delays because the agent must pause at each node to determine the next node to visit. The time spent at a node to make this decision is assessed to determine the total delay, thereby analyzing the benefits of reduced travel costs against the drawbacks of increased waiting time.

In chapter 2 of our study, we present two innovative algorithms for addressing the SS-SA-SD problem. In chapter 3, we introduce two novel algorithms that address the MAD-CTP, MAD-OMT, and MAD-HOP problems. Furthermore, in chapter 4, we extend the existing state-of-the-art algorithm UCT to tackle the MAD-CTP problem and compare it with MAD-HOP.

Bibliography

Alseth, A. (2020). *-CTP: Utilizing Multiple Agents to Find Efficient Routes in Disrupted Networks*. University of Arkansas.

Eyerich, P., Keller, T., and Helmert, M. (2010). High-quality policies for the canadian traveler's problem. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

FEMA (2020). FEMA Preliminary Damage Assessment Guide. Technical report.

Lita, L. V., Schulte, J., and Thrun, S. (2001). A system for multi-agent coordination in uncertain environments. In *Proceedings of the fifth international conference on Autonomous agents*, pages 21–22.

NCEI (2023). Billion-Dollar Weather and Climate Disasters.

2. Single Source Single Agent Single Destination CTP

2.1 Introduction

It is well-established that the frequency and economic consequences of disasters have drastically escalated over the past several decades. According to data from the Institute for Economics and Peace (IEP), there has been a tenfold increase in the annual number of disasters from 1960 to 2019 (IEP). The United Nations Office for Disaster Risk Reduction (UNDRR) anticipates that the average number of disasters occurring worldwide will reach 1.5 per day by 2030 (UNDRR, 2022). The National Centers for Environmental Information (NCEI) has documented over 300 billion-dollar disasters in the United States since 1980, resulting in a cumulative cost of over $2.275 trillion in damages (NCEI). This indicates that disasters frequently result in extensive damage to physical buildings such as homes and facilities as well as disruptions to critical infrastructures such as road networks, power grids, and water systems (FEMA, 2020). These disruptions often leave impacted populations in need of basic life-sustaining support such as food, water, and shelter; also referred to as *community lifelines* (FEMA, 2020).

Logistics plays a vital role in providing *community lifelines* to survivors, minimizing suffering and loss of life attributed to disaster events (FEMA, 2019). In the United States, logistical support for a federally-declared disaster response operation is coordinated from a Federal Staging Area (FSA). A disaster qualifies for a FSA when it is beyond the capabilities of local and state agencies to effectively respond and recover from the disaster (FEMA). This could include situations such as major hurricanes, earthquakes, wildfires, and other large-scale emergencies (FEMA). Disaster response resources, including food, water, search and rescue personnel, and trucks and trailers equipped with supplies are positioned at the FSA, ready for deployment to areas affected by disasters (FEMA, 2022). To support impacted populations, truckloads of emergency response supplies are dispatched from the FSA to critical destinations (FEMA, 2022). These destinations may include temporary shelters, hospitals, and local food and water distribution locations (FEMA, 2022).

Delivering emergency supplies within a region impacted by a disaster may be complicated by disrupted road networks. Factors such as collapsed bridges, water accumulation on roads, debris from landslides, fallen trees, and damaged buildings can make travel difficult or impossible (FEMA, 2020). A truck and driver dispatched from a FSA to deliver a truckload of supplies will typically follow a predetermined path to their destination. However, it is possible that they may encounter unexpected blockages along the way due to road network disruptions. These blockages may only become apparent to the driver once they are in sight. In some cases, the driver may be able to repair or clear the blocked road and continue along their original path, such as a fire engine team responding to a person in distress. In other cases, the driver may need to alter their route and find an alternative path to their destination, leaving the repair work to other resource types. This paper will consider the latter scenario in which the only option available to the driver when encountering a blockage is to plan a different route.

This paper investigates a path planning problem under uncertainty where any road in a network may have a blocked or available status, and further, that status may be known or unknown to the agent at a particular point in time. It is assumed that probabilistic information regarding road blockages is available, as various techniques capable of providing such information in advance are being developed. For example, techniques capable of predicting road blockages due to earthquakes include the remote sensing monitor method (which uses aerial/satellite images) (Santarelli et al., 2018), the omnidirectional buffer-debris model (Toma-Danila, 2018), probabilistic and statistical models (Moya et al., 2020; Santarelli et al., 2018), and geometrical models (Argyroudis et al., 2005). For flooding events, an example technique for predicting road blockages is available in Yuan et al. (2021). Therefore, the focus of this paper is on developing algorithms that do not rely on the exact road status and instead use probabilistic information about road failure, recognizing that this is an area of ongoing research and development.

The problem of finding the shortest path between a source and a destination given an uncertain road network is referred to as the Canadian Traveler Problem (CTP). In CTP variants, the status of a road as either available or blocked is only revealed when an agent reaches one of its

endpoints. This paper focuses on a single agent stochastic CTP variant in which the edge blockage probabilities are available and known to the agent. Specifically, the agent has knowledge of a graph $G = (N, E)$ with a set of nodes N and a set of edges E, as well as the associated edge cost c_e and blockage probability p_e for each edge $e \in E$. Initially, the agent does not have full knowledge of the weather $w \subseteq E$, defined as the set of edges that are available for travel in the given instance. The problem is for the agent to traverse G from source v_s to destination v_d, using only those edges $e \in w$. Knowledge of weather w becomes dynamically available as the agent observes road statuses while traversing the graph. In other words, whether an edge e is available ($e \in w$) or blocked ($e \in E \setminus w$) is revealed to the agent upon reaching a node incident to it. The objective of the agent is to minimize the cost of the path from v_s to v_d.

There are various policies and algorithms available in literature for solving variants of CTP. These algorithms differ in how they utilize the availability status of the roads gathered during their traversal for planning. The algorithm designed in this paper is inspired by the scenario-based planning technique used in Bent and Van Hentenryck (2004). Originally introduced for a dynamic and stochastic vehicle routing problem variant, the algorithm in Bent and Van Hentenryck (2004) generates scenarios by sampling from problem-specific stochastic distributions. Then, solutions (a set of routes, in this case) are developed for each scenario and a distinguished solution is selected from among the scenario solutions to guide the vehicle's next movement(s) along the physical network. The distinguished solution is selected via a consensus function that looks across scenario solutions to identify common solution attributes (i.e., customer visit sequences). In this paper, a scenario solution is an agent path rather than a set of routes. Two consensus functions for choosing a distinguished plan for the agent to follow are explored; these are referred to as Maximum Likely Node (MLN) and Maximum Likely Path (MLP).

Scenario-based planning approaches MLN and MLP are compared in a computational study against the best-known algorithms for single agent stochastic CTP, appearing in Eyerich et al. (2010) and Alseth *et al.* (2020). The computational study consists of 30,000 weathers across 30 Delaunay graphs (Eyerich et al., 2010). On a per-weather basis, MLN and MLP found

new best-known solutions for 10,243 and 10,185 weathers respectively, compared with solutions from Alseth (2020). A similar comparison against the results presented in Eyerich et al. (2010) was not performed due to absence of weather-level data. On a per-graph basis and compared against both Alseth (2020) and Eyerich et al. (2010), approaches MLN and MLP were able to find new average best costs for two and one of these graphs respectively, but these differences were not statistically significant. This article furnishes all graphs and weathers utilized in the experiment to facilitate future researchers in evaluating the efficacy of their novel policies and algorithms.

The structure of the remainder of this paper is as follows: In Section 2.2 we present a survey of pertinent literature. In Section 2.3 we propose our methodology for addressing the problem at hand. The experimental details and results are described in Section 2.4 and Section 4.4 respectively. Finally, we offer concluding remarks in Section 4.5.

2.2 Literature Review

The original CTP as defined by Papadimitriou and Yannakakis (1991) involves a network structure that is known to the agent, but the statuses of the edges in that network are subject to uncertainty and no probabilistic information is available for them. Papadimitriou and Yannakakis (1991) proved that devising a travel strategy for CTP is PSPACE-complete. After its initial introduction, additional variants of CTP were introduced in the literature. These include, for example, k-CTP, where only k network edges are blocked, and stochastic CTP, where probabilities of edge blockage are assumed to be available. Variants of CTP with multiple agents simultaneously traversing the graph have also been introduced, leading us to organize this review into single-agent CTP variants in Section 2.2.1 and multi-agent CTP variants in Section 2.2.2. Because the focus of this paper is on a single-agent variant, the review provided in Section 2.2.1 is more detailed than that in Section 2.2.2. Some additional problems related to CTP are reviewed in Section 2.2.2 as well.

2.2.1 Single-agent CTP Variants

The k-CTP, first introduced in Bar-Noy and Schieber (1991), is the most studied variant of CTP. In this variant, there is a single agent to traverse the graph, at most k edges may be obstructed, and probabilistic information about these blockages is not available. Like the original CTP, k-CTP is also PSPACE-complete (Bar-Noy and Schieber, 1991). In Nikolova and Karger (2008), two heuristic algorithms for solving instances of k-CTP with up to 50 vertices are presented. The better of the two heuristic algorithms has a $\Omega(log|N|)$ gap from the optimal policy. In Westphal (2008), a lower bound of $2k+1$ on competitive ratios for deterministic online algorithms for k-CTP is proven. Further, a simple and deterministic BACKTRACK algorithm that achieves a competitive ratio of $2k+1$ is presented, showing that the lower bound on k-CTP deterministic algorithm competitive ratios is tight. In BACKTRACK, the agent follows a planned path unless a blocked edge in the path is encountered; at which point the agent returns to the source node and plans a new path. In Xu et al. (2009) two deterministic adaptive strategies for k-CTP are proposed: (1) a greedy strategy in which the agent follows the shortest path to the destination from their current location avoiding any blocked edges observed at the current location; and (ii) a comparison strategy in which greedy and BACKTRACK are combined. In the comparison strategy, the agent revisits the choice of whether to follow the greedy strategy or BACKTRACK each time a blockage is encountered. From a current node in which a blockage is discovered, greedy is selected if the cost from the current node to the destination under the greedy strategy is less than or equal to the optimal cost to reach the destination from the source node. Otherwise, BACKTRACK is selected. The competitive ratios for greedy and comparison are $2^{(k+1)}-1$ and $2k+1$, respectively (Xu et al., 2009).

In Demaine et al. (2014), it is shown that a randomized algorithm for k-CTP can outperform the best deterministic algorithms, surpassing the $2k+1$ deterministic competitive ratio lower bound by an $O(1)$ factor. A randomized algorithm with a competitive ratio of $(1+\frac{\sqrt{2}}{2})k+1$ having pseudo-polynomial runtime is presented (Demaine et al., 2014). The lower bound on compet-

itive ratios for randomized algorithms is $k+1$ and is tight (Westphal, 2008; Bender and Westphal, 2015).

Bergé and Salaün (2019) proves that a competitive ratio lower than $2k+1$ is possible to achieve by exploiting small maximum (s,t) – cuts. The DETOUR algorithm proposed in Bergé and Salaün (2019) achieves a competitive ratio of $2\mu_{max} + \sqrt{2}(k-\mu_{max}) + 1$, where μ_{max} is the maximum (s,t) cut size. This competitive ratio is possible to achieve only when $\mu_{max} < k$; otherwise, DETOUR achieves a competitive ratio of $2k+1$. Shiri and Salman (2019b) shows that the randomized backtrack strategy (RBS) proposed in Bender and Westphal (2015) for node-disjoint paths cannot be implemented in some cases and thus is not optimal. Further, Shiri and Salman (2019b) presents a modification of RBS that is optimal for all problem instances on graphs that contain only node-disjoint paths . In Demaine et al. (2021), the theoretical analysis presented in Demaine et al. (2014) is revisited and it is shown that a randomized algorithm can achieve a competitive ratio of $(1+\frac{\sqrt{2}}{2})k+\sqrt{2}$ in pseudo-polynomial time when there are at least two blockages.

In the stochastic CTP, a single agent is assumed to have access to probabilistic information about edge failures (Eyerich et al., 2010; Fried et al., 2013; Alseth, 2020). In Fried et al. (2013) the stochastic CTP is shown to be PSPACE-complete. The focus in Eyerich et al. (2010) is on developing path-finding policies that minimize expected travel cost; four online deterministic policies for stochastic CTP are proposed. The first of the four policies is optimistic in both name and nature (Optimistic (OPT)), in that it assumes a perfect graph, finds the shortest path in that graph, and traverses the planned path until a discovered blockage necessitates planning a new path. The remaining three policies are Hindsight Optimization (HOP), Optimistic Rollout (ORO) and Upper Confidence applied to Trees (UCT). All three use rollout weathers sampled from edge blockage probability distributions to develop paths, similar to the scenario-based planning strategy motivating our algorithm. For example, according to HOP, the next node to visit is the node where the cost to reach it from the current node plus the average shortage path cost from that node to the destination across all rollouts is minimized. Motivated by HOP in particular, a policy named A*-HOP is proposed in Alseth (2020). A*-HOP uses path cost estimations from HOP as the heuristic

function in the A* framework. While A*-HOP outperforms HOP in the computational study in Alseth (2020), it does not outperform UCT.

Sahin and Aksakalli (2015) performs a comparison of penalty and rollout-based algorithms for the stochastic CTP. Four rollout-based algorithms from Eyerich et al. (2010) are compared against a penalty-based algorithm called Distance-to-Termination (DT). The computational results suggest DT ran faster while also providing better results. Thus, they concluded that DT is a better candidate for solving the Stochastic TSP than rollout-based algorithms. Bai et al. (2018) focuses on posterior sampling for Monte Carlo planning under uncertainty and proposes two algorithms for Monte Carlo planning in Markov decision processes (MDPs) and partially observable Markov decision processes (POMPDs). One of the algorithms named Dirichlet-NormalGamma based Monte Carlo tree search (DNG-MCTS) is implemented on the 20-node Delaunay graph CTP instances taken from Eyerich et al. (2010). The results indicate that while DNG-MCTS produces better solutions than some UCT-based approaches, it is outperformed by UCTO (Bai et al., 2018). Because finding an exact solution approach for the Stochastic CTP is difficult, Aksakalli et al. (2016) focuses on the Stochastic k-CTP and proposes an AO* based exact algorithm called CAO*. This algorithm is shown to be optimal but does not have polynomial run time, making it unsuitable for large problem instances.

A number of additional CTP variants that assume probabilistic information regarding edge failures is available have been introduced. In the Bayesian CTP, the statuses of edges are assumed to be correlated with prior known probabilities. Lim et al. (2017) presents a polynomial-time randomized approximation algorithm for the Bayesian CTP called Hedged Shortest Path under Determinization (HSPD). This algorithm is compared against a widely used optimistic algorithm and a UCT-based algorithm developed for Bayesian CTP by modifying the UCT policy from Eyerich et al. (2010). Experimental results indicate HSPD outperforms both these algorithms on Bayesian CTP variants. In the Robust CTP, travel cost variability resulting from a policy is conisdered. An approximate online algorithm for Robust CTP is proposed in Guo and Barfoot (2019). It balances the mean and variation of travel cost when computing an agent's travel policy.

Bar-Noy and Schieber (1991) introduces three additional variants of CTP: the Recoverable CTP, a stochastic variant of Recoverable CTP, and the k-Vital Edges Problem. The Recoverable CTP is a variant of the generic CTP in which a blocked road, once encountered by the agent, may become traversable if the agent waits at the location for a specified amount of time. In the stochastic variant of the Recoverable CTP, the probability of edge failure is taken into account. The k-Vital Edges Problem (k-VEP) is the dual of the generic CTP. It involves identifying certain edges that if blocked would result in the highest travel cost between the source and the destination. Bar-Noy and Schieber (1991) presents a travel strategy for the Recoverable CTP and the stochastic variant of Recoverable CTP and proves that k-VEP is NP-hard. Su et al. (2008) provides a competitive ratio lower bound of $\frac{(3-\beta)}{2}$ where $\beta = max\ \beta_i$, $\beta_i = \frac{t_{i,1}}{t_{i,2}}$, and $t_{i,1}$ and $t_{i,2}$ are the first and second shortest travel times from the current node i to node 1 and 2 respectively on the special network. This bound is shown to be tight by using a comparison strategy. For a risk seeking traveler, Su et al. (2008) provides a risk-reward strategy $\hat{A}$ and analyzes its competitive ratio. Bnaya et al. (2009) introduces an additional variant of the CTP, called CTP-sensing in which the agent can sense blockages before encountering them but this comes at a sensing cost. Two heuristic policies that minimize both travel and sensing cost are presented (Bnaya et al., 2009).

Yildirim et al. (2019) and Alkaya et al. (2021) both study the CTP with Neutralization which is a result of combining the CTP with the Obstacle Neutralization Problem (ONP). In the CTP with Neutralization, the agent can clear the blocked edge using their limited neutralization capacity. Yildirim et al. (2019) is the first to introduce this problem and proposes a Markov Decision Process (MDP) formulation. They also provide an optimal algorithm based on AO* search algorithm called CAON* and compare its performance with well-known value iteration and AO* algorithms by testing it on Delaunay graphs. Alkaya et al. (2021) proposes a new heuristic to solve large instances and computational experiments showed the heuristic had extremely small run times of less than a second in all cases. The expected path length was also observed to be improved by 58%.

2.2.2 Multi-agent CTP

The multi-agent CTP was introduced in Lita et al. (2001). It involves a group of agents starting from a single source and traveling to a single destination. The objective is to have at least one agent reach the destination as quickly as possible. There has been significant interest in the theoretical complexity of this problem (Zhang et al., 2013; Bnaya et al., 2015; Zhang et al., 2015; Shiri and Salman, 2017; Bergé et al., 2019; Shiri and Salman, 2019a, 2020), as well as in the development of online algorithms to solve it (Bnaya et al., 2015; Shiri and Salman, 2020; Akbari and Shiri, 2022; Shiri and Salman, 2020; Alseth, 2020). While some research has focused on variants of the problem that involve a maximum of k failed edges (Zhang et al., 2013, 2015; Shiri and Salman, 2017; Bergé et al., 2019; Shiri and Salman, 2019a), others have considered the problem on specific types of graphs (Bnaya et al., 2015). Additionally, one study has examined the case of multiple destinations in which the goal is to reach these destinations as soon as possible (Lita et al., 2001).

2.2.3 Related Problems to CTP

Motivated by the destruction disasters caused to bridges, Blei and Kaelbling (1999) introduces two problems similar to CTP, named the static and the dynamic bridge problems (BP). The static BP can be thought of as a stochastic CTP in which all edges are associated with edge failure probabilities. In the static BP, the status of the bridge once observed remains unchanged, while in the dynamic BP, as the name suggests, the status may change even after it has been observed. The dynamic BP can be considered as a dynamic stochastic CTP. The main difference between CTP and BP lies in their motivations for addressing different situations. In Papadimitriou and Yannakakis (1991) the application is a snowfall event and thus roads are considered to be blocked, while in Blei and Kaelbling (1999) the scenario is a cluster of islands connected by bridges that are damanged by storms. Another similar problem to CTP studied in literature is the stochastic shortest path problem (SSPP), but the stochasticity pertains to the travel times rather than road status. Recent papers that have studied the SSPP include Randour et al. (2014), Shahabi

et al. (2015), Cheng et al. (2016), Trevizan et al. (2017), Zhang et al. (2018), and Ahmadi et al. (2018). Lastly, Huang and Liao (2012) presents a touring strategy for the metric Travelling Salesman Problem (TSP). This problem is similar to the generic TSP except that the salesman can encounter a blocked edge in the planned tour.

In this paper, we focus on the single agent stochastic CTP and propose two deterministic online heuristics that aim to minimize the expected travel cost, similar to the Eyerich et al. (2010) and Alseth (2020).

2.3 Solution Approach

In a CTP instance, an agent travels through a sequence of nodes in order to reach its destination. At each node, the agent undergoes a cycle that consists of three stages: sensing, planning, and moving. During the sensing stage, the agent determines the status of its adjacent edges. The planning stage involves the agent determining the next node to visit. In the moving stage, the agent begins traversing the edge to reach the planned node. These three stages are described in greater detail in the following section.

2.3.1 Sensing

The process of sensing, or gathering information about the state of the edges in the network only occurs when the agent visits a node for the first time. This is because revisiting a node does not yield new information as the edge availability remains static. When an agent gathers new information about the status of its adjacent edges, it is said to be in a "belief state". Consequently, a belief state is defined as a node where the agent discovers new information. As the agent gathers information about the status of its adjacent edges, they are either classified as "available" or "failed" and added to sets E_A or E_F, respectively. This sensing operation does not incur any cost.

2.3.2 Planning

At every new belief state, planning is performed. To decide the next belief state to visit, a scenario-based planning method is used in which r rollout weathers are generated by sampling from the edge failure probability distributions for every edge in the graph. The set of available edges in each rollout weather $j \in \{1,2,3,\ldots,r\}$ is denoted R_j. While the real weather is the actual realization of the true scenario, rollout weathers are sampled scenarios from all $2^{|E|}$ possibilities. As the agent traverses the network, some rollout weathers may become inconsistent with the real weather the agent has observed. In other words, some edges that the agent has discovered as blocked may be available in some of the rollout weathers and vice versa. Thus, to have consistent rollout weathers, each j is updated to $(R_j \cup E_A)/E_F$. After these updates, the shortest path σ_j from the current node to the destination using only available edges in R_j is determined for each j using Dijkstra's algorithm. The rollout path σ_j is represented as the sequence of nodes $\sigma_j = \langle b, k_1, k_2, ..., k_n, v_d \rangle$, where n is the number of intermediate nodes contained in the path between the current node b and destination node v_d. It is known that $(b,k_1) \in E_A$, as it is visible from b. Edges (k_i,k_{i+1}) for $i = 1 \ldots n-1$ and (k_n,v_d) are available in rollout weather j but may not be available in the true weather. Letting $c_{l,m}$ denote the cost on edge (l,m), the rollout path cost $C(\sigma_j)$ is calculated by summing the costs of all edges between consecutive nodes in σ_j as shown in Equation 2.1:

$$C(\sigma_j) = c_{b,k_1} + \sum_{i=1}^{n-1} c_{k_i,k_{i+1}} + c_{k_n,v_d}. \tag{2.1}$$

Let $\omega = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ be the set of all rollout paths. Paths in the set ω are not necessarily unique since a particular path may be optimal for multiple rollout weathers. To select the next node to visit, a consensus function that examines all rollout paths is defined. Two separate consensus functions are introduced in this study. For "maximum likely node" (MLN) described in Section 2.3.2.1, the consensus function is node string-based whereas for "maximum likely path" (MLP) described in Section 2.3.2.2, it is path-based. Here, the term node string is being used to

refer to a node sequence (for a subset of nodes) embedded in a path. Both consensus functions consist of two parts, with the first part focused on node string *frequency* (F) and the second part focused on path *cost* (C). To balance these two components, weight parameters W_F and W_C are introduced for frequency and cost, respectively.

The frequency component of both consensus functions identifies strings of nodes appearing in rollout paths, beginning with a new belief state and continuing for τ additional nodes. Specifically, in any rollout j, the next belief state b_j is the first node in the sequence σ_j that has not yet been visited by the agent. It is guaranteed to be reachable from the current node b along a path that uses only edges known to be available (i.e., in the set E_A). The string of nodes of interest in a rollout path, denoted T_j, begins at b_j and continues through the next τ nodes. For example, if $b_j = k_1$ in $\sigma_j =< b, k_1, k_2, \ldots, k_n, v_d >$ and $\tau = 2$, then $T_j =< k_1, k_2, k_3 >$. The set of all such strings across all rollout paths is denoted $T = \cup_{j=1}^{r} T_j$. Just as the set of rollout paths ω does not necessarily contain r unique paths, the set T does not necessarily contain r unique strings (i.e., $|T| \leq r$). For each $i \in T$, a frequency F_i is computed. It is defined as the number of rollout paths containing string i. That is, $F_i = count(\sigma_j : T_j = i)$. Sections 2.3.2.1 and 2.3.2.2 outline how each consensus function uses F_i, and also how each function considers path cost.

2.3.2.1 Maximum Likely Node

The consensus function in MLN chooses a preferred string i^* from T and sets the next belief state (i.e., node) to visit as the first node in i^*. The frequency component of the consensus function uses F_i, computed as described in the previous section, directly. The frequency F_i is then normalized using Equation 3:

$$F_i^{'} = \frac{F_i - \min_{k \in T}\{F_k\}}{\max_{k \in T}\{F_k\} - \min_{k \in T}\{F_k\}}. \tag{2.2}$$

To associate a path cost with a string $i \in T$, path costs $C(\sigma_j)$ are averaged across all rollouts j for which $T_j = i$, as shown in Equation 2.3, and normalized according to Equation 2.4:

$$C_i = \frac{\sum_{\{j:T_j=i\}} C(\sigma_j)}{F_i}, \tag{2.3}$$

$$C_i^{'} = \frac{C_i^{'} - \min_{k\in T}\{C_k\}}{\max_{k\in T}\{C_k\} - \min_{k\in T}\{C_k\}}. \tag{2.4}$$

Finally, Equation 2.5 is used to combine F_i and C_i in the consensus function. The preferred string i^* is the string in T that maximizes $S^{MLN}(i)$, as shown in Equation 2.6:

$$S^{MLN}(i) = F_i^{'} W_F + (1 - C_i^{'}) W_C, \tag{2.5}$$

$$i^* = argmax_{i\in T}\ S^{MLN}(i). \tag{2.6}$$

2.3.2.2 Maximum Likely Path

In the maximum likely path approach, the consensus function value $S^{MLP}(j)$ is computed for all weathers j, in contrast to the maximum likely node approach, which computed $S^{MLN}(i)$ for all unique strings i. The consensus function chooses a preferred path σ^* from ω and sets the next belief state (i.e., node) to visit as b_j, the first unvisited node in σ^*. The frequency score F_j for rollout j is set to the frequency score of the string T_j; that is, $F_j = \{F_i : i = T_j\}$. It is normalized using Equation 2.7:

$$F_j^{'} = \frac{F_j - \min_{k=1,\ldots,r}\{F_k\}}{\max_{k=1,\ldots,r}\{F_k\} - \min_{k=1,\ldots,r}\{F_k\}}. \tag{2.7}$$

With the consensus function being defined on rollout weather j, there is no need to average path costs across a variety of paths. Instead, the cost C_j in $S^{MLP}(j)$ is simply the rollout path cost $C(\sigma_j)$ and is normalized using:

$$C_j^{'} = \frac{C_j - \min_{k=1,\ldots,r}\{C_k\}}{\max_{k=1,\ldots,r}\{C_k\} - \min_{k=1,\ldots,r}\{C_k\}}. \tag{2.8}$$

Finally, Equation 2.9 is used to combine F_j and C_j in the consensus function. The preferred path σ^* is the rollout path that maximizes $S^{MLP}(j)$, as shown in Equation 2.10:

$$S^{MLP}(j) = F_j^{'} W_F + (1 - C_j^{'}) W_C, \tag{2.9}$$

$$\sigma^* = argmax_{j=1,\ldots,r} S^{MLP}(j). \tag{2.10}$$

2.3.3 Moving

After the agent has determined the next node to visit using either MLN or MLP, it begins traversing the shortest path constructed from the known edges, moving from its current location to the next node. This cyclical process of sensing, planning, and movement continues until the agent reaches its final destination.

2.4 Experimental Details

The maximum likely node and maximum likely path approaches are tested on various graphs. Section 2.4.1 provides details on the graphs and the test instances used to measure the performance of these approaches. Section 2.4.2 describes the tuning process for parameters required in the computational experiments.

2.4.1 Delaunay Graphs

The graphs used in this computational experiment are from Eyerich et al. (2010) and were created using Delaunay triangulation. A total of 30 graphs from Eyerich et al. (2010) are used: ten each of graph size 20, 50 and 100 nodes. The blockage probabilities p_e and edge costs c_e are from uniform [0,1) and uniform [0,50] distributions, respectively. The source v_s in each graph is the lowest index node and the destination v_d is the highest index node. The number of weathers per graph is 1000. The preceding parameters are identical to those in Eyerich et al. (2010). As the

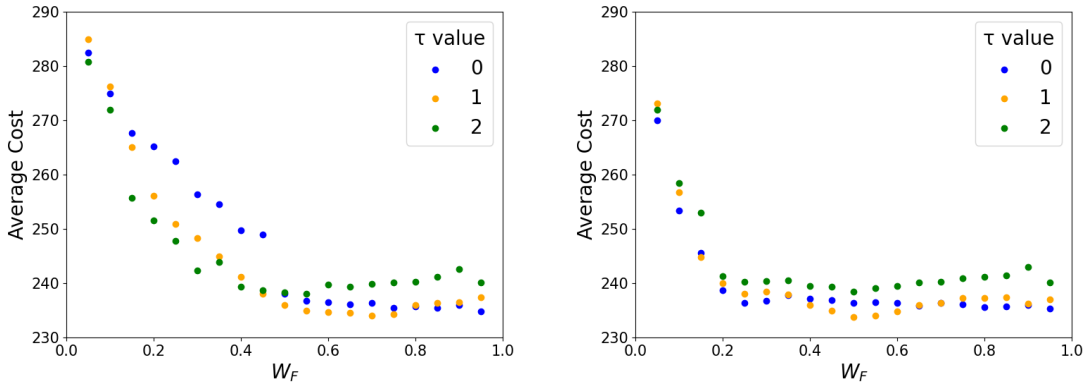
weathers in Eyerich et al. (2010) were not possible to obtain, they are instead taken from Alseth (2020). They were generated for each graph by sampling from a Bernoulli distribution with parameter p_e for each edge; an identical process to that described in Eyerich et al. (2010). The total number of test instances is 30,000. Hypothesis testing results are provided in Alseth (2020) to establish the 30,000 test instances in Alseth (2020) are not statistically different from the 30,000 test instances in Eyerich et al. (2010) at a level of significance of 0.05.

2.4.2 Parameter Tuning

The parameters requiring tuning are the number of rollout weathers, the weight parameters W_F and W_C for consensus function components, and $T = \tau + 1$, the number of nodes inspected in a sequence σ_j after belief state node b_j. Rather than tuning the number of rollouts directly, we adopt a result from Alseth (2020), which indicates 3000 rollouts are sufficient to achieve convergence in average travel cost across weathers. Thus, 3000 rollouts are used in our computational study.

To tune the frequency and cost weight parameters W_F and W_C in the consensus function, values from 0.05 to 0.95 at increments of 0.05 are tested. Parameters W_F and W_C are related by the relationship $W_F + W_C = 1$. For parameter τ, only values in $\{0,1,2\}$ are tested, as initial results indicated increasing τ beyond 2 would increase average travel cost. In total, 19*3 combinations of W_F, W_C, and τ are examined on a subset of Delaunay graphs and weathers. Specifically, for each graph size 20, 50, and 100 nodes, three graphs were included with 100 weathers each.

The results of the tuning experiments are depicted in Figure 2.1, where each data point provides the average solution cost for either MLN (Figure 2.1a) or MLP (2.1b) over 3*3*100 weathers for the (W_F, W_C, τ) combination indicated in the x-axis, y-axis, and legend. The combination providing the smallest average cost is preferred. For MLN, the minimum cost settings are W_F = 0.7, W_C = 0.3, and $\tau = 1$ with an average travel cost of 234.01 units. For MLP, the minimum cost settings are W_F = 0.5, W_C = 0.5, and τ = 1 with an average travel cost of 233.78 units.



(a) Tuning parameter MLN (b) Tuning parameter MLP

Figure 2.1: Parameter tuning MLN

2.4.3 Computational Experiments

The results obtained from using MLN and MLP approach are compared against OMT, HOP, ORO, UCTB, UCTO, and A*-HOP ((Eyerich et al., 2010; Alseth, 2020)) in the next section. Recall that in Section 2.2, an approach named Distance-to-Termination (DT) for solving the Stochastic CTP was shown to produce better solutions than UCTO (Sahin and Aksakalli, 2015). However, the computational experiments on which those results are based have different edge cost and blockage probability distributions than the test instances in this study. Further, there is limited ability to reproduce the results from Sahin and Aksakalli (2015) due to lack of information regarding the number of rollout weathers employed. Thus, the DT approach is not used as an additional reference approach in our computational study. Besides reporting the average cost at the graph level in Table 3.1 for all approaches mentioned in Eyerich et al. (2010) and Alseth (2020), weather-level comparisons against A*-HOP from Alseth (2020) are also reported in Tables 2.2 and 2.3 due to the availability of weather-level data.

2.5 Computational Results

Table 3.1 reports the average travel costs of solutions obtained from MLN, MLP, and comparison approaches from the literature, including OMT, HOP, ORO, UCTB and UCTO from Eye-

Table 2.1: Sample mean and 95% CI on travel cost from various policies on 20, 50, and 100 node graphs

Graph	OMT	HOP	ORO	UCTB	UCTO	A*-HOP	MLN	MLP
20-1	205.90 ± 7	171.60 ± 6	176.30 ± 5	210.70 ± 7	169.00 ± 6	**163.70 ± 5**	168.74 ± 5	168.53 ± 5
20-2	187.00 ± 5	155.80 ± 3	**150.30 ± 3**	176.40 ± 4	**148.90 ± 3**	**151.30 ± 4**	152.26 ± 3	153.01 ± 4
20-3	139.50 ± 6	138.70 ± 6	134.20 ± 6	150.70 ± 7	**132.50 ± 6**	135.20 ± 6	133.53 ± 6	133.55 ± 6
20-4	266.20 ± 8	286.80 ± 8	264.20 ± 7	264.80 ± 9	**235.20 ± 7**	**232.80 ± 7**	246.05 ± 7	246.41 ± 7
20-5	163.10 ± 7	**113.30 ± 5**	**113.00 ± 6**	123.20 ± 7	**111.30 ± 5**	**112.90 ± 6**	116.97 ± 6	117.59 ± 6
20-6	180.20 ± 6	142.00 ± 4	**134.40 ± 4**	165.40 ± 6	**133.10 ± 3**	**133.30 ± 4**	137.23 ± 4	139.09 ± 4
20-7	172.20 ± 5	150.20 ± 4	168.80 ± 4	191.60 ± 6	**148.20 ± 4**	**148.90 ± 5**	149.91 ± 5	150.13 ± 5
20-8	150.10 ± 6	**133.60 ± 5**	**137.70 ± 5**	160.10 ± 7	**134.50 ± 5**	**136.70 ± 5**	138.23 ± 5	149.66 ± 5
20-9	222.00 ± 5	**177.10 ± 4**	**176.40 ± 4**	235.20 ± 6	**173.90 ± 4**	**174.90 ± 4**	182.26 ± 4	182.38 ± 4
20-10	178.20 ± 6	188.10 ± 6	**166.30 ± 5**	180.80 ± 7	**167.00 ± 5**	174.90 ± 6	173.85 ± 6	173.92 ± 6
Avg	186.50 ± 2	165.70 ± 2	162.20 ± 2	185.90 ± 2	**154.20 ± 2**	**156.40 ± 2**	159.90 ± 2	161.43 ± 2
50-1	255.50 ± 10	250.60 ± 9	214.30 ± 7	229.40 ± 12	**186.10 ± 7**	215.10 ± 8	195.01 ± 8	195.06 ± 8
50-2	467.10 ± 11	375.40 ± 7	406.10 ± 8	918.00 ± 16	**365.50 ± 7**	378.50 ± 8	372.61 ± 8	371.95 ± 8
50-3	281.50 ± 9	294.50 ± 7	**268.50 ± 7**	382.10 ± 15	**255.60 ± 7**	**267.50 ± 8**	272.54 ± 8	271.43 ± 8
50-4	289.80 ± 9	263.90 ± 7	**241.60 ± 7**	296.60 ± 12	**230.50 ± 7**	**239.30 ± 7**	249.50 ± 8	256.33 ± 8
50-5	285.50 ± 10	239.50 ± 8	**229.50 ± 7**	290.80 ± 11	**225.40 ± 7**	230.00 ± 7	229.95 ± 7	231.25 ± 7
50-6	251.30 ± 10	253.20 ± 9	**238.30 ± 9**	405.20 ± 21	**236.30 ± 8**	240.00 ± 9	239.12 ± 9	238.35 ± 9
50-7	242.20 ± 9	221.90 ± 7	209.30 ± 7	250.50 ± 11	**206.30 ± 7**	211.50 ± 7	207.61 ± 7	209.61 ± 8
50-8	355.10 ± 11	**302.20 ± 9**	**300.40 ± 8**	462.60 ± 15	**277.60 ± 8**	**296.70 ± 9**	308.73 ± 9	309.35 ± 9
50-9	327.40 ± 13	281.80 ± 11	238.10 ± 9	295.20 ± 18	**222.50 ± 9**	241.00 ± 10	236.08 ± 9	250.06 ± 9
50-10	281.60 ± 8	271.20 ± 7	**249.00 ± 6**	390.80 ± 15	**240.80 ± 6**	259.90 ± 7	251.74 ± 7	249.18 ± 7
Avg	303.70 ± 3	275.40 ± 3	259.50 ± 3	392.10 ± 6	**244.70 ± 2**	257.90 ± 3	256.29 ± 3	258.26 ± 3
100-1	370.90 ± 11	319.30 ± 9	326.80 ± 9	464.50 ± 21	286.80 ± 7	284.10 ± 7	283.86 ± 8*	284.60 ± 8
100-2	160.60 ± 8	**154.50 ± 7**	**153.20 ± 7**	185.90 ± 12	**151.50 ± 7**	**156.00 ± 7**	157.40 ± 6	157.65 ± 6
100-3	550.20 ± 18	488.10 ± 15	451.30 ± 14	811.10 ± 39	**412.20 ± 13**	**423.80 ± 14**	424.41 ± 14	424.28 ± 14
100-4	420.10 ± 10	329.80 ± 7	348.70 ± 8	552.30 ± 20	**314.30 ± 7**	**322.90 ± 8**	324.07 ± 7	330.18 ± 7
100-5	397.00 ± 16	452.40 ± 18	**348.10 ± 13**	654.60 ± 43	**348.30 ± 13**	366.50 ± 14	360.21 ± 14	360.08 ± 14
100-6	455.00 ± 12	487.90 ± 11	**399.90 ± 10**	741.70 ± 29	**396.20 ± 9**	**413.70 ± 10**	415.12 ± 11	416.30 ± 11
100-7	431.40 ± 15	403.90 ± 14	**370.10 ± 12**	716.20 ± 39	**358.20 ± 12**	394.90 ± 14	394.29 ± 14	396.26 ± 14
100-8	335.60 ± 12	322.00 ± 12	295.70 ± 11	405.70 ± 25	293.30 ± 10	291.20 ± 10	285.87 ± 10*	289.45 ± 10*
100-9	327.50 ± 14	366.10 ± 15	**273.80 ± 11**	382.10 ± 27	**262.00 ± 10**	**272.70 ± 11**	289.86 ± 12	291.74 ± 12
100-10	381.50 ± 11	388.40 ± 11	**347.10 ± 9**	735.10 ± 32	**342.30 ± 9**	**350.10 ± 9**	355.30 ± 10	352.34 ± 9
Avg	383.00 ± 5	371.30 ± 4	331.50 ± 4	564.90 ± 10	**316.50 ± 3**	**327.60 ± 4**	329.04 ± 4	330.29 ± 4

rich et al. (2010), and A*-HOP from Alseth (2020). Thirty unique graphs are included in rows, with the first column indicating the graph name (graph size - graph number). The values in table cells represent the sample mean and 95% confidence interval for travel cost, averaged across 1000 weathers per graph. For example, the OMT approach achieves a sample mean travel cost of 205.90 units across 1000 weathers for graph 20-1, with a 95% CI of ± 7 units. In addition, there are three summary rows with the row header "Avg", one for each graph size, that reports sample mean and 95% confidence intervals for travel cost, averaged across 10,000 weathers. For example, the OMT approach achieves a sample mean travel cost of 186.50 units across 10,000 weathers for graphs of size 20. Bold text is used in columns OMT through A*-HOP to indicate

graphs for which the policy in that column produces solutions with lower sample mean travel cost than MLN and MLP. For example, for graph 20-1, A*-HOP is the only reference approach from the literature outperforming MLN and MLP, with a sample mean travel cost of 163.70 compared with sample means of 168.74 and 168.53 from MLN and MLP, respectively. The values marked with "*" in Table 3.1 indicate instances in which MLN and MLP outperform all other reference approaches. This occurs in graphs 100-1 and 100-8.

For 28 out of 30 graphs, at least one reference approach from the literature provides lower average travel cost solutions than MLN and MLP. The approach most frequently doing so is UCTO (27 graphs). Next are ORO (18 graphs) and A*-HOP (17 graphs). However, the confidence intervals of the best reference approach and our new approaches are non-overlapping in 25 out of 28 of these graphs, indicating that we cannot conclude whether the mean travel costs are significantly different at a level of significance of 0.05. The exceptions are graphs 50-8, 100-7, and 100-9, for which UCTO provides solutions with lower mean travel cost.

OMT is regarded as the simplest policy, being a deterministic approach, and is used as a base to compare against other policies in literature. As expected, MLN and MLP outperform OMT on all 30 graphs, by an average of 14.13% and 13.48% respectively. The HOP policy performs better than MLN and MLP in 5 out of 30 graphs; in the remaining 25 graphs, our approaches outperform HOP and improve travel cost by an average of 6.62%. ORO is a more realistic policy in terms of how rollout paths are determined (it does not assume perfect information about the rollout weather, as HOP does). In spite of this, MLN outperforms ORO in 8 of 30 graphs with an average travel cost improvement of 0.74%. MLN also outperforms UCTO in 3 out of 30 graphs (20-1, 100-1, 100-8), despite UCTO being a sophisticated technique tailored for application to CTP. Finally, we compare MLN with A*-HOP which is different than all other policies as it replans only when a blockage is discovered in the pre-determined path rather than replanning at every belief state. MLN achieves better average performance than A*-HOP in 13 of 30 graphs.

Because the travel cost for each of the 30,000 instances is available for A*-HOP, a weather

Table 2.2: Individual weather comparison

	A*-HOP vs MLN			A*-HOP vs MLP		
Graph	A*-HOP	MLN	Tie	A*-HOP	MLP	Tie
20-1	**127**	180	693	**129**	175	696
20-2	119	262	619	**144**	255	601
20-3	52	**80**	868	52	**79**	869
20-4	**249**	369	382	**239**	349	412
20-5	**81**	42	877	**84**	42	874
20-6	**238**	207	555	**339**	214	447
20-7	122	109	769	126	110	764
20-8	73	106	821	**414**	192	394
20-9	**343**	382	275	**368**	384	248
20-10	75	166	759	76	165	759
Total	**1479**	1903	6618	**1971**	1965	6064
50-1	398	**552**	50	395	**555**	50
50-2	316	**472**	212	307	**469**	224
50-3	366	386	248	342	375	283
50-4	**352**	390	258	**410**	384	206
50-5	144	245	611	145	232	623
50-6	149	217	634	152	219	629
50-7	305	**478**	217	311	480	209
50-8	**436**	444	120	**447**	434	119
50-9	183	**321**	496	**423**	321	256
50-10	367	**511**	122	347	**508**	145
Total	3016	**4016**	2968	**3279**	3977	2744
100-1	221	543	236	235	535	230
100-2	88	86	826	**95**	91	814
100-3	402	479	119	403	479	118
100-4	330	396	274	**372**	379	249
100-5	361	466	173	363	462	175
100-6	300	336	364	321	330	349
100-7	317	465	218	319	460	221
100-8	312	651	37	343	626	31
100-9	**400**	395	205	**405**	395	200
100-10	456	507	37	466	486	48
Total	**3187**	4324	2489	**3322**	4243	2435

level travel cost comparison is performed and presented in Table 2.2. Each row contains counts of the number of weathers, out of 1,000, in which the three approaches, or combinations of them, identified the best-known solution for the weather. The first column indicates graph name. The second through fourth columns provide the results of head-to-head comparisons between A*-HOP and MLN. We can see for example in column 2 that A*-HOP alone identifies the best-known solution for 127 weathers for graph 20-1, in column 3 that MLN alone identifies the best-known solution for 180 weathers for graph 20-1, and in column 4 that the two approaches tie on

693 weathers. Columns 5 through 7 provide an analogous head-to-head comparison between A*-HOP and MLP. Total rows are also provided for each graph size. To test whether travel cost differences between our approaches and $A*-HOP$ were significant at the individual weather level and with a level of significance of 0.05, we conduct paired t-tests. For MLN for example, we define the differences, $d_i = C_i^{MLN} - C_i^{A*-HOP}$ for weather i. Here, C_i^{π} represents the travel cost in weather i when policy π is implemented. The null hypothesis is $H_0 : \mu_d = 0$, where μ_d is the mean difference over 1000 weathers for each graph. Because we are interested in knowing which policy performs better, we use two alternative hypotheses; $H_1 : \mu_d > 0$ and $H_2 : \mu_d < 0$. Bold values in columns "A*-HOP" and "MLN" indicate H_0 is rejected in favor of H_1 and H_2 respectively. Besides performing a paired t-test for each graph, we also performed it over each graph size across 10,000 weathers (see "Total" row). MLN statistically outperformed A*-HOP in 6/30 graphs (20-3, 50-1, 50-2, 50-7, 50-9, and 50-10), whereas A*-HOP outperformed MLN in 8/30 graphs (20-1, 20-4, 20-5, 20-6, 20-9, 50-4, 50-8, and 100-9). In the remaining 16/30 graphs no statistical difference was observed. It is interesting to note that for graph size 20 and 100, A*-HOP is statistically superior but for graph size 50, MLN is statistically superior. Thus, it remains unclear how the two approaches would compare on larger graphs. On the other hand, the MLP results are not statistically significant against A*-HOP at any graph size when results are averaged across all graphs of each size.

Table 2.3 provides a 3-way comparison between A*-HOP, MLN, and MLP to analyze the number of weathers (out of 30,000) in which a new best known solution was found by at least one of the two new approaches. This is deduced by summing the columns MLN, MLP and MLN-MLP. For graph 20-1 for example, new best -known solutions were discovered for 6 weathers (MLN by itself) plus 1 weather (MLP by itself) plus 173 weathers (both MLN and MLP discovered it) for a total of 181 new best-known solutions for weathers for graph 20-1. For all graphs of size 20, 104+158+1788=2,050 new best-known solutions are discovered. For graphs of size 50 and 100, these totals are 4,199 and 4,246, respectively. Over 30,000 instances total, MLN and MLP combined identify new best-known solutions for 10,715 weathers. As the graph size

increases, the number of weathers in which all three policies (MLN, MLP, and A*-HOP) tie decreases, according to data presented in the last column of Table 3. Further, the number of weathers in which MLN and MLP have the same travel cost and outperform A*-HOP increases as the graph size increases which can be deduced from column MLN-MLP. There are only a few cases in which A*-HOP and MLN have a tie and outperform MLP and vice-versa as observed from column A*-HOP-MLN and A*-HOP-MLP.

Table 2.3: MLN, MLP, and A*-HOP comparison

	# of weathers in which an approach outperforms other two approaches			# of weathers in which there is a tie in travel cost for two approach and outperforms third approach			# of weathers in which all three approaches are tied
Graph	A*-HOP	MLN	MLP	MLN-MLP	A*-HOP-MLN	A*-HOP-MLP	A*-HOP-MLN-MLP
20-1	126	6	1	173	0	1	693
20-2	118	**11**	5	249	16	1	600
20-3	49	1	0	79	3	3	865
20-4	228	29	8	339	3	17	376
20-5	80	**2**	2	39	4	1	872
20-6	232	**36**	35	169	87	5	436
20-7	121	1	2	108	4	1	763
20-8	70	**10**	95	95	339	0	391
20-9	335	7	9	373	30	3	243
20-10	75	1	1	164	0	0	759
Total	1434	**104**	158	1788	486	32	5998
50-1	391	13	12	534	0	0	50
50-2	303	13	22	442	1	8	211
50-3	325	72	32	309	0	18	244
50-4	335	**53**	68	300	40	4	200
50-5	138	**23**	4	220	5	5	605
50-6	139	11	8	204	9	4	625
50-7	291	**47**	19	424	10	4	205
50-8	427	34	25	396	3	1	114
50-9	174	**116**	88	186	185	2	249
50-10	339	24	**23**	477	0	15	122
Total	2862	**406**	301	3492	253	61	2625
100-1	212	29	25	499	8	2	225
100-2	85	3	13	78	7	1	813
100-3	393	26	25	437	3	3	113
100-4	297	**98**	57	273	31	6	238
100-5	356	18	13	440	2	0	171
100-6	283	52	31	270	20	6	338
100-7	305	**27**	18	429	1	3	217
100-8	300	**100**	45	518	6	0	31
100-9	389	**17**	17	372	5	0	200
100-10	421	78	53	405	1	9	33
Total	3041	**448**	297	3721	84	30	2379

Because MLP and MLN have the same travel cost in many instances and columns 3-4 do not provide a clear idea as to which of the two policies is superior, we perform a paired t-test on travel cost difference d_i defined as $d_i = C_i^{MLN} - C_i^{MLP}$ for weather i. The null hypothesis is $H_0 : \mu_d = 0$ where μ_d is the mean difference in travel cost over 1000 weathers between MLN and MLP. The significance level is 0.05 and the alternative hypotheses are $H_1 : u_d > 0$ and $H_2 : u_d < 0$. Bold values in MLN column indicate that H_0 is rejected in favor of H_2, meaning that MLN statistically has better travel cost than MLP policy. Similarly, a bold value in MLP indicates that H_0 is rejected in favor of H_1. In 12/30 graphs MLN policy is statistically better than MLP, whereas MLP policy is statisticallly better than MLN in only 1 graph. In the remaining 17 graphs, the difference in travel cost was statistically insignificant. When a paired t-test is performed over all weathers for a graph size, it is clear that MLN statistically outperforms MLP on all three graph sizes.

The time taken by an agent to decide the subsequent node to be visited, also known as "replan time" and the frequency of replanning during the course of traversing to the destination is analyzed. The recorded replan times were approximately 4 sec, 6 sec, and 10 sec for 20, 50, and 100 node graphs for both MLP and MLN. The replan time for OMT, HOP, ORO, UCTB, and UCTO for 100 node graph was reported to be 0.00, 0.88, 7.71, 7.38, and 2.87 seconds respectively by Eyerich et al. (2010). The average number of replans made by the agent during the journey to the destination was found to be approximately 6, 11, and 15 times for 20, 50, and 100 node graphs, for both MLP and MLN, respectively. Thus, for a 100 node graph there will be a total delay of 10*15 = 150 seconds.

2.6 Conclusion and Future Work

Results of the computational study suggest the proposed algorithms, MLN and MLP, offer better average-case performance than OMT and UCTB from the literature. However, when compared with other reference approaches, there are mixed results. UCTO offered better average-case performance on all but two graphs and ORO and A*HOP offered better average-case performance

on just over half of the graphs tested. The new approaches did provide better average-case performance than all reference approaches on two graphs, 100-1 and 100-8, but these differences are not statistically significant when the differences are computed on the sample means, instead of pairs of observations. Paired t-testing to compare performance using weather-level data is only possible for A*-HOP. In that case, there are 6 graphs for which paired differences are statistically better for MLN than A*-HOP, and 4 graphs for which paired differences are statistically better for MLP than A*HOP. Together, MLN and MLP together find new best known solutions for 10,715 out of 30,0000 instances.

Although simple and intuitive in nature, both proposed approaches provided promising improvements and a significant number of new best known solutions. One limitation of the new policies is that computational work needs to be performed at every new belief state to determine the next node to visit. This is also true of the rollout-based polices in Eyerich et al. (2010). When such policies are used in real scenarios, it is important to understand whether those decisions can be both computed and communicated to the agent in a timely manner such that emergency vehicles will need to wait at every intersection for directions. For the biggest size graph studied in this paper consisting of 100 nodes, an average delay of 150 seconds was estimated.

A prospect for future research involves increasing the number of weathers and rollout weathers for a larger graph size, instead of employing the same number of weathers and rollout weathers as in the current study. This increase is expected to diminish the variance in the average cost as observed in Table 3.1. Another direction of future work includes extending the problem to CTP variants that include more than one destination, to mimic the real-world operations of Federal Staging Areas (FSAs) for disaster response. Another possibility is to model edges as recoverable in nature, where either the agent has capabilities to clear the blocked roads, and/or edges recover on their own, for example as flood waters subside. Although recoverable CTP has been introduced nearly two decades ago, there is no computational work for the problem variant in the literature at the time of this writing.

Bibliography

Ahmadi, E., Süer, G. A., and Al-Ogaili, F. (2018). Solving Stochastic Shortest Distance Path Problem by Using Genetic Algorithms. *Procedia Computer Science*, 140:79–86.

Akbari, V. and Shiri, D. (2022). An online optimization approach for post-disaster relief distribution with online blocked edges. *Computers & Operations Research*, 137:105533.

Aksakalli, V., Sahin, O. F., and Ari, I. (2016). An ao* based exact algorithm for the canadian traveler problem. *INFORMS Journal on Computing*, 28(1):96–111.

Alkaya, A. F., Yildirim, S., and Aksakalli, V. (2021). Heuristics for the canadian traveler problem with neutralizations. *Computers & Industrial Engineering*, 159:107488.

Alseth, A. (2020). *-CTP: Utilizing Multiple Agents to Find Efficient Routes in Disrupted Networks*. M.S.I.E., University of Arkansas, United States – Arkansas.

Argyroudis, S., Pitilakis, K., and Anastasiadis, A. (2005). Roadway network seismic risk analysis in urban areas: The case of thessaloniki-greece. In *Proceedings of the Geoline Conference, Lyon, France*.

Bai, A., Wu, F., and Chen, X. (2018). Posterior sampling for monte carlo planning under uncertainty. *Applied Intelligence*, 48(12):4998–5018.

Bar-Noy, A. and Schieber, B. (1991). The Canadian Traveller Problem,". In *SODA '91: Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms,*, pages 261–270.

Bender, M. and Westphal, S. (2015). An optimal randomized online algorithm for the $$k$$ k -Canadian Traveller Problem on node-disjoint paths. *Journal of Combinatorial Optimization*, 30(1):87–96.

Bent, R. W. and Van Hentenryck, P. (2004). Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. *Operations Research*, 52(6):977–987.

Bergé, P., Desmarchelier, J., Guo, W., Lefebvre, A., Rimmel, A., and Tomasik, J. (2019). Multiple Canadians on the road: Minimizing the distance competitive ratio. *Journal of Combinatorial Optimization*, 38(4):1086–1100.

Bergé, P. and Salaün, L. (2019). Improved deterministic strategy for the canadian traveller problem exploiting small max-(s, t)-cuts. In *International Workshop on Approximation and Online Algorithms*, pages 29–42. Springer.

Blei, D. M. and Kaelbling, L. P. (1999). Shortest Paths in a Dynamic Uncertain Domain. *IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, page 7.

Bnaya, Z., Felner, A., Fried, D., Maksin, O., and Shimony, S. E. (2015). Repeated-task Canadian Traveler Problem. *AI Communications*, 28(3):453–477.

Bnaya, Z., Felner, A., and Shimony, S. E. (2009). Canadian Traveler Problem with Remote Sensing. *International Joint Conference on Artificial Intelligence*, page 7.

Cheng, J., Leung, J., and Lisser, A. (2016). New reformulations of distributionally robust shortest path problem. *Computers & Operations Research*, 74:196–204.

Demaine, E. D., Huang, Y., Liao, C.-S., and Sadakane, K. (2014). Canadians should travel randomly. In *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I 41*, pages 380–391. Springer.

Demaine, E. D., Huang, Y., Liao, C.-S., and Sadakane, K. (2021). Approximating the canadian traveller problem with online randomization. *Algorithmica*, 83(5):1524–1543.

Eyerich, P., Keller, T., and Helmert, M. (2010). High-Quality Policies for the Canadian Traveler's Problem. *In Twenty-Fourth AAAI Conference on Artificial Intelligence*, page 8.

FEMA. How a Disaster Gets Declared.

FEMA (2019). Supply Chain Resilience Guide. Technical report.

FEMA (2020). FEMA Preliminary Damage Assessment Guide. Technical report.

FEMA (2022). Distribution Management Plan Guide 2.0. Technical report.

Fried, D., Shimony, S. E., Benbassat, A., and Wenner, C. (2013). Complexity of Canadian traveler problem variants. *Theoretical Computer Science*, 487:1–16.

Guo, H. and Barfoot, T. D. (2019). The Robust Canadian Traveler Problem Applied to Robot Routing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5523–5529, Montreal, QC, Canada. IEEE.

Huang, Y. and Liao, C.-S. (2012). The canadian traveller problem revisited. In *Algorithms and Computation: 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings 23*, pages 352–361. Springer.

IEP. Institute for Economics and Peace.

Lim, Z. W., Hsu, D., and Lee, W. S. (2017). Shortest Path under Uncertainty: Exploration versus Exploitation. *Uncertainty in Artifical Intelligence*, page 10.

Lita, L. V., Schulte, J., and Thrun, S. (2001). A system for multi-agent coordination in uncertain environments. In *Proceedings of the Fifth International Conference on Autonomous Agents - AGENTS '01*, pages 21–22, Montreal, Quebec, Canada. ACM Press.

Moya, L., Mas, E., Yamazaki, F., Liu, W., and Koshimura, S. (2020). Statistical analysis of earthquake debris extent from wood-frame buildings and its use in road networks in japan. *Earthquake Spectra*, 36(1):209–231.

NCEI. National Centers for Environmental Information.

Nikolova, E. and Karger, D. R. (2008). Route Planning under Uncertainty: The Canadian Traveller Problem. *Association for the Advacement of Artifical Intelligence*, page 6.

Papadimitriou, C. H. and Yannakakis, M. (1991). Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150.

Randour, M., Raskin, J.-F., and Sankur, O. (2014). Variations on the Stochastic Shortest Path Problem.

Sahin, O. F. and Aksakalli, V. (2015). A comparison of penalty and rollout-based algorithms for the canadian traveler problem. *International Journal of Machine Learning and Computing*, 5(4):319.

Santarelli, S., Bernardini, G., and Quagliarini, E. (2018). Earthquake building debris estimation in historic city centres: From real world data to experimental-based criteria. *International journal of disaster risk reduction*, 31:281–291.

Shahabi, M., Unnikrishnan*, A., and Boyles, S. D. (2015). Robust optimization strategy for the shortest path problem under uncertain link travel cost distribution. *Computer-Aided Civil and Infrastructure Engineering*, 30(6):433–448.

Shiri, D. and Salman, F. S. (2017). On the online multi-agent O–D k-Canadian Traveler Problem. *Journal of Combinatorial Optimization*, 34(2):453–461.

Shiri, D. and Salman, F. S. (2019a). Competitive analysis of randomized online strategies for the multi-agent k-Canadian Traveler Problem. *Journal of Combinatorial Optimization*, 37(3):848–865.

Shiri, D. and Salman, F. S. (2019b). On the randomized online strategies for the k-canadian traveler problem. *Journal of Combinatorial Optimization*, 38(1):254–267.

Shiri, D. and Salman, F. S. (2020). Online optimization of first-responder routes in disaster response logistics. *IBM Journal of Research and Development*, 64(1/2):13:1–13:9.

Su, B., Xu, Y., Xiao, P., and Tian, L. (2008). A risk-reward competitive analysis for the recoverable canadian traveller problem. In *International Conference on Combinatorial Optimization and Applications*, pages 417–426. Springer.

Toma-Danila, D. (2018). A gis framework for evaluating the implications of urban road network failure due to earthquakes: Bucharest (romania) case study. *Natural Hazards*, 93(Suppl 1):97–111.

Trevizan, F., Teichteil-Konigsbuch, F., and Thiebaux, S. (2017). Efficient Solutions for Stochastic Shortest Path Problems with Dead Ends. *Uncertainty in Artifical Intelligence*, page 10.

UNDRR (2022). Our World at Risk.

Westphal, S. (2008). A note on the k-Canadian Traveller Problem. *Information Processing Letters*, 106(3):87–89.

Xu, Y., Hu, M., Su, B., Zhu, B., and Zhu, Z. (2009). The canadian traveller problem and its competitive analysis. *Journal of Combinatorial Optimization*, 18(2):195–205.

Yildirim, S., Aksakalli, V., and Alkaya, A. F. (2019). Canadian traveler problem with neutralizations. *Expert Systems with Applications*, 132:151–165.

Yuan, F., Mobley, W., Farahmand, H., Xu, Y., Blessing, R., Dong, S., Mostafavi, A., and Brody, S. D. (2021). Predicting road flooding risk with machine learning approaches using crowdsourced reports and fine-grained traffic data. *arXiv preprint arXiv:2108.13265*.

Zhang, H., Xu, Y., and Qin, L. (2013). The k-Canadian Travelers Problem with communication. *Journal of Combinatorial Optimization*, 26(2):251–265.

Zhang, H., Xu, Y., and Wen, X. (2015). Optimal shortest path set problem in undirected graphs. *Journal of Combinatorial Optimization*, 29(3):511–530.

Zhang, Y., Song, S., Shen, Z.-J. M., and Wu, C. (2018). Robust Shortest Path Problem With Distributional Uncertainty. *IEEE Transactions on Intelligent Transportation Systems*, 19(4):1080–1090.

3. Single Source Multiple Destination Multiple Agent

3.1 Introduction

The population in the United States has experienced significant growth in recent years (US Census Bureau). This growth has placed a burden on available resources, leading to increased deforestation and fuel consumption that have caused drastic changes in global weather patterns (UNDRR, a; Bertrand, 2021). These changes have increased man-made and natural disasters in recent decades causing not only damage to the infrastructure but also loss of human life (UNDRR, a; Bertrand, 2021). Records from the five-year period between 2018 and 2022 show that such disasters have resulted in the loss of approximately 1751 lives and caused a staggering amount of $121.4B in damages per year (NCEI). By 2030, the world is expected to face 1.5 disasters per day on average (UNDRR, b). Given this situation, it is reasonable to expect the frequency with which governmental organizations such as the Federal Emergency Management Agency (FEMA) need to respond to disasters to increase. Thus, there is a pressing need to address the operational challenges faced by agencies like FEMA during responses to federally-declared disasters in order to minimize the suffering and loss of life. Many of these challenges arise due to the collapse of vital infrastructure, including communication networks, transportation networks, road networks, power grids, and water supplies, which leaves the affected population without access to life-saving supports (FEMA, 2020, 2022). One particularly critical challenge is the delivery of essential supplies and services to affected areas in the midst of damaged road networks. This paper is motivated especially by the supply chain structure employed by FEMA to facilitate disaster response supply delivery for disasters occurring within the continental United States.

An important component of a FEMA response supply chain is a Federal Staging Area (FSA). According to FEMA, a FSA is a base located near a disaster-impacted area that provides logistical support to the disaster response operation (FEMA, 2019). The FSA is established only in circumstances where the disaster overwhelms the capabilities of regional and state agencies to respond and the event has been officially designated as a major disaster (FEMA). The location

of a FSA is determined based on various criteria, such as proximity to the Area of Operations (AOR), layout of the site, and any support required for equipment staging, feeding, and sanitation (FEMA, 2022). Emergency supplies are brought to the Federal Staging Area (FSA) from regions outside the disaster-impacted area. From the FSA, emergency supplies are delivered to disaster survivors in the AOR who are in need. These deliveries might be made door-to-door, or, more often, to congregate locations such as public places, hospitals, and temporary shelters (FEMA, 2022). While some isolated areas may require support from water or air transportation, road transportation is the most commonly used method for fulfilling survivor needs (Ertem et al., 2017). As such, a fleet of trucks is staged at the FSA and is dispatched to make deliveries in the AOR.

Use of road transportation to supply essential resources to survivors requires the implementation of an appropriate supply chain model. The three models commonly used by FEMA are the hub-and-spoke model, fixed location model, and cross-docking model (FEMA, 2022). In the hub-and-spoke model, supplies are transported from a central fixed location to multiple locally operated Commodity Points of Distribution (C-PODs) (FEMA, 2022). In the fixed location model, a warehouse is used to store and distribute supplies without any reconfiguration of the supply order sizes, while in the cross-docking model, supply consolidation takes place and smaller delivery sizes are possible (FEMA, 2022). All three configurations can be modeled as single source multiple destination shortest path problems, where multiple vehicles are dispatched from the source; one to each destination. Therefore, this problem involves the development of paths rather than multi-stop routes, in contrast to vehicle routing problem variants.

While transporting emergency supplies from outside the disaster-hit area to the FSA is usually reliable, moving the supplies from the FSA to local staging areas or C-PODs can be difficult. This is because the road network is uncertain and although there are established routes and backup paths (FEMA, 2022), a significant challenge for emergency planners is determining how to adjust a route quickly when an unexpected road blockage is encountered. Uncertainties re-

garding which road segments contain blockages are only resolved when a vehicle reaches a road segment endpoint and the driver observes that the segment is blocked. Hence, there is a need for an online algorithm that can dynamically update vehicle paths as blockages are discovered. We refer to a vehicle and driver pair as an *agent*. Agents may be capable of communicating with each other either fully, partially, or not at all, depending on the application. For example, some agents may be able to both send and receive information (full communication), some may be able to receive but not send information (partial communication), and some may be disconnected from communication networks and unable to both send and receive.

The problem of finding the shortest path from a source to a destination in an uncertain road network is known as the Canadian Traveler Problem (CTP) and was first introduced by Papadimitriou and Yannakakis (1991). In this problem the agent is given a graph $G = (N, E)$, where N represents a set of nodes and E represents a set of edges. Each edge $e \in E$ has a travel cost c_e and a probability of blockage p_e associated with it. Because each edge is stochastic in nature, this problem is a stochastic variant of CTP. As each edge can either be available or blocked, there are $2^{|E|}$ possible scenarios. Any scenario in which a source and a destination is disconnected is discarded from the study. A scenario, also known as a weather w, consists of only the edges that are available for traversal ($w \subseteq E$). The objective of the agent is to find the shortest path from the source v_s to the destination v_d in the given weather. The travel cost of the shortest path is represented by z. As the agent starts traversing in the unknown weather w, the statuses of edges are revealed when agents reach one of their endpoints. In other words, the unknown weather w is dynamically revealed to the agent. This problem can be extended to a multi-destination and multi-agent variant, where each agent is assigned a single destination and the problem is to find shortest paths for all agents from a single source to their respective destinations. We refer to this problem variant as MAD-CTP, for multiple agents and destinations CTP. The objective in this case is to minimize the total cost of all the agents, defined as $\sum_{l \in L} z_l$, where z_l is the cost of the path traversed by agent l and L is the set of agents. Minimizing the total travel cost across agents is also equivalent to minimizing the average travel cost across agents, represented by $Z^* = \frac{Min \sum_{l \in L} z_l}{L}$.

In the MAD-CTP, as the agents move, they are also capable of communicating new information about the graph with one another. A related problem is an online traveling salesman problem, which involves developing a tour to visit multiple destinations in a network (Liao and Huang, 2014; Zhang et al., 2015a; Zhang and Xu, 2018). In contrast, agents in MAD-CTP are only concerned with visiting their destinations in the shortest amount of time possible. To the best of our knowledge, there is only one paper in literature addressing this problem variant at the time of this writing (Lita et al., 2001).

In this paper, a framework for MAD-CTP that utilizes two different approaches, MAD-OMT and MAD-HOP, is developed. It is tested for various numbers of agents on two types of graphs; Delaunay graphs from Eyerich et al. (2010) and Euclidean graphs from Shiri and Salman (2019b). Two communication levels, sharing (analogous to full communication) and no sharing (no communication), are considered. Results of the computational study indicate the MAD-HOP policy consistently outperforms the MAD-OMT policy for both communication levels by an average of 14% to 19% on Delaunay graphs and of 3% to 8% for Euclidean graphs. Further, sharing information proved beneficial in reducing the average travel cost across agents compared to the no sharing case. Thus, the benefit of sharing increases with the number of agents.

The subsequent sections are organized as follows. Section 2 comprises a review of relevant literature, Section 3 outlines the methodology employed, and Section 4 describes the design of the computational study. Computational findings are presented in Section 5 and finally, Section 6 provides conclusions and directions for future research.

3.2 Literature Review

Variants of CTP can be broadly grouped into three categories: (1) single agent and destination; (2) multiple agents and single destination, and (3) multiple agents and destinations. Given the focus in this paper on the latter domain, we restrict our attention in this literature review to CTP variants with multiple agents and/or multiple destinations, as outlined in Sections 3.2.1 through

3.2.3. Additional details regarding single agent and destination variants are available in Chapter 2.

3.2.1 Multiple Agents and Single Destination

In the k-CTP, at most k edges can fail (Bar-Noy and Schieber, 1991). Like its single agent counterpart, the multiple agent variant of k-CTP is among the most widely studied CTP variants in literature. Zhang et al. (2013) investigates two different versions of multi-agent k-CTP; full communication and limited communication. The goal is for at least one agent to reach the destination as soon as possible. In the full communication version, all agents have the capability to send and receive information (denoted RS-type agents). In the limited communication version, some some agents can both send and receive information, while other agents can only receive information (denoted R-type agent). Letting L be the number of agents and L_1 be the number that are RS-type, Zhang et al. (2013) proves that no deterministic online algorithm can achieve competitive ratios less than $2\lfloor\frac{k}{L}\rfloor + 1$ and $2\lfloor\frac{k-1}{L_1}\rfloor + 1$ for full and limited communication variants, respectively. Shiri and Salman (2022) improves the lower bound for the full communication case to $2\lfloor\frac{k}{2log_2 L}\rfloor + 1$. Using these results, it is clear that as the number of agents grows, the lower bound on the competitive ratio decreases. In addition to the theoretical results, Zhang et al. (2013) also presents two multiple agent k-CTP solution strategies referred to as Retrace-Alternative and Greedy. Results of a computational study indicate that the benefits of having multiple agents with full communication depends on the structure of the network.

Shiri and Salman (2017) introduces additional variants with respect to limited communication. Three communication protocols (CP1, CP2, and CP3) and three agent intelligence levels (IL1, IL2, and IL3) are presented. In CP1, RS-type agents can share graph information with RS-type and R-type agents. In CP2, RS-type agents can additionally share their planned paths with RS-type and R-type agents. In CP3, RS-type agents can plan travel paths for R-type agents. A RS-type agent can have any one of three intelligence levels. In IL1, the agent can only make decisions regarding the immediate next travel move, and can use only communication protocols

CP1 or CP2. In IL2, the agent can plan a full path for themselves and has the same communication protocol options as an IL1 agent. IL3 agents have the planning capabilities of IL2 agents, plus they can use the CP3 protocol. Shiri and Salman (2017) proposes two online strategies for these problem variants called the path labelling strategy (PLS) and modified path labeling strategy (mPLS). The PLS strategy uses IL2 agents and the mPLS strategy uses IL3 agents. Competitive ratio lower bounds of $2\lfloor\frac{k}{L_1+1}\rfloor+1$ and $2\lfloor\frac{k}{L}\rfloor+1$ are provided for PLS and mPLS respectively. Further, it is shown that mPLS is optimal in the special case of origin-destination edge disjoint graphs.

Shiri and Salman (2019a) continues the focus on communication with the aim of presenting a randomized solution strategy. A competitive ratio lower bound of $\sum_{j=1}^{k+1}\left(1-\left(\frac{k-(j-1)}{k+1-(j-1)}\right)^{L}\right)\left(\frac{k-(j-2)}{k+1}\right)^{L}(2j-1)$ is derived for the no communication case and $\frac{L}{k+1}(\lfloor\frac{k}{L}\rfloor)^2+\frac{k+1-L\lfloor\frac{k}{L}\rfloor}{k+1}(2\lfloor\frac{k}{L}\rfloor+1)$ for both the limited and full communication cases. An optimal online randomized strategy is presented for limited and complete communication cases on O-D edge disjoint graphs and it is shown that the competitive ratio does not improve when communication is complete compared to when it is limited.

Bergé et al. (2019) analyzes the same problem but from a slightly different perspective. Other competitive ratios in the literature (and discussed in this review) are time-based, including the time the first agent reaches the destination in the numerator and the optimal offline time to reach the destination in the denominator. In contrast, Bergé et al. (2019) introduces a distance-based competitive ratio, including the total travel distance across all agents to reach the destination in the numerator, and the optimal offline travel distance of a single agent in the denominator. A distance-based competitive ratio is provided for both deterministic and randomized strategies under partial and limited communication. These competitive ratios are compared to scenarios in which agents do not communicate at all and those in which they communicate only during initial planning at the source node. A deterministic strategy called multi-alternating is proposed and has a distance-based competitive ratio of $2(k+1)$–$min(k+1,L)$ for T_{first} and $2k+L$ for T_{last} where T_{first} and T_{last} is the time when the first agent and all the agents reach destina-

tion respectively. They also proved that these bounds are tight, and agents were allowed to have full communication. For case of partial communication, the competitive ratio was shown to be $2(k+1)$–$min(k+1, L_1)$ for T_{first} and $2k+L$ for T_{last} where L_1 are the number of agents in L who can both send and receive information. For the case with initial communication, they showed that no deterministic strategy can obtained a competitive ratio lower than $2(k+1)$–$min(k+1, L)$ for T_{first} and $L(k+1)$ for T_{last}. The optimal competitive ratio for case with no communication was shown to be $(2k+1)L$. Further Bergé et al. (2019) showed that no randomized strategy can achieve a competitive ratio less than $\frac{k+2}{2}$ for T_{first} and $k+L$ for T_{last} when $k+1 \leq L$ and $k+2-L$ for T_{first} and $k+L$ for T_{last} when $k+1 > L$ for cases with complete communication. The competitive ratio of the best strategy in case of agents having no communication lies between $(k+1)L$ and $(2k+1)L$ for T_{first} and T_{last} respectively. For case with initial communication, they provided an upper bound of $2k+1$ and $(2k+1)L$ for T_{first} and T_{last} respectively on the competitive ratio.

Zhang et al. (2015b) studies an optimal path set problem, in which the aim is to determine a minimum collection of paths that can assure the fastest arrival of at least one vehicle when k edges are blocked. The Least-Overlap algorithm is proposed for the case where $k = 1$ and the Modified Least-Overlap algorithm for the case where $k > 1$. In the $k > 1$ case, the blocked edges are assumed to be consecutive on a shortest path from the source to the destination, and nodes connecting the blocked edges are also blocked. Letting n be the number of graph nodes and m be the number of graph edges, the runtime complexities of Least-Overlap and Modified Least-Overlap are $O(n^2)$ and $O(mn + k^2 n^2 logn)$, respectively.

3.2.2 Single Agent and Multiple Destinations

This section encompasses an analysis of single agent and multiple destination online path or route planning problem variants in which at most k edges can be blocked. Liao and Huang (2014) study the Covering CTP, in which the goal is to find the shortest route visiting set of locations. This problem is similar to TSP, except the network contains blocked edges. An efficient touring strategy called Cyclic Routing (CR) is developed. Zhang et al. (2015a) study the online Steiner

TSP which is similar to the Covering CTP, with the difference being that in the online Steiner TSP, a blocked edge is revealed to the agent as soon as it becomes blocked. An exponential time algorithm called DISCOVER with a competitive ratio of $k+1$ is presented, and it is shown this algorithm is optimal. A separate algorithm, PIECEMEAL, with polynomial runtime is proposed. It has a competitive ratio of $k+4$.

Zhang and Xu (2018) study an online Covering Salesman Problem where the goal is to find a shortest tour such that each destination vertex is on the tour or within a predetermined distance L from a vertex that is on the tour, and the agent encounters at most k blocked edges during tour traversal. Similar to the Steiner TSP, the blocked edge is revealed to the agent as soon as it becomes blocked. The author presents a lower bound of $\frac{1}{1+(k+2)L}k+1$ where k is the number of blocked edges. Furthermore, the authors propose an online algorithm called CoverTreeTraversal with a competitive ratio of $k+\alpha$, where $\alpha = 0.5 + \frac{(4k+2)L}{OPT} + 2\gamma\delta$, γ is the approximation ratio for the Steiner tree problem, ρ is the maximal number of locations that a customer can be served, and OPT is the optimal value when agent has complete information. The problem is extended to include service cost and in that case, a lower bound on the competitive ratio was shown to be $min\{\frac{k}{1+(k+2)L}+1, \frac{k}{1+0.5(k+2)W}+1\}$ where w is the uniform penalty for each destination vertex. CoverTreeTraversal had a competitive ratio of k+, where $\alpha = 0.5 + \frac{4kL}{OPT} + 2\gamma\delta$ for this extended version.

Zhang et al. (2019) studies a minimum latency problem with online blocked edges where the goal is to find a tour visiting all the nodes such that total latency of these nodes is minimized. A lower bound on competitive ratios for any algorithm for this problem of $2k+1$ is developed. An online algorithm called GoodTreeTraversal is presented, along with a polynomial time algorithm, DETOUR. The efficiency and effectiveness of the two algorithms is evaluated by conducting numerical experiments.

Akbari and Shiri (2021) studies an online minimum latency problem with edge uncertainty (OMLP) and a weighted online minimum latency problem with edge uncertainty (WOMLP). A

lower bound on the competitive ratio of any algorithm for both of these problems is shown to be $2k+1$. The tightness of the bound is proven by providing an optimal online deterministic strategy called Back-to-Root. For a randomized online algorithm, a competitive ratio lower bound of $k+1$ is established. Finally, two heuristic algorithms called Shortcut and Adopted greedy are presented for the solution of OMLP and WOMLP variants, respectively.

3.2.3 Multiple Agents and Multiple Destinations

Akbari and Shiri (2022) study a problem variant with multiple agents, representing relief distribution crews. The goal is to assign nodes to the agents and generate routes in the presence of k-non recoverable blocked edges. The objective of the problem is minimizing total latency of critical nodes. This problem is formally known as the Multiple Online Minimum Latency Problem with Edge Uncertainty (MOMLP). A competitive ratio lower bound of $2\lfloor\frac{k}{L}\rfloor+1$ is provided for the MOMLP, where k and L $(k > L)$ are the number of non-recoverable blocked edges and agents, respectively. A deterministic online algorithm called multi-agent Back-to-Root (BR) is presented, and an upper bound of $2k+1$ for its competitive ratio is proven. Finally, three heuristics for the solution of MOMLP instances are presented

Lita et al. (2001) is the only paper of which we are aware that studies a single source, multiple agent and multiple destination variant of CTP. In it, a BA^* algorithm is proposed. It is based on A^* where the expected value of cost to destination, computed using the future discounted reward, is used as the heuristic function the A^* algorithm requires. Although the number of edges in the graphs and the number of agents used in the computational study are specified, other essential information is omitted, such as the graph type, number of nodes in the graph, number of weathers, edge costs and blockage probabilities. This poses a significant challenge in reproducing the results and utilizing them for comparative purposes.

3.3 Solution Approach

An overview of the online solution algorithms in this paper is as follows. At the source node, each agent plans a path from the source to their respective destinations. Then, agents begin following those plans. At each new node an agent visits, they first complete a sensing operation, in which edges adjacent to the current node are labeled as either available or blocked. Next, depending on whether communication is allowed, this information may or may not be shared with other agents. After that, the agent determines whether replanning is required. If the agent's plan contains a blocked edge, it is required; otherwise, it is not. Finally, the agent moves to the next node in their plan. This process is depicted in Figure 4.1 and additional details for the sensing, communicating, planning and moving operations are provided Sections 3.3.1 through 3.3.4.

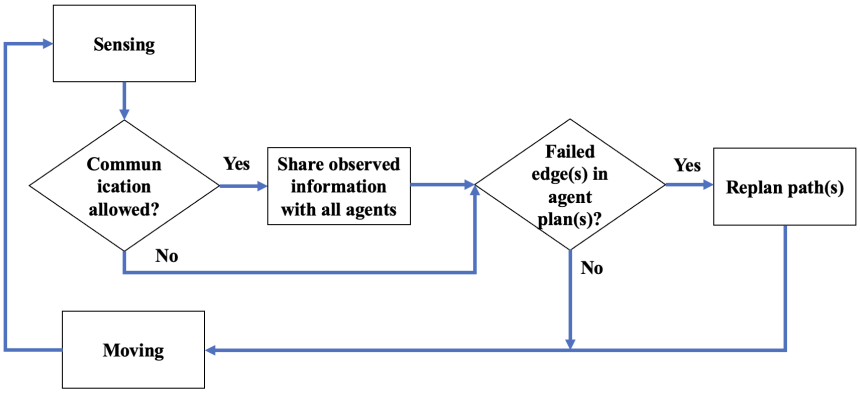


Figure 3.1: Stages of agent

3.3.1 Sensing

An agent is assumed to have the ability to sense the status of edges adjacent to the current node at no cost. Thus, the status of the edge is disambiguated by performing sensing operation when the agent reaches one of its endpoints. Each agent maintains a set of available and failed edges denoted by E_A and E_F respectively. After an edge has been disambiguated it is either added to E_A or E_F depending upon whether the edge is available or failed. The information sensed by the

agent may or may not be shared instantaneously with the other agents, depending on the type of communication policy used. Since communication capabilities of the agents are also considered in this paper, the next section describes the details.

3.3.2 Communicating

Full communication and no communication policies are considered. All agents have the same communication capabilities; that is, either all have full communication, which we refer to as *sharing*, or all have no communication, which we refer to as *no sharing*. In the sharing policy, all information acquired by an agent is shared instantaneously with all other agents. Thus, all agents have identical knowledge of the available and not available edge sets (E_A and E_F) at all points in time. As a result, any time an agent has to perform planning, they are able to access all information acquired by all agents up to that particular moment in time. On the other hand, in the no sharing policy, agents only have local knowledge of the edges they themselves have observed at any point in time. Consequently, replanning is solely based on the information gathered by the individual agent. Replanning in both the sharing and no sharing cases is triggered any time an individual agent becomes aware of a blocked edge in their plan. Thus, in the no sharing case, only the agent who discovered the blockage replans. In the sharing case, all agents replan any time any agent discovers a blockage. Planning is discussed further in the next section.

3.3.3 Planning

In the planning stage, agents each plan a path to their respective destination. A policy is used to generate these plans. Two path planning policies for CTP are adopted from the literature and tailored to the multi-agent multi-destination variant. The first is Optimistic (OMT), which is a deterministic policy that assumes all edges in the graph are available until it is discovered otherwise (Eyerich et al., 2010). The next is Hindsight Optimization (HOP), which is a sample approximation method in which rollout weathers are sampled (Eyerich et al., 2010). Sections 3.3.3.1 and 3.3.3.2 describe these policies in detail.

3.3.3.1 MAD-OMT

The OMT policy is first presented in (Eyerich et al., 2010), for a single agent and single destination variant of the Stochastic CTP. In this policy, all edges not observed by the agent are considered available. Using this assumption, any shortest path algorithm such as Dijkstra's can be used to generate a path from the current node to the destination. Because agents have different destinations, and also different "origins" at any snapshot in time during graph traversal, Dijkstra's shortest path algorithm is implemented for each agent separately. In each case the agent's current node is treated as the source node. Thus, MAD-OMT simply indicates the repeated application of Dijkstra's algorithm, once for each agent, using the free space assumption.

3.3.3.2 MAD-HOP

For a given weather w, the HOP policy samples r rollout weathers. These rollout weathers are generated in the same way as the weather w. The A*-HOP framework proposed in Alseth (2020) is designed for the multi-agent, single-destination CTP. This paper extends the framework to suit it for multi-destination scenarios. This extension aims to tackle two challenges: managing multi-destination and diversifying paths. To address the first challenge, the framework adopts a one-to-one assignment between agents and their respective destinations. This is based on the motivation discussed in Section 4.1. As agents have different destinations, the framework relies on the inherent diversification that arises from having diverse destinations. In order to utilize the A* algorithm, it is necessary to provide a heuristic value (expected cost), as an input in addition to the graph G and its edge costs. A* aims to determine the path that minimizes the function $f_l(v)$, where l represents the agent number, defined in Equation 3.1. The function incorporates the minimum cost to reach node v from the starting node, denoted as $g_l(v)$ as well as the heuristic cost $h_l(v)$ required to reach the destination node from node v. As agents have distinct destinations, the notation in Equation 3.1 incorporates a subscript l to differentiate the values of f, g, and h for agent l:

$$f_l(v) = g_l(v) + h_l(v). \tag{3.1}$$

An ideal heuristic value for function $h(v)$ should be close to the actual expected cost to reach the destination from every node in the network, avoiding overestimation whenever possible, as overestimation can lead to longer paths. To compute this heuristic value, r rollout weathers are used. Each rollout weather j is independent from other rollout weathers and a shortest path to the destination is identified using Dijkstra's algorithm assuming perfect information about the rollout weather. That is, edges that are available in rollout j and failed in rollout j are known to the agent. Let $\sigma_j^l(v)$ denote the cost of the shortest path from from node v to the destination of agent l in rollout j. Let σ_j^l be the set of all such costs σ_j^l for every possible node v. Then for each node v, the average path cost to destination across all rollouts is computed using Equation 3.2. This average cost $h_l(v)$ acts as heuristic value for the A* algorithm for agent l.

$$h_l(v) = \frac{\sum_{j \in 1,2,\ldots,r} \sigma_j^l(v)}{r} \tag{3.2}$$

It is worth nothing that utilizing Equation 3.2 to provide expected cost may lead to overestimations in scenarios where the number of failed edges is less than expected. This implies that the estimated cost may be more than the actual cost.

3.3.4 Moving

In moving stage, the agent moves to the next node in the plan which then becomes the current node of the agent. The cost to traverse the edge is added to the total traversal cost for the agent.

3.4 Experimental Details

Considering two path planning policies and two communication policies, a total of four policy variants are analyzed; MAD-OMT in sharing and no sharing cases, and MAD-HOP in sharing and no sharing cases. The performance of these four policy variants are evaluated on two distinct

type of graphs, namely Delaunay and Euclidean, the details of which are outlined in Section 3.4.1 and Section 3.4.2.

3.4.1 Delaunay Graphs

A total of 30 unique graphs, 10 of each size 20, 50, and 100 nodes, was generated using Delaunay triangulation as mentioned in Eyerich et al. (2010). Edge costs c_e and failure probabilities p_e for edges e were obtained from uniform distributions on the intervals [1,50] and [0,1), respectively. The 30 Delaunay graphs in the computational study in this paper are identical to those in (Eyerich et al., 2010). However, the weathers (instances) in this paper are different and destinations are different; the latter because multiple destinations are involved. Each graph has a source v_s which is the node with the lowest index. For each of these 30 graphs, 1000 weathers (i.e., instances) are generated randomly by sampling from a Bernoulli distribution with parameter p_e for each edge. This process is identical to the one explained in Eyerich et al. (2010). Further, for each problem instance, 1000 rollout weathers are generated, also by sampling from the Bernouli distributions, and made available for future researchers. The number of agents involved in each instance varies depending upon the graph size. For a graph consisting of 20 nodes, 2 agents are used. For a graph consisting of 50 nodes, 3 and 5 agents are used. For a graph consisting of 100 nodes 3, 5, and 10 agents are used. The number of destinations for each instance is equal to the number of agents involved. These destinations are selected using the process outlined in Section 3.4.3. Thus, with 3 different graph sizes, 10 different graphs for each size, 1000 weathers for each graph, and various numbers of agents, a total of 60,000 Delaunay Graph instances test instances are included.

3.4.2 Euclidean Graphs

To generate Euclidean graphs, a 100 x 100 grid is utilized, similar to the approach taken in Shiri and Salman (2019b). To generate a set of graphs, five different sizes are considered with 100, 200, 300, 400, and 500 nodes. For each graph size, three unique graphs are generated. The gen-

eration process involves randomly generating nodes in the grid, where each node is defined by its (x, y) coordinates that are integer values in the range [1,100]. Subsequently, edges are randomly added between nodes to form the desired graph. The number of edges generated are 300, 600, 900, 1200, and 1500 for 100, 200, 300, 400, and 500 node graph sizes, respectively. A total of 15 unique graphs are included. The lowest numbered node in each graph is regarded as its source node v_s. The cost of the edge connecting a pair of nodes in this graph is its Euclidean distance, determined from the endpoint node coordinates. For each of 15 graphs, four edge failure scenarios are generated, using edge blockage probabilities of 10%, 20%, 30%, and 40%. In a given graph and scenario, all edges have the same probability of failure. For example, in the 10% blockage scenario, $p_e = 0.10$ for all e. Thus, there are a total of 60 graphs. For each of these graphs, 100 weathers (i.e., instances) are generated by randomly sampling from the Bernoulli distribution with blockage probability p_e for each edge. For each instance, 1000 rollout weathers are used. The number of agents in an instance varies depending on graph size. For each graph size, three numbers of agents are tested. For the graph size of 100, the numbers of agents are 3, 6, and 9; for the graph size of 200, the numbers of agents are 4, 8, and 12; for the graph size of 300, the numbers of agents are 5, 10, and 15; for the graph size of 400, the numbers of agents are 6, 12, and 18; and for the graph size of 500, the numbers of agents are 7, 14, and 21. The number of destinations are equal to the number of agents involved and selected using the process outlined in the Section 3.4.3. Thus, with 60 graphs, 100 weathers for each graph, and 3 values for numbers of agents, there are a total of 60*100*3 = 18,000 test instances. It is important to note that the Euclidean weathers used in this study are different from those used by Shiri and Salman (2019b). In that paper, the source and destination nodes are randomly selected for each test instance.

3.4.3 Destinations

Instead of randomly selecting l destinations from among the nodes included in a graph, a specific method is employed to select destinations aim for the distance between the source node and an agent's destination is comparable across agents. To achieve this for a given graph, first the short-

est path distances from the source to all other nodes is computed. Nodes are then sorted in non-increasing order of these distances. Finally, l nodes are randomly selected from the top 20% of the list.

3.5 Computational Results

This section provides an analysis of the computational results obtained through the application of the MAD-OMT and MAD-HOP algorithms on both Delaunay and Euclidean graphs for both the sharing and no sharing communication scenarios. Table 3.1 presents the average travel cost $\bar{Z}^*$ for all four variants. Each value in the table is an average computed over 10,000 weathers in the Delaunay graph case and 1200 weathers in the Euclidean graph case. The first column of Table 3.1, labeled T-$|V|$-l, specifies the graph type T (D for Delaunay and E for Euclidean), the number of nodes in the graph $|V|$, and the number of agents l in the test instance. The columns under MAD-OMT provide the average travel cost of solutions produced by MAD-OMT in the no sharing (NS) and sharing (S) scenarios, along with the average percent improvement in average travel cost afforded by sharing versus not sharing information (NS vs S). Analogous columns are provided for MAD-HOP. The final two columns provide the average percent improvement in average travel cost afforded by the MAD-HOP approach, compared with the MAD-OMT approach, for both no sharing (NS (%)) and sharing (S (%)) scenarios. Section 3.5.1 provides commentary comparing the two proposed algorithms is evaluated. Section 3.5.2 discusses the benefits derived from information sharing. Finally, in Section 3.5.3, describes the total delay in agent movement induced by replanning events. This analysis aims to provide a comprehensive understanding of the efficiency and effectiveness of the proposed algorithms.

3.5.1 Approach Comparison

In this section, a comparative analysis of the performance of MAD-OMT and MAD-HOP under the same communication policy is presented. Specifically, when the communication policy is NS, columns 2 and 5 of Table 3.1 are compared, whereas when the communication policy is

Table 3.1: Average cost for Delaunay and Euclidean graphs

	MAD-OMT			MAD-HOP			MAD-OMT vs MAD-HOP	
T-$\|V\|$-l	NS	S	NS vs S (%)	NS	S	NS vs S (%)	NS (%)	S (%)
D-20-2	184.08	182.80	0.70	156.55	155.57	0.63	14.96	14.90
D-50-3	294.45	287.21	2.46	242.14	238.07	1.68	17.77	17.11
D-50-5	288.21	277.15	3.84	235.82	230.77	2.14	18.18	16.73
D-100-3	368.37	358.97	2.55	304.97	299.31	1.86	17.21	16.62
D-100-5	363.41	350.44	3.57	300.60	291.54	3.01	17.28	16.81
D-100-10	357.15	339.47	4.95	296.88	283.78	4.42	16.87	16.41
E-100-3	274.94	272.26	0.98	265.38	263.14	0.84	3.48	3.35
E-100-6	274.72	267.11	2.77	263.40	255.81	2.88	4.12	4.23
E-100-9	267.47	258.33	3.42	255.76	247.88	3.08	4.38	4.05
E-200-4	311.01	307.32	1.18	295.13	291.11	1.36	5.11	5.27
E-200-8	304.24	295.77	2.78	288.11	280.38	2.68	5.30	5.20
E-200-12	309.87	296.30	4.38	294.83	282.08	4.32	4.85	4.80
E-300-5	340.93	334.87	1.78	316.20	311.24	1.57	7.26	7.06
E-300-10	343.74	332.69	3.22	324.49	314.00	3.23	5.60	5.62
E-300-15	346.40	329.07	5.00	324.40	309.88	4.48	6.35	5.83
E-400-6	351.95	342.70	2.63	329.45	320.84	2.61	6.39	6.38
E-400-12	345.61	329.91	4.54	324.41	308.90	4.78	6.14	6.37
E-400-18	347.74	325.08	6.52	327.78	307.68	6.13	5.74	5.35
E-500-7	370.39	359.38	2.97	342.23	332.68	2.79	7.60	7.43
E-500-14	367.79	349.06	5.09	344.62	327.16	5.07	6.30	6.28
E-500-21	367.96	341.15	7.29	344.41	320.07	7.07	6.40	6.18

S, columns 3 and 6 are compared. For instance, considering a Delaunay graph with 20 node and two agents (D-20-2), the average travel costs obtained using MAD-OMT and MAD-HOP are 182.80 units and 155.57 units under the sharing communication policy. This indicates an improvement of 14.90% in favor of MAD-HOP, reported in column 9. Interestingly, when information sharing is not allowed, MAD-HOP continues to exhibit a similar level of improvement, with an average cost improvement of 14.96% over MAD-HOP (column 8). Thus, these results suggest MAD-HOP offers superior performance compared with MAD-OMT, regardless of communication policy. When considering graphs of differing sizes, the percentage improvement of MAD-OMT over MAD-HOP is relatively stable, ranging between approximately 15% to 18% for Delaunay graphs and approximately 3% to 8% for Euclidean graphs. When considering different number of agents and holding graph size constant, it can still be observed that the percentage improvement of MAD-HOP over MAD-OMT is relatively stable. These results suggest that neither the size of the graph nor the number of agents employed appear to have a discernible impact on

the percentage improvement obtained in MAD-HOP over MAD-OMT.

The performance difference between MAD-HOP and MAD-OMT can be analyzed by level of disruption for the Euclidean graphs, as four scenarios with blockage probabilities of 10% to 40% are included. Figure 3.2 depicts the percentage improvement of MAD-HOP over MAD-OMT for 500-node Delaunay graphs on the y-axis and blockage probability on the x-axis. Each data point in the figure is an average over 300 weathers (3 unique graphs each with 100 weathers). It can be observed that when the blockage probability is 10%, MAD-HOP and MAD-OMT exhibit comparable performance. However, as p_e increases, MAD-HOP increasingly outperforms MAD-OMT. For example, when $p_e = 0.40$, percentage improvements of 11.40%, 10.20%, and 8.80% were recorded for 7, 14, and 21 agents respectively.

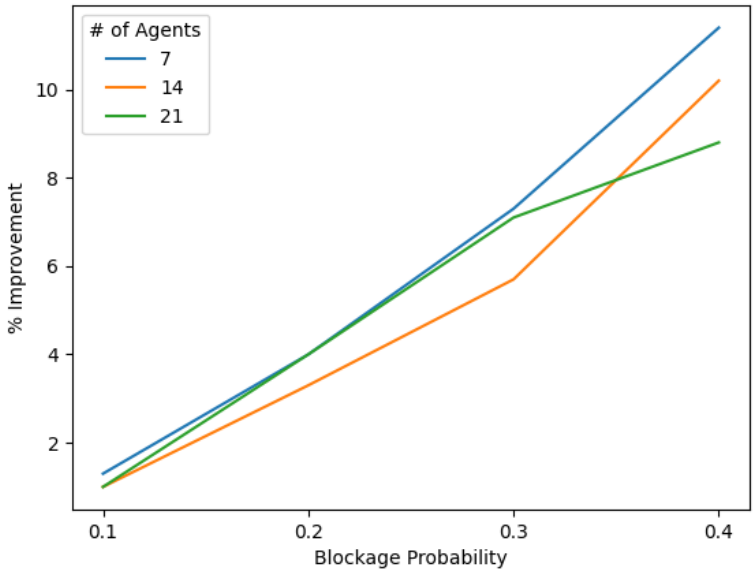


Figure 3.2: Effect of blockage probability p_e in performance of MAD-HOP over MAD-OMT on 500 node Euclidean graphs

3.5.2 Communication Levels

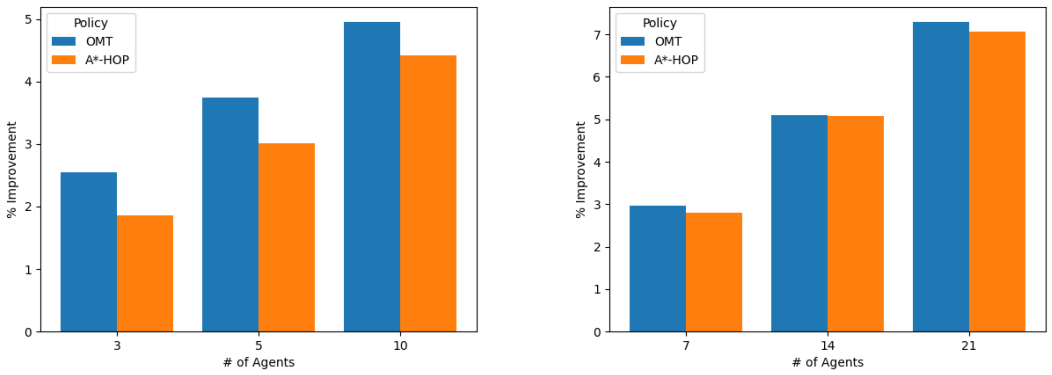
In this Section, the impact of communication policy is examined. The data to support these observations are in Table 3.1, in the NS vs S (%) columns, under both MAD-OMT and MAD-HOP.

For example, for test instances of type D-20-2, the benefit of sharing is 0.70% when MAD-OMT is used and 0.63% when MAD-HOP is used. These improvements become larger as the graph size increases and also as the number of agents increases. For instance, for a 100-node Delaunay graph with 10 agents (D-100-10), percentage improvements of 4.95% and 4.42% are observed for MAD-OMT and MAD-HOP, respectively. These improvements further increase to 7.29% and 7.07% for a Euclidean graph of size 500 with 21 agents (E-500-21), for MAD-OMT and MAD-HOP, respectively. Thus, it is evident that sharing information provides a considerable benefit over no sharing.

To better understand how the improvements afforded by information sharing increase with the number of agents, Figure 3.3 is considered. It depicts the percentage improvement observed for 100 node Delaunay and 500 node Euclidean graphs across increasing numbers of agents. In Figure 3.3a, it can be observed that for MAD-OMT on 100 node Delaunay graphs, the percentage improvement increases from 2.55% to 3.75% as the number of agents increase from 3 to 5. This improvement further increases to 4.95% with 10 agents. This same trend is observed for MAD-HOP on 100 node Delaunay graphs, and also for both MAD-OMT and MAD-HOP on 500 node Euclidean graphs, as seen in the Figure 3.3b. Hence, as the number of agents increases, the benefit from sharing information increases for both MAD-OMT and MAD-HOP algorithms.

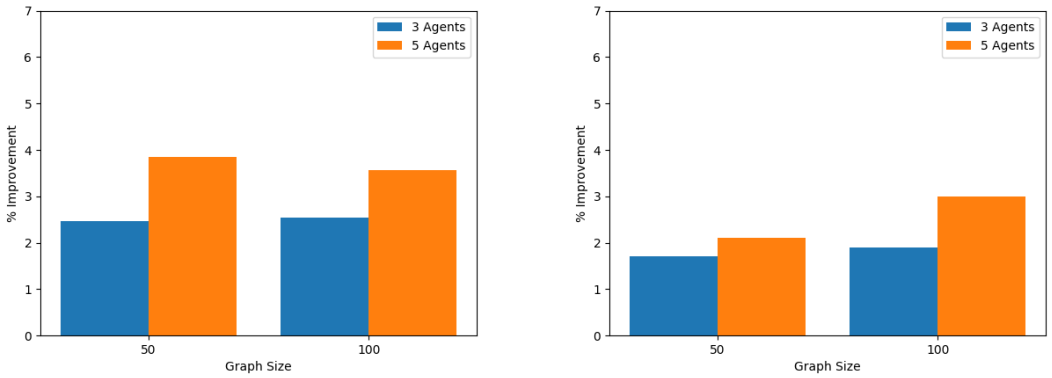
To better understand how the improvements afforded by information sharing increase as the graph size grows, Figure 3.4 is considered. It depicts the percentage improvement observed for 50 and 100 node Delaunay graphs under 3 and 5 agents. The MAD-OMT results indicate a complex relationship between graph size and the benefit obtained by sharing information. In Figure 3.4a, specifically for 3 agents, the percentage improvement increases from 2.46% to 2.55% as the graph size increases from 50 to 100. However, when 5 agents are employed, the percentage improvement decreases from 3.84% to 3.57% as the graph size increases. Thus, the relationship between graph size and the benefit derived from information sharing remains unclear for MAD-OMT. In contrast, the MAD-HOP results suggest a pattern. Figure 3.4b reveals that as the graph size increases the benefit marginally increases for both 3 and 5 agents on Delaunay graphs.

Consequently, it can be inferred that for MAD-HOP, an increase in the graph size yields a corresponding improvement in the benefit achieved through information sharing.



(a) 100 node Delaunay graph (b) 500 node Euclidean graph

Figure 3.3: Impact of number of agents



(a) MAD-OMT (b) MAD-HOP

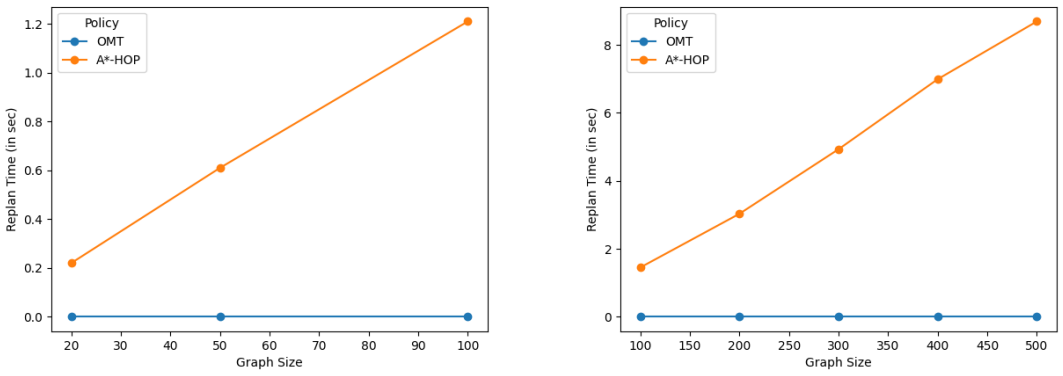
Figure 3.4: Impact of graph size

3.5.3 Agent Movement Delays Due to Replanning

Whenever an agent engages in replanning, computational time is required to identify a new path to destination. Consequently, this process leads to a delay in agent movement, where they are stopped during replanning, and not progressing towards the intended destination. It is therefore imperative to conduct a comprehensive analysis of the number of replanning events that

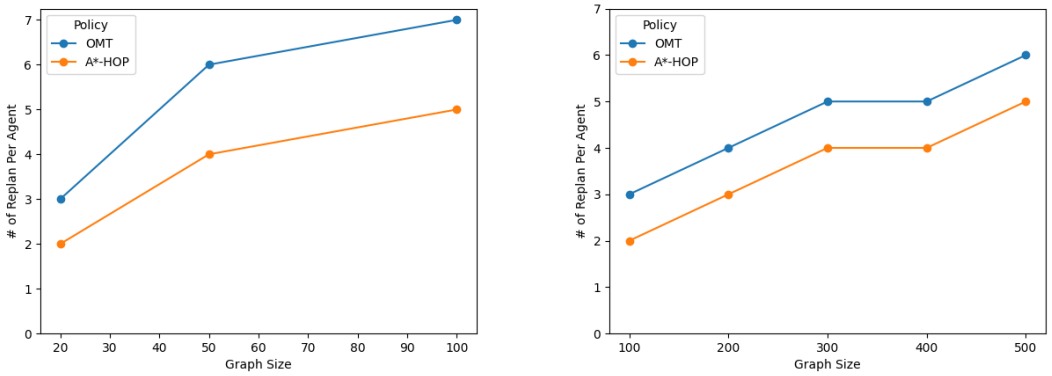
an agent undergoes during graph traversal, as well as the total replanning time. Figure 4.4a and Figure 4.4b depict the replan time per replanning event for Delauany and Euclidean graphs respectively. For MAD-OMT, the replan time is constant with respect to graph size and is less than 0.001 second, practically causing no delay. However for MAD-HOP, a linear relationship can be observed between graph size and replan time both for Delaunay and Euclidean graphs. For a 20 node Delaunay graph, MAD-HOP requires 0.22 seconds to replan a path whereas for a 500 node Euclidean graph, MAD-HOP requires 8.69 seconds. The total delay to reach the destination is equal to the total number of times an agent undergoes replanning event during the traversal multiplied by the replan time. Thus, the tradeoff between reduction in travel cost and delay should be carefully considered before choosing the policy for implementation in practical scenarios.

Figure 4.5 shows the average number of replanning event per agent for different graph sizes. From Figure 4.5b, for a 500 node graph, an agent on average replans 5 times which leads to a total replanning time of 5*8.69 = 43.45 seconds. This means that the total delay caused by computational work performed is less than 1 minute for a 500 node Euclidean graph. There are some identified areas in MAD-HOP where more efficient implementation would reduce the replan time further making it a more practical approach.



(a) Replan time for Delaunay graph (b) Replan time for Euclidean graph

Figure 3.5: Replan time



(a) Replan count for Delaunay graph (b) Replan count for Euclidean graph

Figure 3.6: Replan count

3.6 Conclusion and Future Work

This paper proposes a framework for solving a single source, multiple destination, multiple agent variant of the Stochastic CTP. The proposed framework incorporates two different algorithms, MAD-OMT and MAD-HOP. Both of these path planning algorithms are evaluated on Delaunay and Euclidean graphs under two different communication policies; no sharing and sharing. The computational study indicates the MAD-HOP alogrithm outperforms MAD-OMT on all test instances. The improvement ranges between 14% to 19% and 3% to 8% on Delaunay and Euclidean graphs, respectively, for both communication policies.

Further analysis on the difference in improvement observed among both the graph types concluded that the blockage probability has a significant impact on the performance of the algorithms. MAD-HOP leads to more improvement when the blockage probability is higher in the network. The comparison of travel costs between the two communication levels revealed that sharing has benefits over no sharing, even when agents travel to different destinations. Furthermore, the computational study suggests that the benefits of sharing information increase with the number of agents for both algorithm. A similar conclusion was made with respect to the graph size, but it was only valid for the MAD-HOP algorithm and remained unclear for the MAD-OMT algorithm.

Lastly, the paper evaluates the practicality of implementing MAD-HOP in disaster scenarios where time is a critical factor by conducting a replanning time analysis. The findings reveal that the delay time is less than 1 minute for the largest graph considered in the study which contains 500 nodes and 1500 edges, thus underscoring the practical feasibility of implementing MAD-HOP in such scenarios.

In order to validate the practical feasibility of our approach, it will be necessary to test the framework on a real network and evaluate its performance. To further improve travel cost, more advanced algorithms such as UCT Eyerich et al. (2010) can be implemented. One potential direction is to label edges as deterministic and convert the problem into a k-CTP variant which is widely studied in the literature. Real-life scenarios often involve interdependent edges, so it is not uncommon to encounter such cases. Therefore, to evaluate the effectiveness of our approach, it would be beneficial to test it on variants that incorporate edge dependence. Finally, similar to the recoverable variant of the single agent single destination, the MAD-CTP can also be extended to its recoverable variant.

Bibliography

Akbari, V. and Shiri, D. (2021). Weighted online minimum latency problem with edge uncertainty. *European Journal of Operational Research*, 295(1):51–65.

Akbari, V. and Shiri, D. (2022). An online optimization approach for post-disaster relief distribution with online blocked edges. *Computers & Operations Research*, 137:105533.

Bar-Noy, A. and Schieber, B. (1991). The canadian traveller problem. In *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pages 261–270.

Bergé, P., Desmarchelier, J., Guo, W., Lefebvre, A., Rimmel, A., and Tomasik, J. (2019). Multiple canadians on the road: minimizing the distance competitive ratio. *Journal of Combinatorial Optimization*, 38(4):1086–1100.

Bertrand, S. (2021). Fact Shett | Climate, Environmental, and Health Impacts of Fossil Fuels (2021).

Ertem, M. A., İşbilir, M., and Şahin Arslan, A. (2017). Review of intermodal freight transportation in humanitarian logistics. *European Transport Research Review*, 9(1):1–11.

Eyerich, P., Keller, T., and Helmert, M. (2010). High-quality policies for the canadian traveler's problem. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

FEMA. How a Disaster Gets Declared.

FEMA (2019). FEMA Region-X Logistics Branch. Technical report.

FEMA (2020). FEMA Preliminary Damage Assessment Guide. Technical report, FEMA.

FEMA (2022). Distribution Management Plan Guide 2.0. Technical report, FEMA.

Liao, C.-S. and Huang, Y. (2014). The covering canadian traveller problem. *Theoretical Computer Science*, 530:80–88.

Lita, L. V., Schulte, J., and Thrun, S. (2001). A system for multi-agent coordination in uncertain environments. In *Proceedings of the fifth international conference on Autonomous agents*, pages 21–22.

NCEI. Billion-Dollar Weather and Climate Disasters.

Papadimitriou, C. H. and Yannakakis, M. (1991). Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150.

Shiri, D. and Salman, F. S. (2017). On the online multi-agent o–d k-canadian traveler problem. *Journal of Combinatorial Optimization*, 34(2):453–461.

Shiri, D. and Salman, F. S. (2019a). Competitive analysis of randomized online strategies for the multi-agent k-canadian traveler problem. *Journal of Combinatorial Optimization*, 37(3):848–865.

Shiri, D. and Salman, F. S. (2019b). Online optimization of first-responder routes in disaster response logistics. *IBM Journal of Research and Development*, 64(1/2):13–1.

Shiri, D. and Salman, F. S. (2022). An improved lower bound on the competitive ratio of deterministic online algorithms for the multi-agent k-canadian traveler problem. *Findings*.

UNDRR. Environmental Degradation.

UNDRR. Our World at Risk.

US Census Bureau. Nation's Population Growth Slowed This Decade.

Zhang, H., Tong, W., Lin, G., and Xu, Y. (2019). Online minimum latency problem with edge uncertainty. *European Journal of Operational Research*, 273(2):418–429.

Zhang, H., Tong, W., Xu, Y., and Lin, G. (2015a). The steiner traveling salesman problem with online edge blockages. *European Journal of Operational Research*, 243(1):30–40.

Zhang, H. and Xu, Y. (2018). Online covering salesman problem. *Journal of Combinatorial Optimization*, 35(3):941–954.

Zhang, H., Xu, Y., and Qin, L. (2013). The k-canadian travelers problem with communication. *Journal of Combinatorial Optimization*, 26(2):251–265.

Zhang, H., Xu, Y., and Wen, X. (2015b). Optimal shortest path set problem in undirected graphs. *Journal of Combinatorial Optimization*, 29(3):511–530.

4. Single Source Multiple Destination Multiple Agent UCTO

4.1 Introduction

The Canadian Traveler Problem (CTP) requires an agent to traverse an uncertain network from the source node to the destination node with the objective of minimizing the travel cost. The uncertainty lies in the traversability status of the roads. This problem can be modeled as a Markov Decision Process (MDP) (Nikolova and Karger, 2008), allowing for the use of standard reinforcement learning techniques such as dynamic programming, Monte Carlo methods, or Q-learning to find an optimal policy that minimizes travel cost. One approach to solving CTP is the Upper Confidence Bounds applied to Trees (UCT) algorithm, a rollout-based Monte Carlo planning algorithm proposed in Kocsis and Szepesvári (2006). UCT aims to converge to the best action given sufficient computation time, and reduce the error probability if the algorithm is stopped prematurely. Computational results show that for a stochastic shortest path model applied to a sailing problem, only a relatively small number of samples are required to achieve the same levels of error that is achieved by other approaches in the literature, such as Adaptive Real Time Dynamic Programming (Barto et al., 1991) and a Trajectory-based algorithm (Péret and Garcia, 2004). Because UCT may require a smaller number of samples to achieve convergence and lower errors for problems like the stochastic shortest path problem, it may also be well-suited to solve the related Stochastic CTP. The utilization of the UCT algorithm in research has gained significant prominence since its introduction in Kocsis and Szepesvári (2006). It has been found to have good performance for various problem domains pertaining to decision-making under uncertainty.

Eyerich et al. (2010) devises two variants of the UCT algorithm called UCTB (blind) and UCTO (optimistic) to use in solving the Stochastic CTP with a single agent and single destination. The UCTB algorithm is devoid of any problem-specific information that could potentially influence the rollouts in favor of achieving the goal. On the other hand, the UCTO algorithm incorporates a certain degree of guidance that aids in directing the algorithm towards the goal by leveraging problem-specific knowledge. In a computational study which consisted of Delaunay

graphs with 20, 50, and 100 nodes, it is observed that UCTO outperforms two other rollout-based policies, Hindsight Optimization (HOP) and Optimistic Rollout (ORO) (Eyerich et al., 2010).

Lim et al. (2017) proposes a polynomial-time algorithm, Hedged Shortest Path under Determinization (HSPD), to solve the Bayesian CTP, where edge statuses are correlated with each other. The HPSD algorithm is compared with UCTO in a computational study and interestingly, the findings reveal inferior performance of UCTO compared with HPSD. However, the computational study in Lim et al. (2017) is limited to a hypothetical network on a 10 by 10 grid where all edgse have length 2. The mixed results observed in the relative performance of UCTO compared with other approaches stimulates a compelling possibility of exploring its efficacy in addressing the Stochastic CTP variant that involves multiple agent and multiple destination. This serves as the central subject of investigation in this article.

This study extends the UCTO algorithm to solve a multiple agent and multiple destination Stochastic CTP variant, MAD-CTP. In this version of UCTO, similar to the original UCTO proposed in Eyerich et al. (2010), planning is conducted at every node, regardless of whether or not a blocked edge is detected. This modified UCTO algorithm is compared with the MAD-HOP algorithm from Chapter 3, which only replans when obstructed edges are encountered along the agent's path. Both approaches require computational effort to replan the path, which incurs a certain amount of time delay in reaching the destination. If this delay is significant, it may outweigh the benefits in terms of reduced travel cost. Therefore, it is crucial to evaluate the trade-off between travel cost and the overall time spent on replanning events when comparing policies. This tradeoff is evaluated in a computational study comprised of Delaunay and Euclidean graphs of varying sizes and edge blockage distributions.

The rest of the paper is organized as follows: Section 4.2 outlines the problem definition and proposed solution approach, Section 4.3 describes the experimental design, Section 4.4 presents the computational results, and finally, Section 4.5 offers conclusions and suggestions for future research.

4.2 Problem Definition and Solution Approach

The problem of finding the shortest path from a source to a destination in an uncertain network is defined as the Canadian Traveler Problem and was first introduced in Papadimitriou and Yannakakis (1991). The objective is to minimize the travel cost z to the reach destination. In the MAD-CTP, L agents share a single source node v_s and each agent has their own unique destination v_l. The objective is to minimize the total travel cost of all agents, denoted $Min \sum_{l \in L} z_l$, to reach their destinations. A graph $G = (V, E)$, where V and E denote the sets of nodes and edges in G, is known to all agents. Each edge e has a cost c_e and blockage probability p_e associated with it. The exact status of an edge remains unknown to the agents until an agent reaches one of its endpoints to observe it. Each agent maintains a set of observed available edges E_A and blocked or failed edges E_F throughout their journey. Because each edge can be either available or blocked, a total $2^{|E|}$ scenarios denoted as *weathers* are possible. A weather w is generated by performing a Bernoulli trial on each edge $e \in E$. Only those edges sampled as available (i.e., Bernoulli random variable value is 0) comprise the set $w \subseteq E$. Any weather in which the source and one or more destinations v_l are disconnected are excluded from the study. The objective is to minimize the average travel cost $Z^* = \frac{Min \sum_{l \in L} z_l}{L}$ which is equivalent to minimizing the total travel cost of all agents.

In this paper, the UCT algorithm, originally from Kocsis and Szepesvári (2006) and tailored for the Stochastic CTP as the approach named UCTO in Eyerich et al. (2010), is adapted to solve the multiple agent and multiple destination Stochastic CTP. UCTO refers to the policy used to estimate the cost of moving to new belief states as an agent traverses a network. It is the starting point for the adaptation in this chapter. UCTO is embedded in an online framework as follows. During an agent's journey to its destination, an agent cycles through three stages every time it arrives to a new belief state. A belief state is characterized as a node that has not yet been visited by any agent. The three stages the agent cycles through include sensing and sharing, planning, and moving, as illustrated in Figure 4.1. Agents sense and share information at every belief

state for both UCTO and MAD-HOP. However, UCTO and MAD-HOP differ in what events trigger replanning. For UCTO, every new belief state triggers replanning. For MAD-HOP, only the discovery of one or more failed edges in one or more agents' plans triggers replanning. After replanning (or not), the agent moves to the next belief state and the cycle repeats. These stages are described in more detail in Sections 4.2.1 through 4.2.3. Then in Section 4.2.4 a practical demonstration of the UCTO algorithm is presented.

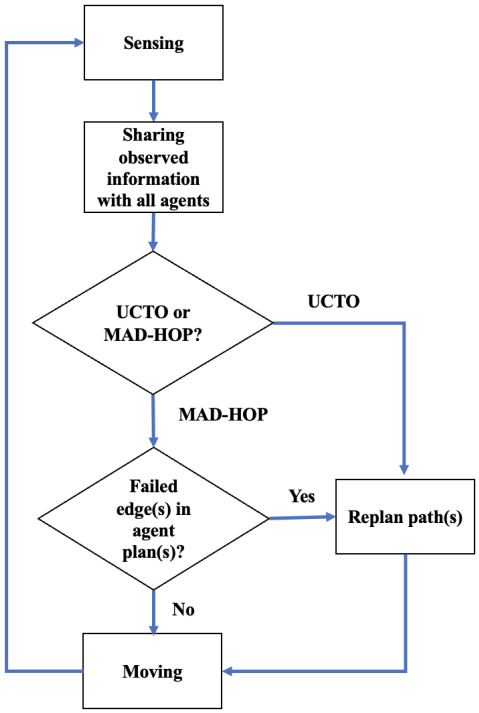


Figure 4.1: Stages agents cycle through at new belief states

4.2.1 Sensing and Sharing

At any given belief state, the agent is capable of disambiguating the status of its adjacent edges to resolve uncertainties regarding unknown roads visible to the agent. The disambiguated road status is categorized as either available or blocked and added to the set E_A or E_F, respectively. This information is simultaneously shared among all agents. Various communication levels are

employed in certain papers, such as no communication, partial communication, and full communication. In no communication scenarios, agents are unable to exchange information, while in partial communication only select agents are permitted to both send and receive information. In contrast, in full communication, all agents can share and receive information. As demonstrated in Chapter 3, the advantages of full communication over no communication have led to the exclusion of other communication levels. That is, only the full communication policy is considered in this chapter.

4.2.2 Planning

During the planning stage, all information collected and shared by all agents up to that point in time is employed to determine the next belief state (i.e., next node) to visit. As MAD-HOP has been explained in Chapter 3, the focus in this chapter is on explaining the mechanics of UCTO. Thus, Section 4.2.2.1 describes how rollouts are generated and how paths are generated for them, and Section 4.2.2.2 describes how the next belief state to visit is selected.

4.2.2.1 Rollout Path Generation

During planning, an agent is at a current node b and information from r rollout weathers is used to determine the agent's next movement. Each rollout weather j contains at least one feasible path from the source node to the destination node and is consistent with all current information the agent has about the graph. That is, the rollout excludes edges known to be failed and includes all edges known to be available. Unlike the MAD-HOP algorithm in which the agent is assumed to have perfect information for edges in the rollout that have not yet been observed, the UCT algorithm assumes the statuses of unobserved rollout edges remain unknown to the agent. The status of the edge is disambiguated only when the agent "virtually" reaches one of its endpoints in rollout weather j while "virtually" following a path for rollout j planned using the UCT algorithm. In other words, inside each rollout j, the agent uses UCT to plan the next node to visit, virtually moves to that node, senses and shares, replans, and then virtually moves again, until the

destination is reached. This cycle takes place while the agent is still physically at node b. This process is repeated for all rollouts, with the rollout path of the j^{th} rollout weather being dependent on all $j-1$ previous rollout paths. The agent only physically moves to a new belief state (i.e., new node) after all such rollout planning and virtual traversal cycles are completed.

To explain how the path of the j^{th} rollout weather is generated, assume rollouts $1,\ldots,j-1$ have already been conducted and those rollout paths are available. Denote the path that will be planned from b to the destination v_d in rollout j as $\sigma_j(v_d)$. Denote an intermediate node between b and v_d in $\sigma_j(v_d)$ as b_i, and the path from b to b_i as $\sigma_j(b_i)$. Assume the agent has virtually traveled to b_i from b in rollout j using the sequence $\sigma_j(b_i) = \langle b, b_1, b_2, ..., b_i \rangle$. Let B_j be a set of nodes reachable from b_i in j using only the edges known to be available in j. Let b_i' be a node in B_j and the travel sequence to reach b_i' from b be $\sigma_j(b_i') = \langle b, b_1, b_2, ..., b_i, k_1, k_2, ...b_i' \rangle$. Here, $k_1, k_2, ...$ represents any nodes along the path from b_i to b_i' that only contains edges known to be available. Let $F(\sigma_j(b_i'))$ be the number of the first $j-1$ rollouts in which the sequence $\sigma_j(b_i')$ was virtually traversed. Let $C_y(b_i', v_d)$ be the cost to reach destination v_d from node b_i' in rollout weather y where $y < j$. It is obtained by adding the edge costs in the sequence $\sigma_y(v_d)$ starting from node b_i'. Denote the average cost to reach the destination v_d from node b_i' across all rollouts in which the sequence $\sigma_j(b_i')$ is virtually traversed as $C(\sigma_j(b_i'))$. It is computed using Equation 4.1:

$$C(\sigma_j(b_i')) = \sum_{y<j:\sigma_y(b_i')=\sigma_j(b_i')} C_y(b_i', v_d). \tag{4.1}$$

Note that $C(\sigma_j(b_i'))$ is an estimate of the uncertain portion of the future travel cost of an agent who has virtually progressed to b_i in rollout j, knows with certainty the cost of moving from b_i to b_i', and does not know with certainty the cost of moving from b_i' to v_d. On the other hand, the certain portion of the agent's future travel cost from b_i to b_i' is denoted $C(b_i, b_i')$; these movements use only edges the agent has observed to be available. Both of these cost components appear in Equation 4.2, as does the frequency metric F. The equation is used to calculate

the UCT value for all b'_i in B_j:

$$UCT = \alpha\sqrt{\frac{logF(\sigma_j(b_i))}{F(\sigma_j(b'_i))}} - C(b_i, b'_i) - C(\sigma_j(b'_i)). \tag{4.2}$$

The parameter α in Equation 4.2 is a bias parameter used to balance exploration and exploitation. The node $b'_i \in B_j$ maximizing the expression in Equation 4.2 is selected as the next node to virtually visit in rollout j. This process continues until the destination is virtually reached. Note that when this process is conducted for the first rollout, $r = 1$, there are no other rollouts available so the frequency metrics F would be zero. To avoid this, $F(\sigma_j(b'_i))$ is initialized to a constant value M and $C(\sigma_j(b'_i))$ is initialized using the optimistic cost of the path to the destination, which is the shortest path distance from b'_i to v_d in rollout 1 assuming all edges in the rollout with unknown status are available. The inclusion of M additional rollouts is how the UCT policy is tailored as the optimistic UCT, called UCTO. The bias parameter α can be initialized to an arbitrary value in the first rollout. This is due to the fact that during the first rollout, the values of $F(\sigma_j(b_i))$ and $F(\sigma_j(b'_i))$ are equal to M for all b'_i in B_j. As a result, the UCT value in Equation 4.2 is not affected by the first term in the equation. Consequently, the selection of the next node in the rollout path is not influenced by this term in the first rollout weather. In other rollouts $j > 1$, α is set to the average cost to reach the destination from the current node b across the previous $j-1$ rollouts, according to Equation 4.3:

$$\alpha = \sum_{y<j} C_y(b, v_d). \tag{4.3}$$

In order to promote and incentivize greater levels of exploration, the value of alpha is divided by a factor of ten.

4.2.2.2 Selecting the Next Agent Movement

The discussion in Section 4.2.2.1 describes how rollout paths are determined and virtually traversed. At the completion of that process, there exists a rollout path $\sigma_j(v_d)$ that begins at the cur-

rent node b and ends at v_d for all rollouts $j = 1, \ldots, r$. The next requirement is to determine the next physical movement (not virtual movement) for an agent from node b to a new belief state. A potential new belief state is determined for each rollout j by scanning the nodes in $\sigma_j(v_d)$ to identify the first node in the sequence that has not been visited before. Denote the set of all potential new belief states across all rollout paths as B. Let b_i be a potential new belief state in B. Define $U(b_i)$ as the total number of rollout paths in which b_i is the next belief state. Define $V(b_i)$ as the average cost of the paths from b_i to v_d in those rollout paths $\sigma_j(v_d)$ where b_i is the next belief state. Then, the expected cost for a belief state $b_i \in B$ is computed using Equation 4.4:

$$E(b_i) = C(b, b_i) + \frac{V(b_i)}{U(b_i)}. \tag{4.4}$$

The belief state $b_i \in B$ minimizing Equation 4.4 is selected as the next belief state for the agent to physically move to.

4.2.3 Moving

During the moving stage, the agent transitions to the subsequent belief state node that was determined during the planning stage.

4.2.4 UCTO in Practice

The example in this section is provided as a simple demonstration of UCTO. The graph in Figure 4.2 is from Eyerich et al. (2010) and includes a source node v_0 and a destination node v^*. For edges with two labels, the first indicates the blocking probability p_e for the edge and the second indicates the edge cost c_e. Edges incident to v_0 have only a single label indicating cost, as those edges are visible to the agent from v_0 and are observed to be available. The symbol ε is used to represent a very small probability of blockage. Given that ε is very small, we assume that the edges $\{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_5, v_6), (v_5, v^*)\}$ are available, while edge (v_6, v^*) is blocked. As a result, the graph is reduced to only three stochastic edges, leading to a total of 2^3 possible weathers. To determine the next node to visit from the source node v_0, we consider all eight pos-

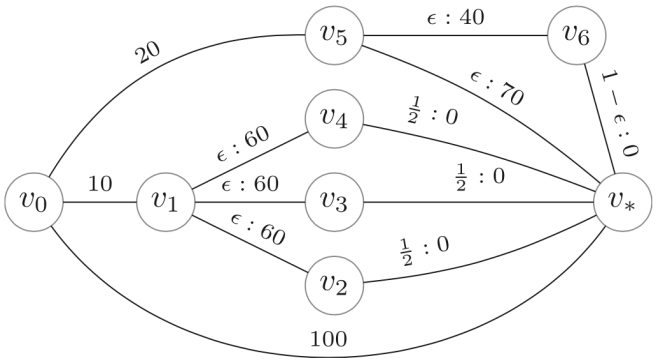


Figure 4.2: UCTO example

sible combinations of rollout weathers, as listed in Table 4.1. For each scenario, the rollout paths are obtained using the UCTO algorithm described in Section 4.2.2.1, starting from node v_0. The values of M and α are initialized in the first rollout to 20 and 1, respectively. In subsequent rollouts, the value of α is determined using Equation 4.3 and these values are provided in Table 4.1.

In the first rollout, the agent has three possible next nodes to visit: v_5, v_1, and v^*. That is, $B = \{v_5, v_1, v^*\}$. If the agent chooses v_5, the UCTO value obtained using Equation 4.2 from v_0 to v^* would be ≈ 60 ($v_0 - v_5 - v^*$). If the agent chooses v_1, the UCTO value would be ≈ 70 ($v_0 - v_1 - v_4 - v^*$). If the agent chooses v^* directly, the UCTO value would be ≈ 100 ($v_0 - v^*$). However, during the first rollout, the agent encounters a blockage at v_6 and must travel back, resulting in a total travel cost of 170. In the next rollouts, the UCTO value of choosing v_5 increases slightly, but not enough to make v_1 or v^* more attractive than v_5. During the second attempt, the agent again encounters blockage at v_6, which suggests that choosing v_5 may not be the best option. In the third rollout, the UCTO value of choosing v_5 increases enough to make v_1 or v^* more attractive than v_5 and hence in the third rollout the agent decides to choose v_1 as the next node to visit. This logic continues for the remaining rollout weathers.

After conducting all the rollout simulations, the expected cost for each node in B is calculated using Equation 4.4. For example, the expected cost for node v_1 is $E(v_1) = 20 + (180 + 300)/2 = 260$, based on the results from rollouts 3 and 6. Similarly, the expected cost for v_5 is $E(v_5) = 10 + (150 \text{ x } 6)/6 = 160$, based on the results from rollouts 1, 2, 4, 5, 7, and 8. Because

v^* didn't appear as the first unvisited node after v_0 in any rollout weather, the expected cost for v^* is not calculated and it is discarded as the next possible node to visit. The expected cost of node v_5 is the least and it is chosen as the next node to be visited. The same process is used to determine the next node to visit from v_5, and it is found that the next node to visit is v^*. Therefore, the agent's path is $\langle v_0, v_5, v^* \rangle$ with a path cost of 90 units.

Table 4.1: Rollout weathers and paths (agent at v_0)

#	Failed Edges	Path	α
1	(v_4, v^*)	$\langle v_0, v_5, v_6, v_5, v^* \rangle$	-
2	(v_3, v^*)	$\langle v_0, v_5, v_6, v_5, v^* \rangle$	170.00
3	(v_2, v^*)	$\langle v_0, v_1, v_2, v_1, v_3, v^* \rangle$	170.00
4	$(v_3, v^*), (v_4, v^*)$	$\langle v_0, v_5, v_6, v_5, v^* \rangle$	176.67
5	$(v_2, v^*), (v_4, v^*)$	$\langle v_0, v_5, v_6, v_5, v^* \rangle$	175.00
6	$(v_2, v^*), (v_3, v^*)$	$\langle v_0, v_1, v_3, v_1, v_2, v_1, v_4, v^* \rangle$	174.00
7	$(v_2, v^*), (v_3, v^*), (v_4, v^*)$	$\langle v_0, v_5, v_6, v_5, v^* \rangle$	196.67
8	–	$\langle v_0, v_5, v_6, v_5, v^* \rangle$	192.86

4.3 Experimental Details

In order to assess the efficacy of the UCTO policy, we conducted performance evaluations on two distinct graph types: Delaunay and Euclidean. Detailed descriptions of these graph types are provided in Sections 4.3.1 and 4.3.2, respectively.

4.3.1 Delaunay Graphs

Three sizes of Delaunay graphs are utilized in this computational study, comprised of 20, 50 and 100 nodes. Ten graphs are generated for each graph size, which are directly obtained from Eyerich et al. (2010). For each graph, the lowest numbered node is denoted as the source node v_s. Different numbers of agents are utilized for each graph size, with 2 agents for the 20 node graphs, 3 and 5 agents for the 50 node graphs, and 3, 5, and 10 agents for the 100 node graphs. Other problem instance parameters are identical to those in Section 3.4.1. For example, destination nodes v_l for each agent are selected from among the nodes located farthest from the source.

Edge costs c_e and blockage probabilities p_e follow uniform distributions on the intervals [1,50] and [0,1), respectively. For each of the 30 graphs, 100 weathers are generated, along with 1000 rollouts for each weather, again identical to those in Section 3.4.1. The value of parameter M is intialized to 20, as in (Eyerich et al., 2010). Overall, a total of 6000 Delaunay graph test instances are included.

4.3.2 Euclidean Graphs

The Euclidean graphs in the computational study are identical to those in Section 3.4.2. Three different sizes of Euclidean graphs consisting of 100, 200, and 300 nodes are generated from a 100 x 100 grid. For each graph size, three distinct graphs are generated. For each graph, nodes are randomly selected with coordinates (x, y) from a discrete uniform distribution on the range $\{1, 2, \ldots, 100\}$. The lowest index node is marked as the source node v_s. The number of agents involved in each graph depends on the size of the graph, with 3, 6, and 9 agents used for the 100 node graphs, 4, 8, and 12 agents utilized for the 200 node graphs, and 5, 10, and 15 agents utilized for the 300 node graphs. Edges are added randomly between generated nodes, with edge costs c_e determined from the Euclidean length of the edge. The number of edges in a particular graph depends on the graph size. For 100, 200, and 300 node graphs, the numbers of edges are 300, 600, and 900 respectively. For each of the 9 distinct graphs, four edge failure scenarios are generated, utilizing edge blockage probabilities of 10%, 20%, 30%, and 40%. In a given graph and scenario, it is assumed that all edges are subject to the same probability of failure. For each of 36 graphs, 32 weathers are generated, along with 1000 rollouts for each weather. The parameter M is intialized to 20 as in Eyerich et al. (2010). With 36 graphs, 32 weathers for each graph and three distinct numbers of agents, a total of 36*32*3 = 3456 Euclidean graph test instances are included. The weathers utilized in this study differ from those employed in Shiri and Salman (2019).

4.3.3 Destinations

In order to maintain consistency across all graphs examined, a standardized method is adopted for generating l destination nodes. This method entails assigning the lowest index node as the source node and computing the shortest distance from the source node to every other node in the graph. These shortest path distances are subsequently sorted in a non-increasing list. Then, l destination nodes are randomly selected from the top 20% of the list.

4.4 Computational Results

This section presents a performance evaluation of the UCTO algorithm proposed for the MAD-CTP variant. In Section 4.4.1, the average cost $\bar{Z}^*$ is compared to that of MAD-OMT and MAD-HOP algorithms discussed in 3. In addition to this cost comparison, Section 4.4.2 presents a re-planning event duration and count analysis.

4.4.1 Average Cost

Table 4.2 provides the average travel cost across all agents for Delaunay graphs obtained using the UCTO algorithm from this chapter and MAD-OMT and MAD-HOP from Chapter 3. Each value in the table under the MAD-OMT, MAD-HOP, and UCTO column headers represents an average cost over 100 weathers, except for the *Avg* row, which is an average over 1000 weathers (10 graphs with 100 weathers each). The first column of the table provides instance names T-$|V|$-L-Q, where T indicates graph type (D for Delaunay), $|V|$ is the number of nodes in the graph, L is the number of agents, and Q is the graph replicate number. Bold values are used to indicate which algorithm provides the best average travel cost for each graph replicate. For example, in instance D-20-2-1, the average cost of 137.29 units found under column MAD-HOP and in bold indicates MAD-HOP obtains better performance than MAD-OMT and UCTO for this 20-node Delaunay graph with 2 agents.

In comparative analysis, it can be observed that UCTO consistently outperforms MAD-

Table 4.2: Average cost for Delaunay graphs

T-$\|V\|$-L-Q	MAD-OMT	MAD-HOP	UCTO	T-$\|V\|$-L-Q	MAD-OMT	MAD-HOP	UCTO
D-20-2-1	159.07	**137.29**	141.42	D-100-3-1	331.97	**248.64**	259.74
D-20-2-2	140.44	**127.94**	129.51	D-100-3-2	305.51	**247.53**	260.89
D-20-2-3	177.06	**165.78**	168.76	D-100-3-3	440.83	**350.55**	366.93
D-20-2-4	285.49	194.18	**191.33**	D-100-3-4	335.52	**292.36**	306.74
D-20-2-5	185.05	156.81	**151.66**	D-100-3-5	482.63	**389.36**	409.03
D-20-2-6	165.94	136.10	**132.41**	D-100-3-6	416.21	**369.54**	384.35
D-20-2-7	178.07	**155.40**	159.00	D-100-3-7	346.10	**320.65**	323.92
D-20-2-8	224.84	**172.35**	175.22	D-100-3-8	276.27	223.00	**218.54**
D-20-2-9	190.18	149.04	**147.29**	D-100-3-9	344.22	**301.62**	308.56
D-20-2-10	145.29	144.79	**143.80**	D-100-3-10	328.71	**292.43**	304.12
Avg	185.14	**153.96**	154.04	Avg	360.80	**303.57**	314.28
D-50-3-1	226.71	195.16	**184.58**	D-100-5-1	362.48	**268.15**	277.63
D-50-3-2	429.66	**335.22**	340.61	D-100-5-2	264.78	**221.18**	237.16
D-50-3-3	305.94	**241.01**	245.10	D-100-5-3	438.33	**362.24**	367.75
D-50-3-4	233.89	**211.63**	217.08	D-100-5-4	338.02	**283.80**	290.97
D-50-3-5	332.27	258.61	**247.00**	D-100-5-5	408.98	**310.84**	321.93
D-50-3-6	246.19	215.31	**209.32**	D-100-5-6	386.41	**342.95**	355.59
D-50-3-7	229.50	**203.93**	208.27	D-100-5-7	346.40	315.62	**311.03**
D-50-3-8	345.99	281.08	**279.35**	D-100-5-8	276.12	**231.45**	233.76
D-50-3-9	277.93	240.93	**230.69**	D-100-5-9	339.85	**284.40**	302.96
D-50-3-10	266.65	**223.13**	226.29	D-100-5-10	327.17	299.57	**291.41**
Avg	289.47	240.60	**238.83**	Avg	348.85	**292.02**	299.02
D-50-5-1	232.04	204.88	**202.90**	D-100-10-1	306.02	**246.11**	254.01
D-50-5-2	366.61	**299.58**	305.49	D-100-10-2	265.40	**227.65**	232.58
D-50-5-3	294.81	253.76	**251.33**	D-100-10-3	373.00	**308.31**	327.34
D-50-5-4	251.23	**207.74**	218.07	D-100-10-4	344.49	**292.88**	302.79
D-50-5-5	321.93	252.85	**248.76**	D-100-10-5	449.15	**346.54**	364.28
D-50-5-6	236.50	210.49	**208.27**	D-100-10-6	343.47	**307.89**	316.91
D-50-5-7	204.95	**170.69**	175.11	D-100-10-7	292.48	**257.08**	267.39
D-50-5-8	319.38	264.59	**258.61**	D-100-10-8	305.54	**240.72**	243.73
D-50-5-9	276.88	230.52	**226.82**	D-100-10-9	308.26	**280.15**	289.74
D-50-5-10	237.39	**193.38**	199.50	D-100-10-10	313.60	**266.14**	269.01
Avg	274.17	**228.85**	229.49	Avg	330.14	**277.35**	286.78

OMT in terms of average cost across all instances. However, when compared with the MAD-HOP, the results are mixed. Specifically, out of 60 graphs, UCTO demonstrates superior average cost for 19 graphs. Among these 19 graphs, 5 are size 20, 11 are size 50, and 3 are size 100. The average improvements of UCTO over MAD-HOP for these 19 graphs are 1.9% for the 5 20-node graphs, 2.4% for the 11 50-node graphs, and 2.1% for the 3 100-node graphs. While the average improvement of UCTO over MAD-HOP remains below 2.5% for these 19 graphs, certain individual graphs display greater levels of improvement. For example, for D-20-2-5 graph,

the improvement is 3.3%. For D-50-3-1, D-50-3-5, and D-50-3-9, the improvements are 5.4%, 4.5%, and 4.3%, respectively. On the other hand, in the 41 out of 60 graphs where UCTO shows inferior average performance, the average cost increase over MAD-HOP is 2.0% for the 5 20-node graphs, 2.5% for the 9 50-node graphs, and 3.6% for the 27 100-node graphs. Analyzing the *Avg* row, differences between MAD-HOP and UCTO for 20- and 50-node graphs are negligible. However, for 100-node graphs, MAD-HOP produces solutions with 3% lower average travel cost than UCTO.

Despite being regarded as a more advanced and sophisticated algorithm, the UCTO resulted in increases in average cost when compared with MAD-HOP for the majority of test instances. Notably, UCTO was only executed with 1000 rollouts, which may account for the suboptimal outcomes. Previous research in Eyerich et al. (2010) indicates that increasing the number of rollouts can improve the average cost in single-agent Stochastic CTP variants. However, for the multiple agent and multiple destination test instances in our study, increasing the number of rollouts beyond 1000 results in a substantial escalation in runtime, rendering it impractical.

To better understand the effect of number of rollouts on UCTO performance for this problem variant, we conducted an abbreviated computational study on the 10 20-node graph replicates using 10,000 rollouts instead of 1,000. The average travel cost results from MAD-HOP (with 1,000 rollouts), UCTO-1000RW (with 1,000 rollouts) and UCTO-10000RW (with 10,000 rollouts) are provided in Table 4.3. Bold values are used to indicate which algorithm provides the best average travel cost for each graph replicate. For 5 of 10 replicates, the average travel cost of solutions produced by UCTO with 10,000 rollouts improved, compared with 1,000 rollouts. These are replicates 1, 2, 3, 7, 9 and 10 of the 20-node Delaunay graphs with 2 agents. However, MAD-HOP still offers better performance for 3 of these replicates (numbers 1, 2 and 3), and UCTO with 1,000 rollouts offers better average performance with 10,000 rollouts for 4 replicates (numbers 4, 5, 6 and 8). Therefore, it is unclear whether increasing the number of rollouts will consistently lead to better performance for multiple agent and multiple destination Stochastic CTP test instances.

Table 4.3: Average cost for 20 node Delaunay graphs with 2 agents

T-$\|V\|$-L-Q	MAD-HOP	UCTO-1000RW	UCTO-10000RW
D-20-2-1	**137.29**	141.42	137.30
D-20-2-2	**127.94**	129.51	127.95
D-20-2-3	**165.78**	168.76	167.10
D-20-2-4	197.18	**191.33**	194.13
D-20-2-5	156.81	**151.66**	152.77
D-20-2-6	136.10	**132.41**	133.27
D-20-2-7	155.40	159.00	**153.79**
D-20-2-8	**172.35**	175.22	176.45
D-20-2-9	149.04	147.29	**146.36**
D-20-2-10	144.79	143.80	**143.45**
Avg	153.96	154.04	**153.25**

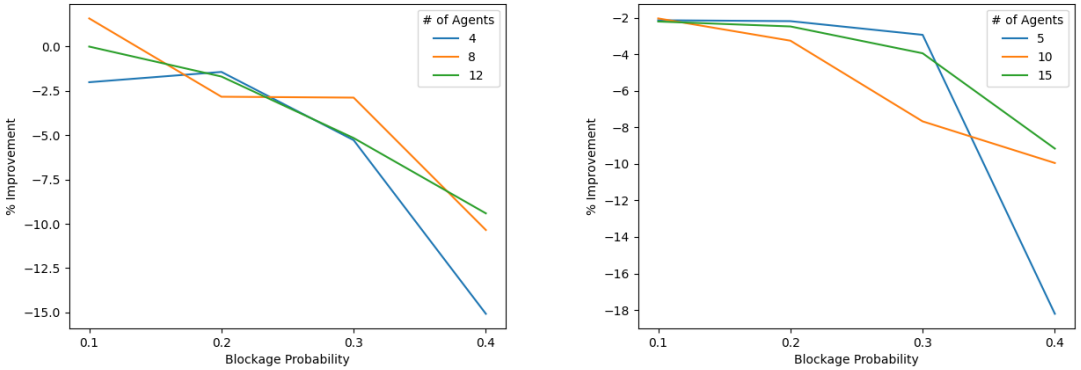
Table 4.4 provides average cost outputs for the Euclidean graph test instances. It can be observed that MAD-HOP consistently outperforms UCTO in all rows. On average, UCTO results in a 4.8% increase in average travel cost compared with MAD-HOP when utilized on these Euclidean graphs.

Table 4.4: Average cost for Euclidean graphs

T-$\|V\|$-L-Q	MAD-OMT	MAD-HOP	UCTO
E-100-3	272.26	**263.14**	274.56
E-100-6	267.11	**255.81**	257.74
E-100-9	258.33	**247.88**	258.05
E-200-4	307.32	**291.11**	312.49
E-200-8	295.77	**280.38**	297.62
E-200-12	296.30	**282.08**	296.00
E-300-5	334.87	**311.24**	336.42
E-300-10	332.69	**314.00**	334.58
E-300-15	329.07	**309.88**	312.70

A further investigation of the data disclosed that as the blockage probability in the graph increases, the difference in average cost between UCTO and MAD-HOP also increases. These results are depicted in Figures 4.3a and 4.3b. For a 200-node graph and blockage probability of 10%, UCTO and MAD-HOP exhibit comparable performance for all numbers of agents, as seen in Figure 4.3a. However, as the blockage probability increases, the percent difference in average cost between the two algorithms becomes more pronounced for all numbers of agents. The aver-

age performance of UCTO at $p_e = 0.40$ in 200-node Euclidean graphs is inferior to MAD-HOP by approximately 15%, 10%, and 9% for 4, 8, and 12 agents, respectively. This trend is also evident in 300-node Euclidean graphs, especially for $p_e = 0.40$, as demonstrated in Figure 4.3b.



(a) 200 node Euclidean graph (b) 300 node Euclidean graph

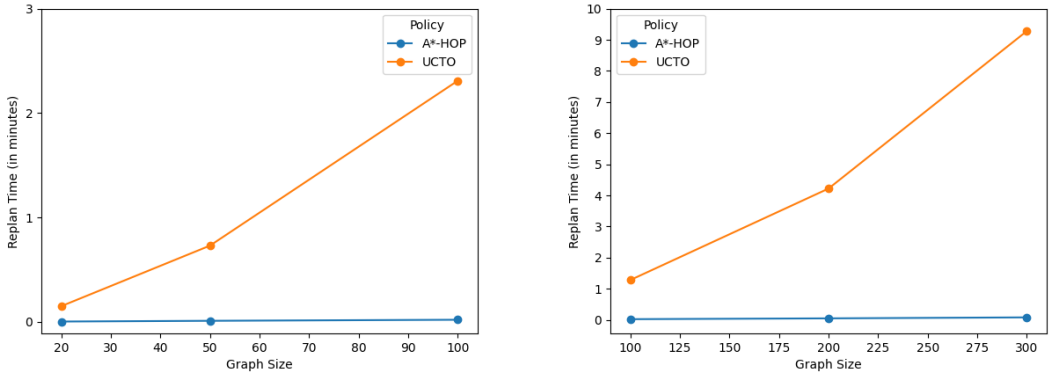
Figure 4.3: Blockage probability

It is noteworthy that as the number of agents increases, the percentage difference in average travel cost between UCTO and MAD-HOP diminishes. For example, in Figure 4.3a, at 40% blockage probability, UCTO's performance was poorest by approximately 15% with four agents. Still, this percentage decreased to approximately 10% with eight agents and further reduced to 9% with twelve agents. This implies that when the number of agents increases, sharing information among them becomes more advantageous, reducing the performance discrepancy between the two algorithms. The subsequent section discusses the replanning time and replanning events.

4.4.2 Number and Duration of Replanning Events

In addition to understanding the solution quality for UCTO and the MAD-HOP reference approach, it is also important to evaluate runtime aspects of the approaches. Rather than focus on total runtime, we investigate computation times that reflect delays in agent movement as they physically traverse a graph. We record the number of replanning events and their durations for both approaches.

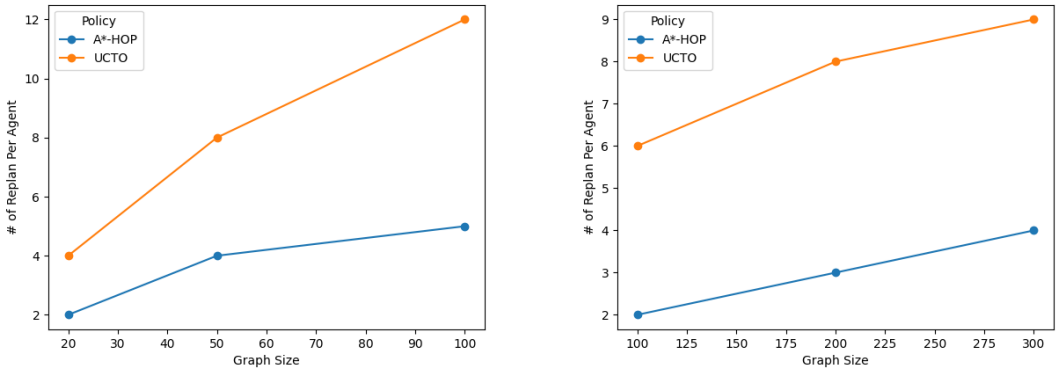
Figure 4.4 depicts the average duration of a replanning event for both approaches for Delaunay graph and Euclidean graph test instances. The findings demonstrate that for 20-node Delaunay graphs, the average duration of a UCTO replanning event is 0.15 seconds, increasing to 2.31 minutes for 100-node graphs, as depicted in Figure 4.4a. As the graph sizes increase further, UCTO replanning event durations also increase, reaching 4.22 and 9.28 minutes for Euclidean graphs with 200 and 300 nodes, respectively, as seen in Figure 4.4b. These durations are significantly higher than MAD-HOP. For example, the duration of MAD-HOP replanning events is less than 5 seconds on all Euclidean graphs.



(a) Delaunay graphs (b) Euclidean graphs

Figure 4.4: Average replanning event durations for Delaunay and Euclidean graphs

The total delay during agent graph traversals can be assessed by examining the number of replanning events required. Figure 4.5 provides replanning event counts for both UCTO and MAD-HOP on Delaunay and Euclidean graph test instances. For 20-node Delaunay graphs, UCTO requires 4 replanning events, on average per agent. As the Delaunay graph size increases to 100 nodes, the number of replanning events required in UCTO increases to 12 (see Figure 4.5a). A similar trend is observed in Figure 4.5b for Euclidean graphs with 40% blockage probability, with 9 UCTO replanning events for the largest graph in the study, consisting of 300 nodes. Notably, MAD-HOP requires fewer replanning events than UCTO, as it only replans when a blocked edge is encountered, unlike UCTO, which replans at every new belief state. The approximate total replanning duration across full graph traversal for 300-node Euclidean graphs



(a) Delaunay graphs (b) Euclidean graphs

Figure 4.5: Numbers of replanning events for Delaunay and Euclidean graph test instances

is 9*9.28 = 83.52 minutes, compared with 3*4.93 = 14.79 seconds for MAD-HOP.

4.5 Conclusion and Future Work

The present study aims to understand the suitability of the UCTO algorithm for the multiple agent multiple destination variant of Stochastic CTP. Unlike other methods, the UCTO algorithm does not prescribe a pre-determined path for agents to follow, but rather replans at every belief state to select the next node to visit. The performance of the UCTO algorithm was compared against MAD-OMT and MAD-HOP from the existing literature, and the evaluation was conducted on Delaunay and Euclidean graphs.

The results indicate that UCTO outperforms MAD-OMT on all instances tested. When compared against MAD-HOP, UCTO exhibits superior performance on 19 out of 60 Delaunay graphs. However, for the remaining Delaunay graphs, the average cost increases by 2.0%, 2.5%, and 3.6% for 20, 50, and 100 node graphs, respectively. Moreover, UCTO shows an increased average cost of 4.8% on all instances of Euclidean graphs. Previous research in Eyerich et al. (2010) suggests that more than 1000 rollouts are required for the UCTO to converge for single-agent CTP variants, and using more rollouts may lead to better performance. A small experiment is conducted with 10,000 rollouts on a 20 node graph. In it, the average cost was only improved

for a subset of test instances, compared to when using 1000 rollouts. Hence, it remains unclear whether more rollouts will provide better performance for other instances of the multiple agent and multiple destination Stochastic CTP.

The study indicates that as the level of disruption in a network increases, evidenced by increasing probabilities of edge blockage, especially in Euclidean graph test instances, the performance disparity between UCTO and MAD-HOP widens. It is interesting to observe that as the number of agents increases and the level of disruption is held constant, the performance gap between UCTO and MAD-HOP narrows, indicating the advantages of sharing information between agents may lead to UCTO convergence faster with smaller number of rollouts. The analysis of replanning durations revealed a substantial computational workload that incurs a delay of over 80 minutes for the largest graphs in the computational study, which may render the UCTO policy impractical for such graphs.

Future research avenues include more efficient implementations of UCTO to reduce its replanning event duration, which would enhance its practicability and allow for more rollouts. A comprehensive analysis of the extent to which the benefits of sharing can reduce the number of rollouts required for convergence of the average cost in UCTO is another possible research direction. In addition, deploying MAD-HOP and UCTO on real-world networks will provide evidence of their practicality for realistically-sized problem scenarios.

Bibliography

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1991). *Real-time learning and control using asynchronous dynamic programming*. University of Massachusetts at Amherst, Department of Computer and ....

Eyerich, P., Keller, T., and Helmert, M. (2010). High-quality policies for the canadian traveler's problem. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*, pages 282–293. Springer.

Lim, Z. W., Hsu, D., Lee, W. S., and Sun, W. (2017). Shortest path under uncertainty: Exploration versus exploitation. In *UAI*.

Nikolova, E. and Karger, D. R. (2008). Route planning under uncertainty: The canadian traveller problem. In *AAAI*, pages 969–974.

Papadimitriou, C. H. and Yannakakis, M. (1991). Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150.

Péret, L. and Garcia, F. (2004). On-line search for solving markov decision processes via heuristic sampling. *learning*, 16:2.

Shiri, D. and Salman, F. S. (2019). Online optimization of first-responder routes in disaster response logistics. *IBM Journal of Research and Development*, 64(1/2):13–1.

5. Conclusions

This dissertation studies Stochastic CTP variants and reports on computational findings obtained for varying problem variants and experimental parameters. We focus on two variants of Stochastic CTP, namely SS-SD-SA and SS-MD-MA, the latter which is also called MAD-CTP. For SS-SD-SA, we propose two novel rollout-based algorithms, Maximum Likely Node and Maximum Likely Path. Both these algorithms utilize a consensus function to determine the next node to visit. Chapter 3 introduces MAD-OMT and MAD-HOP algorithms that plan a path for agents until they encounter a blocked edge, while Chapter 4 discusses a UCT policy where the agent decides on the next node to visit, causing them to wait at every intersection, making it less practical for real-life scenarios. We compare the average cost and replan time of the UCT policy to MAD-HOP algorithm. Additionally, we investigate the benefits of information sharing for MAD-OMT and MAD-HOP algorithms.

Chapter 2 presents a detailed comparison of our proposed algorithms, Maximum Likely Node (MLN) and Maximum Likely Path (MLP), against various policies. Our approaches demonstrate superior performance when compared to OMT and UCTB policies on all test instances studied, and nearly outperform HOP on all instances. Furthermore, our algorithms nearly outperform ORO and A*-HOP on half of the instances evaluated. The experimental analysis also reveals that our algorithm achieves better average costs for two graphs against all reference approaches, although the difference was not deemed to be statistically significant. Additionally, we conduct a comparative analysis of A*-HOP against MLP and MLN, utilizing weather level data, and we observe that MLP and MLN together yield new best-known solutions in 10,715 out of 30,000 instances. The replan time for both algorithms is recorded to be approximately 15 seconds for a 100 node graph size.

In chapter 3 MAD-HOP outperforms MAD-OMT on all Delaunay and Euclidean graph instances with an average improvement of 14% to 19% and 3% to 8% respectively on the travel cost for each agent. The improvement differences between Delaunay and Euclidean graphs was

found to primarily depend on differences in each blockage probabilities. The analysis concludes that the higher the blockage probability, the higher is the improvement against MAD-OMT. Thus MAD-HOP is beneficial to use in the graphs with higher blockage probabilities. Because multiple agents are involved, the benefit of sharing information is analyzed and it is found that even when the agents travel to different destinations, agents benefit by a significant amount. As the number of agents increases the benefit increases. For a 500 node Euclidean graph with 21 agents, the benefit is nearly 7%. In other words, the average cost for each agent improves by 7% over the no sharing case. The replan time is recorded as 1.21 seconds for a 100 node Delauany graph and 8.69 seconds for a 500 node Euclidean graph. This small amount of replan time makes the MAD-HOP more practical.

In chapter 4 of this dissertation, we explore the implementation of the state-of-the-art UCT algorithm to solve MAD-CTP. Surprisingly, our results indicate that UCT is able to provide better costs on only 21 out of 60 Delaunay graphs and on none of the Euclidean graphs. This observation may be attributed to the number of rollouts employed, which we limited to 1000. Increasing the number of rollouts may improve the performance of UCT, as previously demonstrated for the single-agent single-destination variant mentioned in Eyerich et al. (2010). To investigate this possibility, we conduct a small experiment using 10,000 rollouts on 20-node graphs. However, our findings reveal that only on 6 out of 10 graphs does the average cost improve when 10,000 rollouts are used, compared to the results obtained using 1000 rollouts. Due to computational intractability, using 10,000 rollouts for larger-sized graphs is not feasible. Therefore, from the results obtained for 20-node graphs, it remains unclear whether UCT will outperform MAD-HOP or not. Even if UCT outperforms MAD-HOP, the replan time for a 300-node Euclidean graph is recorded to be 9.28 minutes compared to 4.93 seconds for MAD-HOP, which represents a significant increase of 99%.

Future research related to the MAD-CTP variant involves reducing the replan time for the UCT policy, which can enhance its practicality and allow for more rollouts. Additionally, a com-

prehensive analysis of the extent to which the benefits of sharing information could reduce the required number of rollouts for convergence of average cost is a promising research direction. Another vital aspect of future research is the deployment of the MAD-HOP and UCT algorithms on real-world networks and selecting real destination locations. Notably, in this study, the destinations were chosen based on comparable shortest path distances, which may not be realistic in actual scenarios. Therefore, to validate the practical feasibility of the proposed approaches for MAD-CTP, it is essential to test the framework on a real network and evaluate its performance.

Another direction for future work is to include variants that model edges as being recoverable. These variants allow for blocked roads to be cleared either by the agent or on their own, such as when floodwaters recede. Although the recoverable CTP for a single agent and destination was introduced nearly two decades ago, there is a lack of computational research on this problem variant in current literature. Another potential direction is to classify edges as deterministic and transform the problem into a k-CTP variant, which is extensively studied in literature. Interdependent edges are frequently encountered in real-life scenarios, and therefore, it would be beneficial to test our proposed approach on variants that incorporate edge dependence in order to evaluate its effectiveness.