Routing in stochastic networks

Routing in Stochastic Networks

This thesis is number D180 of the thesis series of the Beta Research School for Operations Management and Logistics. The Beta Research School is a joint effort of the departments of Industrial Engineering & Innovation Sciences, and Mathematics and Computer Science at Eindhoven University of Technology and the Center for Production, Logistics and Operations Management at the University of Twente.

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN: 978-90-386-3530-9

Cover design by Fırat Gelbal (Ufak Tefek Tasarım)

Cover Design is inspired from the mobiles of Alexander Calder.

Printed by Proefschriftmaken.nl- Uitgeverij BoxPress

This research has been funded by Eyefreight- ITUDE, The Netherlands

Routing in Stochastic Networks

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen
op dinsdag 21 januari 2014 om 16:00 uur

door

Derya Sever

geboren te Ankara, Turkije

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. A.G.L. Romme
1^e promotor:	prof.dr. T. van Woensel
2^e promotor:	prof.dr. A.G. de Kok
copromotor:	dr.ir. N.P. Dellaert
leden:	dr. L. Zhao (Tsinghua University)
	prof. dr. F.C.R. Spieksma (Katholieke Universiteit Leuven)
	prof.dr.ir. I.J.B.F. Adan
reserve:	prof.dr.ir. J.C. Fransoo

To my mother, sister and father,
to Fırat,
and in the memory of my grandmother...

Annem'e, Deniz'e, Babam'a,
Fırat'a,
ve en derin özlemle Anneannem'e...

Long you live and high you fly
And smiles you'll give and tears you'll cry
And all you touch and all you see
Is all your life will ever be.

Pink Floyd

Acknowledgements

My life in OPAC department started in 2008 with my master studies so without a doubt it would be unfair to consider my PhD as only completing a thesis or publishing papers. This was a long road where my life and my perspectives changed in many ways. This PhD research would not be colorful without the invaluable support of my family, my supervisors, mentors and my friends.

Foremost, I owe my PhD research and this thesis to Tom van Woensel, my first promotor and Nico Dellaert, my copromotor. Thank you Nico and Tom for your encouragement, sharing your invaluable knowledge and experience with me and giving your time endlessly. I really appreciate your valuable feedbacks to improve my skills especially in writing which required patience. I learned a lot from you by different perspectives you brought into the research which made this experience richer and in harmony. Especially, I will always remember how we climbed to the Chinese Wall together.

Tom, I would like to thank you for all your contributions of time, ideas, and funding to make this PhD research productive and interesting. You provided me with gentle encouragement and enough freedom to take initiatives which created a good working environment and a motivation for me to finish. I admire your enthusiasm and patience in dealing with me and teaching me how to do research. I am grateful for mentoring me to understand the way of analyzing and solving routing problems.

Nico, I am indebted and extremely grateful to you for your endless support, contribution and guidance. This thesis would not exist without your ideas and vision. Thank you for all the fruitful discussions in general and ideas on the methodological approaches. I am amazed with your knowledge and skills in operations research. I hope I grasped even a little part of your knowledge and curiosity in exploring. Besides, for me it was really valuable that you helped me whenever I came to your office and shared your ideas and knowledge with me.

I also would like to express my gratitude to my second promotor Ton de Kok for his invaluable feedback and ideas. In our monthly meetings, you brought a broader perspective to this research. Especially in writing the thesis, your feedback improved my perspective in writing and considering our research from a macro level.

I would like to deeply thank Lei Zhao for hosting my visit in Tsinghua University in Beijing and his invaluable contribution to the Chapters 4 and 6 of this thesis. During this visit, you ensured that I was feeling at home. You showed me the endless hospitality of China. I owe the most of my knowledge on approximate dynamic programming to your knowledge and experience. With your supervision, I learned how to be more critical and detailed in analyzing and writing. This is really valuable for me who needs improvement in different ways. I am extremely grateful for your valuable feedback and spending your time for this research even in your holidays. I would also like to thank Chen and Zhaoxia for helping me in Beijing and introducing me to the Chinese culture.

I would like to thank Ivo Adan and Frits Spieksma for being part of my thesis committee and for your insightful comments and invaluable feedback on my thesis. Your feedback provided me to consider my thesis in a more rigorous way and helped me to improve it substantially. Furthermore, I would like to thank Jan Fransoo for being part of my defense committee.

I also gratefully acknowledge EyeFreight- Itude for funding my research and making all of these moments possible.

I would like to express my gratitude to Carlo van de Weijer for sharing his invaluable experiences and knowledge with me. With his help, I have a clearer vision on how TomTom deals with uncertainties and the future of navigation.

I would like to thank all my current and former colleagues in OPAC who provided me with a cool and relaxed working environment that facilitated doing research. When I look back, I see that I had many roommates which made me lucky while sharing and exchanging more stories. I would like to thank my roommates over these years: Frank, Duygu, Kristina, Kristel, Maxi and Hande. Thanks for the friendly atmosphere and dealing up with my high entropy. Also, I really appreciate the nice conversations with Gönül, Anna, Kasper, Sjaña, Zümbül, İpek, Engin, Maryam, Chiel, Stefano, Joachim, Frank, Josué and Baoxiang.

Many thanks to Florida, José, Claudine, Ineke and Christel for the nice and relieving chats and helping me by solving the most difficult scheduling problems.

My life here became joyful and interesting with the presence of my friends who became my family in the Netherlands for more than 6 years now. I feel really lucky to have you all. Not only we enjoyed our times together but we all shared our dreams, concerns and views with each other. This is really precious to me. I wish our moments will continue to be joyful and colorful together: Memo, Melike, İrem, Görkem, Sinan, İsmail, Beste, Seda, Nilhan, Merih, Ekin, Önder, Ezgi, Ulaş, Nimet, Tunç, Hakkı, Seda, Levent, Güneş, Judith, Nathalie :), Nesrin, Can and Wiebe. Sevim thank you for always being a close friend and supporting me with your mails and nice postcards from long distances.

Fırat thanks for motivating me with your love and care during these years. I guess the

most affected person from my PhD is you. You enlightened me with new eyes about life. Without all those trips we had, music we listened, very postmodern drawings we made, my life would be dull and boring. It was a great motivation for me every time you share a new technological trend about my research. Besides, thank you for designing my colorful cover with enthusiasm. Towards a long trip together to discover other cultures in the world...

I am grateful to my family especially to my mom and dad for encouraging me in all of my new starts in life and supporting me to follow my dreams. My deepest thanks goes to my mom who devotes her life to guide me and my sister to be a good and a caring person. You provided me with the best opportunities for not only for a better education but also a creative path. She always supported me and never imposed me while mostly reminding me that the most important thing in life is to be happy and free wherever we are. I learned how to smile from my mother. Anneciğim iyi ki varsın!

My sister, Deniz, thanks for always supporting me and bringing another color into my life. I am looking forward our new trips :) My uncle you are always special to me who helped to gain vision and widened my perspective to think bigger and pursue my dreams. Finally, I thank my grandmother who raised me with deep care and gave me hope. Rest in peace...

I wish peace, love and freedom to all people in the world. I hope there will be times where each of us will live in harmony with each other and with nature without greed and selfishness.

Derya Sever
December, 2013

Contents

Chapter 1

Introduction

Consider a delivery company delivering products to its customers who have high penalties in case of late deliveries. The driver of the company travels every day from The Hague to Utrecht to make a delivery to a set of customers via the highway A12. The average speed of a vehicle detected on the highway A12 is shown for any day in Figure 1.1. On the 26^{th} September 2013 at one of the intersections of A12 an accident took place at $14:30$ where travelers were stuck at the road for about 2 hours (Figure 1.2). These two figures show that the speed of vehicles changes dynamically depending on random events such as accidents and time of the day such as rush hours.

Figure 1.1 Average speed profile at highway A12 on the 19^{th} September 2013 (Regio Delft Project 2013)

Figure 1.2 Average speed profile at highway A12 on the 26^th September 2013 (Regio Delft Project 2013)

Planners of transportation companies and also drivers face such dynamic and uncertain environments every day to figure out the fastest route and to deliver and/or pick-up goods to customers in a timely fashion. As in the specific example, in real-

life, neither traffic conditions nor customer profiles are known with certainty but are rather dynamic and stochastic. They are dynamic because traffic conditions as well as supply and demand fluctuate over space and time dimensions. These dynamics occur due to random events such as road blockages from accidents, weather conditions, uncertain demand, vehicle breakdowns, traveler choices, etc. The information regarding the random events become available over time rather than at once before departure. Transportation networks are characterized by uncertainty of traffic conditions and/or customer demands on the networks. These stochastic and fluctuating elements on transportation networks lead to stochastic travel times and so uncertain travel costs for transportation companies and travelers. Therefore, when we represent transportation networks as a network consisting of edges and nodes, it is wiser to consider the edges as travel times rather than deterministic distances and nodes as customers with random demands. These make networks stochastic. This thesis considers an interesting and challenging problem on developing routing policies at an operational level in such stochastic and dynamic network conditions where travel times and/or demands fluctuate by random events.

In the European Union countries, the volume of passenger transport increased by 24.68% and the freight transport increased by 36.24% from 1995 to 2005 (European Environment Agency 2013). The rising volumes of passenger and freight transportation cause roads to be used more intensely, increasing both the individual and the societal costs. For instance, a recent study by CE Delft shows that in 2008 the total external costs of transport (costs of emissions, congestion and accidents) in the EU amounted to more than 4% of the total GDP (CE Delft 2013). Therefore, improving the efficiency of passenger and freight transportation routing is needed for reducing both the individual and the societal costs while considering dynamic and stochastic nature of traffic conditions and customer profiles on the networks.

Traditionally, most routing algorithms and software consider networks in a static way with deterministic or stochastic information, i.e. the route is determined before going en-route with all model inputs known with certainty or the inputs are random variables that follow certain probability distributions based on historical characteristics, respectively. Obviously, the world is dynamic and stochastic and does not fit into a deterministic and static straitjacket. All these dynamic and stochastic phenomena are rarely considered or are treated in a static way in the current transportation planning tools. Any schedule built on these unrealistic assumptions results in higher transportation costs.

Today, for transportation companies the market is becoming competitive with higher customer service expectations and drivers want to be on time at their destinations in this dynamic and stochastic environment. Fortunately, the advances in information technologies provide travelers and dispatchers with real-time information on the location of the vehicles, traffic network conditions and customer demand realizations. These present operations researchers an opportunity to tackle dynamic and stochastic conditions on transportation networks to provide travelers and

transportation companies low transportation costs via dynamic routing. Considering high quality and frequently updated real-time information, planners can replan and reroute travelers and vehicles during traveling. In other words, dynamic routing allows planners to change their routing decisions during traveling as more accurate information becomes available to handle fluctuating demands and travel times.

For example, consider a delivery company traveling from The Hague to Utrecht to make a delivery to a set of customer with high delay penalties on the 26^{th} September 2013. From The Hague, there are two alternative highways, i.e. A12 and A4, (Figure 1.3). Normally, the navigation device suggests taking highway A12, which is the fastest route and takes around on average 50 minutes. Regarding this a priori information, the vehicle driver planned to depart at $15:00$ that would make the expected arrival time $15:50$ (Figure 1.4). When the vehicle driver started the journey, the navigation device showed that there was a disruption on A12 creating a long traffic jam and the arrival time was expected to be delayed at least 90 minutes, meaning that the driver would be too late (Figure 1.5). Considering the real-time information and the probability distribution on the successor roads, the navigation device suggested to take an alternative route at the Hague, A4 and then N11 (Figure 1.6). When the driver followed this alternative route, the total travel time was 55 minutes which would be 120 minutes if the static route (A12) was followed. The rerouting in case of a disruption is an example of dynamic routing. This example shows that dynamic routing reduces the transportation significantly via rerouting by considering up-to-date information about the real-world. Obviously, this example is also relevant for a driver company planning routes for the vehicle to deliver goods to its customers where the random element can be stochastic customer demands that fluctuate during traveling or new customer arrivals.

Figure 1.3 Alternative routes

Figure 1.4 Everyday route

Figure 1.5 Accident

Figure 1.6 Reroute

This thesis addresses routing problems in networks where the state of the network changes regarding to stochastic and dynamic events which is motivated by real-life settings. We develop *dynamic and stochastic routing models* at an operational level

to handle the network conditions that are inherently dynamic and stochastic. We limit our scope to networks where the stochasticity arises from either the travel time changing due to traffic disruptions (event that increase the travel time significantly) or stochastic customer demands that are realized upon arrival.

Dynamic routing models can capture real-life dynamics more realistically. According to recent studies, dynamic decision making by considering uncertainty and real-time information lead to significant cost savings (up to 47% cost savings when compared to the routing decisions without real-time information, Lu et al. (2011), Kim et al. (2005b)). However, these savings are at the expense of computational complexity as the routing problem becomes combinatorial. According to Cordeau et al. (2002), a good routing algorithm should be accurate, fast, simple to implement and flexible enough to adapt to real life situations. Also, for practical purposes, we need fast and simple dynamic routing algorithms that provide the travelers with high quality routing decisions. In this thesis, balancing the trade-off between computational time and solution quality is of particular importance because fast, simple and accurate routing decisions prevent travelers wasting time for retrieving decisions with lower transportation costs and higher customer service levels for transportation companies.

The rest of this chapter is organized as follows: In section 3.2, we describe how transportation networks are modeled based on information and modeling type. In Section 1.1.1, we further describe type of information being considered in routing models. In Section 1.1.2, we explain how the routing decisions are made considering these information types. Next, in Section 1.2, we present the dynamic routing literature together with the models and solution methods being studied. Lastly, we give an overview of the thesis with an outline.

1.1. Modeling the Dynamic and Stochastic Transportation Networks

As described in the previous section, real-world is complex and dynamic. To develop high quality and fast routing decisions, we need to model the real-world in a realistic but not too complex way (lower computation times). In this section, we provide the types of input information used for the routing models. Then, we present both static and dynamic routing models depending on the information being used.

1.1.1 Information in Routing Models

Before describing the information types considered in routing models, we start with the information availability in transportation networks for developing routing decisions.

For efficient routing decisions and reducing the effects of uncertainty during travel, travelers need more information than the offline estimated knowledge of the networks. Recent advances in communication, information and location technologies

allow travelers to get benefit from a wide variety of network information. In recent years, Intelligent Transportation Systems (ITS) and Advanced Fleet Management Systems (AFMS) provide real-time and historical information on the road networks and customers (Pillac 2012). For example, Advanced Traveler Information Systems (ATIS) provide travelers with updated information about the road network conditions. In case of an incident on the road, with the information from ATIS travelers can adapt their routes accordingly.

In stochastic networks, travelers or fleets for companies make their decisions (route, mode, departure time) based on the information on the network conditions, retrieved from ATIS and/or AFMS (Huang 2012). Specifically, the information on stochastic networks can be classified in two main categories: a priori (offline) and real-time information (online). A priori information provides travelers or companies with long-term or short-term fluctuations in the network based on historical observations. This can be a probability distribution of travel time on a specific road or customer demand or the probability that a certain blockage on a road is repaired. For example, on a specific day of the week, the travel time between The Hague and Utrecht is 50 minutes on average, but in the morning peak it becomes 55 minutes.

Real-time information consists of the network conditions on a specific day and time, e.g. the real-time information about the accident when the traveler is en-route or the realization of the customer demand during the visit. For example, on the 26^{th} September 2013 at $14:30$, the real-time information about expected travel time from The Hague to Utrecht using A12 is 120 minutes due to the specific disruption. ATIS provides both the real-time information and historical information on the network conditions.

Stochastic versus Deterministic Information

The transportation networks are uncertain environments where the planners may not know with certainty customer demands, travel times or other network conditions. According to the modeling choice, information about the state of the transportation network is modeled as either stochastic or deterministic.

A stochastic process provides a mathematical representation of the states of the physical system evolves over time. Consider Figure 1.2 where the speed of the vehicle so the travel time on a road changes frequently over time. If one considers the fluctuations and significant changes in travel time as states that have different likelihood of occurrence or consider travel time as a random variable following a specific probability distribution, then information is stochastic. Considering the fluctuating speed profile, travel time has the following values with some known probability of occurrence: 50 minutes when there is no disruption, 55 minutes during rush hours and 120 minutes when there is a disruption. Stochastic processes are also modeled in a routing network through which a vehicle distributes goods to customers with uncertain demand. The demand information is stochastic when the demand is represented with a probability distribution, where each value occurs with some

likelihood.

When the travel time is seen as the average time over all fluctuations without considering any fluctuations due to the disruptions in Figure 1.2, the travel time information is deterministic. For instance, when we consider that travel time in the road is always 5 minutes on average, then this information is deterministic. When only realizations are considered, this type of information is also deterministic as no probabilistic information about the future is used. Average customer demand or considering demand realizations without any probabilistic information about the future demands are also examples of deterministic information.

Mostly, the deterministic network assumption is unrealistic when the routing takes place a priori and not adapted to the changes because it is obvious that world is changing within time. However, in case of highly uncertain and unpredictable networks, deterministic information may be used with real-time information to represent the condition at the specific moment. This is intuitive because one may hardly predict the range or probability distributions of the uncertain element. In the literature, this is done via online optimization where only the real-time information without any knowledge about the future is considered in the routing decisions (Ferrucci 2013).

In this thesis, we focus on both deterministic information and stochastic information for the routing models. We consider realization on uncertain elements without exploiting any stochastic information for obtaining information on the current events. For estimating values of uncertain elements in the future, we consider stochastic information.

Dynamic versus Static Information

The networks are inherently dynamic due to the presence of an uncertain environment. In the routing model, the value of an uncertain element can be considered as either a real-time realization or a constant value or a random variable following a certain probability distribution. Information in routing models is either modeled as dynamic or static depending on the modeling choice.

Considering our routing example, if the input travel time on the highway A12 is used as an average over all fluctuations or the expected value with known variance considering the uncertainty, we are able to suggest travelers an a priori path which is to follow A12 from the origin to the destination. At each intersection, the travel time on A12 will be considered as the same in the model. This type of information is denoted as static. For instance, when the planner considers that the customer demand is on average 20 units and this does not change over the planning horizon, the demand information is also static. The static information to the routing model does not change during the planning horizon.

Consider the routing example where realizations of travel times are used for the routing model. In this case, the travel time as in Figure 1.2 changes during the

planning horizon. The travelers observe the travel time differently when they arrive at different intersections in time. When the value of the uncertain element in the network changes over the problem horizon, the input information on that uncertain element is dynamic. Dynamic information can be in other forms. For instance, the travel time on the road changes when a random accident happens and also the impact of the accident decreases in time due to the repair activities. This means that the travel time is different at the very next moment of the accident and after a long time of the accident. Consider a vehicle filling ATM's from a central bank or distributing oil to individual houses. The amount of money each ATM needs or amount of oil for each house is not precisely known in advance and it is possible that the demand is more than the amount predicted before departure.

In the literature, dynamic information is also referred to time-dependent information where environment changes with function of time but *known in advance* by configuring historical data (Powell et al. 1995). Time-dependent data can be travel times on particular days of the week and times of the day.

In this thesis, we do not focus on dynamic information due to time-dependency. We focus on dynamic information where this information changes over the horizon with a known probability distribution. We gather information and make decisions at specific decision moments in the planning horizon. The dynamic input information follows an a priori probability distribution that becomes deterministic and known when realized. This means that we exclude the information on the events that happen instantly and in an unpredictable manner. For example, when we consider incidents on road transportation, we know the probability distribution of how the impact of an incident is repaired and reduced within time. The information is dynamic in two aspects. First, realizations on uncertain elements (deterministic information) change as we move to different decision moments. Second, stochastic information on uncertain elements change in the short-term horizon of the decision moment. For example, when there is a disruption observed, the probability distribution of having the disruption changes in time according to the repair time. In some of the routing models we develop, we also use static information in terms of expected travel times or time-invariant probability distributions for the long-term horizon of the decision moment. We should note that in the routing models, the long-run probability distributions do not change as we assume that the changes in the long-run probability distributions are very small during the planning horizon. However, in another departure time, the information on probability distributions may change depending on the dynamic network conditions.

1.1.2 Routing Models in Stochastic Networks

Routing models are classified based on which type of network information is considered in the model as explained in Section 1.1.1. In Figure 1.7 represents a matrix where routing models are represented according to the types of information considered in the model. In this section, we will explain each quadrant of this matrix.

Traditionally, in most research and software, routing problems are modeled in a static

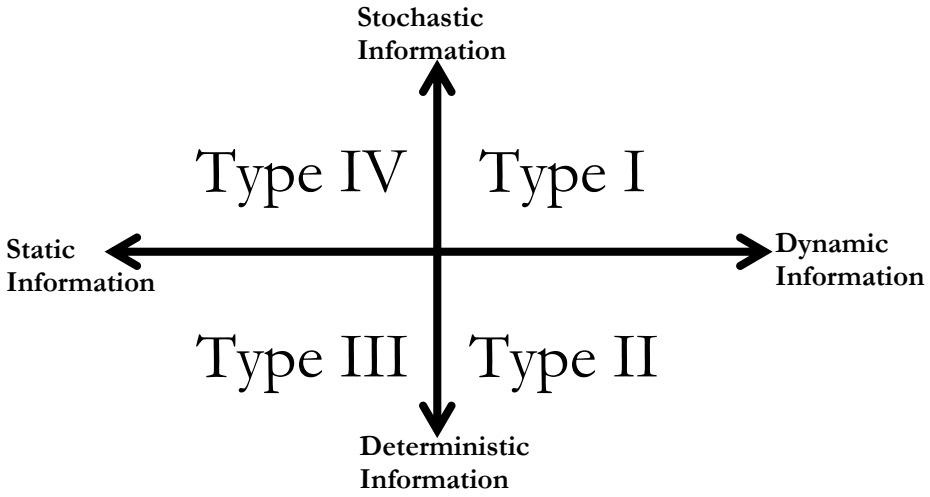


Figure 1.7 Classification of routing models

way where a fixed a priori path is developed at the start of the trip and followed no matter the realizations of the network. This type of routing is also referred as a static routing or non-adaptive (fixed) routing. The static route is determined before going en-route (in offline setting) with all static model inputs known with certainty or with probabilistic information. In the thesis, we refer to a routing algorithm as static, when the planned routes are not re-optimized and the routes are determined from the information that is known before going en-route (Psaraftis 1995). Typically, in the offline setting, one can plan the static routes either taking into account deterministic information (Type III) or stochastic information (Type IV). Furthermore, the application of the static model is also static (Ferrucci 2013). In a static routing, a single path is determined before departing according to a priori probabilistic information. During travel even if there is real-time delay information, the planned path is followed. Note that in the literature, there are also approaches such as two-stage stochastic programming where static routes are determined at offline level in the first stage. When the routes are executed in the second stage, recourse actions are applied to the first stage solution by using real-time information at online level.

The advances in availability of real-time and probabilistic information provide planners/researchers new opportunities to develop efficient dynamic routing models. Dynamic routing is the replanning of the routes based on the updated dynamic information becoming available during the travel. In static routing, the output is a set of routes determined at the offline level. However, in dynamic routing the output is rather a policy where the routes are defined as a function of network state that becomes known in real-time during the travel (Psaraftis 1988, Larsen and Madsen

2000).

The vehicle routing problem with stochastic demand (VRPSD) is an example to both static and dynamic routing depending on how we model the input information (static or dynamic way). In VRPSD, customer demands are random variables with known probability distributions. However, the exact demand is observed when the customer is visited. So, the vehicle capacity may be insufficient to fulfill the customer demands. If there is not enough capacity during service, a failure occurs and the vehicle needs to return back to the depot for replenishment. In the static case, a fixed route is modeled according to the static information on the known a priori demand distributions (Type IV). There is a fixed sequence and in case there is a failure, the restocking decisions are done either with predefined policies or with proactive restocking decisions (Gendreau et al. 1995, Laporte et al. 2002, Christiansen and Lysgaard 2007). In the dynamic routing approach, the demand of the customer becomes available during the visit and these realizations are considered in the model. In other words, dynamic realizations with stochastic information about the future events are considered in the model (Type I). After serving the demand of a customer, the driver makes a routing and a replenishment decision dynamically. This decision is based on the available vehicle capacity and the set of not served customers. The realized demand and where to go next can be communicated with the driver with AFMS technologies.

In the literature, there are two main classes of dynamic routing models:

Routing with dynamic and deterministic information (Type II): In case of highly dynamic networks, the information is not known before the departure. When the information is unknown from the beginning and learned with certainty during travel, then the routes are planned and re-optimized at real-time repeatedly (online optimization). The model considers real-time information and does not consider the available stochastic knowledge about future expected changes. These are called reactive, pure online or real-time approaches (Figure 1.8).

Routing with dynamic and stochastic information (Type I): When the realization of an uncertain element is learned during travel and the real-time information is exploitable with probability distributions, the problem is modeled in both dynamic and stochastic way. This type of routing is called pro-active real time or adaptive approaches (Figure 1.8). In the literature, there are two main classes of dynamic and stochastic models. First one is making decisions and observing outcomes on a continuous, rolling horizon basis based on non-stationary information while exploiting the stochastic information on future events. Second one is based on a priori (offline) optimization where routing policies are generated for stochastic problem regarding the realization of the random elements prior to the real-time information retrieval. The routing policy is determined at the offline level but the execution of the routes depending on the realization of the random element is done at the online level where real-time information is retrieved.

In this thesis, we focus on the latter type of dynamic and stochastic models (Type

I). This modeling choice is valid with reality where we assume that the network is stochastic and the random element follows a prescribed probability distribution which is known beforehand. We do not consider any information on the non-stationary fluctuations occurring within time. In other words, we exclude any information that is totally unknown and that cannot be defined before going en-route. We assume that either travel time or customer demand profiles that are inherently dynamic and stochastic, are already configured and explicitly modeled via known probabilistic information based on historical data. For instance, dynamism arising from changes in travel times is nowadays also well predictable due to the advances in recent technologies such as floating car data (Ehmke et al. 2009, 2010). Furthermore, traffic congestions and traffic states with their impacts are detectable by traffic sensors (Attanasio et al. 2007, Ferrucci 2013). Thus, in our dynamic and stochastic models, the dynamic changes in the network are limited to network states that are defined before going en-route. The network states consider the values of dynamic and stochastic elements that may occur during travel. These states can change from one departure time to another depending on the dynamic network conditions and updated forecasts.

In the literature, mostly first type of dynamic routing models is considered. The second type is much harder to solve due to the computational burden in computing the policies for all realizations. One of the contributions of this thesis is to develop fast and high quality algorithms to deal with computational challenges while developing routing policies for all possible realizations.

To compare the effect of different routing modeling approaches, we focus on both dynamic and static routing models. We develop dynamic routing models considering a variety of information as explained in Section 1.1.1. We also model various static (with static and deterministic or static and stochastic information) and dynamic routing models (dynamic and deterministic information) as a benchmark to measure the value of dynamic decision making in dynamic and stochastic routing problems with computational time measure. Therefore, we consider the routing models: Type I, II, III and IV.

In this thesis, we deal with two types of dynamic events: dynamic travel times and stochastic customer demands realized upon the visits. We address two different types of dynamic routing problems: dynamic shortest path problems and vehicle routing problems with stochastic demands. (Eksioglu et al. 2009).

In Chapters 3-5, we tackle dynamic shortest path problems with stochastic network disruptions. Before departure, travel time is only known with a probability distribution. The information on the disruption states and thus the travel time is realized as the traveler travels along the network. The input travel information is dynamic because the travel time on a road changes dynamically due to the stochastic events over the planning horizon. The output is a policy that prescribes how the routes evolve as a function of the state of the network.

In Chapter 6, we tackle a vehicle routing problem with stochastic demands. The demands of customers are realized as the vehicle visits the customers. The randomness of the customer demand leads to failure where the capacity of the vehicle is not sufficient to fulfill the demand. The vehicle then returns to the depot to replenish. The information on customer demand and the remaining capacity of the vehicle is dynamic as this information changes during the travel. The demands of future customer demands are modeled as a stochastic process and follow a known probability distribution. The output is a policy with set of decisions on which customer to visit next or whether go back to depot to replenish based on the realized demand information and remaining capacity.

1.2. Dynamic Routing Problems- a Literature Review

In this section, we briefly summarize the existing literature on routing in stochastic networks that is relevant to the thesis. We address two different types of routing problems: dynamic shortest path problems and stochastic vehicle routing problems.

Consider the routing example as a graph, $G(N, A)$ where a traveler wants to traverse G from an initial node, The Hague, to a destination Utrecht. The arcs are the roads and the nodes are the intersections of multiple roads. We consider that the arc traversal cost (travel time on a road) is stochastic and dynamic. Briefly, this means that the cost of traversing an arc (i, j) on G is a known function: $f_{(i,j)}(e^i)$ of the state e^i of a certain environment variable at node i. The actual state of e^i is revealed to the traveler when it is at node i. Once departing from node i, the traveler incurs a cost $f_{(i,j)}(e^i)$ depending on the state of the network. Based on this definition, dynamic shortest path problem is to find a policy that minimizes the expected total cost of a traversal from initial node to destination (Psaraftis and Tsitsiklis 1993). For example, let e^i denote the information on the occurrence of the disruption on the road and the associated cost. Then, our problem becomes finding a routing policy with dynamic decisions from The Hague to Utrecht with minimum expected total travel time.

The vehicle routing problem (VRP) is first introduced by Dantzig and Ramser (1959). In general, a VRP considers finding a set of routes for a set of vehicles departing from the depot, such that each of the customers is visited exactly once and the vehicles return back to depot after serving all the customers. The objective is to minimize the overall routing cost. When the customer demands and travel times are stochastic the general VRP problem is denoted as stochastic Vehicle Routing problem (SVRP). When the customer demands, arrival of customers and travel times becomes known and evolve during travel and the routes are determined dynamically, we denote this type of VRP as a dynamic vehicle routing problem.

We provide the literature review on DSPP and VRP based on two properties of the problem: Types of Dynamic Event and Modeling Perspectives and Solution Methods. We only focus on two types of dynamic events: dynamic travel time and stochastic customer demand. For information on other type of dynamic events such as new

customer arrivals and vehicle breakdowns, we refer the reader the surveys by Psaraftis (1995), Gendreau et al. (1996a), Gendreau and Potvin (1998), Ichoua et al. (2000), Ghiani et al. (2003), Jaillet and Wagner (2008), Larsen and Madsen (2000), Pillac et al. (2012), Ferrucci (2013).

Types of Dynamic Events

We first focus on the dynamic routing literature based on the typed of dynamic events in the routing problem. In dynamic routing, dynamic events lead to change in network information during travel. In the literature, four types of dynamic events are considered (Ferrucci 2013, Pillac 2012): dynamic travel times due to network disruptions, dynamically revealed demands, new customer arrivals and vehicle availability. In this thesis, we focus on the first two types.

Dynamic travel times due to network disruptions: In traffic networks, disruptions due to accidents, bad weather and road bottlenecks lead to a drastic increase in travel times and decrease the probability of being on-time at the destination. In case of freight transportation, the service level decreases due to late arrivals to customers. The occurrence of disruptions becomes known during travel. The travel times given the disruption information change during planning horizon with random disruptions. To reduce travel time and increase the quality of passenger transportation, rerouting the travelers to less disrupted roads based on the information about the network disruptions is necessary. For VRP, to increase customer service level and decrease transportation cost, rerouting of vehicles and reassignment of customers are essential. In dynamic routing literature, this type of dynamic event is taken into account as ATIS and AFMS technologies provide travelers network information with higher reliability and availability. The literature considering dynamic travel times can be seen in Figure 1.8. The papers that are relevant with this thesis are on dynamic shortest path problems with dynamic travel times. Gao and Huang (2012), Thomas and White (2007) and Kim et al. (2005b) consider dynamic travel times in terms of dynamically occurring disruptions in traffic such as accidents. Güner et al. (2012) explicitly analyze both time-dependent travel times and uncertain events that change the travel time. Psaraftis and Tsitsiklis (1993), Polychronopoulos and Tsitsiklis (1996), Cheung (1998) consider the dynamic travel times via real-time information where the travel time can change due to both dynamic disruptions and time-dependency.

Stochastic customer demands: In stochastic VRPs, stochastic customer demand is considered with known probability distributions. Each customer has a given and known demand distribution and the actual demand realization is unknown until the vehicle arrives at the customer. The actual demand is known when the customer is visited. We denote these problems as vehicle routing problems with stochastic demands (VRPSD). In the VRPSD, the vehicle may be unable to satisfy the actual customer's demand realization when visiting the customer and the vehicle needs to return to the depot for a refill and return back to the partially served customer. The network state changes depending on the realized customer demands, current location

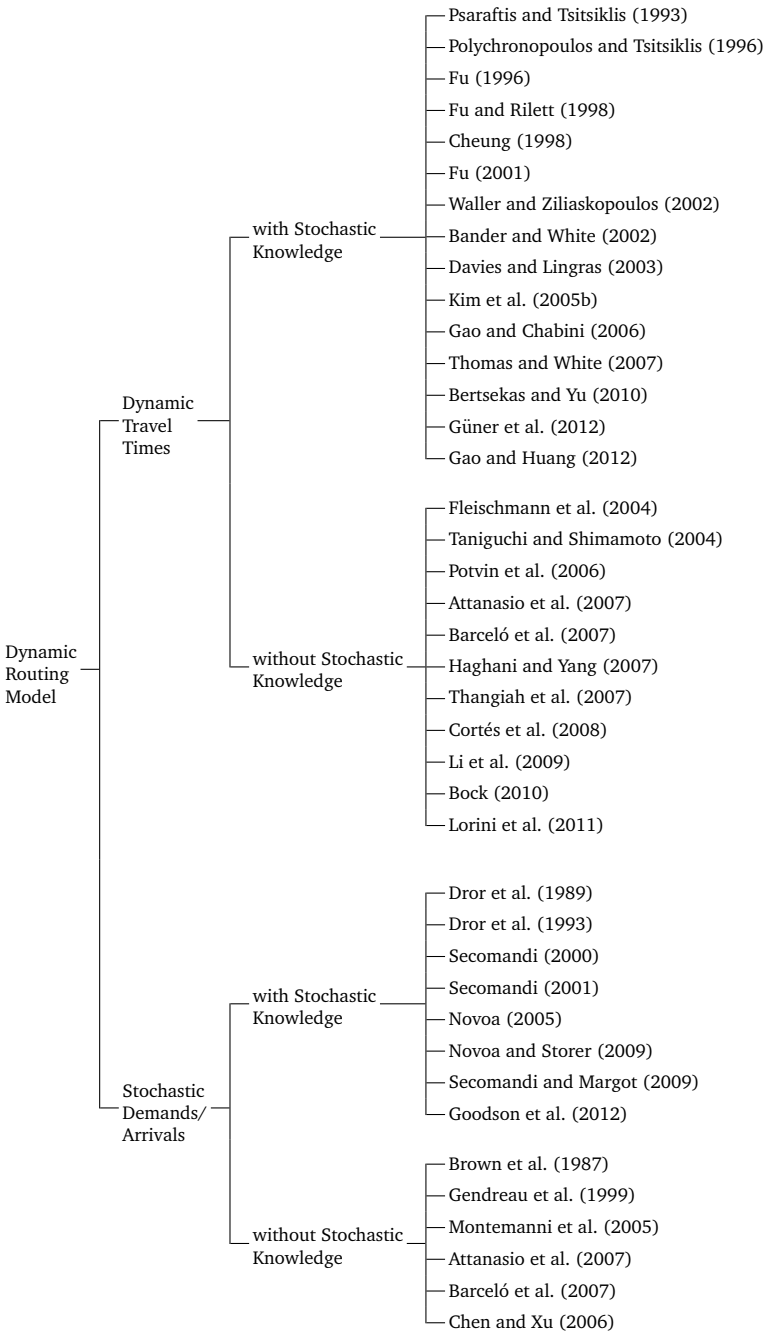


Figure 1.8 The literature based on types of dynamic events

of the vehicle and available capacity. The cost function depends on realization of the customer demand, state of the remaining capacity of the vehicle and unvisited customers that change in time. The dynamic decision is then whether to return back to depot and which customer to visit next given the state of the system. The real-life examples for this type of uncertainty are: beer distribution to retail outlets, the re-supply of baked goods at food stores, replenishment of liquid gas at research laboratories, local deposit collection from bank branches, less-than-truckload package collection, garbage collection, home heating oil delivery, and forklift routing. The VRP considering dynamically revealed stochastic demands are considered in Dror et al. (1989, 1993), Secomandi (2000, 2001), Novoa (2005), Novoa and Storer (2009), Secomandi and Margot (2009) and Goodson et al. (2013) where the realization of customer demand is known when the customer is visited.

Modeling Perspectives and Solution Methods

In this section, we explain in general the modeling perspectives and related solution methods for solving dynamic and stochastic problems. For this, we use the categorization in the literature survey of Ferrucci (2013) based on whether the dynamic routing model considers deterministic or stochastic information (Section 1.1.2). For a detailed review on solution methods used in dynamic routing problems, we refer to the literature surveys by Ferrucci (2013), Ichoua et al. (2006, 2007), Larsen and Madsen (2000) and Pillac et al. (2012).

Routing model without stochastic information

In dynamic and deterministic routing problems, routing models are built in the absence of stochastic information. The information on the network is revealed over time and used in the model without considering any probabilistic information on future events.

The traveler observes the information by the time the vehicle arrives to a node and then the information is used for optimization. However, the information may change after making a decision. Therefore, the solution from the deterministic model is only optimal at the current state and time and the solution may not be optimal at the next decision epoch. Due to this, mostly re-optimization and meta-heuristics are used as solution methodologies.

Re-optimization is the update and the sequential optimization of the model whenever new information on the network is retrieved. The re-optimization algorithm periodically solves a static optimization problem for the current state based on the real-time information. This can be done in decision moments or time intervals. The advantage of the re-optimization method is that it can benefit from the static methods that are widely studied in the literature. The drawback of the method is that it heavily depends on the availability of the real-time information and it does not involve the stochastic information. The re-optimization is also needed frequently for updating the routing plan. This creates the necessity for faster algorithms to prevent delays in

routing decisions.

In vehicle routing literature, most of the re-optimization and meta-heuristics are used for handling dynamic customer arrivals together with stochastic customer demands. Chen and Xu (2006) consider a DVRP with dynamic customer arrivals to be picked up within their time windows. The dispatcher does not have any deterministic or probabilistic information on the location and size of a customer order until it arrives. The objective is to minimize the sum of the total distance of the routes used to cover all the orders. A dynamic approach is used for the problem where single-vehicle trips are built over time in real-time at each decision epoch. Montemanni et al. (2005) consider DVRP with dynamic customer arrivals using Ant Colony System (ACS) with time buckets. The static VRP is solved repeatedly in the beginning of each time bucket. In ACS, information about promising routing solutions are saved when the optimization problem evolves. A parallel Tabu Search (TS) algorithm is introduced by Gendreau et al. (1999). In TS, good routes are pooled in the adaptive memory as the dynamic information evolves in time. The parallelization is done by optimizing the routes in independent threads. The other variations of TS are also applied in Attanasio et al. (2007) and Barceló et al. (2007). Brown et al. (1987) a computer assisted dispatching system is designed for real-time dispatch of mobile tank trucks. The customer requests are handled via this dispatching system considering the customer order realizations. Fleischmann et al. (2004) consider a dynamic pickup and delivery problem where dynamism is from newly arriving customers and travel times. Insertion heuristics and modified Dijkstra's algorithm are used to handle the dynamism in the problem during execution of travel. Potvin et al. (2006) develop reactive routing strategies with insertion heuristics for dynamic VRP with dynamic travel times.

Routing model with stochastic information

In dynamic and stochastic routing problems, in addition to the real-time information, stochastic information is also available in the input. In stochastic models the planners take into account the transition probabilities and the expectations considering the uncertain elements.

In the dynamic routing literature either sampling or stochastic modeling strategies are used. In sampling strategies, scenarios are generated based on the realizations from the random variable distributions (Pillac et al. 2012). Multiple Scenario Approach (MSA) is a sampling approach where the scenarios are generated, re-optimized and added to the scenario pool as time evolves. Bent and Van Hentenryck (2004b) consider MSA algorithms in DVRP problems with dynamic arrival of customers by selecting customers appearing first and most frequently. Additionally, Bent and Van Hentenryck (2004a) approximate the cost of visiting each customer for all scenarios to avoid re-optimization of all scenarios. Various algorithms are used for defining how the information from the scenario pool is used to make a routing decision. In Pillac (2012), an event driven optimization framework is applied based on MSA.

In the shortest path literature, Hall (1986) first shows that traditional shortest path algorithms fail to find the minimum expected travel time path on a network with random travel times. With this paper, stochastic and time-dependent shortest path problems are introduced. It is shown that the optimal route choice is not a simple path but an adaptive decision rule. Fu (1996) develops an optimal adaptive routing algorithm that updates the travel times according to the real-time information and perform re-optimization by finding the path with the minimum expected travel time. Later, Fu and Rilett (1998) provide a heuristic approach where where arc traversal times are continuous functions of time. Miller-Hooks and Mahmassani (2000) develop an exact solution algorithm for finding the least expected time path in stochastic time-dependent networks.

The papers on shortest path problems discussed above do not use real-time information. In these papers, only travel-time distributions are taken into account. Polychronopoulos and Tsitsiklis (1996) relax the assumption of time-dependent model and introduce dynamic shortest path problem (DSPP). In this problem, arc costs are randomly distributed, but the realization becomes known once the vehicle arrives to a node. The objective is to find the routing policy that has the minimum cost. Given the location of the vehicle and the cumulative information, the arc to traverse is selected. The shortest path algorithm is applied every time the cost of the route is different than expected. Cheung (1998) develops an iterative algorithm for DSPP. In Psaraftis and Tsitsiklis (1993), the arc travel times evolve over time according to a Markovian Process. Davies and Lingras (2003) extend this model by developing genetic algorithms for dynamical rerouting. Fu (2001) study DSPP where travel time realizations are used for developing for adaptive routing strategies. In this paper, the time-dependency of link travel times is not explicitly considered. An efficient approximation is developed to solve the problem and the advantage of adaptive routing systems is shown. Bander and White (2002) also develop adaptive routing policy for stochastic shortest path problems where travel times are modeled as a stochastic process. An optimal algorithm is developed to find routing decision which is dependent on the (time, node) pair. Waller and Ziliaskopoulos (2002) deal with disruptions by using online-recourse models where every time real-time information becomes available, the remaining path until the destination is re-evaluated. They proposed algorithms for the online shortest path problems with limited arc-cost dependencies.

Gao and Chabini (2006) study optimal routing policies in the stochastic time-dependent networks with with both time-wise and link-wise dependency and perfect online information. They designed approximation algorithms for time and arc dependent stochastic networks. Gao and Huang (2012) extend this research with partial or no-online information case. They analyze generic forms of the real-time information availability in the routing decisions. They also developed a heuristic algorithm for the adaptive routing problem by using different online information levels based on a set of necessary conditions for optimality. The dynamism in the model arises from the link dependencies in case of traffic disruptions. When there is

an accident on a road (learnt with real-time information), through correlations the travel time distribution of the neighbor roads changes.

Most of the dynamic and stochastic routing problems are modeled by using a Markov Decision Process (MDP). In this model, the dynamic routing policies are found based on the state of the system at specified decision epochs by considering the expected routing cost. An MDP formulation provides a practical framework to find dynamic routing decisions for each decision epoch which becomes easier in practice. For the solution procedure, dynamic programming (DP) algorithms are used. However, for large scale networks with stochastic travel times, obtaining the optimal solution faces the curses of dimensionality, i.e., states, outcomes, and decisions (Powell 2007, 2011). Therefore, in the dynamic routing problem literature with an MDP formulation, either approximation algorithms are developed to deal with the curses of dimensionality or the structure of the optimal solution is investigated to reduce the state space. Approximate Dynamic Programming (ADP) algorithm is a well-known approximation approach to effectively respond to the various levels of disruptions while reducing the computation time significantly. ADP is a powerful method (Powell 2007, 2011) to solve large-scale stochastic problems.

Kim et al. (2005b) deal with dynamic shortest path problems with anticipation using an MDP for the optimal vehicle routing problem. Here, whenever there is new information about a disruption, the model takes into account the congestion dissemination and anticipates the route accordingly. The travel time is stochastic and time-dependent. The model used in this thesis for dynamic shortest path problems is similar to the one used in this paper; however, we use time-invariant disruptions with dynamically evolving probability distributions for travel time. For larger networks, as the formulation becomes intractable, Kim et al. (2005a) propose state space reduction techniques where they identified the traffic data that has no added value in decision making process. Thomas and White (2007) also formulate the dynamic shortest path problem as an MDP. They provided conditions on which the optimal routing decision does not change even the network state changes. The MDP model in this thesis is also similar to the model used in this paper. Yet, the authors only consider unpredictable disruptions in their analysis. Bertsekas and Yu (2010) formulate the stochastic shortest path problem as an MDP. For solving large scale problems, they developed a Q-learning algorithm with a policy iteration technique. They showed that their stochastic Q-learning algorithm is bounded and converges to the optimal at any initial solution. Güner et al. (2012) considers non-stationary stochastic shortest path problems with both recurrent and non-recurrent congestions using real time traffic information. They formulated the problem as an MDP that generates a dynamic routing policy based on the state of the system. To prevent the state explosion, they limit the formulation to two-arc-ahead formulation where they retrieve the state information for only two links ahead of the current location.

In DVRP literature, the stochastic modeling is considered mostly in the problems with dynamically revealed demands (VRPSD). Dror et al. (1989, 1993) are the

early papers that introduce the re-optimization strategies. They formulate a fully dynamic, single-vehicle VRPSD as an MDP. They optimally re-sequence the unvisited customers whenever a vehicle arrives at a customer and observes the demand. Secomandi (2000) presents a stochastic vehicle routing problem formulation based on an MDP, and develops two heuristics: a rollout algorithm and an approximate policy iteration. Secomandi (2001) gives more details on the rollout algorithm, which uses a nearest insertion and a 2-int heuristic as its base sequence, and a cyclic heuristic to generate new partial routes. Novoa and Storer (2009) extend the rollout algorithm by implementing different base sequences, two-step look-ahead policies and pruning schemes. Secomandi and Margot (2009) also consider a VRPSD under re-optimization. They formulate the problem as a finite horizon MDP for the single vehicle case. They develop a partial re-optimization methodology to compute suboptimal re-optimization policies for the problem. In this methodology, they select a set of states for the MDP by using two heuristics: the partitioning heuristic and the sliding heuristic. They compute an optimal policy on this restricted set of states by a backward dynamic programming. Goodson et al. (2013) present a rollout policy framework for general stochastic dynamic programs and apply the framework to solve for VRPs with stochastic demands and duration limits.

In this thesis, we model the stochastic routing problems with MDP for both DSPP with dynamic travel times and SVRP with stochastic demands. MDP is chosen as it provides a practical framework to find dynamic routing decisions for each decision epoch. For the large instances, due to the curses of dimensionality, we develop efficient and fast stochastic lookahead strategies and ADP algorithms.

1.3. Thesis Overview

This thesis addresses routing problems in stochastic networks with stochastic disruptions on roads or stochastic customer demands. By considering the predictability of these events, we propose a multi-stage stochastic dynamic programming model based on a discrete-time, finite MDP formulation as in Kim et al. (2005b), Thomas and White (2007), Güner et al. (2012). In our models, we couple the real-time network information with probabilistic information on the stochastic elements to improve the decision making process (Ichoua et al. 2006, Powell et al. 1995). In a real-time setting, the network information (e.g. travel times, customer demands, disruption realizations and current location) is realized during the execution of the routing process. Before the departure, we develop fast and efficient algorithms to obtain *stationary routing policies* based on network state realizations. As the traveler collects information on the realizations during travel, the relevant policy is selected. In other words, given the stochastic information, we develop *dynamic decisions* depending on the realizations of the stochastic elements to minimize the total expected travel time. These decisions may change from one departure to another based on changing network conditions.

We solve the problem with a Dynamic Programming (DP) approach. For the

routing decisions, the DP approach provides a practical framework to find routing decisions for multiple stages. However, for large-scale networks, finding the optimal solution suffers from the curse of dimensionality. Throughout the thesis, we provide approximations to reduce the computation time significantly while providing high quality solutions. In Chapters 3-5, we focus on dynamic shortest path problems in stochastic networks where the stochasticity comes from travel time. In these chapters, we develop dynamic routing policies and we compare various offline and online algorithms as well. In the last chapter, we deal with vehicle routing problem where we have stochastic customer demands.

In Chapter 2, we describe the traffic networks that we focus on for dynamic shortest path problems. In this chapter, we explain how we implement the dynamic algorithms from input information retrieval to output generation. This chapter is an introduction for Chapters 3-5.

In Chapter 3, we consider dynamic shortest path problems where there are stochastic disruptions in the network due to accidents and bottlenecks that cause congestions that lead to significantly higher travel time. For developing routing policies, we both consider real-time traffic information and stochastic information. Furthermore, we also consider the probability distribution of the duration of a congestion caused by a disruption. In this chapter, we analyze the effect of considering different types of information used for different parts of the network on quality of routing decisions and the computation time. We develop a framework based on a DP in which we formulate and evaluate various online and offline routing policies in the literature. Next to this, we develop computationally efficient hybrid routing policies using both real-time and historical information. To test the efficiency of different routing policies, we develop a test bed of networks based on a number of characteristics and analyze the results in terms of routes, solution quality and calculation times. Our results show that a significant part of the cost reduction can be obtained by considering only a limited part of the network in detail at online level.

In Chapter 4, we consider the same dynamic shortest path problem considered in Chapter 3. However, in this chapter we extend the problem for larger network sizes with multiple levels of disruptions. We model this as a discrete time, finite MDP. For large-scale networks with many levels of disruptions, MDP suffers from the curse of dimensionality. To mitigate the curses of dimensionality, we apply Approximate Dynamic Programming (ADP) algorithm. In this algorithm, instead of computing exact value functions (computationally challenging), we consider value function approximations. We develop a hybrid ADP algorithm with efficient value function approximations based on a clustering approach. In this hybrid ADP algorithm, we combine a deterministic lookahead policy with a value function approximation. We provide various algorithmic design variations for the ADP where multiple initial solutions and various update methods are used. We show insights about the performance of these variations based on different network structures. Furthermore, we develop a test bed of networks to evaluate the efficiency of our approximation with

standard ADP algorithm and hybrid routing policy based on a stochastic lookahead algorithm (developed in Chapter 3). The results show that the hybrid ADP algorithm reduces the computational time significantly while providing high quality solutions.

In Chapter 5, we consider another dynamism in traffic networks which is the propagation effect of disruptions on the dynamic shortest path problems. When a disruption occurs at a certain link, due to limited capacity, the effect of the disruption propagates to upstream links. We denote this as the spillback effect. To have better routing decisions, we should capture the dynamics of spatial and temporal correlations. In this chapter, we model the dynamic shortest path problem with stochastic disruptions as a discrete time, finite MDP. To reduce the state-space, we only use real-time information for a limited part of the network (Hybrid routing policy developed in Chapter 3). We analyze the effect of considering and not considering the spillback effect in our routing decisions.

Chapter 6 deals with the vehicle routing problem with stochastic demands, considering a single vehicle. In this problem, the actual demand realization is unknown until we visit a customer. We build a stochastic dynamic programming model and implement an ADP algorithm to overcome the curses of dimensionality. The ADP algorithms are based on the Value Function Approximations (VFA) with a lookup table. The standard VFA with lookup table is extended and improved for the VRP with stochastic demands. The improved VFA algorithm reduces the computation time significantly with good quality of solutions. A significant reduction of computational time enables us to conduct systematic larger scale numerical experiments, important for real-life decision making. Several test instances found in the literature are used to validate and benchmark our obtained results.

1.4. Thesis Outline

In the thesis, we model all the dynamic routing problems using an MDP and solve them using a DP. In Chapters 3-5, we consider dynamic travel times due to disruptions in the network. Chapter 3, analyzes the value of information in DSPP where we develop efficient stochastic lookahead strategies. In Chapter 4, we tackle DSPP problems with higher levels of dynamism by developing ADP algorithms. In Chapter 5, we consider link correlations during traffic disruptions such as spillback effect in DSPP. Considering stochastic and dynamically revealed demand is a focal element in Chapter 6 where we develop ADP algorithms (Figure 1.9).

The chapters of the thesis are based on the following papers:

Chapters 3: Sever, D., Dellaert, N., van Woensel, T., de Kok, T. (2013) Dynamic shortest path problems: Hybrid routing policies considering network disruptions. Computers & Operations Research 40 (12), $2852 - 2863$.

Chapter 4: Sever, D., Zhao, L., Dellaert, N., van Woensel, T., de Kok, T. (2013) Dynamic shortest path problems with stochastic network disruptions: Hybrid approx-

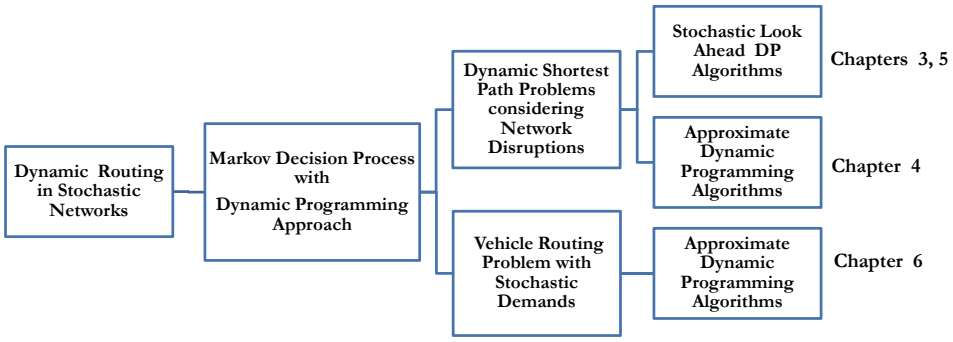


Figure 1.9 Outline of the thesis

imate dynamic programming algorithms with a clustering approach. Beta Working Paper number wp 423, School of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology.

Chapter 5: Sever, D., Dellaert, N., van Woensel, T., de Kok, T. (2013) Influence of spillback effect on dynamic shortest path problems with stochastic network disruptions. Beta Working Paper number wp 424, School of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology.

Chapter 6: Zhang, C., Dellaert, N., Zhao, L., van Woensel, T., Sever, D. (2013) Single vehicle routing with stochastic demands: Approximate dynamic programming. Beta Working Paper number wp 425, School of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology.

Chapter 2

A Prologue to Dynamic Routing with Network Disruptions

Traffic congestions and longer travel times have become one of the major problems in the world. According to IBM, traffic congestion costs more that 1% GDP of European Union which is more than 100 billion Euros. The latest traffic data of TomTom and Inrix show that traffic congestion due to disruptions in the European traffic has enormously increased over the past years due to inefficient infrastructure and planning. According to the traffic report of Inrix (Inrix, 2013), in 2012, an average driver in the Netherlands wasted in total 51 hours in traffic congestion. Furthermore, regarding the IBM's survey on 20 global cities, 91% of the traffic commuters got stuck in the traffic with 1.3 hours of delay on average (IBM, 2013). About 50% of the commuters request for better routing to improve the quality of their lives. These results show that we need efficient routing policies to tackle the increasing congestion rates.

Most of the wasted time in traffic congestion is a result of uncertain traffic environment because of traffic disruptions such as accidents, weather conditions and breakdowns. The negative effect of these uncertain traffic disruptions can be reduced by acquiring more information (Lu et al. 2011, Schrank et al. 2012). The travelers using more information have better routing decisions while reducing their commitment times in traffic. Traditional traffic information informs travelers only average values. However, in real-life travel-times can be significantly higher than the average due to unexpected disruptions (Figure 2.1). In this case, if the routing decision is done based on the average historical value, delay time may increase significantly. When the traveler has the right information at the right time, the traffic delay can be reduced significantly with higher quality routing decisions.

As discussed in Chapter 1, the advances in the current technology provides travelers

a wider range of traffic information through Advanced Traveler Information Systems (ATIS). ATIS provide users a priori (offline) information based on historical information, real-time (online) information and predictive information considering the prediction of disruptions in near future. The aim of Chapters 3-5 is to develop fast and efficient dynamic routing policies in case of traffic disruptions such that the travelers are less affected from the congestion by considering richer traffic information.

In this section, we present the traffic network properties that we focus on in Chapters 3-5. First, we give an overview of which types of disruptions we tackle and how we model the travel time. Then, the traffic information and routing strategies provided by the current navigation technologies will be explained. We also provide information on how we model dynamic routing decisions considering network disruptions. In terms of application, we will give the dynamic routing mechanism from input information retrieval to how the routing policies are implemented.

2.1. Modeling Travel Time with Disruptions

In general, there are two sources of congestions in traffic networks: recurrent congestions and non-recurrent congestions. Recurrent congestions occur due to recurrent disruptions such as the insufficient capacity of the road during peak hours and inefficient traffic management when the density on the roads increases. A good example for a recurrent congestion is the travel time increase during peak hours. When the traffic volume in the rush hours is greater than the road capacity, then a traffic congestion occurs at specific time of the day and day of the week. Also, the physical capacity of the roads such as number of lanes and merge areas influence the recurrent congestion. On the other hand, non-recurrent congestion occurs under network disruptions such as accidents, weather conditions, vehicle breakdowns, road works and special events. Non-recurrent disruptions dramatically reduce the available capacity and reliability of the entire transportation system.

In Snelder et al. (2012), the recurrent and non-recurrent congestions are analyzed

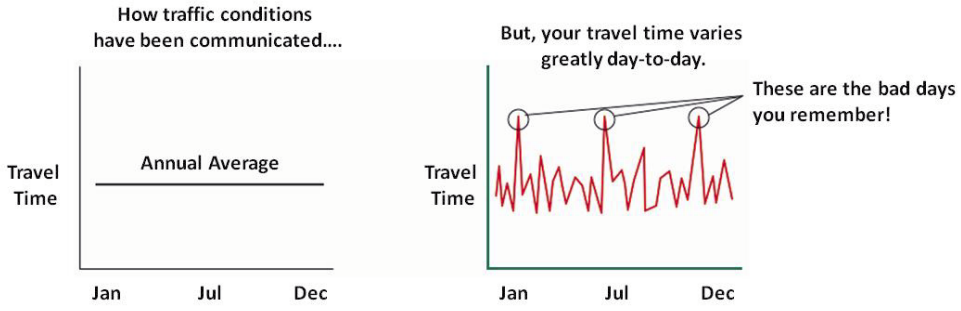


Figure 2.1 Perceiving travel time with average and real-time values (Schrank et al. 2012)

in terms of the degree of recurrence by providing the information about the predictability of the traffic disruptions (Table 2.1). Accordingly, both recurrent and non-recurrent disruptions can be predicted with information about occurrence and their impacts (e.g. peak hour congestions, maintenance activities, special transport, holiday traffic, weather conditions). The non-predictable disruptions also cause both recurrent and non-recurrent congestions. The occurrence and frequency of non-predictable disruptions are either analyzed using mathematical formulations and simulations or learnt though real-time information and their impact is estimated via ATIS or traffic authorities.

Table 2.1 The classification of traffic disruptions

	Predictable	Non-Predictable
Recurrent	Peak hour congestion, off peak hours, weekend traffic, bridge openings, small maintenance activities.	Small incidents
Non-Recurrent	Holiday traffic, big events, large maintenance activities, special transport, extreme weather conditions	Calamities, big accidents, defective infrastructures, crisis, attacks, road closures

The percentage occurrence of recurrent and non-recurrent congestions is different for each country. In the Netherlands, the recurrent congestions account for on average 80% and non-recurrent congestions account for on average 20% of the total traffic congestions (Rijkswaterstaat, 2013). The percentage of non-recurrent congestions increase up to 64% and 55% in Germany and USA, respectively (Medina, 2010). According to the studies (Medina 2010, Suson 2010), non-recurrent congestions cause most of the delays in Germany. According to Suson (2010), using alternative roads guided with the real-time information is the most useful when there are non-recurrent congestions.

In this thesis, we tackle both recurrent and non-recurrent disruptions that can be described with a known probability distribution. We assume that for these disruptions, we are able to specify disruption types, their disruption transition rates and their impacts on travel time through real-time, predictive and historical information. Note that the consideration of the recurrent congestion is not in terms of time-dependency which will be explained in the following section.

Time-dependency versus Travel-time Dependency

In most of the routing algorithms, the travel time is modeled as time-dependent. This means that, travel time changes during the time of the day and even from one day of the week to another (Figure 2.2). The recurrent congestions are modeled in terms

of time-dependency where the traffic flow and average travel times at specific time intervals (e.g. rush hours) can be estimated via the historical information and known before going en-route.

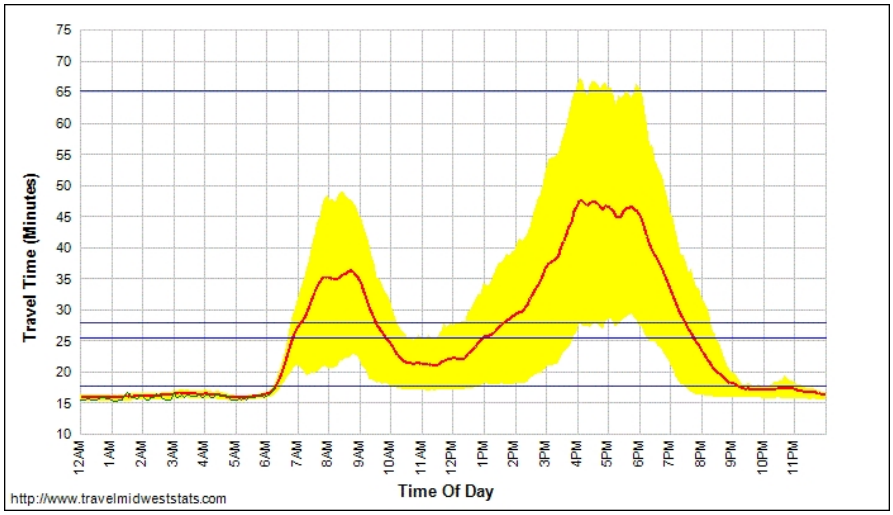


Figure 2.2 Sample data for time-dependency

Another way to model travel time is to consider the probability distribution of the duration of a congestion caused by a disruption. In the literature, when there is a disruption on a road, the distribution of travel time considering disruptions is modeled according to Increasing Failure Rate (IFR) and Decreasing Failure Rate (DFR) models (Thomas and White 2007). The theory states that the longer a road has been in congestion, the more likely the congestion will clear. For instance, the disruptions are repaired in time due to incident maintenance activities and the length and capacity of the road. Suppose that we observe a disruption in a successor road. The sooner we arrive to that road, the probability of observing the road still in disruption is high. The later we arrive at that road, the probability of finding the road in disruption decreases and converges to the steady-state (Figure 2.3). In this sense, the travel time distribution affected from disruptions is dependent on the current travel time. We denote this as "travel-time-dependency". In this thesis, we use this property to model the impact of disruptions on the travel time distribution.

The travel-time-dependency enables us to predict the future disruption state information with transition rates to other disruption states when we arrive at each intersection. In this thesis, we denote such information as "predictive information". This type of information depends on the current location of the traveler, the real-time information on the current disruptions and the decision.

In Chapters 3-5, we focus only on travel-time-dependency for recurrent and non-recurrent congestions. We do not consider recurrent congestions in terms of time-

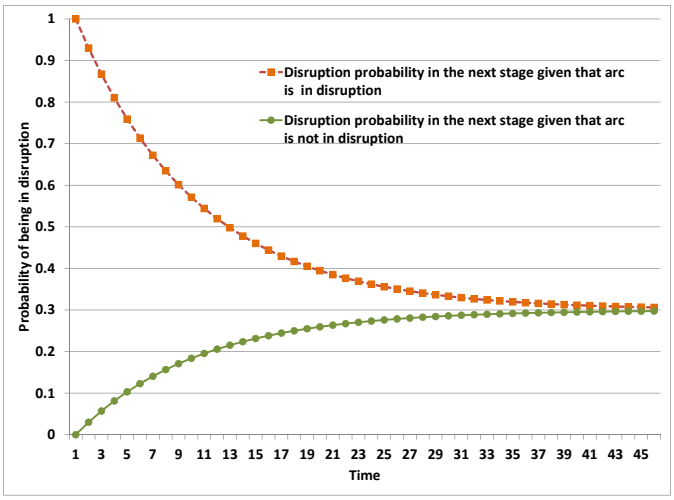


Figure 2.3 Travel-time-dependency

dependency. Rather, we model the system in terms of disruption states and transition rates. We argue that by measuring the flows of the road depending on the disruption, the delay time can be predicted (This will be discussed in Section 2.3). Furthermore, we collect the travel time information real-time where we observe the actual disruption state. In this way, the impact of each congestion is measured dependent on the disruption type instead of being time-variant. Therefore, the routing model we use is travel-time and state dependent.

Note that time-dependency can be adapted to the dynamic routing algorithms by including a time dimension to the travel-time-dependent travel times and disruption probabilities. In this case, the travel times, disruption types, steady state disruption probabilities and transition probabilities will be dependent on the time that the traveler passes the route.

2.2. Traffic Information in Practice

Advanced traveler information systems (ATIS) provide travelers updated information about the network conditions for better routing decisions to reduce traffic congestions. ATIS can both provide real-time and historical traffic information. The real-time information consist of the network conditions at the time of traveling. It can be either variable message signs on the roads (VMS), radio information or information from navigation technologies. The historical information can be estimated in terms of speed profiles considering time-dependent travel times with daily fluctuations.

The navigation technologies such as TomTom, Inrix and IBM provide travelers historical and real-time information. Additionally, predictive data is also provided

where the near-term traffic flows can be estimated. Recently, Inrix has offered travelers dynamic predictive traffic information in the United States where flow patterns in short intervals (15 minutes) to a longer horizon (1 year) can be predicted (Inrix 2012). This type of information leads to better precision of the network knowledge. Furthermore, IBM launched a traffic prediction tool to predict traffic flows over pre-set durations of 10, 15, 30, 45 and 60 minutes (IBM 2013). Integration of such information in the route planning can be used in new ways to build a smarter transportation systems. These advances show that the availability and accuracy of traffic information is getting better which presents researchers an opportunity to develop intelligent routing algorithms to reduce delays wasted in traffic disruptions.

The navigation technologies also provide routing software to handle the increasing rate of congestion on the roads. For instance, TomTom has launched a route planner where both historical and real-time information are being used (TomTom 2013). The historical information consists of speed profiles with expected travel times, analysis of bottlenecks, time-dependent travel times and estimated traffic volumes. In this route planner, historical information is considered to be a good approximation for the roads that are far away from the current position when there is no predictive data available. The real-time information considers data from GSM/ GPS probes with feed on delays and incidents as well as flow conditions, travel time and delay information on customer defined routes. The real-time information is used mostly when there is a disruption.

In this thesis, we assume that we are able to use the traffic information from ATIS as an input information. We use three types of traffic information. Real-time information, historical information (time-invariant disruption probabilities) and travel-time-dependent disruption information where we compute and predict the probability of having a disruption when we arrive at any link. We assume that the real-time information provided by ATIS is accurate and timely.

2.3. Dynamic Routing Mechanism

In this chapter, we develop dynamic and stochastic routing models to suggest travelers routing policies for each network condition. For this, we need traffic information as an input. As input information, traffic information about the disruption type, transition probability and travel time based upon disruption types are considered. To solve these complex dynamic and stochastic routing models, fast and efficient routing algorithms are developed. The output of the routing algorithms is routing policies dependent on each network condition. These policies can be computed at offline level as a table of policies versus states. As travelers receive the real-time information, the relevant routing policy can be chosen. Alternatively, the algorithms are computationally fast enough to be implemented at online level as well.

In this chapter, first, we explain how to collect the traffic information for our routing models and then, we will explain how we build the models and finally we will

illustrate how to implement the routing policies.

2.3.1 Traffic Information Retrieval

As stated in Section 2.1, both recurrent and non-recurrent disruptions can be predicted before going en-route. Furthermore, there are also disruptions that can not be predicted in advance but occurring frequently like accidents. The research papers that will be discussed in this section show that the likelihood of experiencing non-predictable disruptions and their impacts on travel time can be estimated. The practitioners from navigation industry also tell that the non-recurrent and non-predictable disruptions information can be handled at real-time. In this case, when these disruptions occur, real-time predictive information is received by considering the capacity reductions (e.g. how long it will take and how fast it will be cleared, etc.). In this section, we will explain how to retrieve information about these disruptions that we use in our dynamic routing algorithms.

The traffic state is defined with three parameters: speed, flow and density. Within time, these parameters change so the traffic state also changes. In this thesis, we define each traffic state that changes the travel time as a "disruption level". When we define the disruption level with respect to the traffic state, we can retrieve the impact of each disruption on travel time and also it becomes possible to compute the transition probabilities from one disruption level to another (Wang et al. 2010).

When density on a road increases, due to the limited capacity, the speed of the vehicles decreases leading to an increase in the travel time. For each road, there is a maximum possible density that can be provided by the available capacity. When this boundary is reached, the speed decreases even to zero. When we define the traffic states in terms of various density intervals, the speed of vehicles will also determined accordingly. In this way, we are able to determine the disruption levels and their impact on the speed and so the travel time.

In the traffic literature, there are various theorems stating that the travel times change in several boundaries of the density of a road. For instance, the three-phase theory (Kerner 2004) states that on a road there is a free flow and two congestion phases. When the density of a road reached its maximum boundary, then the travelers transit from a free flow state into a congested state: first a synchronized flow then a wide moving jam. The current empirical studies on the features of traffic congestion also show that the highways can have multiple levels of disruption which are specified according to the speed level and possible spillbacks (Helbing et al. 2009, Rehborn et al. 2011).

There are also several papers on predicting the probabilities and impacts of non-predictable disruptions such as accidents and road closures. In most of these studies robust networks are developed. The methodologies and observations in these studies provide route planners an opportunity to consider both recurrent and non-recurrent disruptions for routing decisions. Hall (1993) models and simulates

the probabilities and the delays caused by non-recurrent disruptions. It is found out that the delays caused by non-recurrent disruption are significantly higher than the ones from recurrent disruptions. Schreuder et al. (2008) develop a model to estimate the probability of disruptions and compute the vulnerability of the network by considering the total delay and the probability. In Immers et al. (2004) and Snelder (2010), the probability of having disruptions is modeled by considering vehicle kilometers driven and capacity reductions to develop robust networks. There are also statistical techniques and simulation based models which are used to estimate disruption types, impacts and durations (Knoop 2009, Miete 2011). Bottom et al. (1999), Ben-Akiva et al. (1997) developed a simulation based software, DynaMIT, for realtime guidance generation in an operational traffic information center. By this software, network conditions are estimated, network states are estimated and predictions of network conditions are executed and traveler information is generated.

As we model the routing algorithms in terms of a discrete time, finite Markov Decision Process, we adapt the traffic state discretization procedure developed by Wang et al. (2010). In this thesis, we refer this paper for the disruption probability retrieval because the disruptions are considered as random events and traffic state transitions are modeled with a discrete time Markov Chain. The traffic state mechanisms are analyzed by using empirical observations. In this study, the disruption probabilities are linked to how many vehicles are occupying a certain road depending on the state of the road at the time of observation. Because we model travel time as time-invariant and by considering state dependencies, the disruption transition probability retrieval model used in this paper is appropriate for the discrete time, finite MDP model used in this thesis. Therefore, the model is appropriate for modeling both recurrent and non-recurrent disruptions in terms of traffic states. In this model, the density and time ranges are partitioned into discrete intervals with specified density increase. The upper limit of the interval is the jam density where the maximum density level is reached and the cars are stopped.

In this study, M/G/C/C state dependent queuing theory is used to model disruption mechanism. The model is mostly used for vehicular traffic flow. The details of the model can be found in Jain and Smith (1997). In this model, speed is related to the number of vehicles occupying the link and service rate is defined by the ratio of average travel speed to free flow speed. Then, the transition probabilities are computed, by using M/G/C/C state dependent queuing theory. In the queuing model, the service rate can be computed by finding the ratio between the average travel speed to the free flow speed. So, as the speed decreases with respect to the free flow speed, the service rate decreases. It is intuitive that the increasing congestion on the link results in decrease in the service level on that link. When we consider a congestion model using a discrete Markov process with a link capacity, the rate of change in the traffic flow on a link is defined as the ratio between the speed of the link at that stage and the free flow speed (Jain and Smith 1997, Wang et al. 2010). For details, we refer readers to Wang et al. (2010).

For the routing models developed in Chapters 3-5, The steady state information and the transition probability rates can be derived from the M/G/C/C queuing theory. This information conversion is done at offline level by using the historical information. This historical information and the real-time information for the current disruption states, current travel time and disruption rates are used to compute the travel-time-dependent disruption transitions. By this way, we derive the predictive information at offline level.

The historical information may be updated and can be different from departure to departure due to dynamic and stochastic road networks. Therefore, before computing any offline decisions, the information should be updated according to the new network conditions.

2.3.2 Dynamic and Stochastic Routing Model

In this section, we briefly explain how we model routing problems considering network disruptions by considering travel information as described in Section 2.3.1. Due to the structure of the network and our objective of minimizing expected total travel time from origin to destination, we refer the problem as a dynamic shortest path problem (DSPP) as described in Chapter 1. In Chapters 3-5, we propose a multi-stage stochastic dynamic programming model based on a discrete-time, finite Markov Decision Process (MDP) to formulate DSPP. MDP is a widely accepted model for modeling dynamic and stochastic shortest path problems due to practical use for online applications.

Consider a road network where the stochasticity is due to the occurrence of disruptions on the roads. Let the graph $G(N, A, A_v)$ represent a road network, where the set N represents the finite set of nodes (modeling road intersections), A the set of arcs (modeling roads between nodes) and A_v the set of vulnerable arcs (vulnerable roads), potentially in disruption ($A_v \subseteq A$). We now describe the routing model considering the route network example in Figure 2.4 which depicts the network presentation of the roads from The Hague to Utrecht in presence of traffic disruptions. In this network, there are 7 road intersections (intersection of highways or intersection of smaller roads), 8 roads, 2 of which are vulnerable (the dashed roads) where the travel time increases due to stochastic disruptions. The vulnerable roads are identified such that on these roads, the velocities of the vehicles and the travel times significantly fluctuate depending on random disruptions.

The traveler wants to reach to Utrecht from The Hague with minimum expected total travel time. The travel time on a road is assumed to be predictable from historical data and follows a discrete distribution given the disruption status of the road. For instance, the travel time to traverse the vulnerable road between intersections 3 and 4 is 5 minutes if there is no disruption; 8 minutes if there is a disruption level 1; and 90 minutes if there is a disruption level 2. From vulnerability analysis, disruption level 1 can be congestion due to rush hour and disruption level 2 can be congestion due to an incident. As described in Section 2.3.1, the disruption levels are determined based

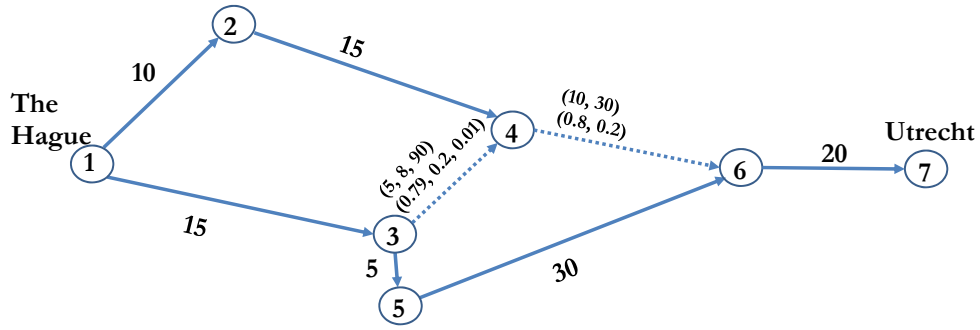


Figure 2.4 A road network example

on the density of roads. For example, for the road $3-4$, the disruption levels are determined as follows:

$$\text{Disruption level} = \begin{cases} \text{No disruption} & \text{Density:} & (0, 25], \\ \text{Disruption level } 1 & \text{Density:} & (25, C), \\ \text{Disruption level } 2 & \text{Density:} & C^*. \end{cases} \tag{2.1}$$

* C is the capacity of the road and vehicles stop at this point.

Transition probabilities for these disruption levels are computed with historical observations by considering queueing theory as described in Section 2.3.1. Furthermore, we receive the real-time information about the disruption statuses of all vulnerable roads when we arrive at an intersection. This is exogenous input information to the model.

The traveler arrives at an intersection and makes a decision at discrete time points in the planning horizon which we denote as a "stage". As traveler moves along the network, the stage represent the number of intersections that have been visited so far from the departure intersection, The Hague. The final stage is reached by arriving at the destination, which is Utrecht in our example.

The network state at any stage, is evaluated by the current intersection and the disruption information of the vulnerable roads. The disruption information consists of which disruption level the vulnerable road is in at the specific stage. For example, when we are at The Hague, we are at the first stage. We observe that the vulnerable road, $3-4$ is in the disruption level 3 and the vulnerable road $4-6$ is in the disruption level 1 (By retrieving the disruption information, we also retrieve the travel time information).

At each intersection, the traveler observes the network state and makes a decision about which intersection to travel next. For example, given the network state at stage 1, we can either go to the intersection 2 or 3. When a decision is made to

travel to an intersection, an immediate cost is incurred for traversing the current road in the current network state. In Chapters 3 and 4, the immediate cost is the travel time from the current intersection to the next intersection given the disruption information at the stage where the decision is made. In The Hague, the traveler receives the disruption information on the network and observes that the current travel time between The Hague and the intersections 2 and 3 is 10 and 15 minutes, respectively. It is assumed that this value does not change until the traveler arrives at the next stage. Different from other chapters in Chapter 5, the immediate cost is the expected travel time while considering the propagation effect of disrupted successor links to the current road. For instance when the road $3-4$ is disrupted, due to the limited capacity of the roads, the congestion will spill back to the road between The Hague and the intersection 3. Thus, the travel time of this road will be longer than 15 minutes due to the spillback effect.

Based on the information about the network state, a routing decision should be made. While making a routing decision, the possible future outcomes should be considered because what will happen to the network at the next stage is uncertain. The disruption statuses of the vulnerable roads may change as we proceed to the next stage due to traffic dynamics and travel-time-dependency. To obtain higher solution quality, this stochastic process should be taken into account by computing the expected travel time of each road in the future stages. As we receive real-time information, depending on a priori information, expected travel times can be computed by summing the product of the transition probabilities to different network states with the resulting travel times. The transition probabilities are computed by considering the travel-time-dependency. For instance, consider that the traveler is at The Hague at stage 1 and observe the network state as: the vulnerable road, $3-4$ is in the disruption level 3 and the vulnerable road $4-6$ is in the disruption level 1. If the repair rate of the road $3-4$ is low, within a short travel time it's highly probable that by the time traveler arrives at the intersection 3, the disruption will be still there. If the travel time between The Hague and node 3 takes too long and/or the repair rate of the road $3-4$ is high, it is quite likely that the disruption will clear out. By considering these transitions and the probabilities of having these transitions, an expected travel time can be computed for each vulnerable arc.

The objective is to minimize the expected total travel time from the origin until the destination node over all stages. To solve this combinatorial problem, we develop efficient algorithms where a stationary routing policy is found by mapping each decision to a network state. In this policy, for each intersection according to the realizations of network states, a decision is provided. The decisions are dynamic as they change depending on the network state realizations. Finding routing policies is done at offline level and during travel as real-time information is retrieved, the relevant decision is selected according to the decision-network state mapping.

Table 2.2 shows the routing policy as an output of the dynamic and stochastic routing algorithm for the described example. This routing policy is obtained before departure.

Table 2.2 The routing policy for the network given in Figure 2.4

Int.\State	(3,4)=ND*; (5,6)= ND	(3,4)=D1*; (5,6)= ND	(3,4)=D2*; (5,6)= ND	(3,4)=ND; (5,6)= D*	(3,4)=D1; (5,6)= D	(3,4)=D2; (5,6)= D
The Hague	3	3	2	3	3	2
2	4	4	4	4	4	4
3	4	4	5	4	4	5
4	6	6	6	6	6	6
5	6	6	6	6	6	6
6	**Utrecht**	**Utrecht**	**Utrecht**	**Utrecht**	**Utrecht**	**Utrecht**

ND: Not in disruption.
D: In disruption.
D1: Disruption level 1.
D2: Disruption level 2.

2.3.3 Implementation of the Dynamic Routing Policies

Chapters 3-5 provide readers with dynamic routing algorithms that change the traffic information into an output of routing policies. In this section, we will describe how to implement these routing policies by considering the example in Figure 2.4.

Figure 2.5 shows how a routing policy is implemented before the departure until the arrival at the destination. Before the departure, updated historical information about the possible disruption levels on the roads, the estimated travel times based on the determined states and the disruption transition rates are retrieved. Then, considering these input information, routing algorithm provides the planners with a stationary policy as shown in Table 2.2. The dynamic decisions in this thesis can be chosen at the online level given a priori routing policy. The traveler receives real time information from ATIS and then the dynamic routing policy is chosen accordingly.

If not all the network state information can be determined at the offline level and if the steady-state probabilities change dynamically during travel, the routing policies may also be determined at the online level. Then, the information is updated and gathered each time the traveler arrives to an intersection and the intersection to travel next is determined by the algorithm during the travel. In this thesis, we ensure that the computational time is fast enough for also online implementation.

Figure 2.5 shows the time-line for the dynamic routing mechanism considering the example in Figure 2.4. Before going en route, we have the routing policy which maps each network state to a decision. The traveler starts traveling at The Hague at time $15:00$. At this intersection, the real-time information is received considering the network state of all roads. Assume that the traveler receives the information that the road $3-4$ is in disruption meaning that it is blocked and its travel time is 90 minutes. Furthermore, the road $5-6$ is also in disruption with travel time 30 minutes. All the other links are not in disruption. The routing policy suggests that the traveler next visits the intersection 2 given this information. When the traveler arrives the intersection 2 at time $15:10$, the real-time information is updated again and the dynamic route is selected according to the new network state. This process continues

until the destination. Considering the dynamic and stochastic routing algorithm the travel time becomes 55 minutes.

If we use static routing algorithms, then no real-time and predictive information is considered. The traveler follows a path computed by the expected values considering stationary probabilities. By only considering the stationary probabilities given in Figure 2.4, the policy would be to follow the path " The Hague-3-4-6-Utrecht". Under the real-time information obtained on the day of implementation, this path causes higher travel time (more than 135 minutes). If we consider only real-time information for the disrupted roads, then the algorithm suggests to travel to the intersection "3" and then "4-5-6" which then gives higher travel time (70 minutes) in total. In this thesis, we show the value of considering real-time information while exploiting the stochastic process in the routing networks by developing dynamic and stochastic models. We also develop static and online models to compare the performances as provided in this specific example.

In Chapters 3-5, we develop dynamic routing algorithms considering the mechanism described above. In these chapters, we show under what network conditions which type of information and routing algorithm provide higher performance. Considering the dynamic and stochastic shortest path literature discussed in Chapter 1, the models used in Chapters 3-5 are similar to the ones used in Kim et al. (2005b), Thomas and White (2007) and Gao and Huang (2012). Kim et al. (2005b) consider only time-dependency, Thomas and White (2007) focus on only non-recurrent disruptions and Gao and Huang (2012) provide analysis on optimal policies in case of time-dependency and link correlations due to non-recurrent disruptions. In this thesis, considering a more advanced traffic data gathering mechanism as discussed above, we extend these studies by modeling the non-recurrent and recurrent disruptions with an MDP using discrete network states. Furthermore, we provide an analysis on the efficient levels of real-time, historical and predictive traffic information for higher solution quality and lower computational time (Chapter 3). In Chapter 4, we develop approximation algorithms to handle large traffic states with many levels of disruptions. In Chapter 5, we extend the concept of link-dependency in Gao and Huang (2012) by providing a dynamic routing model where we consider explicitly the propagation effect of disruptions.

Before Departure	Retrieve information on disruption levels, probabilities and the realizations.	Run the routing algorithm to retrieve the routing policy.

Time	Real-world event	Decision	Graph
15:00	There is vehicle breakdown at A12 between intersections 3 and 4. The travel time is 90 min with disruption level 2. There is also a congestion at road 4-6 with travel time 30 with disruption level 1.	First stage, the vehicle is at The Hague.	
		Observe the network information. Given the policy choose the relevant decision.	
		Decide to go to intersection 2.	
15:10	The disruption at road 3-4 still continues. The travel time is 90. The congestion at road 4-6 still continues with travel time 30 min.	Second stage. Vehicle arrives at the intersection 2.	
		Observe the network information. Given the policy choose the relevant decision.	
		Decide to go to the intersection 4.	
15:25	The disruption at 3-4 still continues. The disruption at road 4-6 is cleared. The travel time is 10 min now.	Third stage. Vehicle arrives at the intersection 4.	
		Observe the network information. Given the policy choose the relevant decision.	
		Decide to go to the intersection 6.	
15:35	The disruption at 3-4 still continues.	Fourth stage. Vehicle arrives at the intersection 6.	
		Observe the network information. Given the policy choose the relevant decision.	
		Decide to go to Utrecht.	
15:55		Termination stage. Destination is reached.	

Figure 2.5 Timeline for dynamic routing

Chapter 3

DSPP: Hybrid Routing Policies considering Network Disruptions

As we mentioned in Chapter 1, in this thesis we focus on dynamic routing policies in stochastic networks. For dynamic routing decisions, planners need network information. In Chapter 2, it is shown that there is a wide variety of traffic information due to advances in information systems. In practice, mostly historical (offline) information and real-time (online) information are used to overcome the negative impacts of traffic disruptions. However, it is crucial to analyze which type of information is useful and efficient under which network types and conditions. In Chapter 3, we start with investigating the efficient levels of network information to consider in developing higher quality and computationally efficient dynamic routing policies. Then, with the observations we make in this chapter, we will be able to develop computationally efficient algorithms with high solution quality for more complex networks.

3.1. Introduction

In traffic networks, link disruptions due to accidents, bad weather and traffic congestion lead to a significant increase in travel times and decrease the probability of being on-time at the final destination. As stated in Chapter 2, the statistics at the Inrix Traffic Scorecard in 2012 (Inrix (2012)) shows that in European countries, (especially at Belgium and Netherlands), an average driver annually experienced in

total at least 50 hours of delay in congestion due to these disruptions. In practice, navigation technologies (like TomTom Route Planner TomTom (2013)) respond to these disruptions by using offline information (i.e. the historical data giving information on the expected state of the network) and/or online information (i.e. the information on the actual real-time state of the network) to create paths that are less affected. As such, considering the stochastic nature of the network with real-time information generates better solutions for the navigation systems. However, this information does not come for free since this may need higher routing calculation times and higher information retrieval costs. Therefore, it is essential to use routing policies that effectively respond to the network disruptions and give high quality solutions within acceptable calculation times. One important question to be answered is thus how to balance the level of online information and historical information needed to obtain higher solution quality with lower calculation times.

In this chapter, we focus on the dynamic stochastic shortest path problems with stochastic disruptions at the network. We are interested in networks where a single traveler has a certain origin-destination pair. We model the problem as a discrete-time, finite Markov Decision Process (MDP). The traveler gets the real-time information about the state of the network before arriving at each node which is an intersection of roads and/or highways. At every intersection, the traveler decides which next link to follow depending on the current state of the network. The next link to travel is the link with the minimum expected travel time until the destination given the current state of the network. The system transits to a new state depending on the travel time of the chosen link. We denote this property as travel-time-dependency (Chapter 2). We focus on networks that represent highways where drivers are able to get online and historical information easily. We assume that travel times can be predictable from historical data and retrieved from online data. The state of the whole network can be retrieved from Intelligent Traffic Systems (ITS). Considering these, we have three possible traffic information to use: online information (real-time travel information of the links retrieved from ITS), travel-time-dependent probability distributions for the links for which we have online information and offline information (time-invariant data using the historical information). The chapter addresses the advantage of using different information for different parts of the network. To test the value of considering different levels of traffic information, we develop various routing policies.

In the dynamic shortest path literature, various routing policies are discussed to handle disruptions on the network. Routing policies can be categorized into offline and online policies. In the offline policies, the generation of paths is done before the traveler goes en-route. For instance, the naive policy considers the deterministic travel times ignoring any disruptions. The disadvantage is that, in case of a disruption, the cost of traveling the static route increases dramatically. The robust routing policy (Chen et al. 2009, Donati et al. 2003, Ferris and Ruszczynski 2000) is an offline policy that considers disruptions at the network. In this policy, a robust path is selected, considering the disruption probabilities. One often-used robust policy is the worst-

case scenario policy, where the robust path is constructed assuming that disruptions occur with 100% certainty. The drawback of this policy is higher expected costs, since the worst-case scenario is only realized with a small probability. Clearly, the disadvantage of offline policies is their inability to efficiently respond to the real-time dynamics resulting from the disruptions. By contrast, online policies update the routes based on the realization of the disruptions, while traveling through the network. Using online recourse policies (Kim et al. 2005b, Fu 2001, Polychronopoulos and Tsitsiklis 1996), the shortest paths are generated and updated during the trip as the online information is retrieved from an in-vehicle communication system. In this case, re-optimization is potentially executed after each retrieval of the online information.

This chapter extends the literature by providing a detailed analysis on the efficient level of online versus offline information for the dynamic routing decisions which has not been done in the literature to our knowledge. Moreover, we develop hybrid policies for the dynamic shortest path problems which will be also considered in Chapters 4 and 5. These hybrid policies combine different levels of real-time information, time-dependent and time-invariant probability distributions of the link travel times at the network. The hybrid policy limits the state-space explosion by using both the time-dependent data (explodes the state space) and the time-invariant data (has limited effect on the state space). The strength of the hybrid policies is the ability to respond to the disadvantages of both the offline and the online routing.

The main contributions of this chapter are as follows:

1. We develop a generic framework based on dynamic programming to formulate and evaluate different policies for the dynamic shortest path problem considering the link disruptions.

2. Using the dynamic programming framework, we model offline and online policies. Next to this, we develop new hybrid policies that combine the strengths of both the online and the offline policies. Specifically, these hybrid policies take into account the online and the offline information about the state of the network by using different levels of both the time-dependent and the time-invariant probability distribution information.

3. We test the efficiency of our hybrid policies and compare this to the offline and the online policies for different network structures. These network structures vary in network size, number of vulnerable links and their disruption rates. The numerical results show that, for more complex networks and under more realistic conditions, where the detailed information about the network is more effective, our hybrid policies perform outstanding. An overwhelming part of the cost reduction can be obtained by only considering a limited amount of the online information. The hybrid routing policies have thus crucial implications for practice, as we do not need to collect the real-time information for the complete network to obtain a good performance.

The structure of the chapter is as follows. In Section 2, we provide a selected literature review on dynamic shortest path problems. Section 3 provides the details of our modeling framework based on dynamic programming. Section 4 discusses the offline, online and hybrid policies within this framework in detail. In Section 5, the experimental design, the numerical results and the important insights are discussed. In Section 6, we provide an overview of the results and future directions.

3.2. Modeling Framework

We model the Dynamic Shortest Path Problem with real-time information on the realizations of disruptions on the network as a discrete time, finite (finite state, finite action and finite decision) Markov Decision Process (MDP). Consider a traffic network represented by the graph $G(N, E, E_v)$ where the set N represents nodes (or intersections), E the set of undirected links (edges) and E_v the set of vulnerable links (edges), potentially in disruption $E_v \subseteq E$. If we consider a real road network, a node represents an intersection or a junction that connects highways such as an entry/exit point. A link represents a segment of a highway. Figure 3.2 shows an example highway network between an origin-destination pair in Netherlands. We choose highways rather than roads because the online information is mostly available for highways, and the number of highway disruptions is much higher. Each of the vulnerable links at the network has a known probability of going into a disruption and a known probability of recovering from this disruption.

We assume that the travel time is discretized. In this chapter, we propose a general discrete time finite MDP where the edges can be both undirected and directed. In case revisiting nodes is advantageous, we allow both forward and backward moves from the current node. For the undirected case, we assume that the travel time between any nodes is symmetric which means that the travel time is the same for both directions. We also assume that waiting at a node is not allowed and does not provide travel time savings. In other words, we assume a first-in-first-out property. We also assume that the disrupted links are independent that indicates that there is no correlation between travel times of neighbor edges.

We refer a decision epoch as stage k where we observe the state of the network and make a decision. Stage k is equivalent to the number of nodes (intersections) that are visited so far from the initial node. The process terminates when the destination is reached, which is referred as the goal state. As the termination is defined by a goal state ($\mathbb{G}$), the end of horizon is a priori unknown but finite. Therefore, the time horizon is a random variable which is represented by K where $K < \infty$.

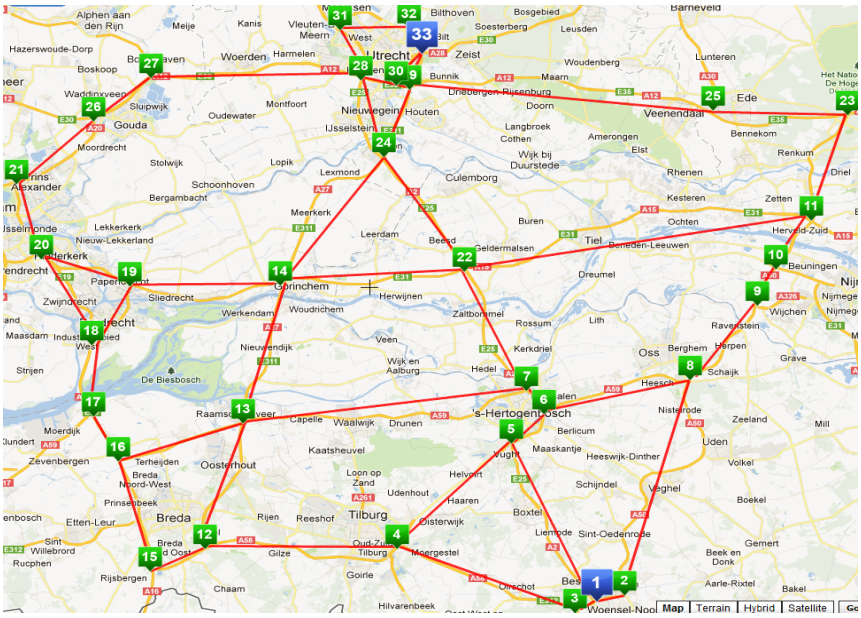


Figure 3.1 Road network

The MDP Formulation

States

The state of the system at stage k, $S_k \in \mathbb{S}$, is represented by two components:

- The current node at stage k, $i_k \in N$.
- The disruption state vector at stage k is denoted by vector D_k. We represent the generic random variable vector that includes the possible disruption states for all vulnerable links at stage k as $D_k = (D_k^1, D_k^2, ..., D_k^R)$ where $R = \|E_v\|$. Here, D_k^r represents the r^{th} element of the disruption vector which is actually the state of the r^{th} vulnerable link at stage k. Each random variable can take any value from the disruption scenario vector U_r at stage k. The disruption levels for each vulnerable link are represented by the disruption level vector U_r where $D_k^r \in U_r$. For each link, there can be M_r different types of disruption levels: $U_r = \{U^1, U^2, ...u^{M_r}\}$. For example, if there are two disruption levels, $M_r = 2$ and the disruption level vector becomes: $U_r = \{0, 1\}$. Note that at each stage, we use a realization of the disruption state vector, $\hat{D}_t$ because at each stage we observe the realizations of the disruptions at the network ($\hat{D}_k^r \in U^r$).

The state of the system at each stage k is then: $S_k = (i_k, \hat{D}_k)$. The process terminates when destination is reached. The possible goal states that include the pairs of

destination and the possible disruption realizations is referred by the goal state set $\mathbb{G} \in \mathbb{S}$. The goal state is assumed to be an absorbing and cost free state.

Actions

The action at stage k, x_k, is to determine which node to travel next such that the total expected travel time until the destination node, d, is minimized given the current state. The action set is limited to the neighbors of the current node i_k. Define the neighborhood as $Neighbor(i_k)$. Then, $x_k \in Neighbor(i_k)$ if there exists a link between the nodes i_k and x_k.

$X_k^\pi(S_k)$: Decision function that determines the node at stage $k+1$, (i_{k+1}) which is determined at stage k under policy π given state S_k.
Π : Set of possible policies. Each $\pi \in \Pi$ refers to a different policy where $\{X_k^\pi(S_k)\}_{\pi \in \Pi}$ is the set of decision functions.

The Cost Function

Traveling from node i_k to x_k with a transition from the state S_k to S_{k+1} results in a non-negative cost which is the deterministic travel time between the two nodes given the current disruption state.

Denote $t_{i_k,x_k}(\hat{D}_k)$ as the known and discrete travel time between i_k and x_k given the disruption state realization. At each node i_k, an immediate cost occurs from choosing action of traveling to x_k. We refer to this cost as $\tilde{c}(S_k, x_k)$ where:

$$\tilde{c}(S_k, x_k) = t_{i_k,x_k}(\hat{D}_k). \tag{3.1}$$

Upon arrival at node i_k, the actual value of the travel time from the current node i_k to all neighbor nodes given the current disruption state is retrieved. In the model, we assume that the travel time of the links emanating from the node i_k is purely based on the disruption state of the network at the moment we enter the link.

The State Transition Function

Given the state S_k and a selected action (e.g. travel to neighbor node x_k), a transition is made to the next state, $S_{k+1} = (i_{k+1}, \hat{D}_{k+1})$. The state transition involves the following transition functions:

$$i_{k+1} = x_k, \tag{3.2}$$
$$D_{k+1} = \hat{D}_{k+1}. \tag{3.3}$$

The disruption status vector transits from $\hat{D}_k$ to $\hat{D}_{k+1}$ according to a Markovian transition matrix. Note that D_{k+1} is the vector of random variables representing the disruption status of each vulnerable arc in the network for the next stage.

Let $p^r_{u,u'}$ denote the unit-time transition probability between any two disruption levels of the vulnerable arc r, $p^r_{u,u'} = P\{\hat{D}^r_{t+1} = u' | \hat{D}^r_t = u\}$. As an input to the model, we define the unit-time-transition matrix for a vulnerable link r, $r \in E_v$, with M_r possible disruption scenarios as follows:

$$\Phi^r_{t,t+1} = \begin{bmatrix} p^r_{u^1,u^1} & p^r_{u^1,u^2} & \cdots & p^r_{u^1,u^{M_r}} \\ p^r_{u^2,u^1} & p^r_{u^2,u^2} & \cdots & p^r_{u^2,u^{K_r}} \\ . & & & \\ . & & & \\ p^r_{u^{M_r},u^1} & p^r_{u^{M_r},u^2} & \cdots & p^r_{u^{M_r},u^{M_r}} \end{bmatrix} \quad \forall t = 0,1,2,..., \quad \forall r \in E_v.$$

The transition from state S_k to S_{k+1} depends on the discrete travel time between nodes i_k to x_k given the disruption state, $t_{i,x_k}(\hat{D}_k)$. This is equal to the time between two decisions at stage k and $k+1$ which is an integer. Due to the nature of most disruptions in traffic, if we know that the successor links are disrupted at stage k, the longer it takes to arrive at those links, the more likely that the disruptions will be cleared in the next stage $k+1$. After sufficient long time, the probability of having a disruption goes to the steady state as shown in the Appendix A. Accordingly, we define the travel-time-dependent transition matrix for vulnerable link r (between nodes i_k and x_k) from stage k to $k+1$ as:

$$\Theta^r(t_{i_k,x_k}(\hat{D}_k)) = \begin{bmatrix} p^r_{u^1,u^1} & p^r_{u^1,u^2} & \cdots & p^r_{u^1,u^{M_r}} \\ p^r_{u^2,u^1} & p^r_{u^2,u^2} & \cdots & p^r_{u^2,u^{K_r}} \\ . & & & \\ . & & & \\ p^r_{u^{M_r},u^1} & p^r_{u^{M_r},u^2} & \cdots & p^r_{u^{M_r},u^{M_r}} \end{bmatrix}^{(t_{i_k,x_k}(\hat{D}_k))} \quad \forall k = 0,1,2..,\forall r \in E_v.$$

The probability of having the new disruption status $\hat{D}_{t+1}$ given $\hat{D}_t$ is then calculated as ($\Theta^r_{u,u'}(t|S_t,x_t)$ indicate the specific row and column index in the matrix $\Theta^r(t_{i_k,x_k}(\hat{D}_k))$):

$$P(S_{k+1}|S_k) = \prod_{r=1}^{R} \Theta^r_{u,u'}(t_{i_k,x_k}(\hat{D}_k)). \tag{3.4}$$

The Objective Function

Our objective is to minimize the expected travel time from the initial node until the goal state in a random but finite horizon K:

$$\min_{\pi \in \Pi} E \sum_{k=1}^{K} \tilde{c}(S_k, X_k^{\pi}(S_k)), \tag{3.5}$$

where $x_k = X_k^{\pi}(S_k)$ is the decision made according to the decision function $X_k^{\pi}(S_k)$ under policy π, given the current state S_k.

Bellman Equations

The optimization problem at Equation (3.5) can be solved by using a backward recursion algorithm. For this reason, we formulate the problem in terms of Bellman equations. We use the value function $V_k(S_k)$ as the value of being in state S_k that is the minimum expected travel time to go from the node S_k to the goal state at the unknown horizon K. The next node to travel to has the state $S_{k+1} = (x_k, \hat{D}_{k+1})$. Then, the total cost function for node i_k given the current state S_k is:

$$V_k(S_k) = \min_{x_k \in \mathcal{X}_k} \tilde{c}(S_k, x_k) + \sum_{S_{k+1}} P(S_{k+1}|S_k) V_{k+1}(S_{k+1}), \tag{3.6a}$$

$$V_K(S_K) = 0. \tag{3.6b}$$

where S_K is the goal state, $S_K \in \mathbb{G}$. Our solution becomes:

$$x_k^* = \arg\min_{x_k \in \mathcal{X}_k} \tilde{c}(S_k, x_k) + \mathbb{E}(V_{k+1}(S_{k+1})). \tag{3.7}$$

3.3. Solution Approach- Value Iteration Algorithm

The MDP formulation considered for the dynamic shortest path problem has a random indefinite horizon. Therefore, there is no definite end horizon to apply a backward recursion algorithm. However, as the states and actions are known and finite and there is a absorbing cost-free goal state, we can still propagate information backwards by considering a value iteration algorithm. The value iteration algorithm stops when all states, $S \in \mathbb{S}$ have finite cost until the goal state, destination. This means that we will iterate the values of each state (each node and disruption level pair) until a stationary cost value until the destination is obtained. For this we define a value function, $V_t(S)$ that is the expected total cost starting from state S for a horizon of t steps until the destination. Note that we search for stationary cost for termination, the index for stage k as shown in Section 3.2 is no longer needed to find the optimal policy.

The aim of the algorithm is to find a stationary optimal policy which minimizes the expected travel time (over a finite but indefinite horizon) incurred to reach the goal state. The algorithm is implemented offline by considering all possible known finite states. For the stochastic shortest path problem, the value iteration is guaranteed to converge to an optimal stationary value function if following conditions are satisfied (Bertsekas and Tsitsiklis 1991, Bonet 2007):

1. The state set contains a goal or a termination state ($\mathbb{G} \in \mathbb{S}$) which is cost free and absorbing:
($P(d, \hat{D}_{k+1}|d, \hat{D}_k) = 1$).
2. All non-goal states are free of absorbing cycles (self loops):
$P(i_{k+1} = i_k, \hat{D}_{k+1}|i_k, \hat{D}_k) = 0, \forall S \in \mathbb{S} \setminus \mathbb{G}$.
3. There exists at least one proper policy which reaches the goal state with a finite cost from each state with a probability of 1 ($V_t(S) < \infty, \quad \forall S \in \mathbb{S}$). This is also defined as connectivity assumption which assures that there is at least one path connecting every state with the goal state.
4. Every improper policy incurs a cost of ∞ from every state that does not reach to the goal state.
5. All the edge costs are nonnegative and discrete.

The general MDP formulation where the graph can be either directed or undirected may bring about complexities in obtaining the optimal routing policy. First of all, the graph considered in this chapter represents a general stochastic shortest path problem which is not necessarily acyclic. This indicates that the optimal policy might contain cycles because we allow detours and and we make decisions dependent on the real-time information at each decision epoch. Retour to the previous node might be profitable when the successor links travel times increases significantly due to a disruption (Polychronopoulos and Tsitsiklis 1996, Bertsekas and Tsitsiklis 1991). However, the value iteration is guaranteed to converge in finite number of stages by satisfying these 5 conditions where the graph cannot contain absorbing cycles for the non-goal states.

Proposition 3.1 If these conditions above are satisfied, the probability that the dynamic shortest path contains a cycle infinitely many times is zero.

Proposition 3.2 If these conditions are met, the general stochastic shortest path problem is shown to converge to an optimal policy with finite number of iterations (Bertsekas and Tsitsiklis 1991, Bonet 2007).

Propositions 3.1 and 3.2 indicate that the value iteration algorithm converges to a stationary cost function for each state. Due to random and indefinite horizon, we consider ϵ-consistency rule for the termination condition where $\epsilon > 0$ and small (0.001). We do this to obtain a policy that is guaranteed within ϵ of optimum. The proof for convergence is found in Appendix B. We refer the reader to Bonet (2007) for a formal proof of finding a bound for the number of iterations for convergence.

Algorithm 1 describes how to find the optimal policy with the minimum expected travel time to the destination from each state by using value iteration algorithm with ϵ-consistency rule. The value iteration algorithm iteratively updates the value of being

in state $S \in \mathbb{S}$ from any state $S' \in \mathbb{S}$ by using the Bellman optimality equations. This is called value update or Bellman update/back-up.

As the termination state is determined by arriving at the destination, we start updating the value of each state from backwards which is from the destination. At iteration $t = 0$, only the values of the goals states $\mathbb{G} = (d, \hat{D})$ are zero. The other values of other non-goal states ($\mathbb{S} \setminus \mathbb{G}$) are ∞ as the destination is not reached in zero steps and only the goal state is cost free. After the initialization of values of each state, in each iteration, the cost value for each state $S \in \mathbb{S}$ is computed by using the Bellman optimality Equations (3.8). Then, we compute the residual value between the value of being in state S at iteration t and the minimum value of being in state S among all iterations so far (Equation (3.10)). If for each state, this residual is less than ϵ (ϵ-consistency is satisfied) and the value of being in each state is less that ∞ (the goal state is reachable from each state), the algorithm stops. Then the optimal policy is computed for iteration t. With this algorithm, we satisfy the 5 conditions for the convergence of value iteration in finite number of iterations.

Algorithm 1 Value Iteration Algorithm

Step 0: Initialize iteration $t = 0$.
Set $\forall S \in \mathbb{S}$

$$V_0(S) = \begin{cases} 0 & \text{if node } i \text{ is the destination node } d, \text{ S is the goal state } (S \in \mathbb{G}), \\ \infty & \text{otherwise} \end{cases}$$

Let $t = 1$.
Step 1: $\forall S \in \mathbb{S}$
Step 1a:

$$V_t(S) = \min_{x \in N, x \in Neighbor(i)} \tilde{c}(i, \hat{D}, x) + \sum_{S'} P(S'|S) V_{t-1}(S'), \tag{3.8}$$

$$V(S) = \min_t V_t(S). \tag{3.9}$$

Step 1b: Compute residual at iteration t:

$$\Delta V_t(S) = |V_t(S) - V(S)| \tag{3.10}$$

Step 2: If $\max \Delta V_t(S) < \epsilon$ and $V_t(S) < \infty \; \forall S \in \mathbb{S}$, Stop and go to Step 4. Otherwise go to Step 3.
Step 3: $t = t + 1$ and go to Step 1.
Step 4: The routing policy becomes:

$$\pi^{V_t}(S) = \underset{x \in N, x \in Neighbor(i)}{\arg\min} \; \tilde{c}(i, \hat{D}, x) + \sum_{S'} P(S'|S) V_t(S') \tag{3.11}$$

3.4. Routing Policies

In this section, we define various routing policies within the dynamic programming framework presented in Section 3.2. We consecutively discuss the optimal, offline, online and our proposed hybrid routing policies. As one of the objectives of the chapter is to compare the developed routing policies that use different levels of online/offline information with respect to the optimal policy, we keep the number of disruption levels constant with two levels: in disruption or not in disruption. Increasing the number of disruption scenarios changes neither the formulation nor the structure of the algorithm to solve the problem. The policies can be easily modeled with more disruption scenarios by changing the one-time-period probability transition matrix. However, the increase in disruption scenarios increases the computational complexity. More complicated disruption scenarios are left for the future research.

3.4.1 The Optimal Routing Policy (Opt)

The optimal routing policy (Opt) takes the disruption state information of the whole network as an input and uses the travel-time-dependent transition probabilities. The state space at stage k is denoted by $S_k = (i_k, \hat{D}_k)$. The state space represents the current node and the disruption state of all vulnerable links. The binary disruption state of a vulnerable link r, $r \in E_v$, is denoted by $\hat{D}_k^r$.

$$\hat{D}_k^r = \begin{cases} 1 & \text{if the link } r \text{ is in disruption state at stage } k, \\ 0 & \text{otherwise.} \end{cases}$$

Given the state is S_k, a transition is made to the next state, $S_{k+1} = (x_k, \hat{D}_{k+1})$ after traveling to the next node x_k. We define the transition matrix for a vulnerable link r (between nodes i_k and x_k) from stage k to $k+1$ as Θ^r. The transition matrix for two disruption states is assumed to be given. Here, α_r is the one-time-period probability of going into a disruption and β_r is the one-time-period repair probability from the disruption. Repair probability is defined as the probability of recovering from the disruption.

$$\Theta^r(t_{i_k,x_k}(\hat{D}_k)) = \begin{bmatrix} (1-\alpha^r) & \alpha^r \\ \beta^r & (1-\beta^r) \end{bmatrix}^{(t_{i_k,x_k}(\hat{D}_k))}$$

The transition probability function to state S_{k+1} from the state S_k given travel-time-dependent transition matrix $\Theta^r(t_{i_k,x_k}(\hat{D}_k))$ is:

$$P(S_{k+1}|S_k) = \prod_{r=1}^{R} \Theta^r_{\hat{D}_k^r, \hat{D}_{k+1}^r}(t_{i_k,x_k}(\hat{D}_k)). \tag{3.12}$$

Note that for any vulnerable link r between the nodes i_{k+1} and x_{k+1} when $t_{i_{k+1},x_{k+1}}(\hat{D}_{k+1})$ is multiplied by the probability transition function $P(S_{k+1}|S_k)$, we

obtain the travel-time-dependent distribution of travel time on the link r. In this way, we obtain the travel-time-dependent probability distribution of the vulnerable links for all possible states. When a policy uses the probability distribution with a travel-time-dependent transition probability, it will be denoted as (PD, T) in the remainder of this chapter. Thus, the Bellman optimality equations for Opt becomes the same as Equation (3.6), and it considers a probability distribution with travel-time-dependent transition probability as in Equation (3.12). Then, we find the optimal solution by using Algorithm 1.

3.4.2 Offline Routing Policies

We consider two offline routing policies: a naive policy and a robust policy. By comparing the naive and the robust routing policies with the dynamic routing policies, we evaluate the value of taking into account the link disruption information and determining the routing policies dynamically.

Naive Routing Policy ($Naive$)

The naive routing policy ($Naive$) determines the optimal path minimizing the deterministic travel times (minimum value) ignoring any disruptions. This policy always follows the deterministic optimal route even if there is a disruption on the path. In determining the route, we only consider the non-disruption state for all vulnerable links. The transition probabilities for all vulnerable links have $\alpha_r = 0$ and $\beta_r = 1$ and the no-disruption state becomes an absorbing state. The state at stage k becomes $S_k = (i_k, \hat{D}_k = (0, 0, \ldots, 0))$. The travel time for each link (i_k, x_k) is $t_{i_k,x_k}(\hat{D}_k = \vec{0})$. The transition matrix between the stage k and $k+1$ is:

$$\Phi^r_{k,k+1} := \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

.

Bellman Equations (3.6) is modified into:

$$V_k(i_k, \hat{D}_k) = \min_{x_k \in \mathcal{X}_k} \tilde{c}(S_k, x_k) + V_{k+1}(S_{k+1}), \tag{3.13}$$

$$V_K(S_K) = 0. \tag{3.14}$$

The route is found by using Algorithm 1 and modifying Equation (3.8) to:

$$V_t(S) = \min_{x_k \in \mathcal{X}_k} \tilde{c}(S, x_k) + V_{t-1}(S). \tag{3.15}$$

Robust Routing Policy ($Robust$)

The robust policy ($Robust$) determines the optimal path assuming that all disruptions will occur certainly. This policy is risk averse to any disruption. In this policy, the

total travel time is minimized but using the maximum link travel times (see also Wald's minimax criterion (Wald 1945)). For the robust policy, there is only one state vector that has a disruption state for all vulnerable links. The transition probabilities for all vulnerable links have $\alpha_n = 1$ and $\beta_n = 0$ and the state vector where all links have disruption becomes the absorbing state. The state at stage k becomes $S_k = (i_k, \hat{D}_k = (1, 1, \ldots, 1))$. The travel time for each link (i_k, x_k) is $t_{i_k,x_k}(\hat{D}_k = \vec{1})$ and for each link the transition matrix between the stage k and $k+1$ is:

$$\Phi^r_{k,k+1} := \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

.

Bellman Equations (3.6) is modified into:

$$V_k(i_k, \hat{D}_k) = \min_{x_k \in \mathcal{X}_k} \tilde{c}(S_k, x_k) + V_{k+1}(S_{k+1}), \tag{3.16}$$

$$V_K(S_K) = 0. \tag{3.17}$$

The route is found by using Algorithm 1 and modifying Equation (3.8) to:

$$V_t(S) = \min_{x_k \in \mathcal{X}_k} \tilde{c}(S, x_k) + V_{t-1}(S). \tag{3.18}$$

3.4.3 Online Routing Policies ($Online(n)$)

Various online policies are used for dynamic shortest path problems. The policies depend on the amount of real-time (online) information used. At each node, we have the online information about the disruption realizations of the vulnerable links that are maximum n links away from the current node i_k. For the links not in the zone of n-stages, no online information is considered and therefore, the expected values based on steady-state probabilities are used.

The vector r_{i_k} consists of the vulnerable links that are maximum n links away from the current node i_k. The cardinality of vector r_{i_k} is denoted by R_{i_k}. The state space of the online model is different for each node as the neighborhood of each node is different. The state space of the hybrid policy for any node i_k is modified as $S_k = (i_k, \hat{D}^{i_k}_k)$ where $\hat{D}^{i_k}_k = (\hat{D}^{1_{i_k}}, .., \hat{D}^{R_{i_k}})$ with $R_{i_k} \leq R$. Here, $\hat{D}^{r_{i_k}}$ represents the actual value of the state of r^{th} vulnerable link of the node i_k which is at most n-stages away from the node and where $r \in r_{i_k}$. When a policy uses the expected values for a group of vulnerable links, it will be denoted as EV. When a policy uses actual values, it will be denoted as AV.

The steady-state probability to be in state $\hat{D}^r$ for any link r is denoted by $P(\hat{D}^r)$. Please note that $\hat{D}^r \in U_r$. The expected value of the link r for traveling from node i_k to node x_k is denoted by $\bar{t}_{i_k,x_k}$:

$$\bar{t}_{i_k,x_k} = \sum_{\hat{D}^r} P(\hat{D}^r) t_{i_k,x_k}(\hat{D}^r). \tag{3.19}$$

At each node i_k, online information of the n-stages from the node plus the expected values of the links until the destination node that are located farther from the n-stages within the node are considered to determine the next node to travel to. The online route can be found using Algorithm 1 and modifying Bellman Equations (3.6) into:

$$V_k(i_k, \hat{D}_k^{i_k}) = \begin{cases} \min\limits_{x_k \in \mathcal{X}_k} & \tilde{c}(i_k, \hat{D}_k^{i_k}, x_k) + V_{k+1}(x_k, \hat{D}_k^{i_k}) & \text{if } S_k \in \mathbb{S} \setminus \mathbb{G} \text{ and link } i_k\text{-}x_k \text{ is in } n\text{-stages} \\ \min\limits_{x_k \in \mathcal{X}_k} & \bar{t}_{i_k,x_k} + V_{k+1}(x_k, \hat{D}_k^{i_k}) & \text{if } S_k \in \mathbb{S} \setminus \mathbb{G} \text{ and link } i_k\text{-}x_k \text{ is NOT in } n\text{-stages} \\ 0 & & \text{if } S_k \in \mathbb{G}\,. \end{cases} \tag{3.20}$$

The routing decision is found by using Algorithm 1 after retrieving each state information. This means that the algorithm is solved each time that a disruption information is retrieved at each node until the termination. This is because the cost function of successor states is dependent on the current disruption information. Equation (3.8) is modified into:

$$V_t(i_k, D^{i_k}) = \begin{cases} \min\limits_{x_k \in \mathcal{X}} & \tilde{c}(i_k, D^{i_k}, x_k) + V_{t-1}(x_k, D^{i_k}) & \text{if link } i_k\text{-}x_k \text{ is in } n\text{-stages} \\ \min\limits_{x_k \in \mathcal{X}} & \bar{t}_{i_k,x_k} + V_{t-1}(x_k, D^{i_k}) & \text{otherwise.} \end{cases}$$

Note that, when $n = N$, the online routing policy uses the online information for the complete network which considers realized disruption information for all links. When $n = 0$, no realized disruption information is used, but only the expected values.

3.4.4 Hybrid Policies ($DP(n, H)$)

Offline policies lack responsiveness to the dynamic nature of disruptions. For instance, when disruptions occur, the naive approach ignoring all disruptions gives higher routing costs. The robust routing policy also gives higher costs as disruptions do not always occur. Online routing policies depend highly on the availability of the needed online information and do not consider transition probabilities between different disruption states. In this section, we develop policies combining the different strong elements from offline and online routing policies. We refer to these policies as $DP(n, H)$, where n denotes the number of stages with online information (with $n \in \{0, 1, 2, \ldots, N\}$). In the hybrid policy, we have online information for n-stages from the node. Therefore, the state space of the hybrid policy for any node i_k at any stage k is modified as $S_k = (i_k, \hat{D}_k^{i_k})$ where $\hat{D}_k^{i_k} = (\hat{D}^{1_{i_k}}, \hat{D}^{2_{i_k}}, .., \hat{D}^{R_{i_k}})$ with $R_{i_k} \leq R$ which is similar to the $Online(n)$ policy.

Given the state is S_k, a transition is made to the next state at stage $k+1$, $S_{k+1} = (x_k, \hat{D}_{k+1}^{x_k})$. For hybrid policies, we use the travel-time-dependent state transition matrix for the vulnerable links for which we have online information. The probability transition function to the state S_{k+1} from the state S_k is the same as the one given in Equation (3.12) except we only consider the limited part of the transition matrix where only the vulnerable links that are maximum n-stages away from the node are included.

For the links beyond the n-stages of the current node (links without any online information), we compute the value of being in state, $S_{k+1} = (x_k, \hat{D}^{x_k}_{k+1})$ until the destination with time-invariant (memoryless) transition probabilities. In other words, for these links, we use probability distributions with time-invariant probabilities. The unit-time transition probabilities of r^{th} link are obtained from the matrix $\Phi^r_{t,t+1}$. Please note that r^{th} vulnerable link is an element of the vector r_{i_k}. For the formulation, we use time-invariant transition probabilities for the vulnerable links at each node i_k as (with two disruption levels):

$$\Theta^{r_{i_k}} = \begin{bmatrix} (1-\alpha^{r_{i_k}}) & \alpha^{r_{i_k}} \\ \beta^{r_{i_k}} & (1-\beta^{r_{i_k}}) \end{bmatrix}$$

.

Then, the transition probability to the state S_{k+1} considering time-invariant probabilities is:

$$P(S_{k+1}|S_k) = \prod_{r=1}^{R_{i_k}} \Theta^r_{\hat{D}^r_k, \hat{D}^r_{k+1}}. \tag{3.21}$$

Now that the new transition matrix has been defined, we can apply Algorithm 1 using this matrix. Note that for any vulnerable link r between the nodes i_k and x_k, when $t_{i_{k+1},x_{k+1}}(\hat{D}^r_{k+1})$ is multiplied by the time-invariant transition probability, i.e. $P(S_{k+1}|S_k)$, we obtain the distribution of traveling the link r. When a policy uses the probability distribution with memoryless transition probabilities for a group of vulnerable links, we denote this value as (PD, M) in the remainder of this chapter.

Note that, when n involves all stages until the destination node, the hybrid routing policy uses the travel-time-dependent transition probabilities for all vulnerable links at each node. So, it yields the same routing policy as the optimal policy. When $n = 0$, only memoryless transition probabilities are used at each node.

3.5. Computational Results and Analysis

By comparing the routing policies' performance for different networks, we identify which routing policies lead to better performance under what conditions. Table 3.1 summarizes the characteristics of the routing policies discussed in Section 3.4. Note that when a policy uses the expected values for a group of vulnerable links, it is denoted as EV. When it uses actual values (or online travel times), it is denoted as AV. When a policy uses a probability distribution with memoryless (transition) probabilities (travel-time-dependent (transition) probabilities) for the group of vulnerable links, we refer them as (PD, M) ((PD, T) respectively). For the naive routing policy, travel time is used ignoring disruptions for the vulnerable links. Likewise, the robust routing policy assumes that the disruptions always occur on the disruption prone links.

Table 3.1 The different routing policies

	Travel time information				
Policy	1^{st} Link	2^{nd} Link	3^{rd} Link	...	n^{th} Link
$Naive$	No disruption	No disruption	No disruption	No disruption	No disruption
$Robust$	disruption	disruption	disruption	disruption	disruption
$Online(N)$	AV	AV	AV	AV	AV
$Online(1)$	AV	EV	EV	EV	EV
$Online(2)$	AV	AV	EV	EV	EV
$Online(3)$	AV	AV	AV	EV	EV
$DP(1,H)$	AV	(PD,M)	(PD,M)	(PD,M)	(PD,M)
$DP(2,H)$	AV	(PD,T)	(PD,M)	(PD,M)	(PD,M)
$DP(3,H)$	AV	(PD,T)	(PD,T)	(PD,M)	(PD,M)
Opt	AV	(PD,T)	(PD,T)	(PD,T)	(PD,T)

In what follows, we first discuss our problem instances, then we give details for a small worked-out example and finally, we give aggregate results over all networks for each routing policy.

3.5.1 Problem Instances

We generate in total 12 different instance types using a number of properties to characterize and construct these instances. We define the instance properties such that they are relevant to the real-life highway networks, i.e. Dutch Road Network (Figure 1). The properties considered are as follows:

Network size

The network size consists of small and large networks with 16 and 36 nodes respectively. The network is designed such that the origin and destination nodes are situated in the top-left corner and bottom-right corner respectively. With this structure, we prevent evaluating unnecessary nodes far from the shortest path. Clearly, this does not limit the applicability of our results, but merely reduces the number of unnecessary calculations to be evaluated.

Furthermore, with this structure each node has at least two neighbor links such that when there is a disruption the routing policies choose among a number of alternative links to travel next. This is important for the dynamic routing decisions where making an alternative decision may have a significant impact on the total cost. This network structure is also relevant to the real networks. The number of nodes represents the junctions of various highways. In highways, for each junction there is more than one alternative route which is relevant to the random network structure.

Network vulnerability

We measure the vulnerability by the number of vulnerable links in the network. Vulnerable links are randomly assigned to the shortest paths that are found with an iterative process where every iteration a new shortest path is found and a new vulnerable links is added to the network until we reach the desired number of vulnerable links. We have instances with low percentage (50%) and high percentage (70%) of vulnerable arcs in the network which are denoted as low and high network vulnerability respectively.

Travel times

Travel time of each arc for non-disrupted state is randomly selected from a discrete uniform distribution $U(1, 10)$. For example, in this uniform distribution a unit can be 1, 3, 5 minutes. In this way, we provide variability in travel times due to the structural differences among roads. Furthermore, we keep the range between $1 - 10$ units because when we analyze the Dutch Road Network, the ratio of the distances between the highway segments is also similar.

The impact of disruption on the travel time is an another network parameter to decide for the numerical experiments. When there is a disruption on a link, the travel time increases at a certain rate. Therefore, we should identify the rate of increase for the experimental study for disrupted links. In this chapter, we focus on higher impact disruptions. So, the historical data on the high impact incidents at the Dutch Network shows that when there is a disruption, the travel time increases more than 3 times of the normal travel time (Snelder 2010). Furthermore, by considering a wider survey, Eliasson (2004) showed that the travel times with disruptions are valued 3 to 5 times higher than the travel time under normal conditions. We also performed preliminary tests to investigate the effect of using different ratio parameters on the performance of the optimal routing policy. The results show that when the impact gets higher (> 5 times the regular travel time), the total expected travel time becomes similar because the routing algorithms choose the risk aversive routes (Figure 2). Figure 2 shows that the average performances of the algorithms (except the Robust and the Naive policy) with respect to each other are similar regardless of the impact of the disruption. For instance, $Online(2)$ performs the same or worse than the optimal algorithm or $DP(2, H)$ for all the cases. Considering these, we set the increase rate in travel time when there is a disruption to 3. In this way, the disruption creates a significant difference in total expected travel time and also this is relevant for the real-life situations.

Disruption rate

We define low disruption rate by a low probability of having disruptions to be between $[0 - 0.3]$, medium disruption rate by a medium probability in the range $[0.4 - 0.6]$ and high disruption rate by a high probability between $[0.7 - 1]$. We chose the probability intervals such that each interval is not much scattered to represent the probability category effectively.

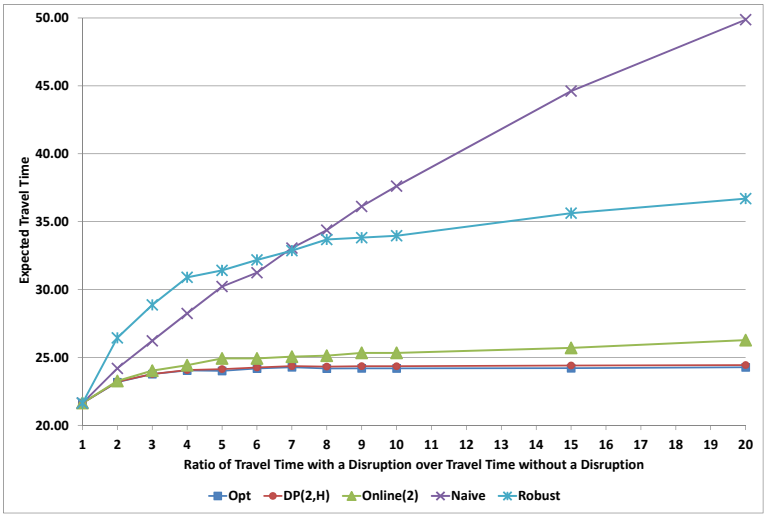


Figure 3.2 The effect of ratio of the travel time with a disruption over the travel time with no disruption on the expected travel time

Evaluation of the policies

For each instance type, we generate 100 replications (1200 instances) and report the average value. For each instance replication, the expected cost is found as follows: using Algorithm 1, we obtain an array of routing policies considering each state. Then, we compute the exact value function using these pre-determined policies with an exhaustive evaluation for all possible states. This value gives the expected cost of the specific routing policy considering all possible states. For each instance, the percentage cost difference relative to the optimal policy for each routing policy is calculated as follows:

$$\Delta(\%) = \frac{E[\text{Cost Alternative Policy}] - E[\text{Cost Optimal Policy}]}{E[\text{Cost Optimal Policy}]} \times 100 \qquad (3.22)$$

The algorithms presented in this chapter are programmed in Java programming language. The computational results in this section are obtained by using IntelCore Duo 2.8 Ghz Processor with 3.46 GB RAM.

3.5.2 An Illustrative Example

Consider a network consisting of 16 nodes and 4 vulnerable links. The vulnerable links are shown in bold with their travel times and their disruption probabilities in Figure 3.3, which also shows the routing decisions made by the different routing policies. At each node, the arcs show the next node to travel to given the state of the

network. From Figure 3.3, we observe that different routing policies lead to different decisions. The online and hybrid policies are more complicated than the naive and the robust policies in the sense that more information is taken into account. Clearly, this extra complexity comes with the advantage of reducing the costs dramatically. Table 3.2 gives the costs for each policy for this specific network and the percentage cost difference relative to the optimal policy. For this specific example, Table 3.2 shows that hybrid policies perform better than the online and the offline policies.

Table 3.2 Overview of costs for each routing policy and the percentage cost difference relative to the optimal policy for the example network

	Opt	$Naive$	$Robust$	$Online(1)$	$Online(2)$	$Online(3)$	$DP(1,H)$	$DP(2,H)$	$DP(3,H)$
Cost	27.13	51.79	35.25	28.25	28.27	28.51	27.65	27.13	27.13
Δ(%)	n.a.	90.93	29.96	4.16	4.20	5.11	1.93	0	0

Figure 3.3 demonstrates that online and hybrid policies choose different routes according to the number of stages in the network with online information. For instance, at node 2, $Online(1)$ chooses to go to node 3 even when the link $6 \rightarrow 7$ is not in disruption. This is because it takes into account the expected travel time on the link $6 \rightarrow 7$ which is higher. However, $Online(2)$, $Online(3)$ and $Online(N)$ choose to go to the node 6 when the link $6 \rightarrow 7$ is not in disruption, as these policies directly take into account the actual value of $6 \rightarrow 7$ in the node 2. But, upon arrival at the link $6 \rightarrow 7$, there is a non-zero probability that it goes into disruption. Hybrid policies, on the other hand, take into account this probability and consequently, choose not to go to the node 6.

The difference between the effect of different levels of stages on the routing decisions is seen at the node 5. When $Online(3)$ and $Online(N)$ observe the actual value of the link $10 \rightarrow 11$ as not in disruption and $6 \rightarrow 7$ is in disruption, they choose to travel to the node 9 next which decreases the total expected cost. However, $Online(1)$ and $Online(2)$ do not observe the actual value of the link $10 \rightarrow 11$. $DP(1,H)$ only sees the probability of the link $6 \rightarrow 7$ going into disruption and chooses to go to the node 9 for all states. $DP(2,H)$ and $DP(3,H)$, however, consider the travel-time-dependent probability distributions so they observe that probability to find $6 \rightarrow 7$ and $10 \rightarrow 11$ in disruption upon arrival is low when initially at node 5 they are not in disruption. So, they choose to go to either node 9 or node 6. In this specific example, $DP(2,H)$ and $DP(3,H)$ follow the same route as the optimal policy.

3.5.3 Computational Results for All Networks

Table 3.3 gives the average, minimum and maximum percentage cost difference (Δ(%)) for all policies relative to the optimal policy. We also provide the sign rankings for each routing policy using the Wilcoxon rank test by using the software IBM SPSS Statistics 20 (provided in Table 3.4). The Wilcoxon rank test compares two routing policies by showing the percentage number of cases a policy is better, worse or equal to another routing policy (Field 2009). In the Table 3.4, positive ($\%$) indicates the

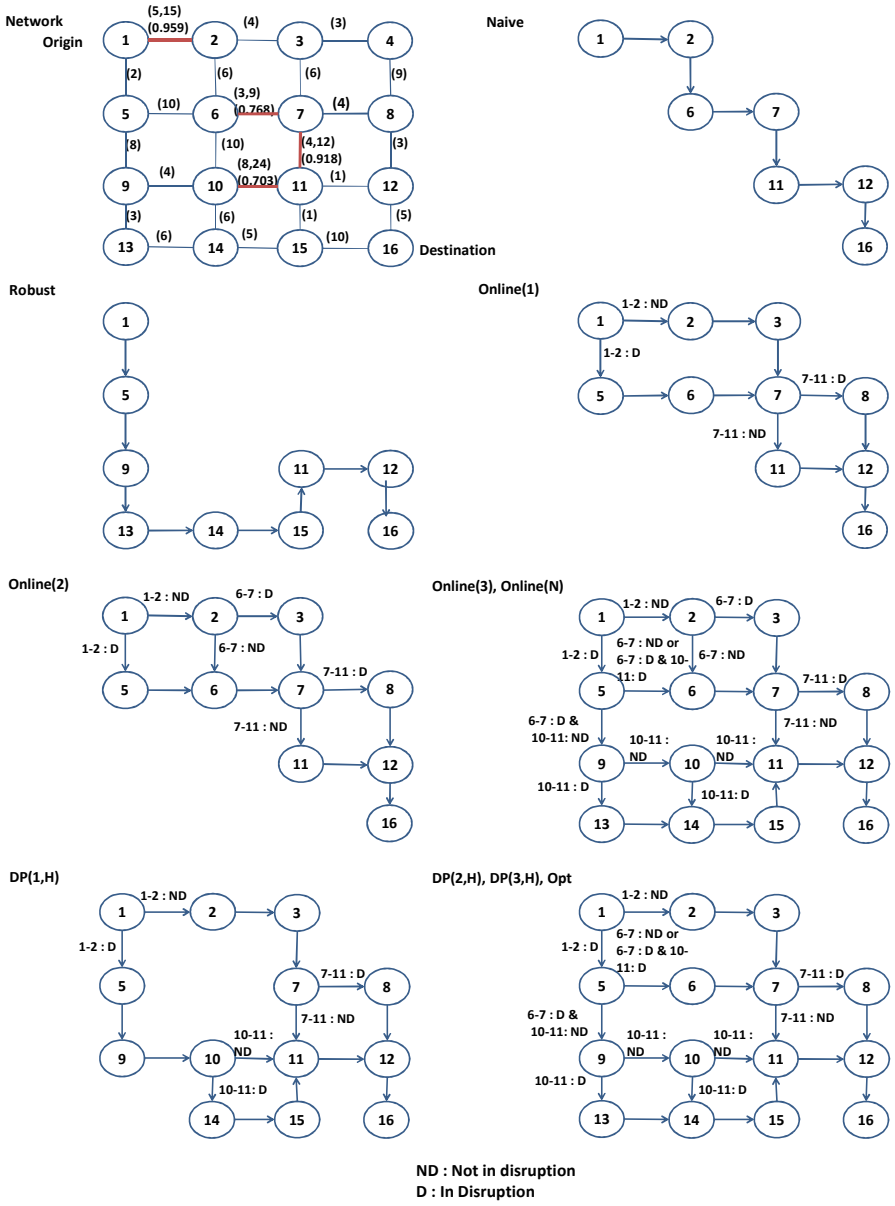


Figure 3.3 Routing decisions for each policy

percentage of how many cases the routing policy (at the row) gives higher total expected cost than the other routing policy (at the column). Similarly, negative (%) (ties (%)) indicates how many times a routing policy better (the same) than the other routing policy. As such, this gives additional insights into the performance of the different policies, next to the average cost differences.

Offline versus Online/Hybrid Routing policies

Looking at the offline policies (naive and robust), Table 3.3 demonstrates that these policies systematically perform worse than the online or the hybrid policies. The robust policy on average is better than the naive policy. The table also demonstrates that the maximum difference between the costs of the offline policies and the cost of the optimal policy is much higher than the one for the online and the hybrid policies. Table 3.4 shows that, for 78% of the evaluated network instances, the robust policy is better than the naive policy. Observe also that the robust policies become worse than the naive policy when the disruption rate is low for this specific set of instances (the robust policy chooses to travel the risk averse path).

Hybrid versus Online Routing policies

Across all network instances and network variants, the hybrid policies perform at least the same or better than the online policies (Table 3.3). On average, the differences between the online policies and the hybrid policies are around 1% to 3%, which is not that large. However, comparing the average performance difference is not revealing the real power of the hybrid policies versus the online policies. Analyzing the maximum and the minimum percentage cost differences relative to the optimal solution (Table 3.3), we see differences between the performances of hybrid versus online policies. This performance difference is driven by the proximity of the locations of the vulnerable links (as in Figure 3.3). When the vulnerable links are located close to each other, online policies perform worse than the hybrid policies with the same number of stages. In real-life, congested links also occur close to each other, as when one link becomes blocked, the connected links are more likely to also become blocked. Table 3.3 also shows that as the disruption rate increases, the difference between the maximum percentage difference of hybrid policies relative to the optimal policy goes up to 11% better than the one of the online policies.

When we compare the hybrid policies with the optimal policy, we see that the hybrid policies perform only marginally worse than the optimal policy in this specific set of instances. However, hybrid policies have significant calculation time savings compared to the optimal policy. For example, the optimal policy finds the optimal solution in average 201.9 seconds whereas as Table 3.3 demonstrates, hybrid policies find the solutions in at most 0.5 seconds. The ranking results point out that the optimal policy is at most in 66.67% of all cases better that the hybrid policies whereas the online policies are at least in 78.58% of all the cases worse than the optimal policy in this specific set of instances (Table 3.4). Furthermore, the relative range of the

Table 3.3 Minimum, maximum and average values of the percentage cost difference relative to the optimal policy and overall average computation times

Disruption Rate		$Naive$	$Robust$	$Online(1)$	$Online(2)$	$Online(3)$	$DP(1,H)$	$DP(2,H)$	$DP(3,H)$
Low Rate	Min $\Delta(\%)$	12.88	3.55	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	116.03	113.24	54.84	54.84	54.84	50.08	50.08	50.08
	Average $\Delta(\%)$	28.64	19.92	2.55	2.07	2.01	1.35	1.08	1.06
Med Rate	Min $\Delta(\%)$	9.87	0.79	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	97.02	153.51	15.41	14.23	14.17	6.56	3.60	3.60
	Average $\Delta(\%)$	27.37	13.92	1.89	1.37	1.29	0.68	0.17	0.17
High Rate	Min $\Delta(\%)$	3.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	132.99	137.67	34.51	34.63	34.68	26.32	26.23	26.23
	Average $\Delta(\%)$	32.76	10.00	1.84	1.87	1.92	1.63	1.37	1.14
Size		$Naive$	$Robust$	$Online(1)$	$Online(2)$	$Online(3)$	$DP(1,H)$	$DP(2,H)$	$DP(3,H)$
Small	Min $\Delta(\%)$	12.88	3.55	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	132.99	73.31	24.02	24.02	24.28	20.15	20.15	20.15
	Average $\Delta(\%)$	32.09	15.56	1.44	1.19	1.19	0.60	0.25	0.24
Large	Min $\Delta(\%)$	10.64	2.73	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	116.03	153.51	54.84	54.84	54.84	50.08	50.08	50.08
	Average $\Delta(\%)$	28.09	13.54	2.50	2.15	2.10	1.65	1.29	1.16
Vulnerability		$Naive$	$Robust$	$Online(1)$	$Online(2)$	$Online(3)$	$DP(1,H)$	$DP(2,H)$	$DP(3,H)$
Low	Min $\Delta(\%)$	6.80	1.52	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	96.57	153.51	14.21	13.76	11.26	10.07	10.07	10.07
	Average $\Delta(\%)$	23.65	12.43	1.11	0.91	0.86	0.46	0.18	0.17
High	Min $\Delta(\%)$	12.88	3.55	0.00	0.00	0.00	0.00	0.00	0.00
	Max $\Delta(\%)$	132.99	106.20	54.84	54.84	54.84	50.08	50.08	50.08
	Average $\Delta(\%)$	35.43	16.17	2.99	2.58	2.56	1.96	1.54	1.38
CPU (Sec)		$Naive$	$Robust$	$Online(1)$	$Online(2)$	$Online(3)$	$DP(1,H)$	$DP(2,H)$	$DP(3,H)$
		0.001	0.001	0.005	0.007	0.009	0.027	0.105	0.498

Table 3.4 The Wilcoxon rank test for the different routing policies

		$Robust$	$Online(1)$	$Online(2)$	$Online(3)$	$DP(1,H)$	$DP(2,H)$	$DP(3,H)$	Opt
$Naive$	Positive(%)	78.42	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Negative (%)	20.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Ties (%)	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$Robust$	Positive (%)		90.83	92.83	93.83	94.50	96.42	96.50	98.58
	Negative (%)		1.84	4.92	5.08	0.00	0.00	0.00	0.00
	Ties (%)		7.33	2.25	1.09	5.50	3.58	3.50	1.42
$Online(1)$	Positive (%)			35.08	43.00	47.33	76.58	76.83	78.58
	Negative (%)			31.25	37.08	0.00	0.00	0.00	0.00
	Ties (%)			33.67	19.92	52.67	23.42	23.17	21.42
$Online(2)$	Positive (%)				25.00	54.42	77.42	77.67	78.67
	Negative (%)				24.33	16.50	0.00	0.00	0.00
	Ties (%)				50.67	29.08	22.58	22.33	21.33
$Online(3)$	Positive(%)					54.94	77.67	78.83	79.92
	Negative (%)					16.58	1.08	0.00	0.00
	Ties (%)					28.48	21.25	21.17	20.08
$DP(1,H)$	Positive (%)						57.50	58.33	66.67
	Negative (%)						0.00	0.00	0.00
	Ties (%)						42.50	41.67	33.33
$DP(2,H)$	Positive (%)							1.83	62.92
	Negative (%)							0.00	0.00
	Ties (%)							98.17	37.08
$DP(3,H)$	Positive (%)								62.08
	Negative (%)								0.00
	Ties (%)								37.92

online policy relative to the optimal policy becomes higher than the range of the hybrid policy at high disruption rate network structures (Table 3.3).

In the numerical results, there are exceptional cases where the online and the hybrid policies perform up to 54.84% and 50.08% worse than the optimal policy respectively. For the extreme case, the expected travel time from the solution of the optimal algorithm is 31.45 while the expected travel time from the solution of $DP(3,H)$ is 47.19. In these exceptional cases, the network size is large with high network vulnerability and low disruption rate ([0- 0.3]). The first vulnerable links are located at least 4 links away from the first node. It is observed that the optimal policy outperforms the other policies by choosing different routes from the beginning of the network. At the initial node, the alternative nodes are less so the horizon of the travel-time-dependent disruption information is really crucial. When a vulnerable link is in disruption, it takes a certain time period to return to its steady state probability. If this time period for a vulnerable link is longer than the horizon for which the hybrid policies consider the travel-time-dependent probabilities, then the vulnerable link will not be in its steady state probability when we arrive there. This shows that we need to include more stages with the travel-time-dependent disruption information in our dynamic routing decision. Appendix A shows how to compute the time needed to reach the steady state probability of being in disruption. In the extreme cases, it is shown that we need at least 47 time units to find the vulnerable link in steady state meaning that we need to use travel-time-dependent probability distributions for at least 5 links. If the suggested information is used, the percentage gap relative to the optimal solution decreases from 50.08% to 11.01%.

The Number of Stages with Online Information

An important observation is that as higher number of stages with detailed information is included, the performance of hybrid policies converges on average to the performance of the optimal solution. This is a nice result for practice because it indicates that we can save calculation time and information retrieval costs. When we compare lower level hybrid policies with higher level online policies, i.e. $Online(3)$ versus $DP(1,H)$; $Online(3)$ versus $DP(2,H)$, the percentage that the higher level online policies are better decreases as the hybrid policies get more information, i.e. 17.58% and 10.92% respectively as shown in Table 3.4.

When we compare $DP(1,H)$ versus $DP(2,H)$; $DP(2,H)$ versus $DP(3,H)$, the value of using travel-time-dependent disruption information for more stages is observed. Table 3.4 shows that $DP(2,H)$ performs better than $DP(1,H)$ for 57.50% of the cases while $DP(3,H)$ performs better than $DP(2,H)$ for only 1.83% of the cases. This shows that value of of using travel-time-dependent disruption information for more stages increases at a decreasing rate such that $DP(2,H)$ and $DP(3,H)$ perform the same for most of the cases.

3.6. Conclusions

In this chapter, we analyzed different routing policies dealing with network disruptions. To build and evaluate various routing policies, we developed a modeling framework based on dynamic programming. We analyzed some well-known offline and online policies. In addition to this, we developed hybrid policies that use different levels of online and offline information. We formulated the hybrid routing policies in terms of dynamic programming models. We compared the performance of the current known heuristics, such as the offline and the online policies, with the hybrid routing policies. By testing different policies using different levels of stages, we analyzed the effect of having a sufficient level of online and offline information about the network for different networks. Moreover, we gained insights into the network conditions in which it is better to use what type of policies.

We found that the usage of only limited online information leads to a significant decrease in computation time compared to the policies that use higher levels of information. We observed that the performance of hybrid policies only deteriorates marginally compared to the optimal solution. We showed that a large part of the cost reduction can already be obtained by considering just a part of the network in detail. This result is beneficial when availability and reliability of online information is low and the cost of retrieving this information is high. This can be an implementation direction for the current navigation systems, i.e. usage of online and historical data to develop probabilistic information on travel time. Furthermore, we observed that under more complex networks, the more detailed information about the disruption's state becomes effective and cost efficient even if the network knowledge is limited with few stages with online information for the hybrid policies. This chapter has practical relevance as the results show that using a moderate level of network knowledge is generally sufficient to have acceptable performance compared to the optimal policy while computation time and information retrieval cost decrease significantly. The drivers can benefit from this finding in reducing the computation and information retrieval costs. Furthermore, the developed hybrid policies can be implemented easily by using the navigation technologies. In practice, the disruption scenarios can be determined by using threshold speed levels. If the speed is under the threshold level, the link is blocked otherwise not. The transition probabilities can be dealt by using these threshold levels.

Future research includes the analysis of the policies with many disruption scenarios, considering more than two disruption scenarios and considering link dependencies. As discussed above, the computation effort increases exponentially with the number of disruption scenarios. In order to handle this complexity, space reduction techniques and approximations could be used.

3.A. Appendix

A A note on the time needed to find a vulnerable link in steady state

Consider a a two state Markov Chain with a unit-time transition probability matrix for a vulnerable link r:

$$\Theta^r = \begin{bmatrix} (1-\alpha^r) & \alpha^r \\ \beta^r & (1-\beta^r) \end{bmatrix}$$

For this matrix, the eigen values are $(1-\alpha^r-\beta^r)$ and 1. So, it is shown that:

$$\Theta^r(t) = U \begin{bmatrix} 1 & 0 \\ 0 & (1-\alpha^r-\beta^r)^t \end{bmatrix} U^{-1}$$

We are interested in the time that a disrupted link will be found in its steady state when we arrive there. So, we need the transition probability to a disrupted state at time t given that it is disrupted at time $t=0$. This can be written as $p_{22}^t = a+b((1-\alpha^r-\beta^r)^t)$ where $p_{22}^0 = a+b = 1$ and $p_{22}^1 = 1-\beta^r$. So, $1-\beta^r = a+b(1-\alpha^r-\beta^r)$ and $a = \frac{\alpha^r}{\alpha^r+\beta^r}$ and $b = \frac{\beta^r}{\alpha^r+\beta^r}$. So, p_{22}^t is:

$$P(w_t^r = 2 | w_1^r = 2) = p_{22}^t = \frac{\alpha^r}{\alpha^r+\beta^r} + \frac{\beta^r}{\alpha^r+\beta^r}(1-\alpha^r-\beta^r)^t \tag{A1}$$

Note that this converges exponentially to $\frac{\alpha^r}{\alpha^r+\beta^r}$ which is the steady state probability of being in disruption. So, $\frac{\beta^r}{\alpha^r+\beta^r}(1-\alpha^r-\beta^r)^t$ converges to zero when $t \to \infty$. Consider that the expression approaches to a very small number ϵ:

$$\frac{\beta^r}{\alpha^r+\beta^r}(1-\alpha^r-\beta^r)^t = \epsilon \tag{A2}$$

If we solve the Equation (A2) for t:

$$t = log_{(1-\alpha^r-\beta^r)}(\frac{\epsilon(\alpha^r+\beta^r)}{\beta}) \tag{A3}$$

Equation (A3) indicates that what determines this time period is the probability to go into a disruption and the probability to recover from the disruption. Figure 3.4 shows the effect of the steady state probability of being in disruption on the time needed to converge to the steady state. It is shown that as the probability to go into a disruption decreases, it takes longer to be in steady state probability.

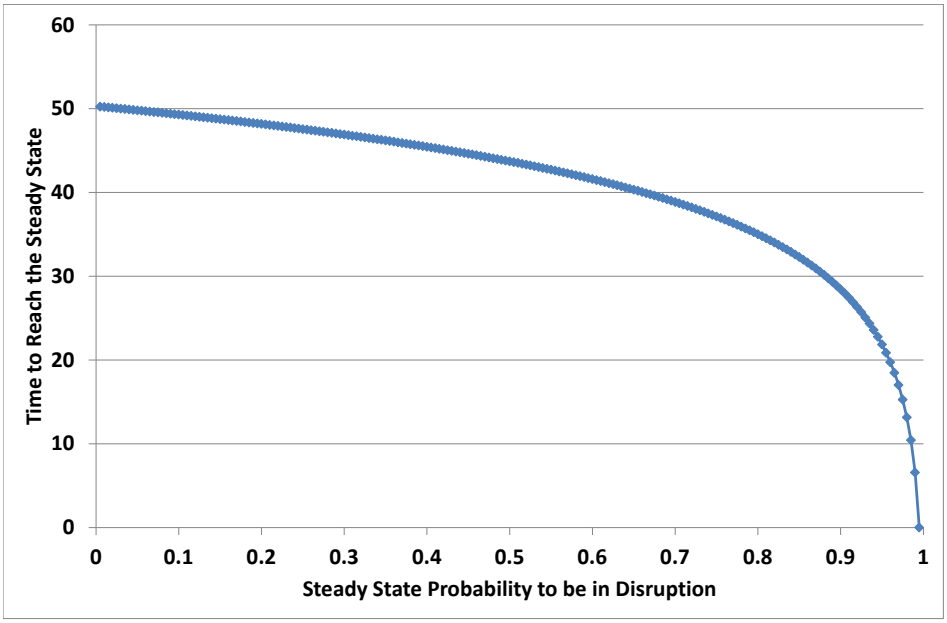


Figure 3.4 The time to reach the steady state probability versus the steady state probability to be in disruption

B Proof: Termination of value iteration algorithm

Proposition 3.3 For any $\epsilon > 0$ there exists a finite number of iterations $\mathbb{N}$ such that $\|V_t - V^*\| < \epsilon$ for all $t > \mathbb{N}$.

Proof: If the value iteration algorithm in Algorithm 1 terminates after a finite number of iterations and toward a fixed point, then $\Delta V_t(S)$ should be decreasing in t.

Let $TV(S)$ be the dynamic programming operator that maps value functions into value functions:

$$TV(S) = \min_{x \in N, x \in Neighbor(i)} \tilde{c}(i, \hat{D}, x) + \sum_{S'} P(S'|S)V(S')). \tag{B1}$$

We define that a value function estimate converges if it reaches a fixed point (ΔV_{t-1} is really small and converges to zero). Then:
$V_t = TV_{t-1} = V_{t-1} + \Delta V_{t-1}$

We prove the convergence of any dynamic programming operator that is a contraction mapping and it satisfies the equations below:

$$\|Tv - Tu\| \leq \gamma \|v - u\| \tag{B2}$$

where $\|.\|$ is the max norm, $\gamma = 1$ as in the stochastic shortest path problem and the termination is guaranteed with the goal state. For conciseness, we modify the value function into vector and matrix notation ($d \in D$ is the Markovian decision rule mapping each state into an action (Puterman 2009)):

$$V_t = \min_{d \in D} \tilde{c}_d + P_d V_{t-1}. \tag{B3}$$

Now, we show that value iteration is a contraction mapping for any state (the max-norm of a probability matrix (P_d) is one, and the max-norm of scalar immediate cost (C_d) is again a scalar):

$$\begin{aligned}
\|TV_t - TV_{t+1}\| &\leq \|V_t - V_{t+1}\| && \text{(B4)}\\
\|V_{t+1} - V_{t+2}\| &\leq \|V_t - V_{t+1}\| && \text{(B5)}\\
\|\tilde{c}_d + P_d V_t - \tilde{c}_d - P_d V_{(t+1)}\| &\leq \|V_t - V_{t+1}\| && \text{(B6)}\\
\|P_d\|\|V_t - V_{(t+1)}\| &\leq \|V_t - V_{t+1}\| && \text{(B7)}\\
\|V_t - V_{(t+1)}\| &\leq \|V_t - V_{t+1}\| && \text{(B8)}
\end{aligned}$$

This indicates that $\|V_{t+1} - V_{t+2}\| \leq \|V_t - V_{t+1}\|$ is monotonically decreasing. These equations show that as the dynamic programming operator is applied with certain amount of iterations, then the largest difference between two value function shrinks. This means that the contraction mapping tends towards a fixed point:
$V_t = V_{t-1} + \Delta V_{t-1} = TV_{t-1} \Rightarrow V_t = V^*$ where V^* is the optimal value.

Next we show that by considering relatively small $\epsilon > 0$, the value iteration converges to the optimal value, V^* in a finite number of iterations. We refer the reader to Bonet (2007) for the formal proof of number of exact iterations for convergence.

$$\begin{aligned}
\|V^* - V_t\| &= \|V^* - TV_{t-1}\| && \text{(B9)}\\
&= \|V^* - V_{t-1} - \Delta V_{t-1}\| && \text{(B10)}\\
&= \|V^* - V_{t-1}\| - \|\Delta V_{t-1}\| && \text{(B11)}\\
& && \text{(B12)}
\end{aligned}$$

This indicates that $\|V^* - V_t\| \leq \|V^* - V_{t-1}\|$, which is the gap with respect to the optimal, is monotonically decreasing. □

Chapter 4

DSPP: Hybrid Approximate Dynamic Programming Algorithms with a Clustering Approach

In Chapter 3, we investigated the efficient levels of network information for higher quality and computationally efficient dynamic routing policies. However, we focused on only two disruption levels: in disruption or not in disruption. As it is described in Chapter 2, in traffic there are multiple types of disruption levels leading to different impacts on travel time. In this chapter, we analyze dynamic shortest path problem for large-scale networks with multiple disruption levels. In this problem, as the state- and outcome-space increases exponentially with disruption levels, we develop efficient approximate dynamic programming algorithms (ADP) to reduce computational time and improve solution quality.

4.1. Introduction

In traffic networks, disruptions due to accidents and traffic bottlenecks cause traffic congestions that lead to a drastic increase in travel times and decrease the probability of being on-time at the destination. The travel time in traffic networks depends on the disruption levels (types). Furthermore, as the uncertainty in traffic networks increases due to recurrent and non-recurrent incidents, the planners need to take into

account many disruption levels. Having real-time traffic information from intelligent navigation systems and taking into account the stochastic nature of disruptions for the dynamic routing decisions can significantly reduce delays. As a trade-off, for networks with many disruption levels, computing dynamic routing decisions considering detailed information takes very long. Note that in real-life applications (as in navigation devices), low computation times enabling fast and high quality routing decisions are very important. Therefore, we need to develop fast and efficient techniques to make the dynamic routing decisions considering stochastic disruptions.

In this paper, we consider dynamic shortest path problems with stochastic disruptions on a subset of arcs. We develop dynamic routing policies by using both historical information and real-time information of the network. We observe the disruption status of the network when we arrive at each node. The disruption status of the following stage is dependent on the travel time of the current arc. We denote this property as travel-time-dependency. We model the problem as a discrete-time, finite Markov decision process (MDP). In our model, the dynamic routing policies are found based on the state of the system which can be retrieved with real time information.

MDP formulation provides a practical framework to find dynamic routing decisions at each decision epoch. However, for large scale networks with many disruption levels, obtaining the optimal solution faces the curses of dimensionality, i.e., states, outcomes, and decisions (Powell 2007, 2011). Therefore, in the dynamic shortest path problem literature with a MDP formulation, either approximation algorithms are developed or the structure of the optimal solution is investigated to deal with the curses of dimensionality .

The approximations or reduction techniques in the dynamic shortest path literature were shown to deal with binary levels of disruption, i.e., congested or not congested. The current empirical studies on the features of traffic congestion show that highways can have more than two levels of disruption which are specified according to the speed level and possible spillbacks (Helbing et al. 2009, Rehborn et al. 2011). However, increase in disruption levels leads to an exponential state- and outcome-space growth causing the well-known curses of dimensionality. Therefore, we need efficient approximation techniques to deal with many disruption levels.

In this paper, we focus on networks with many disruption levels which are similar to the real-life situations. We use an Approximate Dynamic Programming (ADP) algorithm which is a well-known approximation approach to effectively deal with the curses of dimensionality. In the literature, ADP is shown to be a powerful method to solve large-scale stochastic problems (e.g., Powell et al. 2002, Powell and Van Roy 2004, Simão et al. 2009).

The ADP algorithms in this paper are based on value function approximations with a lookup table representation. First, we employ a standard value function approximation algorithm with various algorithmic design variations for updating state values with efficient exploration/exploitation strategies. Then, we improve the

standard value function approximation algorithm by developing a hybrid ADP with a clustering approach. In this hybrid algorithm, we combine a deterministic lookahead policy with a value function approximation (Powell et al. 2012). We exploit the structure of disruption transition function to apply the deterministic lookahead policy. We form a cluster at each stage consisting of the nodes that are within the two-arc-ahead neighborhood of the current node. The decision is chosen from the cluster depending on an exploration/exploitation strategy. Then, we apply value function approximation until the destination.

Hybrid ADP algorithms with a lookahead policy and value function approximation are suggested by Powell et al. (2012). Similar approaches can be found in the literature on the hierarchal solution for large MDPs and using factored MDPs. In the factored representation of MDP, the states and/or decisions are clustered according to their specific properties. The value functions are then computed by using these compact representations (Boutilier et al. 2000, Givan et al. 2003, Kim and Dean 2002, Barry et al. 2011).

In this paper, instead of clustering the states, we expand the action set from the neighboring nodes into the set of nodes in a cluster. By a learning process, the hybrid algorithm clusters the nodes such that the approximate value until the destination is lower given the current disruption status. The state variables considered in the value function approximation are then visited and updated more frequently such that their state values become more accurate. This leads to higher quality state values and reduces the computational cost significantly by eliminating the steps for updating and exploring the states that are already included in the lookahead policy. To our knowledge, the hybrid ADP algorithm with the clustering approach has not been investigated so far for the dynamic shortest path problems.

The main contributions of this paper are as follows:

1. We construct a hybrid ADP with a clustering approach considering a lookahead policy and a value function approximation to solve for traffic networks with many disruption levels. The results show that the hybrid ADP algorithm with a clustering approach reduces the computational time significantly. Furthermore, the quality of the solution is higher compared to the standard ADP algorithm. We also provide a test bed consisting of random networks to compare the performance of the ADP algorithms with a stochastic lookahead policy shown to perform well in binary disruption levels with significantly lower computational times.

2. We provide various algorithmic design variations for the ADP where multiple initial solutions and both single and double pass algorithms are used with efficient exploration/exploitation strategies. We provide insights about the performance of these variations based on different network structures.

4.2. The Model Formulation

We model the dynamic shortest path problem with stochastic disruptions as a discrete-time, finite Markov decision process (MDP). Consider a traffic network represented by the graph $G(N, A, A_v)$ where N represents the set of nodes (or intersections), A the set of directed arcs and A_v the set of vulnerable arcs, potentially in disruption ($A_v \subseteq A$). The number of vulnerable arcs is R: $R = |A_v|$. Each of these vulnerable arcs has a known unit-time transition probability matrix for each disruption level. Each vulnerable arc, $r \in A_v$, can take any value from the disruption level vector, U^r, whose dimension depends on the specific vulnerable arc.

The travel time on an arc is assumed to be predictable from historical data and follows a discrete distribution, given the disruption level of the arc. We receive the real-time information about the disruption statuses of all vulnerable arcs when we arrive at a node. At each node, we make a decision on the next node to visit. Our objective is to travel from the origin node to the destination node with the minimum expected travel time. We derive the optimal routing decision using the discrete-time, finite MDP formulation. Stage k represents the number of nodes that have been visited so far from the origin node. The end of horizon is reached by arriving at the destination. The end of horizon is represented by K where K is a random variable depending on the network states and is not known before the departure.

States

The state of the system at stage k, S_k, is represented by two components:

- The current node, $i_k \in N$.
- The disruption status vector, $\hat{D}_k$, gives the disruption statuses of all vulnerable arcs. Each vulnerable arc, r, can take any value from the disruption level vector U^r. For each vulnerable arc, there can be M_r different types of disruption levels: $\hat{D}_k(r) \in U^r$: $U^r = \{u^1, u^2, ...u^{M_r}\}, \forall r \in A_v$. Note that at each stage, we use a realization of the disruption vector, $\hat{D}_k$.

The state of the system at stage k is then: $S_k = (i_k, \hat{D}_k)$. We set the initial state as: $S_0 = (\text{origin node}, \hat{D}_0)$ and the final goal state as: $S_K = (\text{destination node}, \hat{D}_K)$, where $\hat{D}_0$ and $\hat{D}_K$ are the realizations of the disruptions for all vulnerable arcs at the initial and final stage, respectively. Note that the goal state is absorbing and cost free.

Actions

The action, x_k, is the next node to visit given the state S_k. We note that each action, x_k, is an element of the set of all possible actions $\mathcal{X}(S_k)$ which is the neighbor set of the current node:

$$x_k = i_{k+1}. \tag{4.1}$$

Exogenous Information

The disruption statuses of the vulnerable arcs may change as we proceed to the next stage. The exogenous information consists of the realization of the disruption statuses of all the vulnerable arcs. Let $\hat{D}_{k+1}$ denote the disruption status realization that becomes known between stages k and $k+1$.

$$W_{k+1} = \hat{D}_{k+1}. \tag{4.2}$$

Cost Function

The cost is calculated as the travel time from the current node, i_k, to the next node, $x_k = i_{k+1}$, given the realized disruption status, $\hat{D}_k$:

$$C(S_k, x_k) \quad = \quad t_{i_k,x_k}(\hat{D}_k). \tag{4.3}$$

State Transition Function

At stage k, the system is in state S_k, we make a decision x_k and then observe the exogenous information W_{k+1}. The system transits to a new state S_{k+1} according to the transition function: $S_{k+1}= S^M(S_k, x_k, W_{k+1})$. Note that "M" represents model in Powell (2007). The state transition involves the following transition functions:

$$i_{k+1} = x_k, \tag{4.4}$$

$$D_{k+1} = \hat{D}_{k+1}. \tag{4.5}$$

The disruption status vector transits from $\hat{D}_k$ to $\hat{D}_{k+1}$ according to a Markovian transition matrix. Note that D_{k+1} is the vector of random variables representing the disruption status of each vulnerable arc in the network for the next stage. We define the transition matrix of a vulnerable arc r from stage k to $k+1$ as $\Theta^r(k|S_k, x_k)$. This transition matrix is dependent on the state and the travel time of the current arc: the probability of being in the disruption status of the next stage $\hat{D}_{k+1}$ depends on the travel time between i_k and i_{k+1} given the disruption status realization $\hat{D}_k$. Note that the travel time given disruption status is a positive integer. Let $p^r_{u,u'}$ denote the unit-time transition probability between any two disruption levels of the vulnerable arc r, $p^r_{u,u'} = P\{\hat{D}_{k+1}(r) = u' | \hat{D}_k(r) = u\}$. $\Theta^r(k|S_k, x_k)$ is the transition matrix of the vulnerable arc r considering the travel-time-dependency given the current travel time based on the disruption status realization, $\hat{D}_k$:

$$\Theta^r(k|S_k, x_k) = \begin{bmatrix} p^r_{u^1,u^1} & p^r_{u^1,u^2} & \cdots & p^r_{u^1,u^{M_r}} \\ p^r_{u^2,u^1} & p^r_{u^2,u^2} & \cdots & p^r_{u^2,u^{M_r}} \\ . & & & \\ . & & & \\ p^r_{u^{M_r},u^1} & p^r_{u^{M_r},u^2} & \cdots & p^r_{u^{M_r},u^{M_r}} \end{bmatrix}^{t_{i_k,x_k}(\hat{D}_k)} . \tag{4.6}$$

The probability of having the new disruption status $\hat{D}_{k+1}$ given $\hat{D}_k$ is then calculated as ($\Theta^r_{u,u'}(k|S_k, x_k)$ indicate the specific row and column index in the matrix $\Theta^r(k|S_k, x_k)$):

$$P(\hat{D}_{k+1}|\hat{D}_k) = \prod_{r=1}^{R} \Theta^r_{u,u'}(k|S_k, x_k). \tag{4.7}$$

Objective Function

The objective is to minimize the expected total travel time from the origin node until the destination node (where K is reached by arriving at the destination):

$$\min_{\pi \in \Pi} \mathbb{E}(\sum_{k=1}^{K} C(S_k, \mathcal{X}^\pi(S_k))), \tag{4.8}$$

where π denotes a policy or decision rule and Π denotes the set of all policies.

4.3. Approximate Dynamic Programming Approach

The optimization problem in Equation (4.8) can be solved using the Bellman Equations:

$$V_k(S_k) = \min_{x_k \in \mathcal{X}_k} C(S_k, x_k) + \sum_{S_{k+1}} P(S_{k+1}|S_k) V_{k+1}(S_{k+1}), \tag{4.9a}$$

$$V_K(S_K) = 0. \tag{4.9b}$$

Here, $V_k(S_k)$ is the value of being in state S_k. Our solution becomes:

$$x_k^* = \arg\min_{x_k \in \mathcal{X}_k} C(S_k, x_k) + \mathbb{E}(V_{k+1}(S_{k+1})). \tag{4.10}$$

The MDP faces the curses of dimensionality (states, outcomes and decisions) with the increase in the number of disruption levels and the number of nodes. To solve large scale problems with many disruption levels, the dynamic programming approach for solving the Bellman's equations becomes computationally intractable. In the literature, many techniques such as state-reduction techniques (Kim et al. 2005a, Thomas and White 2007) and stochastic lookahead algorithms (Güner et al. 2012, Sever et al. 2013) are used to solve for the MDP problems with reduced computational time. As an alternative the Approximate Dynamic Programming (ADP) approach is a powerful tool to overcome the curses of dimensionality, especially for complex and large scale problems (Powell 2007, 2011).

In this paper, we use an ADP approach with value iteration (Powell 2007, 2011). The essence of this approach is to replace the actual value function $V_k(S_k)$ with an

approximation $\bar{V}_k(S_k)$. ADP proceeds by estimating the approximate value $\bar{V}_k(S_k)$ iteratively. For our problem, this means that we traverse the roads repeatedly to estimate the value of being in a specific state. Let $\bar{V}_{k+1}^{n-1}(S_{k+1})$ be the value function approximation after $n-1$ iterations. Instead of working backward through time as in the traditional DP approach, ADP works forward in time.

The optimization problem using the ADP approach is given in Equation (4.11):

$$\hat{v}_k^n = \underset{x_k \in \mathcal{X}_k}{\arg\min} \quad C(S_k^n, x_k^n) + \sum_{S_{k+1}} P(S_{k+1}|S_k)\bar{V}_{k+1}^{n-1}(S_{k+1}). \tag{4.11}$$

Post-decision State Variable

The approximate value of being in state S_k^n at iteration n, $\bar{V}_k^n(S_k^n)$, contains the expectation over all possible states at the next stage. For large scale problems with many disruption levels, the state- and outcome-space explodes and optimization with the expectation can be computationally intractable. We adopt the post-decision state variable as suggested by Powell (2007) and Powell (2011). We define the post-decision state as S_k^x which represents the state immediately after we make a decision.

$$S_k^x = S^{M,x}(S_k, x_k) = (x_k, \hat{D}_k) = (i_{k+1}, \hat{D}_k), \tag{4.12}$$

where M represents model (Powell 2011). The post-decision state eliminates the expectation in Equation (4.11) by using the deterministic value of choosing an action, x_k, given the current state S_k.

Value Function Approximation with Lookup Table Representation

We use lookup table representation for the value function approximation. This means that for each discrete state S_k, we have an estimate $\bar{V}_k(S_k)$. We, then, update our estimate when we visit the particular state. As we use a post-decision state, we should update the value of being in the post-decision state of the previous stage by using $\hat{v}_k^n$:

$$\bar{V}_{k-1}^n(S_{k-1}^{x,n}) = (1-\alpha_{n-1})\bar{V}_{k-1}^{n-1}(S_{k-1}^{x,n}) + \alpha_{n-1}\hat{v}_k^n. \tag{4.13}$$

The Equation 4.13 represents *the harmonic step size rule* where the state value is updated by using a weighted average of the current estimate and the estimate from the previous iteration.

Exploration/Exploitation Strategy

In the ADP algorithm, we estimate a value function by visiting states to estimate the value of being in a certain state. If we explore, we visit states to improve the estimates of the value of being in the state regardless the decision gives the best value. However, if we exploit, we use our best estimate values and we make the decision that gives the best value given the current information. One of the challenges in ADP is that we should determine a right balance between exploration and exploitation when making a decision given a certain state. If we only exploit, we

may trap into the local optima by choosing the decision that "appears" to have the best value. If we only explore, then we visit every state randomly that may increase the computation time while leading to poor value estimates. Therefore, we need an efficient exploration/exploitation strategy for higher quality estimates.

In the early iterations, the value of being in a state is highly dependent on the sample path and the initial solution. Therefore, we need to use the exploration more in the early iterations to improve the quality of the state values. For this purpose, we use a mixed exploration/exploitation strategy (Powell 2007). To do this, we select a random probability of choosing the best action or choosing alternative actions. We ensure that the probability of choosing the best action increases as we visit the state more. Therefore, we choose to use exploration rate, ρ_k, for choosing the next best alternative as $\rho_k = b/n'(S_k)$. We should note that the exploration rate decreases with the number of visits to the particular state, $n'(S_k)$, and increases with the parameter "b".

Initialization Heuristics

According to Powell (2007) and Powell (2011), initial values play an important role in the accuracy of the approximate values and the computational performance of an ADP algorithm. For this purpose, we investigate the effect of two initialization heuristics: a deterministic approach and a stochastic two-arc-ahead policy with memoryless probability transitions (denoted as $DP(2, M)$).

In the deterministic approach, we determine initial state values by solving a deterministic shortest path problem, assuming that the probability of having a disruption is zero. In determining the route, we only consider the non-disruption state for all vulnerable arcs. In the transition probabilities of all vulnerable arcs, the no-disruption state becomes an absorbing state in Equation (4.6). Then, we solve Equation (4.9) (The output from this initialization heuristic is independent of the disruption state).

However, the dynamic shortest path problem is travel-time-dependent, so including disruption status dependent initial values in the ADP algorithm may improve the quality of the solutions. As the problem involves disruption that may occur during travel, solving for the stochastic shortest path problem with the time-invariant (memoryless) probability transitions "for all vulnerable arcs" in the network may provide lower solution quality. Therefore, we use a lookahead policy. According to results of preliminary tests, we observe that considering lookahead policy considering the detailed information of only the two-arc-ahead neighborhood reduces the expected total travel time by the range $0.5\% - 0.8\%$ on average for the networks with 16 and 36 nodes (We refer the reader to Chapter 3 for more information on the effect of travel information on the quality of routing policies).

In this policy, we have real-time information of only two-arc-ahead from the current node. Therefore, the state space of the current node i_k is modified as $S_k = (i_k, \hat{D}_k^{i_k})$ where $\hat{D}_k^{i_k} = \{u_k^{r_{i_k 1}}, u_k^{r_{i_k 2}}, .., u_k^{r_{i_t R_{i_k}}}\}$ with r_{i_k}, the vector of the vulnerable arcs that

are in the two-arc-ahead neighborhood of node i_k and $R_{i_k} = |r_{i_k}|$, $R_{i_k} \leq R$. For the rest of the arcs, we calculate expected travel times. The transition probability vector, in Equation (4.6) is no longer travel-time-dependent and we only consider the part of the transition matrix with only the vulnerable arcs in the two-arc-ahead neighborhood (The output from this initialization heuristics is dependent on the disruption state).

Updating the value function

In this paper, we investigate the computational effect of both single-pass and double-pass approaches to update the state values (Powell et al. 2012).

4.3.1 The Generic Approximate Dynamic Programming Algorithm

In the generic ADP, we adopt the value function approximation algorithm with the post-decision state variable. We use a mixed exploration/exploitation strategy to update the value function approximations. In this paper, we denote the generic ADP algorithm as "Standard ADP" algorithm.

ADP Algorithm: Single-Pass Version (ADP_S)

In the ADP algorithm with single-pass, updating the value function takes place as the algorithm progresses forward in time. According to the Algorithm 2, in Step 2a, at stage k, we obtain an updated estimate of the state value of being in state S_k. In Step 2b, we update the estimate of $\bar{V}_{k-1}^n$ by using the state value from the previous iteration, $\bar{V}_{k-1}^{n-1}$, and the current value estimate $\hat{v}_k^n$. Note that we collect information as we proceed in time due to the fact that the disruption transition rates for the future stage is dependent on the current travel time.

ADP Algorithm: Double-Pass Version (ADP_D)

As an alternative to the single-pass value iteration algorithm, we also use a double-pass algorithm (Algorithm 3). In this algorithm, we step forward in time by creating a trajectory of states, actions and outcomes (Step 2). Then, we update the value function by stepping backwards through the stages (Step 3). The update of the value function is conditional on whether we exploit or explore. If we exploit at stage k (choosing the action with the minimum value), then we update the value function of the state at stage k. If we explore at stage k, then we update the value function only if the explored action improves the value function compared to the exploitation (Step 3b).

Algorithm 2 ADP algorithm with single-pass

Step 0: Initialization
Step 0a: Initialize $\bar{V}_k^0$, $\forall k$ by using the value from the initialization heuristic.
Step 0b: Set n=1.
Step 1: Choose a sample disruption vector with Monte Carlo simulation for n and for $k = 0$, and initialize $S_0^n = (\text{origin node}, \hat{D}_0^n)$.
Step 2: Do for all $k = 0, ..., K-1$ (where K is reached by arriving at the destination node)
Step 2a: Choose a random probability p and exploration rate ρ_k as $\rho_k = 0.2/n'(S_k)$

$$\hat{v}_k^n = \min_{x_k \in N, x_k \in \mathcal{X}_k} \quad C(S_k^n, x_k^n) + \bar{V}_k^{n-1}(x_k, \hat{D}_k^n). \tag{4.14}$$

The node x_k that gives the optimal value is denoted as x_k^{*n}.

$$\hat{v}_k^n = \begin{cases} \text{solve Equation (4.14) for } x_k \in N & \text{if} \quad p \geq \rho_k, \\ \text{solve Equation (4.14) for } x_k \in N \setminus x_k^{*n} & o.w. \end{cases}$$

The node that is chosen is denoted as x_k^n and becomes the next node to visit.
Step 2b: If $k > 0$, update $\bar{V}_{k-1}^n(S_{k-1}^{x,n})$ using:

$$\bar{V}_{k-1}^n(S_{k-1}^{x,n}) = (1 - \alpha_{n-1})\bar{V}_{k-1}^{n-1}(S_{k-1}^{x,n}) + \alpha_{n-1}\hat{v}_k^n. \tag{4.15}$$

We use the harmonic stepsize: $\alpha_{n-1} = \frac{a}{a+n'(S_k)-1}$ and $n'(S_k)$ is the total number of visits to state S_k.

Step 2c: Find the post-decision state: $S_k^{x,n} = (i_{k+1}, \hat{D}_k^n)$.
Step 2d: Find the next pre-decision state: $S_{k+1}^n = (i_{k+1}, W_{k+1}^n)$.
Step 3: Increment n. If $n \leq \mathbb{N}$ go to Step 1. Note that $\mathbb{N}$ denotes the pre-set maximum number of iterations.
Step 4: Return the value functions $(\bar{V}_k^{\mathbb{N}})_{k=1}^K$.

Algorithm 3 ADP algorithm with double-pass

Step 0: Initialization
Step 0a: Initialize $\bar{V}_k^0$, $\forall k$ by using the value from the initialization heuristic.
Step 0b: Set n=1.
Step 1: Choose a sample disruption vector with Monte Carlo simulation for n and for $k = 0$, and initialize $S_0^n =$ (origin node, $\hat{D}_0^n$).
Step 2: (Forward pass) Do for all $k = 0, 1, ..., K-1$ (where K is reached by arriving at the destination node)
Step 2a: Solve:

$$\hat{v}_k^n = \min_{x_k \in N, x_k \in \mathcal{X}_k} C(S_k^n, x_k^n) + \bar{V}_k^{n-1}(x_k, \hat{D}_k^n). \tag{4.16}$$

The node x_k that gives the optimal value is denoted as x_k^{*n}.

$$\hat{v}_k^n = \begin{cases} \text{solve Equation (4.16) for } x_k \in N & \text{if} \quad p \geq \rho_k, \\ \text{solve Equation (4.16) for } x_k \in N \setminus x_k^{*n} & o.w. \end{cases}$$

The node x_k that gives the optimal value is denoted as x_k^{*n} and $x_k^{\Delta n}$ if it is from the exploration.
Step 2b: If we choose exploration with $x_k^{\Delta n}$ compute :

$$\bar{V}_{k-1}^{*n}(S_{k-1}^{x,n}) = (1-\alpha_{n-1})\bar{V}_{k-1}^{n-1}(S_{k-1}^{x,n}) + \alpha_{n-1}\hat{v}_k^{*n}, \tag{4.17}$$

$$\hat{v}_k^{*n} = C(S_k^n, x_k^{*n}) + \bar{V}_k^{n-1}(x_k^{*n}, \hat{D}_k^n). \tag{4.18}$$

Step 2c: Find the post-decision state: $S_k^{x,n} = (i_{k+1}, \hat{D}_k^n)$.
Step 2d: Find the next pre-decision state: $S_{k+1}^n = (i_{k+1}, W_{k+1}^n)$.
Step 3: (Backward pass) Do for all $k = K-1, ..., 1$
Step 3a: Compute $\hat{v}_k^n$ using the decision x_k^n from the forward pass:

$$\hat{v}_k^n = C(S_k^n, x_k^n) + \hat{v}_{k+1}^n. \tag{4.19}$$

Step 3b: If $k > 1$, update $\bar{V}_{k-1}^n(S_{k-1}^{x,n})$ using:

$$\bar{V}_{k-1}^n(S_{k-1}^{x,n}) = \begin{cases} (1-\alpha_{n-1})\bar{V}_{k-1}^{n-1}(S_{k-1}^{x,n}) + \alpha_{n-1}\hat{v}_k^n & \text{if} \quad x_k^{*n}, \\ (1-\alpha_{n-1})\bar{V}_{k-1}^{n-1}(S_{k-1}^{x,n}) + \alpha_{n-1}\hat{v}_k^n & \text{if} \quad x_k^{\Delta n} \& \quad \bar{V}_{k-1}^{\Delta n}(S_{k-1}^{x,n}) < \bar{V}^{*n}_{k-1}(S_{k-1}^{x,n}). \end{cases}$$

We compute $\bar{V}_{k-1}^{\Delta n}(S_{k-1}^{x,n})$ if we have explored at stage k with action $x_k^{\Delta n}$:

$$\bar{V}_{k-1}^{\Delta n}(S_{k-1}^{x,n}) = (1-\alpha_{n-1})\bar{V}_{k-1}^{n-1}(S_{k-1}^{x,n}) + \alpha_{n-1}\hat{v}_k^n. \tag{4.20}$$

Where we use the harmonic stepsize: $\alpha_{n-1} = \frac{a}{a+n'(S_k)-1}$ and $n'(S_k)$ is the number of visits to the current state.
Step 4: Increment n. If $n \leq \mathbb{N}$ go to Step 1. Note that $\mathbb{N}$ denotes the pre-set maximum number of iterations.
Step 5: Return the value functions $(\bar{V}_k^{\mathbb{N}})_{k=1}^K$.

4.3.2 Hybrid ADP with a Clustering Approach : a Deterministic lookahead policy with Value Function Approximations

The state variable in our MDP formulation is a representation of the nodes and the disruption statuses of the vulnerable arcs. When the network size and the number of disruption levels increase, the number of states and estimated state values in the lookup table also increase. To reduce the computational time while maintaining good solution quality for large size instances, we need to improve the standard ADP algorithm. To do this, we exploit the structure of the disruption transition function in Equation (4.6) which is travel-time-dependent. The structure of the disruption transition function shows that the shorter the travel time between two nodes, the disruption statuses of the vulnerable arcs at the next stage remain similar to the disruption statuses at the current stage. In other words, in a close neighborhood of the current node, i_k, the disruption statuses of the observed vulnerable arcs at stage k do not change much as compared to those at stage $k-1$. This structure is a good motivation to cluster the nodes that are close to each other. Using this clustering idea, we develop a hybrid ADP algorithm. In this hybrid algorithm, we first apply a deterministic lookahead policy within the cluster and we estimate the cost outside the cluster until the destination using value function approximations.

Figure 4.1 illustrates how we cluster and perform the hybrid ADP algorithm. At stage k, we have the state variable $S_k = (i_k, \hat{D}_k)$. We form the cluster of the current node i_k by simply considering the nodes that are two-arc-ahead from i_k. We denote the cluster of i_k as cl_{i_k}. We assume that the disruption status $\hat{D}_k$ does not change within the cluster and we apply a deterministic lookahead policy within the cluster. The lookahead policy consists of solving the Dijkstra's algorithm (Dijkstra 1959) within the cluster. In this way, we determine the shortest path from i_k until the next node $i_{k+1}^{cl_{i_k}}$ in the cluster cl_{i_k}. The cost from the current node to the selected next node given the disruption status is determined by the Dijkstra's shortest path algorithm and is denoted as $\tilde{C}_k(i_k, i_{k+1}^{cl_{i_k}}, \hat{D}_k)$. Then, we estimate the cost outside the cluster from the node $i_{k+1}^{cl_{i_k}}$ until the destination using value function approximations.

We find the next node to visit with the exploration/exploitation strategy defined in Sections 4.3.1 and 4.3.1. Then, we approximate the current state value, $\bar{V}_k(S_k)$, using the harmonic stepsize rule. Then, we continue to find the next cluster consisting of the two-arc-ahead nodes of the next node and approximate its value. We always go further towards the destination node. We apply the same process until arriving at the destination node. By a learning process, the hybrid algorithm clusters the nodes such that the approximate value until the destination is lower given the current disruption status. The state variables considered in the value function approximation are then visited and updated more frequently such that their state values become more accurate. This leads to higher quality approximate values for the state variables and reduces the computational cost significantly by eliminating the steps for updating and exploring the states that are already included in the lookahead policy.

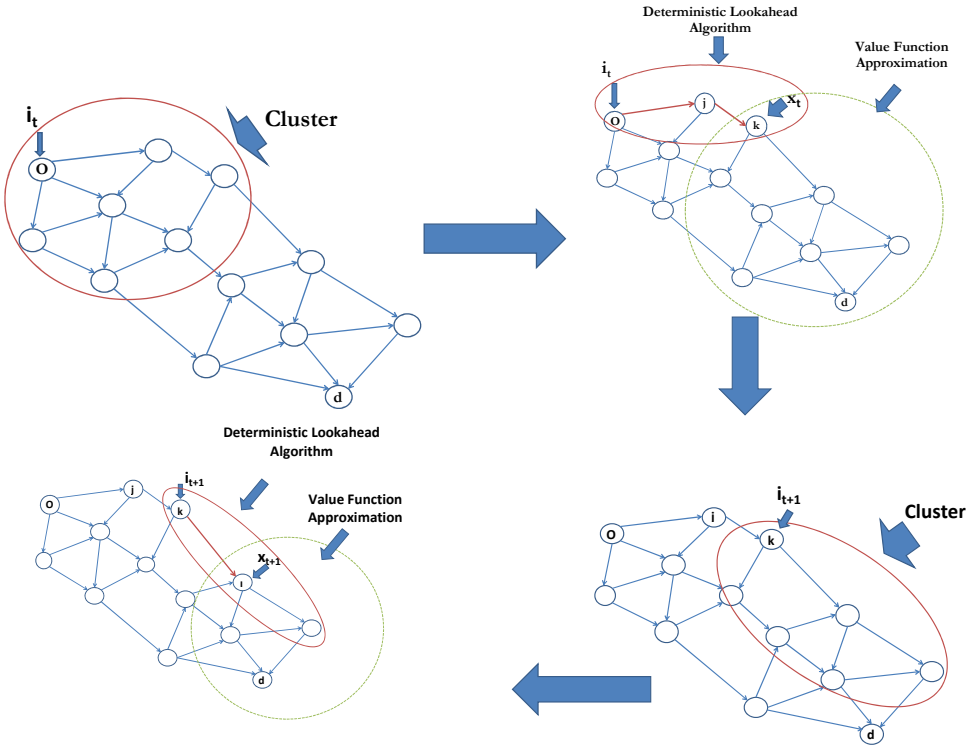


Figure 4.1 Hybrid ADP with a clustering approach

The hybrid ADP algorithm with the clustering approach has three types of algorithmic variations: by the size of the cluster, by deciding on which nodes to update when we determine a shortest path within the cluster and whether we use single-pass or double-pass to update state values.

Hybrid ADP with Clustering Approach: Algorithmic Variations

The size of the cluster
In this paper, we analyze the effect of cluster size by developing hybrid ADP algorithms with clusters considering two-arc-ahead neighborhood and three-arc-ahead neighborhood of the current node. We refer the hybrid ADP algorithms with different network sizes as CL_x where $x \in 2, 3$. In the hybrid ADP with the clustering approach CL_2, the clusters are formed by considering the nodes that are two-arc-ahead from the current node. Similarly in CL_3, the cluster contains the nodes within the three-arc-ahead neighborhood of the current node.

We apply the deterministic lookahead policy to obtain the cost between i_k and $i_{k+1}^{cl_{i_k}}$, $\tilde{C}_k(i_k, i_{k+1}^{cl_{i_k}}, \hat{D}_k^n)$. $i_{k+1}^{cl_{i_k}}$ is an element from the cluster of two-arc-ahead or three-arc-ahead neighborhood of the node i_k for hybrid ADP algorithms CL_2 and CL_3, respectively. We then add the approximate value from the node $i_{k+1}^{cl_{i_k}}$ to the destination node. Here, we should note that we take the deterministic travel time within the cluster.

The update of the state values in a shortest path within the cluster
Another variation of the hybrid algorithm is to decide whether to update the state values of the nodes on the shortest path or not. For each shortest path consisting of nodes to travel until $i_{k+1}^{cl_{i_k}}$, we also investigate whether updating the state values of the path obtained from the lookahead policy improves our solution or not. For instance, our current node is i_k and we choose to go to node $i_{k+1}^{cl_{i_k}}$ by using the shortest path $i_k - z - i_{k+1}^{cl_{i_k}}$. If we only update the state value of node i_k, we call this as "No-Update", NU. If we also update the state value of node z, then we call this as "Update", U.

The update of the state values
We study both single-pass and double-pass approach to update the state values. For the single-pass algorithm with CL_2 and CL_3 approach with or without updating the values of the nodes on shortest path within the cluster ($CL_{(x,S,U)}$, $CL_{(x,S,NU)}$ and $x \in 2, 3$), we use algorithm 2. Similarly, for the double-pass algorithm with CL_2 and CL_3 approach ($CL_{(x,D,U)}$, $CL_{(x,D,NU)}$), we use algorithm 3. For the algorithms, the value function in Equations (4.14) and (4.16) becomes:

$$\hat{v}_k^{*n} = \min_{i_{k+1}^{cl_{i_k}}} \tilde{C}_k(i_k, i_{k+1}^{cl_{i_k}}, \hat{D}_k^n) + \bar{V}_k^{n-1}(i_{k+1}^{cl_{i_k}}, \hat{D}_k^n). \tag{4.21}$$

A Benchmark Heuristic: Dynamic Programming with Stochastic Two-arc-ahead Policy $(DP(2,H))$

For large instances, where the optimal solution are not obtainable, we benchmark with the stochastic two-arc-ahead policy ($DP(2,H)$) as proposed in Chapter 3, which is shown to perform only slightly worse than the optimal algorithm. This algorithm considers limited online information for each stage which reduces the computational time significantly. In $DP(2,H)$ it is assumed that by the time we arrive at the further arcs, we will experience the time-invariant probabilities. Therefore, we eliminate the computational burden to calculate the transition probabilities for all vulnerable arcs.

In this policy, we retrieve online information of the arcs that are two-arc-ahead from the current node. Therefore, the state space of the hybrid policy for any current node i_k is modified as $S_k = (i_k, \hat{D}_k^i)$ where $\hat{D}_k^i = \{u_k^{1_{i_k}}, u_k^{1_{i_k}}, .., u_k^{\bar{R}_{i_k}}\}$ only includes the vector of the vulnerable arcs that are in the two-arc-ahead neighborhood of node i_k and $R_{i_k} = |r_{i_k}|$. In this heuristic, we use the same travel-time-dependent transition matrix in Equation (4.6) except we only consider the limited part of the transition matrix where only the vulnerable arcs that are maximum two-arc-ahead from the current node are included. For the vulnerable arcs outside the neighborhood, we calculate the time-invariant probability distributions. We solve Equation (4.9) by using a backward recursion algorithm given in Chapter 3.

4.4. Computational Experiments and Analysis

In our computational experiments, we analyze the performance of various algorithms with different network instances. In what follows, we first describe our test instances. Then, we analyze the effect of algorithmic design variations of ADP algorithms (the standard and the hybrid ADP algorithms) in Section 4.4.3. At the end of the analysis, we choose the design variations for both the standard ADP and the hybrid ADP algorithms that provide high quality solutions. Considering these design variations, in Section 4.4.4, we compare the performance of the hybrid ADP algorithms with the standard ADP algorithms, the benchmark heuristic and the optimal algorithm with respect to the total realized travel time and the computational time.

Throughout the experiments, we fix the parameters used in the ADP algorithms. After performing several preliminarily tests, we set the constant in the harmonic stepsize rule to $a = 5$. We also set the exploration rate as $0.2/n'(S_k)$, where $n'(S_k)$ is the number of visits to the state. Also, we fix the total number of iterations to $\mathbb{N} = 100,000$.

4.4.1 Design of Test Instances

We generate our test instances based on the following network properties:

Network size: We generate small, medium and large size networks with 16, 36 and 64 nodes, respectively. The network is designed such that the origin-destination pairs are situated from the top-left to the bottom-right corners. With this structure, we prevent evaluating unnecessary nodes far from the shortest path. Clearly, this does not limit the applicability of our algorithms.

Network vulnerability: We measure the vulnerability by the percentage of vulnerable arcs in the network. Vulnerable arcs are randomly assigned to the shortest paths that are found with an iterative process where every iteration a new shortest path is found and a new vulnerable arc is added to the network. We have instances with low (50%) and high percentage (up to 80%) of vulnerable arcs which are denoted as low and high network vulnerability, respectively.

Number of disruption levels: The vulnerable arcs on a network have either $K = 2$, 3 or 5 disruption levels. Note that the disruption levels also include the non-disrupted case where there is no disruption at all. (For each vulnerable arc the possible disruption levels are kept the same).

Disruption rate: Disruption rate is defined by the steady state probability of having a disruption on a vulnerable arc. We use a low probability of having disruptions to be between $[0.1 - 0.5)$ and a high probability between $[0.5 - 0.9)$ which are denoted as low and high disruption rates, respectively.

Travel times: Travel time of each arc for the non-disrupted level is randomly selected from a discrete uniform distribution $U(1, 10)$. The steady state probability of the non-disrupted level for each vulnerable arc is randomly selected based on the disruption rate. If there is a disruption, the travel time for the non-disrupted level is multiplied by a scalar depending on the disruption level.

The probabilities of the disrupted levels are computed to make sure that the expected travel time of the vulnerable arc is the same for all K. For example, the expected travel time of a vulnerable arc in a network with 2 disruption levels is kept the same in the same network with 3 and 5 disruption levels with the same disruption rate.

We define an "instance type" as the networks that have the same set of network properties as described above. For instance, small size network with low network vulnerability and low disruption rate is a unique instance type. In this paper, we generate 36 different instance types with properties as in Table 4.1. For each instance type, we randomly generate 50 replications and in total we generate 1800 test instances.

In the experimental analysis, we report the results by aggregating them according to these network properties: disruption rate, network size, vulnerability of the network and number of disruption levels.

Table 4.1 Summary of instance types

Number of Nodes	Number of vulnerable arcs		Disruption rate		Number of disruption levels
	Low	High	Low	High	K
16	3	5	[0.1 - 0.5)	[0.5 - 0.9)	2, 3, 5*
36	5	7	[0.1 - 0.5)	[0.5 - 0.9)	2, 3, 5*
64	7	9	[0.1 - 0.5)	[0.5 - 0.9)	2, 3, 5*

(*) In the first experimental analysis where we analyze the effect of different algorithmic variations, we exclude the network instances with $K = 5$ due to computational issues.

4.4.2 Evaluation of the Algorithms

We analyze the performance of the standard ADP algorithms, the Hybrid ADP algorithms, the dynamic programming algorithm with stochastic two-arc-ahead policy ($DP(2, H)$) and the optimal algorithm via an MDP. For the evaluation of the algorithms, we use two procedures: exact evaluation and evaluation via simulation.

We perform an exact evaluation for the networks where the optimal algorithm is computationally tractable within the computational time limit of 1 hour. This specific time limit is selected due to practical reasons because in real-life before departure it is too long and impractical to wait for an offline policy more than an hour. In the exact evaluation, for each algorithm, we obtain routing policies considering each possible state. Then, we compute the exact value function with these pre-determined policies by enumerating for all states via a backward recursion. This value gives the expected cost of the algorithm considering all possible states.

For the networks where MDP is not tractable within the time limit of 1 hour (so the exact evaluation), we simulate each algorithm with a sample size of 5000 for each test instance. We ensure that each algorithm uses the same sample of transition probabilities for each instance. For each test instance, we find the average travel time and computational time by calculating the average values of the simulation.

For each instance type, we report the average travel time, which is the average values of the 50 replications, the standard deviation of the 50 replications and the percentage gap with respect to the benchmark heuristic, i.e., $DP(2, H)$.

The algorithms presented in this paper are programmed in Java. All experiments are conducted on a personal computer with an IntelCore Duo 2.8 Ghz Processor and 3.46 GB RAM.

4.4.3 Experimental Results and Discussion- Effect of Algorithmic Design Variations of ADP Algorithms

In this section, we compare the performance of the ADP algorithms that use different algorithmic design variations: initialization heuristics, updating state values with either single-pass or double-pass and the update or no-update of the state values in a shortest path within the cluster. The ADP algorithms with respect to the relevant design variations are summarized in Table 4.2. The analysis is done considering the total realized cost for different network sizes, vulnerabilities and disruption levels. As the observations show that disruption rate has no distinctive effect on these variations, for conciseness we do not make a separate analysis for the effect of disruption rate. Rather, we analyze all the test instances with both low and high disruption rate depending on network sizes, vulnerabilities and disruption levels. The cost for each algorithm (C[ADP algorithm]) represents the realized cost which is the total travel time computed from the relevant evaluation method (expected total travel time or average travel time over simulation replications). To analyze the statistical performance differences, we also apply paired samples t-test using SPSS and we report mean differences between ADP algorithms with different design variations and its confidence interval with the significance level of 0.05 (Ross 1999). Note that due to the computational purposes and conciseness, for the algorithmic design variations experiments we leave out the instance types with 5 disruption levels. After choosing a good algorithmic design variation, we will provide a thorough analysis over all test instances.

Table 4.2 The ADP algorithms and algorithmic design variations

	Single-pass	Double-pass
Standard ADP	ADP_S	ADP_D

(a) Standard ADP algorithms

	Cluster Size			
	Two-arc-ahead		Three-arc-ahead	
	No-update	Update	No-update	Update
Single-pass	$CL_{(2,S,NU)}$	$CL_{(2,S,U)}$	$CL_{(3,S,NU)}$	$CL_{(3,S,U)}$
Double-pass	$CL_{(2,D,NU)}$	$CL_{(2,D,U)}$	$CL_{(3,D,NU)}$	$CL_{(3,D,U)}$

(b) Hybrid ADP algorithms

Effect of Initialization Heuristics

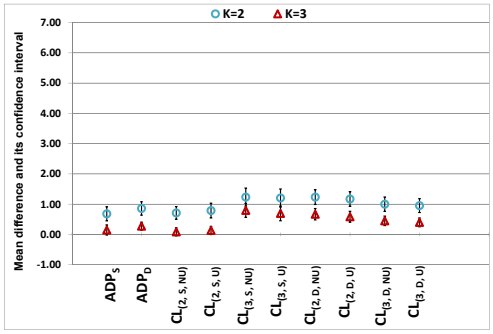
In the ADP algorithms, we use two different initialization heuristics for obtaining initial value function estimates: a deterministic approach and a stochastic two-arc-ahead policy with memoryless probability transitions ($DP(2, M)$).

To analyze whether there is a significant performance difference between the deterministic approach and $DP(2, M)$, we applied paired samples t-test with significance level of 0.05 (Ross 1999). In this test, we analyze the realized cost difference between ADP algorithms considering deterministic approach and $DP(2, M)$ as the initialization heuristic ($(C[\text{ADP with deterministic approach}] - C[\text{ADP with } DP(2, M)])$). Figure 4.2 shows the mean difference and the variation of the mean difference with a 95% confidence interval. A positive mean difference values and a positive confidence interval indicate that the ADP algorithms considering the initialization heuristic $DP(2, M)$ on average outperform the ones using initial values from the deterministic approach. The test instances are analyzed depending on the number of disruption levels and network sizes.

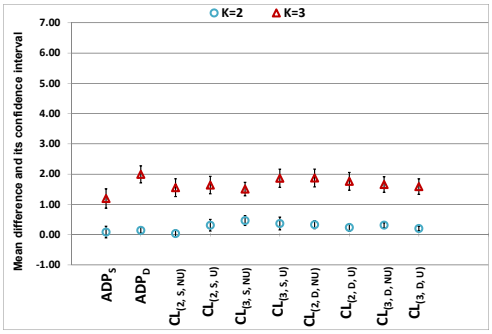
Note that the ADP algorithms with double-pass approach considering the initialization heuristic $DP(2, M)$ perform significantly different from the ADP algorithms using the initial values from the deterministic approach (Appendix Table A1). This is not the case in for the single-pass approach. For instance, in ADP_S (small network with K=3 and medium network with K=2) and $CL_{(2,S,NU)}$ (small network with K=3, medium and large network with K=2) algorithms, there is no significant difference between the two initialization heuristics. This shows that the update of state values in the backward-pass (only if there is an improvement) exposes the benefits of considering state-dependent initial values with $DP(2, M)$.

The positive mean difference values in Figure 4.2 indicate that the ADP algorithms considering the initialization heuristic $DP(2, M)$ on average outperform the ones using initial values from the deterministic approach. In Figure 4.2, we observe that the confidence interval is strictly positive in the cases where the two initialization heuristics perform significantly different from each other. This shows that regardless of network properties ADP algorithms considering the initialization heuristic $DP(2, M)$ either significantly outperform the ones using initial values from the deterministic approach or perform without any significant difference. When we compare Figure 4.2-(a), (b) and (c), we observe that as disruption level and network size increase, the performance of ADP algorithms using the initialization heuristic $DP(2, M)$ significantly increases. This indicates that when we have more states, starting with the state-dependent initial values (as in $DP(2, M)$) provides more efficient exploration/exploitation such that we obtain higher quality value function estimates.

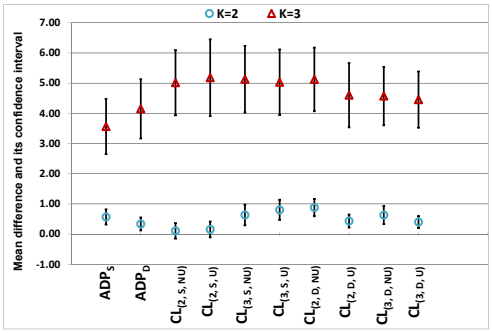
For the rest of the numerical analysis, we adopt $DP(2, M)$ as the initialization heuristic for the ADP algorithms because it gives significantly higher solution quality.



(a) Small network



(b) Medium network



(b) Large network

Figure 4.2 Mean difference between ADP algorithms with deterministic and $DP(2, H)$ initialization heuristics (with different network size). The error bars represent the 95% confidence interval around the mean difference.

Single-Pass versus Double-Pass

To investigate the effect of how we update value functions, we use both the single-pass and the double-pass procedures in ADP algorithms. Figure 4.3 and Figure 4.4 illustrate the percentage cost improvement of ADP algorithms considering the double-pass relative to the single-pass approach for different network sizes and different network vulnerability, respectively. First, the percentage cost improvement for each test instance is computed as follows:

$$\Delta(\%)_{instance} = \frac{(C[\text{ADP with single-pass}] - C[\text{ADP with double-pass}])}{C[\text{ADP with single-pass}]} * 100 \quad (4.22)$$

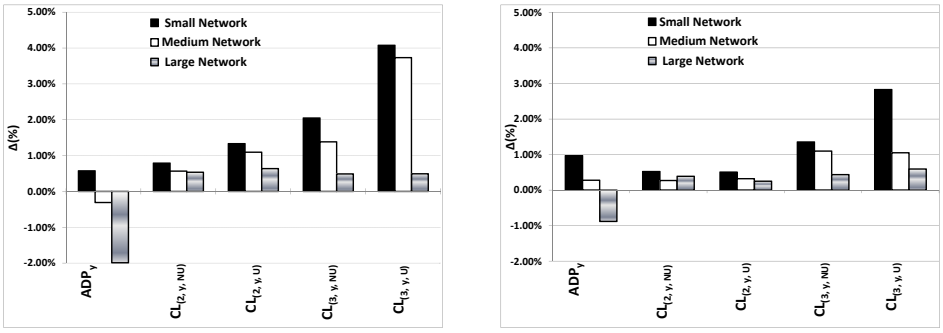
Then, we report the average value of $\Delta(\%)_{instance}$'s over the test instances ($\Delta(\%)$) based on the given network property. The positive $\Delta(\%)$ indicates that ADP algorithms with double-pass approach outperform the single-pass approach.

Note that in ADP_y (medium network with K=2), $CL_{(2,y,NU)}$ (large network with K=2) and $CL_{(2,y,U)}$ (large network with K=3) algorithms, there is no significant difference between the two update procedures (Appendix Table A2). For the other ADP algorithms in the remaining network instances, double-pass algorithms perform significantly different from single-pass algorithms.

Considering the significant differences, Figure 4.3 shows that the hybrid ADP algorithms with double-pass significantly outperform their single-pass counterparts for all network sizes. When the network size is smaller and the number of disruption levels is lower, in the hybrid ADP algorithms the double-pass approach perform much better than the single-pass approach. Furthermore, when we compare Figure 4.3-(a) and (b), the performance difference between the double-pass and the single-pass algorithms decreases. This indicates that as there are more disruption levels and as the number of states increases, the performance of the double-pass and single-pass approaches becomes similar. This is not necessarily true for the standard ADP algorithms where the double-pass approach for the standard ADP algorithm does not perform well on large size networks and the networks with low and high vulnerability.

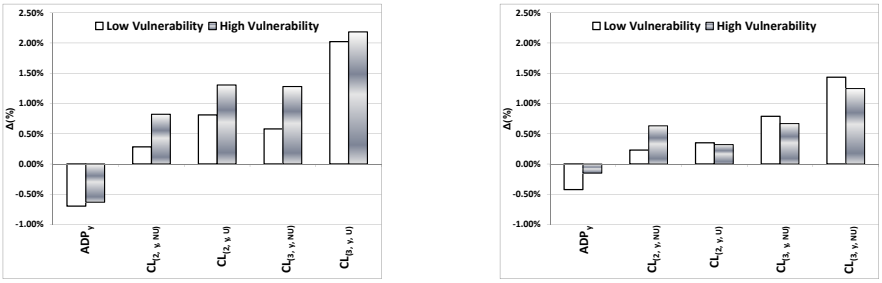
The effect of using single-pass and double-pass approaches in our ADP algorithms is also visible when we aggregate the networks according to the network vulnerability. We do this by taking the average over all different network sizes with respect to the network vulnerability. Figure 4.4 shows that the hybrid ADP algorithms with double-pass again significantly outperform their single-pass counterparts for all types of network vulnerability where $\Delta(\%)$ values are always positive. On the other hand, the standard ADP with double-pass performs worse than its counterpart with single-pass approach in both high and low network vulnerability. For all of the ADP algorithms as the network vulnerability and/or disruption levels increases, the performance difference between double-pass and single-pass decreases.

For the rest of the numerical analysis, we adopt the double-pass approach for the



(a) 2 disruption levels (b) 3 disruption levels

Figure 4.3 Average percentage cost improvement of ADP algorithms with double-pass procedure relative to single-pass (with different network sizes). Note that "y" stands for single-pass or double-pass approach.



(a) 2 disruption levels (b) 3 disruption levels

Figure 4.4 Average percentage cost improvement of ADP algorithms with double-pass procedure relative to single-pass (with different network vulnerability)

hybrid ADP algorithms as the algorithms with this design variation give consistently higher solution quality. For the standard ADP algorithms we adopt both the single-pass and the double-pass approaches.

The update or no-update of the state values in a shortest path within the cluster

In the hybrid algorithms, we either update the state values within the deterministic lookahead policy or not. Figures 4.3 and 4.4 show that the performance of the double-pass approach is significantly higher than the single-pass approach for the hybrid ADP algorithms. Therefore, in this analysis, we focus on the effect of the update of state

values considering only the double-pass approach.

Figure 4.5 and 4.6 illustrate the percentage cost improvement of the hybrid ADP algorithms with double-pass approach considering the update of the state values within the deterministic lookahead policy relative to their counterpart with the no-update approach. This is done for different network sizes and different network vulnerability, respectively. First, the percentage cost improvement of the hybrid ADP algorithms with update relative to their no-update counter parts for each test instance is computed as in Equation (4.22).

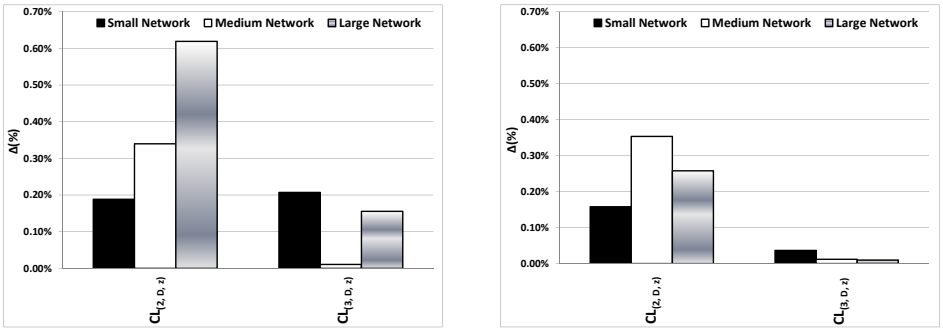
Then, we report the average value of $\Delta(\%)_{instance}$'s over the test instances ($\Delta(\%)$) based on the given network property. The positive $\Delta(\%)$ indicates that hybrid ADP algorithms with double-pass and update approach outperform the no-update approach.

To analyze whether there is a significant performance difference between the no-update and update procedures, we also applied paired samples t-test. In this test, we analyze the cost value difference between $CL_{(x,D,NU)}$ and $CL_{(x,D,U)}$ $((C[CL_{(x,D,NU)}] - C[CL_{(x,D,U)}]))$. $CL_{(2,D,U)}$ on average performs significantly different from $CL_{(2,D,NU)}$ regardless of network size and disruption levels (Appendix Table A3). This is not necessarily true for $CL_{(3,D,z)}$ because there is no significant difference between $CL_{(3,D,U)}$ and $CL_{(3,D,NU)}$ in medium and large networks with 2 disruption levels and networks with 3 disruption levels. Yet, in small networks with 2 disruption levels, $CL_{(3,D,U)}$ significantly outperforms $CL_{(3,D,NU)}$.

In Figures 4.5 and 4.6, the positive percentage relative cost considering the significant differences indicate that $CL_{(2,D,U)}$ algorithm significantly outperforms $CL_{(2,D,NU)}$ for all of the network sizes, disruption levels and vulnerability (Except for the case where $CL_{(2,D,U)}$ and $CL_{(2,D,NU)}$ performs similar in low network vulnerability with 3 disruption levels, Appendix Table A4). Similarly, $CL_{(3,D,U)}$ either significantly outperforms $CL_{(3,D,NU)}$ or there is no significant difference between the two algorithms (in the cases where $\Delta(\%)$ is really small, Appendix Tables A3 and A4).

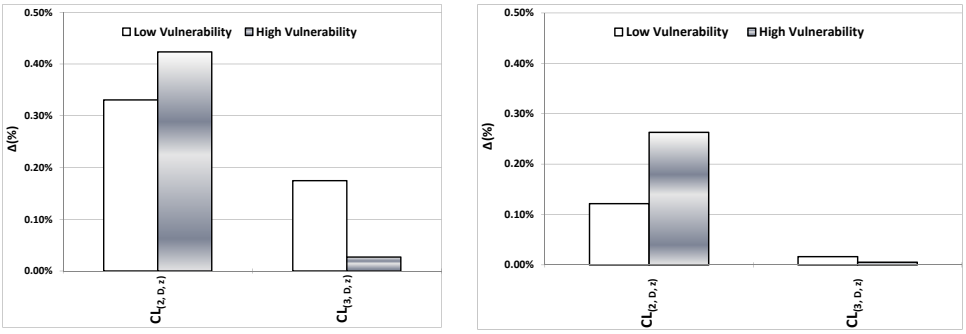
Not that the percentage relative improvements are relatively small on average for double-pass algorithms with no-update and update. This because in the double-pass approach, we update the states values in the path, only if they improve the solution. Therefore, on average a decision considered with no-update procedure leading to higher state values, most probably is not chosen. Note that there are also test-instances for which the improvement of update procedure relative to the no-update procedure is more than 10%.

For the rest of the numerical analysis, we adopt the double-pass with update approach for the hybrid ADP algorithms as the algorithms with this design variation give consistently higher solution quality (considering the significant cases).



(a) 2 disruption levels (b) 3 disruption levels

Figure 4.5 Average percentage cost improvement of the hybrid ADP algorithms with update relative to no-update of the state values in the shortest path (with different network size). Note that "z" stands for no-update or update approach.



(a)2 disruption levels (b) 3 disruption levels

Figure 4.6 Average percentage cost improvement of the hybrid ADP algorithms with update relative to no-update of the state values in the shortest path (with different network vulnerability).

4.4.4 Experimental Results and Discussion- Comparison of Algorithms

In this section, we analyze the hybrid ADP algorithms (with the selected algorithm design) with the standard ADP, MDP (the optimal algorithm when tractable within 1 hour of computational time limit), and the benchmark heuristic $DP(2, H)$. The comparison is done with respect to their estimated total cost defined by the total travel time computed according to the relevant evaluation method. We also provide the percentage gap of each algorithm with respect to the benchmark heuristic, $DP(2, H)$.

To analyze the statistical performance differences, we also apply paired samples t-test using SPSS and we report mean differences between ADP algorithms with different design variations and confidence intervals with the significance level of 0.05 (Ross 1999). We show the average results over all 1800 test instances depending on the network size, the disruption rate and the network vulnerability. For each of these network properties, we investigate the performance of the algorithms on the networks with different sizes, disruption rates and vulnerability.

Tables 4.3 and 4.4 show the average realized cost values of algorithms based on the total travel time, standard deviation of the cost values and the percentage gap with respect to $DP(2,H)$ in test instances with 2, 3 and 5 disruption levels for different network vulnerability considering all network sizes (Negative sign in the percentage gap means that the algorithm outperforms the benchmark heuristic).

We also provide the paired samples t-test results to analyze the significance between the performance of algorithms (Appendix B). Here, 95% confidence intervals (lower and upper bounds) are given with the significance level (p-value). The t-test is executed for the cost differences between the hybrid ADP, standard ADP, $DP(2,H)$ and MDP algorithms of each test instance based on the specific instance type. $p-value < 0.05$ indicates that there is a significant difference between the given algorithm pairs. If not, we conclude that there is no significant difference between them.

The hybrid ADP algorithm with clusters considering two-arc-ahead ($CL_{(2,D,U)}$) versus three-arc-ahead ($CL_{(3,D,U)}$) neighborhood

When we compare the hybrid ADP algorithms with different clustering sizes, we observe that in most of the network sizes $CL_{(2,D,U)}$ significantly outperforms $CL_{(3,D,U)}$. Note that solution quality of $CL_{(2,D,U)}$ is higher than or equal to the solution quality of $CL_{(3,D,U)}$ for all network sizes (Tables 4.3, 4.4 and Appendix Tables B1-B3). As the network size increases, the performance of $CL_{(2,D,U)}$ increases. When we consider small networks with low disruption rate, there is no significant difference between $CL_{(2,D,U)}$ and $CL_{(3,D,U)}$ with 2 disruption levels. However, as the network size is large, $CL_{(2,D,U)}$ outperforms $CL_{(3,D,U)}$ by 0.56% with 2 disruption levels (Table 4.3 and Appendix Tables B1, B3). Note that these percentages are computed from the gap differences.

Similar to the results regarding the network size, the solution quality of $CL_{(2,D,U)}$ algorithm is significantly higher than or equal to the solution quality of $CL_{(3,D,U)}$ algorithm for both low and high vulnerable networks (Table 4.3). When we compare Tables 4.3 and 4.4, we observe that as network vulnerability becomes high, the performance difference between two algorithms decreases. Significance results also support this such that the significant differences in low vulnerability become insignificant in high vulnerability considering the same networks size, disruption rate and disruption level.

Table 4.3 The estimated total cost value, standard deviation (Std) and percentage gap with respect to $DP(2, H)$ in different network sizes with low network vulnerability

Network Size	Disruption Rate	Algorithms	$DP(2,H)$	MDP	ADP_S	ADP_D	$CL_{(2,D,U)}$	$CL_{(3,D,U)}$
Small	Low	Cost Value	25.37	**25.29**	**26.16**	**26.08**	**25.48**	**25.48**
		Std	5.21	**5.23**	**5.34**	**5.23**	**5.23**	**5.28**
		Gap(%)	–	**-0.29%**	**3.12%**	**2.81%**	**0.43%**	**0.44%**
	High	Cost Value	28.22	**28.15**	28.87	28.70	**28.28**	28.42
		Std	5.32	**5.28**	5.41	5.42	**5.29**	5.30
		Gap(%)	–	**-0.25%**	2.32%	1.69%	**0.21%**	0.69%
Medium	Low	Cost Value	39.48	39.41	41.07	41.00	39.61	39.66
		Std	6.38	6.39	6.38	6.38	6.36	6.35
		Gap(%)	–	-0.17%	4.02%	3.86%	0.32%	0.45%
	High	Cost Value	42.94	42.90	43.27	44.08	43.07	43.21
		Std	6.26	6.27	6.22	6.24	6.24	6.25
		Gap(%)	–	-0.11%	0.76%	2.65%	0.30%	0.62%
Large	Low	Cost Value	54.88	**54.46**	**56.56**	**57.59**	**54.69**	**55.00**
		Std	6.84	**6.55**	**6.34**	**6.84**	**6.62**	**6.61**
		Gap(%)	–	**-0.76%**	**3.07%**	**4.95%**	**-0.33%**	**0.23%**
	High	Cost Value	56.89	56.63	58.61	59.03	56.90	57.41
		Std	7.31	7.17	7.19	7.33	6.99	7.14
		Gap(%)	–	-0.46%	3.04%	3.76%	0.03%	0.92%

(a) 2 disruption levels

Network Size	Disruption Rate	Algorithms	$DP(2,H)$	MDP	ADP_S	ADP_D	$CL_{(2,D,U)}$	$CL_{(3,D,U)}$
Small	Low	Cost Value	24.46	24.31	25.13	24.89	**24.56**	**24.57**
		Std	5.41	5.40	5.48	5.53	**5.40**	**5.40**
		Gap(%)	–	-0.61%	2.74%	1.79%	**0.41%**	**0.45%**
	High	Cost Value	27.38	27.33	28.01	27.76	27.46	27.63
		Std	5.71	5.67	5.79	5.81	5.59	5.74
		Gap(%)	–	-0.16%	2.32%	1.40%	0.31%	0.91%
Medium	Low	Cost Value	39.59	39.42	42.03	40.86	39.68	39.89
		Std	6.13	6.03	6.57	6.30	6.21	6.23
		Gap(%)	–	-0.45%	6.16%	3.19%	0.21%	0.74%
	High	Cost Value	41.24	41.19	42.89	42.23	41.35	41.65
		Std	6.28	6.17	6.31	6.33	6.41	6.48
		Gap(%)	–	-0.14%	3.99%	2.40%	0.25%	0.99%
Large	Low	Cost Value	53.23	N/A	56.28	56.30	**53.07**	**53.12**
		Std	6.22	N/A	6.15	6.11	**6.07**	**6.02**
		Gap(%)	–	N/A	5.73%	5.77%	**-0.31%**	**-0.20%**
	High	Cost Value	55.38	N/A	58.21	59.18	55.47	55.54
		Std	6.15	N/A	6.35	6.35	6.23	6.26
		Gap(%)	–	N/A	5.11%	6.87%	0.17%	0.29%

(b) 3 disruption levels

Network Size	Disruption Rate	Algorithms	$DP(2,H)$	MDP	ADP_S	ADP_D	$CL_{(2,D,U)}$	$CL_{(3,D,U)}$
Small	Low	Cost Value	26.72	26.62	**27.30**	**27.09**	**26.78**	**26.95**
		Std	5.60	5.57	**5.46**	**5.63**	**5.59**	**5.53**
		Gap(%)	–	-0.37%	**2.16%**	**1.40%**	**0.22%**	**0.86%**
	High	Cost Value	28.33	28.25	29.04	28.63	28.37	28.55
		Std	5.47	5.47	5.73	5.54	5.48	5.42
		Gap(%)	–	-0.27%	2.51%	1.06%	0.16%	0.79%
Medium	Low	Cost Value	40.20	N/A	43.05	42.97	40.68	40.87
		Std	7.54	N/A	7.74	7.86	7.77	7.90
		Gap(%)	–	N/A	7.09%	6.89%	1.19%	1.66%
	High	Cost Value	42.01	N/A	44.86	44.69	42.34	42.59
		Std	5.38	N/A	5.53	5.35	5.73	5.83
		Gap(%)	–	N/A	6.77%	6.36%	0.77%	1.38%
Large	Low	Cost Value	53.46	N/A	**56.85**	**57.73**	**53.33**	**54.06**
		Std	6.15	N/A	**6.37**	**6.44**	**6.36**	**6.80**
		Gap(%)	–	N/A	**6.34%**	**7.99%**	**-0.24%**	**1.12%**
	High	Cost Value	56.30	N/A	59.40	60.41	**56.15**	**56.90**
		Std	6.16	N/A	6.44	6.30	**6.32**	**6.32**
		Gap(%)	–	N/A	5.51%	7.31%	**-0.27%**	**1.06%**

(c) 5 disruption levels

Table 4.4 The estimated total cost value, standard deviation (Std) and percentage gap with respect to $DP(2, H)$ in different network sizes with high network vulnerability

Network Size	Disruption Rate	Algorithms	$DP(2,H)$	MDP	ADP_S	ADP_D	$CL_{(2,D,U)}$	$CL_{(3,D,U)}$
Small	Low	Cost Value	26.56	**26.38**	27.86	27.80	**26.65**	**26.66**
		Std	5.39	**5.38**	5.66	5.84	**5.41**	**5.42**
		Gap(%)	–	**-0.71%**	4.87%	4.63%	**0.33%**	**0.34%**
	High	Cost Value	30.90	30.81	31.16	30.82	30.81	30.90
		Std	5.51	5.83	5.44	5.41	5.57	5.63
		Gap(%)	–	-0.29%	0.84%	-0.24%	-0.29%	-0.01%
Medium	Low	Cost Value	40.38	40.28	42.57	42.39	40.62	40.64
		Std	6.34	6.32	6.40	6.48	6.34	6.34
		Gap(%)	–	-0.24%	5.44%	4.98%	0.61%	0.65%
	High	Cost Value	45.12	45.04	46.65	46.63	45.42	45.61
		Std	6.65	6.66	6.94	6.87	6.86	6.85
		Gap(%)	–	-0.16%	3.40%	3.34%	0.68%	1.09%
Large	Low	Cost Value	57.01	56.46	59.48	61.46	**56.87**	**56.90**
		Std	6.34	6.55	6.62	6.73	**6.53**	**6.46**
		Gap(%)	–	-0.97%	4.32%	7.80%	**-0.25%**	**-0.20%**
	High	Cost Value	59.97	59.63	62.33	63.57	59.52	60.22
		Std	7.70	7.17	7.55	7.49	7.08	7.18
		Gap(%)	–	-0.57%	3.94%	6.00%	-0.74%	0.42%

(a) 2 disruption levels

Network Size	Disruption Rate	Algorithms	$DP(2,H)$	MDP	ADP_S	ADP_D	$CL_{(2,D,U)}$	$CL_{(3,D,U)}$
Small	Low	Cost Value	**26.70**	**26.38**	**28.10**	**27.47**	**26.81**	26.91
		Std	**5.26**	**5.17**	**5.43**	**5.59**	**5.36**	5.28
		Gap(%)	**–**	**-1.22%**	**5.21%**	**2.85%**	**0.39%**	0.78%
	High	Cost Value	28.60	28.43	**28.94**	**28.74**	**28.39**	28.54
		Std	5.66	5.63	**5.75**	**5.79**	**5.79**	5.71
		Gap(%)	–	-0.59%	**1.18%**	**0.51%**	**-0.73%**	-0.20%
Medium	Low	Cost Value	40.30	N/A	43.15	42.14	40.51	40.62
		Std	6.20	N/A	6.29	6.29	6.21	6.23
		Gap(%)	–	N/A	7.07%	4.56%	0.51%	0.80%
	High	Cost Value	44.03	N/A	45.19	45.06	44.12	44.41
		Std	6.10	N/A	6.03	6.06	6.22	6.25
		Gap(%)	–	N/A	2.64%	2.34%	0.21%	0.87%
Large	Low	Cost Value	54.21	N/A	57.02	57.48	**53.91**	53.99
		Std	5.61	N/A	5.88	5.86	**5.83**	5.88
		Gap(%)	–	N/A	5.19%	6.04%	**-0.55%**	-0.40%
	High	Cost Value	**57.48**	N/A	60.21	60.84	**57.42**	57.63
		Std	**6.07**	N/A	6.48	7.00	**6.52**	6.67
		Gap(%)	**–**	N/A	4.75%	5.85%	**-0.11%**	0.26%

(b) 3 disruption levels

Network Size	Disruption Rate	Algorithms	$DP(2,H)$	ADP_S	ADP_D	$CL_{(2,D,U)}$	$CL_{(3,D,U)}$
Small	Low	Cost Value	28.12	30.00	29.91	**28.39**	**28.51**
		Std	5.95	6.33	6.42	**6.46**	**6.36**
		Gap(%)	–	6.67%	6.38%	**0.96%**	**1.38%**
	High	Cost Value	29.45	30.95	30.09	**29.69**	**29.87**
		Std	5.94	6.11	6.05	**6.14**	**6.19**
		Gap(%)	–	5.08%	2.18%	**0.82%**	**1.44%**
Medium	Low	Cost Value	41.18	43.85	43.22	41.41	41.63
		Std	6.56	6.88	6.72	6.67	6.84
		Gap(%)	–	6.48%	4.94%	0.55%	1.08%
	High	Cost Value	44.16	46.37	46.09	44.31	44.61
		Std	5.40	5.35	5.38	5.33	5.34
		Gap(%)	–	5.00%	4.37%	0.34%	1.02%
Large	Low	Cost Value	54.42	**56.49**	**56.92**	**54.03**	54.44
		Std	6.63	**7.35**	**7.28**	**6.79**	7.68
		Gap(%)	–	**3.81%**	**4.60%**	**-0.72%**	0.03%
	High	Cost	58.50	**59.63**	**60.37**	**58.44**	58.53
		Std	7.48	**8.58**	**8.82**	**8.02**	8.75
		Gap(%)	–	**1.94%**	**3.20%**	**-0.09%**	0.06%

(c) 5 disruption levels

In both low and high network vulnerability, as disruption rate increases, $CL_{(2,D,U)}$ performs significantly better than $CL_{(3,D,U)}$ at a higher rate. For example, in high vulnerability with 3 disruption levels and small network size, there is no significant difference between two algorithms (Appendix Table B1). However, as disruption rate becomes higher, $CL_{(2,D,U)}$ significantly outperforms $CL_{(3,D,U)}$ by 0.53% (Table 4.4).

Although, the average percentage improvement of $CL_{(2,D,U)}$ relative to $CL_{(2,D,U)}$ is relatively small (less than 2%), we observe that there are also test instances (with 3 and 5 disruption levels in small size, high disruption rate instance types) for which the percentage improvement increases up to 29%.

These results indicate that considering shorter horizon for the deterministic lookahead policy increases the performance of the hybrid ADP algorithms especially for higher number of disruption rates, larger network sizes and higher disruption levels. Thus, considering clusters of two-arc-ahead neighborhood provides higher quality solutions than considering three-arc-ahead neighborhood. In the following algorithm comparisons, we will focus on $CL_{(2,D,U)}$ as it gives consistently higher quality solutions than $CL_{(3,D,U)}$.

The hybrid ADP algorithm with clusters considering two-arc-ahead neighborhood ($CL_{(2,D,U)}$) versus the benchmark heuristic ($DP(2,H)$)

Tables 4.3 and 4.4 show that the hybrid ADP algorithm with clusters considering two-arc-ahead neighborhood ($CL_{(2,D,U)}$) performs within -0.73% and 0.96% away from the benchmark heuristic ($DP(2,H)$). As the network size becomes larger, the significant difference between $CL_{(2,D,U)}$ and $DP(2,H)$ decreases such that in large size networks the significant solution quality of $CL_{(2,D,U)}$ is higher than or equal to $DP(2,H)$ (Appendix Table B3). As vulnerability and network size increase, $CL_{(2,D,U)}$ outperforms $DP(2,H)$. For instance, in networks with low network vulnerability and low disruption rate, the performance of $CL_{(2,D,U)}$ relative to $DP(2,H)$ increases from small to large networks by the following gap percentages: 0.43% to -0.33%, 0.41% to -0.31% and 0.22% to -0.24% for 2, 3 and 5 disruption levels, respectively (Tables 4.3).

This is also true for high network vulnerability as shown in Table 4.4. As network vulnerability becomes higher, the performance of $CL_{(2,D,U)}$ relative to $DP(2,H)$ increases. For large and highly vulnerable networks, on average $CL_{(2,D,U)}$ outperforms $DP(2,H)$ for all disruption levels and disruption rates.

In the networks with high disruption rates, in general the performance gap between $CL_{(2,D,U)}$ and $DP(2,H)$ on general diminishes when compared to the networks with low disruption rates. For instance, in Table 4.3, in small networks with 2 disruption levels, $DP(2,H)$ outperforms $CL_{(2,D,U)}$ by 0.43% (with significant difference) in low disruption rate and by 0.21% (with no significant difference) in high disruption rate. This is because as disruption rate increases, the routing algorithms tend to make similar routing decisions considering more risk aversive links.

The hybrid ADP algorithm with clusters considering two-arc-ahead neighborhood ($CL_{(2,D,U)}$) versus the MDP

The comparison between $CL_{(2,D,U)}$ and MDP is limited to the instance types where we can compute the optimal policy within 1 hour. Tables 4.3-(a) and 4.4-(a) show that as disruption rate increases, the performance gap between $CL_{(2,D,U)}$ and MDP decreases. In high network vulnerability and small network size, the performance gap decreases from 1.04% (significant difference) to 0% (no significant difference) for low and high disruption rates, respectively (Tables 4.4-(a) and Appendix B1-(a)). On the other hand, as the network vulnerability becomes higher, the performance gap between $CL_{(2,D,U)}$ and MDP increases. For instance, in small networks with low disruption rate with 3 disruption levels, the performance gap increases from 1.02% to 1.61% for low and high network vulnerability, respectively (Tables 4.3-(b) versus Tables 4.4-(b)). Note that these percentages are computed from the percentage gap differences.

The hybrid ADP algorithm with clusters considering two-arc-ahead neighborhood ($CL_{(2,D,U)}$) versus the standard ADP algorithms

The paired samples t-test results show that with 95% confidence $CL_{(2,D,U)}$ performs statistically different than the standard ADP algorithms for all the test instances regardless of network size and vulnerability (Appendix Tables B1- B3). Tables 4.3 and 4.4 show that $CL_{(2,D,U)}$ outperforms the standard ADP algorithms in all of the network types regardless of network size, vulnerability, disruption rate and disruption level. This shows that the combination of deterministic lookahead policy with the value function approximation provides better updates of the value functions. This performance change is more significant when the number of disruption levels increases.

As the network size increases, the performance gap between $CL_{(2,D,U)}$ and standard ADP algorithms increases. For instance, Table 4.3-(a) shows that $CL_{(2,D,U)}$ outperforms ADP_S (ADP_D) by 2.69% (2.38%) (small network, low disruption rate) and 3.40% (5.28%) (large network, low disruption rate). The difference increases even more as the disruption level increases from $K = 2$ to $K = 5$. Table 4.3-(c) shows that $CL_{(2,D,U)}$ outperforms ADP_S (ADP_D) by 1.94% (1.18%) (small network, low disruption rate) and 6.60% (8.33%) (large network, low disruption rate). This result is also true for high network vulnerability (Table 4.4). The paired samples t-test results show that these differences are all significant with 0.05.

Tables 4.3 and 4.4 also indicate that in high disruption rate networks, on average the performance gap between the hybrid algorithms and the standard algorithms is lower when compared to low disruption rate networks. This is because the ADP algorithms find similar risk aversive routing policies as the vulnerable links become highly disrupted. For instance, Table 4.4-(a) shows that $CL_{(2,D,U)}$ outperforms ADP_S (ADP_D) by 5.20% (4.93%) (small network, low disruption rate) and 1.13% (0.05%, no significant difference) (small network, high disruption rate). The is also true for

3 and 5 disruption levels. Table 4.4-(c) shows that $CL_{(2,D,U)}$ outperforms ADP_S (ADP_D) by 4.13% (5.32%) (large network, low disruption rate) and 2.03% (3.29%) (large network, high disruption rate).

Computational Time

Figure 4.8 shows the computational times (in minutes) of the standard ADP algorithm (ADP_D), the hybrid ADP algorithm ($CL_{(2,D,U)}$) and the benchmark heuristic $DP(2,H)$ with respect to different network sizes with the given disruption level (considering both low and high vulnerability). When the network size increases from small to large with 2 disruption levels, Figure 4.8-(a) shows that the hybrid ADP algorithm is slower (still less than 0.1 minutes) than the standard ADP and $DP(2,H)$ due to search in clusters and update of the state values in the deterministic shortest path. As disruption level increases, the hybrid ADP mitigates the effect of the increase in the network size better than the standard ADP and $DP(2,H)$ with lower computational times. Figure 4.8-(c) shows that when the network size increases with higher disruption levels, the computational time for the hybrid ADP algorithm increases at a slower rate than the standard ADP and $DP(2,H)$.

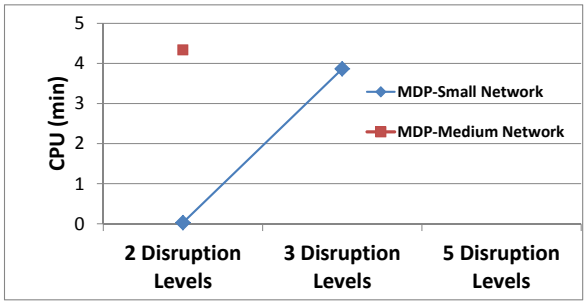
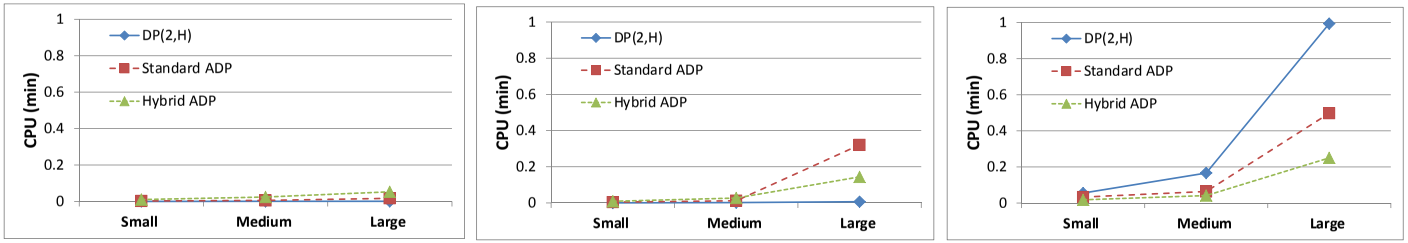


Figure 4.7 Computational time for MDP in instances with different disruption levels and network sizes (within CPU limit of 1 hour)

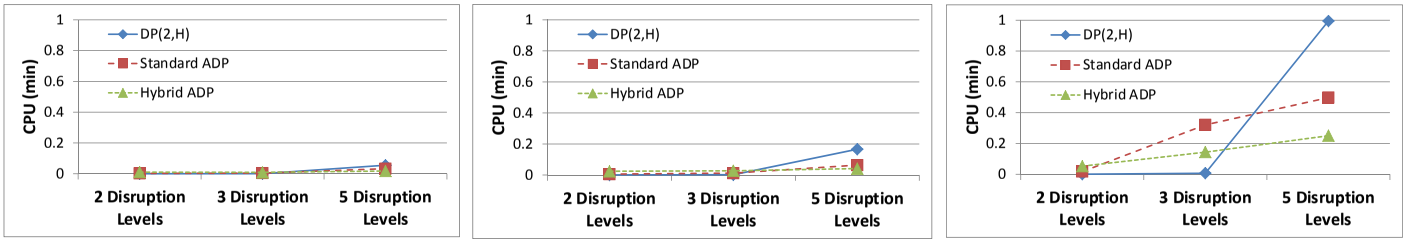
The computational time is also significantly affected from the number of disruption levels as the state- and outcome-space increases exponentially with the increase in disruption levels. For instance, Figure 4.7 shows the computational times (in minutes) of the MDP with respect to different disruption levels with the given network size where we can solve MDP in an hour (the non-existent data for medium network 3 and 5 disruption levels indicates that CPU is more than an hour). We observe that the computational time increases exponentially with the increase in disruption levels and network size. Figure 4.9 shows the computational times (in minutes) of the standard ADP algorithm (ADP_D), the hybrid ADP algorithm ($CL_{(2,D,U)}$) and the benchmark heuristic $DP(2,H)$ with respect to different disruption levels with the given network size. The figure illustrates that both the ADP algorithms and $DP(2,H)$ significantly reduces the computational time significantly. However, we

also observe that the hybrid ADP algorithm with the clustering approach's ($CL_{(2,D,U)}$) computational time has the lowest rate of increase as the number of disruption levels increases (Figure 4.9-(a) and (c)). The computational time of $DP(2,H)$ increases at a higher rate when the number of disruption levels increases. This is true for all network sizes. However, as the network size increases with the increase in disruption level the difference between the hybrid ADP and $DP(2,H)$ increases even more. This shows that the hybrid ADP with clustering approach mitigates the effect of state- and outcome-space explosion when compared to $DP(2,H)$. Although $DP(2,H)$ has relatively good solution quality as discussed in Section 4.4.4, for large scale networks with many disruption levels the hybrid ADP algorithms with the clustering approach becomes more attractive with lower computational time and high solution quality for practical implementation.



(a) 2 disruption levels (b) 3 disruption levels (c) 5 disruption levels

Figure 4.8 Computational time for different algorithms in instances with different network sizes



(a) Small network (b) Medium network (c) Large network

Figure 4.9 Computational time for different algorithms in instances with different disruption levels

4.5. Conclusion

In this paper, we consider dynamic shortest path problems with stochastic disruptions on a subset of arcs. Both historical and real-time information for the network are used in dynamic routing decisions. The disruption states of the following stages are dependent on the travel time on the current arc. We denote this property as travel-time-dependency. We model the problem as a discrete-time, finite Markov decision process (MDP).

MDP formulation provides a theoretical framework to find dynamic routing decisions for each decision epoch. However, for large scale networks with disruption levels, obtaining the optimal solution faces the curses of dimensionality, i.e., states, outcome, and decisions. Therefore, we use an Approximate Dynamic Programming (ADP) algorithm. First, we employ a standard value function approximation algorithm with various algorithmic design variations for updating the state values with efficient exploration/exploitation strategies. Then, we improve the value function approximation algorithm by developing a hybrid ADP with a deterministic lookahead policy and value function approximations using a clustering approach. We develop two types of hybrid ADP algorithms considering a shorter horizon (two-arc-ahead neighborhood) and a longer horizon (three-arc-ahead neighborhood) for the deterministic lookahead policy.

We develop a test bed of networks to evaluate the efficiency (both in computational time and solution quality) of our algorithms. The generated networks vary in network size, network vulnerability, number of disruption levels, and disruption rate on the vulnerable arc. We use a benchmark heuristic, a stochastic limited lookahead policy ($DP(2,H)$), shown to perform well in binary disruption levels as shown in Chapter 3. In our numerical analysis, we show that the hybrid ADP algorithms with the clustering approach significantly outperforms the standard ADP algorithms. The solution quality of hybrid ADP algorithms is higher than or equal to the solution quality of the benchmark heuristic when the network size gets large and the disruption level gets higher. The computational time of the hybrid ADP algorithms shows the slowest rate of increase with respect to the increase in network size and disruption level. The computational time of $DP(2,H)$ increases at a higher rate when the number of disruption levels increases. Although $DP(2,H)$ has relatively good solution quality, for large scale networks with many disruption levels the hybrid ADP algorithms become more attractive with lower computational time and high solution quality for practice.

This paper provides an exploratory study on solving the dynamic shortest path problems using hybrid ADP algorithms in large scale networks with many disruption levels. Furthermore, we provide an extensive analysis on the effect of using different algorithmic design variations of ADP algorithms for the problem on hand. Future research involves integrating the hybrid ADP algorithms for the dynamic shortest path problems into the stochastic vehicle routing problem with dynamic travel times.

4.A. Appendix

A Paired samples t-test results for ADP algorithmic design variations analysis

Table A1 Paired significance t-test for the difference between ADP algorithms with deterministic and $DP(2, H)$ initialization heuristics with different network sizes

	Small Network		**Medium Network**		**Large Network**	
	K=2	K=3	K=2	K=3	K=2	K=3
	p-value	p-value	p-value	p-value	p-value	p-value
ADP_S: Deterministic - DP(2,M)	0.00	**0.09**	**0.38**	0.00	0.00	0.00
ADP_D: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00
$CL_{(2,S,NU)}$: Deterministic - DP(2,M)	0.00	**0.21**	**0.52**	0.00	**0.42**	0.00
$CL_{(2,S,U)}$: Deterministic - DP(2,M)	0.00	**0.05**	0.00	0.00	**0.26**	0.00
$CL_{(3,S,NU)}$: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00
$CL_{(3,S,U)}$: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00
$CL_{(2,D,NU)}$: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00
$CL_{(2,D,U)}$: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00
$CL_{(3,D,NU)}$: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00
$CL_{(3,D,U)}$: Deterministic - DP(2,M)	0.00	0.00	0.00	0.00	0.00	0.00

Table A2 Paired significance t-test for the differences between ADP algorithms with single-pass and double-pass approach with different network sizes

	Small Network			**Medium Network**			**Large Network**		
	Mean	Std	p-value	Mean	Std	p-value	Mean	Std	p-value
$ADP_S - ADP_D$	0.16	0.06	0.01	-0.13	0.10	0.18	-1.17	0.18	0.00
$CL_{(2,S,NU)} - CL_{(2,D,NU)}$	0.21	0.03	0.00	0.24	0.04	0.00	0.05	0.09	0.63
$CL_{(2,S,U)} - CL_{(2,D,U)}$	0.37	0.05	0.00	0.84	0.08	0.00	0.43	0.07	0.00
$CL_{(3,S,NU)} - CL_{(3,D,NU)}$	0.58	0.06	0.00	0.59	0.05	0.00	0.23	0.08	0.00
$CL_{(3,S,U)} - CL_{(3,D,U)}$	1.18	0.09	0.00	1.64	0.11	0.00	0.56	0.10	0.00

(a) 2 disruption levels

	Small Network			**Medium Network**			**Large Network**		
	Mean	Std	p-value	Mean	Std	p-value	Mean	Std	p-value
$ADP_S - ADP_D$	0.33	0.06	0.00	0.74	0.10	0.00	-0.52	0.13	0.00
$CL_{(2,S,NU)} - CL_{(2,D,NU)}$	0.14	0.04	0.00	0.11	0.03	0.00	0.14	0.07	0.04
$CL_{(2,S,U)} - CL_{(2,D,U)}$	0.14	0.03	0.00	0.13	0.02	0.00	0.09	0.38	0.81
$CL_{(3,S,NU)} - CL_{(3,D,NU)}$	0.37	0.06	0.00	0.46	0.04	0.00	0.31	0.06	0.00
$CL_{(3,S,U)} - CL_{(3,D,U)}$	0.78	0.08	0.00	0.69	0.07	0.00	0.35	0.07	0.00

(a) 3 disruption levels

Table A3 Paired significance t-test for the differences between hybrid ADP algorithms with no-update and update approach with different network sizes

	Small Network			**Medium Network**			**Large Network**		
	Mean	Std	p-value	Mean	Std	p-value	Mean	Std	p-value
$CL_{(2,D,NU)} - CL_{(2,D,U)}$	0.05	0.02	0.01	0.14	0.02	0.00	0.36	0.06	0.00
$CL_{(3,D,NU)} - CL_{(3,D,U)}$	0.06	0.03	0.03	0.00	0.01	0.72	0.09	0.07	0.22

(a) 2 disruption levels

	Small Network			**Medium Network**			**Large Network**		
	Mean	Std	p-value	Mean	Std	p-value	Mean	Std	p-value
$CL_{(2,D,NU)} - CL_{(2,D,U)}$	0.04	0.02	0.02	0.15	0.03	0.00	0.14	0.04	0.00
$CL_{(3,D,NU)} - CL_{(3,D,U)}$	0.01	0.01	0.22	0.00	0.01	0.66	0.01	0.01	0.69

(b) 3 disruption levels

Table A4 Paired significance t-test for the differences between hybrid ADP algorithms with no-update and update approach with different network vulnerability

	Low Vulnerability			**High Vulnerability**		
	Mean	Std	p-value	Mean	Std	p-value
$CL_{(2,D,NU)} - CL_{(2,D,U)}$	0.13	0.02	0.00	0.17	0.03	0.00
$CL_{(3,D,NU)} - CL_{(3,D,U)}$	0.07	0.03	0.03	0.01	0.02	0.58

(a) 2 disruption levels

	Low Vulnerability			**High Vulnerability**		
	Mean	Std	p-value	Mean	Std	p-value
$CL_{(2,D,NU)} - CL_{(2,D,U)}$	0.03	0.02	0.20	0.10	0.02	0.00
$CL_{(3,D,NU)} - CL_{(3,D,U)}$	0.00	0.01	0.58	0.00	0.01	0.83

(a) 3 disruption levels

B Paired samples t-test results for comparison of algorithms

Table B1 95% Confidence interval (95% Lower bound (LCI), 95% Upper bound (UCI)) for the differences between the algorithms in small network instances

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.18	-0.04	0.00	-0.14	0.02	0.14	-0.14	-0.05	0.00	-0.16	0.37	0.41
$MDP - CL_{(2,D,U)}$	**-0.29**	**-0.07**	**0.00**	-0.19	-0.07	0.00	**-0.38**	**-0.18**	**0.00**	**-0.21**	**0.21**	**1.00**
$ADP_S - CL_{(2,D,U)}$	**0.47**	**0.90**	**0.00**	0.33	0.87	0.00	**0.94**	**1.47**	**0.00**	**0.02**	**0.68**	**0.04**
$ADP_D - CL_{(2,D,U)}$	**0.35**	**0.86**	**0.00**	0.22	0.62	0.00	**0.86**	**1.42**	**0.00**	**-0.19**	**0.21**	**0.89**
$CL_{(3,D,U)} - CL_{(2,D,U)}$	**-0.06**	**0.06**	**0.89**	0.03	0.24	0.01	-0.05	0.05	1.00	-0.01	0.18	0.07

(a) 2 disruption levels

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.17	-0.03	0.01	-0.14	-0.04	0.00	-0.25	0.04	0.16	0.05	0.36	0.01
$ADP_S - CL_{(2,D,U)}$	0.38	0.76	0.00	0.26	0.84	0.00	0.94	1.64	0.00	0.36	0.73	0.00
$ADP_D - CL_{(2,D,U)}$	0.14	0.53	0.00	0.12	0.47	0.00	0.42	0.89	0.00	0.21	0.49	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	-0.04	0.06	0.65	-0.05	0.37	0.12	**-0.02**	**0.23**	**0.10**	**0.05**	**0.25**	**0.00**

(b) 3 disruption levels

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.20	0.08	0.39	-0.13	-0.02	0.00	-0.07	0.62	0.00	-0.39	-0.10	0.00
$ADP_S - CL_{(2,D,U)}$	**0.31**	**0.73**	**0.00**	0.33	1.00	0.00	1.42	1.81	0.00	0.83	1.68	0.00
$ADP_D - CL_{(2,D,U)}$	**0.20**	**0.43**	**0.00**	0.09	0.42	0.00	1.32	1.73	0.00	0.22	0.58	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	0.03	0.31	0.01	0.05	0.31	0.01	0.05	0.19	0.00	0.09	0.27	0.00

(c) 5 disruption levels

Table B2 95% Confidence interval (95% Lower bound (LCI), 95% Upper bound (UCI)) for the differences between the algorithms in medium network instances

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.21	-0.05	0.00	-0.20	-0.05	0.00	-0.31	-0.19	0.00	-0.44	-0.17	0.00
$MDP - CL_{(2,D,U)}$	**-0.28**	**-0.11**	**0.00**	-0.25	-0.09	0.00	-0.42	-0.27	0.00	-0.51	-0.25	0.00
$ADP_S - CL_{(2,D,U)}$	1.20	1.72	0.00	-0.45	0.85	0.01	1.68	2.22	0.00	0.98	1.47	0.00
$ADP_D - CL_{(2,D,U)}$	1.10	1.69	0.00	0.77	1.25	0.00	1.51	2.02	0.00	0.98	1.42	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	-0.03	0.13	0.21	0.06	0.23	0.00	-0.06	0.09	0.69	0.05	0.33	0.01

(a) 2 disruption levels

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.21	0.04	0.17	-0.22	0.01	0.07	-0.40	-0.01	0.04	-0.33	0.14	0.43
$ADP_S - CL_{(2,D,U)}$	1.82	2.89	0.00	1.12	1.96	0.00	2.19	3.09	0.00	0.66	1.48	0.00
$ADP_D - CL_{(2,D,U)}$	0.91	1.45	0.00	0.70	1.08	0.00	1.39	1.88	0.00	0.70	1.18	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	0.08	0.34	0.00	0.18	0.43	0.00	-0.03	0.26	0.11	0.18	0.40	0.00

(b) 3 disruption levels

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.67	-0.30	0.00	-0.45	-0.20	0.00	-0.48	-0.06	0.02	-0.49	-0.11	0.00
$ADP_S - CL_{(2,D,U)}$	1.71	2.72	0.00	2.18	2.87	0.00	1.79	3.09	0.00	2.52	3.61	0.00
$ADP_D - CL_{(2,D,U)}$	1.84	2.62	0.00	1.92	2.78	0.00	1.43	2.18	0.00	1.42	2.15	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	0.01	0.35	0.04	0.19	0.34	0.02	0.05	0.39	0.02	0.10	0.50	0.01

(c) 5 disruption levels

Table B3 95% Confidence interval (95% Lower bound (LCI), 95% Upper bound (UCI)) for the differences between the algorithms in large network instances

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.19	0.56	0.33	-0.36	0.33	0.93	-0.34	0.49	0.55	-0.14	1.02	0.13
$ADP_S - CL_{(2,D,U)}$	**1.33**	**2.40**	**0.00**	1.17	2.25	0.00	2.20	3.02	0.00	2.35	3.26	0.00
$ADP_D - CL_{(2,D,U)}$	**2.11**	**3.69**	**0.00**	1.54	2.72	0.00	3.91	5.27	0.00	3.43	4.66	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	**0.09**	**0.54**	**0.01**	0.19	0.82	0.00	-0.29	0.21	0.77	0.42	0.98	0.00

(a) 2 disruption levels

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.20	0.53	0.37	-0.31	0.12	0.11	-0.07	0.68	0.30	-0.73	0.87	0.00
$ADP_S - CL_{(2,D,U)}$	2.77	3.66	0.00	2.34	3.14	0.00	2.58	3.66	0.00	2.35	3.24	0.00
$ADP_D - CL_{(2,D,U)}$	2.66	3.80	0.00	3.17	4.25	0.00	3.06	4.09	0.00	2.91	3.94	0.00
$CL_{(3,D,U)} - CL_{(2,D,U)}$	-0.07	0.19	0.36	-0.06	0.20	0.27	-0.28	0.13	0.43	-0.08	0.51	0.15

(b) 3 disruption levels

	Low Vulnerability						High Vulnerability					
	Low Disruption Rate			High Disruption Rate			Low Disruption Rate			High Disruption Rate		
	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value	95% LCI	95% UCI	p-value
$DP(2,H) - CL_{(2,D,U)}$	-0.25	0.51	0.51	-0.56	0.86	0.02	-0.38	1.17	0.65	-1.19	1.32	0.01
$ADP_S - CL_{(2,D,U)}$	**2.76**	**4.27**	**0.00**	2.71	3.81	0.00	**1.71**	**3.22**	**0.00**	**0.83**	**1.55**	**0.00**
$ADP_D - CL_{(2,D,U)}$	**3.63**	**5.17**	**0.00**	3.72	4.82	0.00	2.09	**3.69**	**0.00**	**1.38**	**2.47**	**0.00**
$CL_{(3,D,U)} - CL_{(2,D,U)}$	0.09	1.37	0.03	0.55	0.95	0.02	0.72	2.13	0.00	-0.04	0.22	0.16

(c) 5 disruption levels

Chapter 5

Influence of Spillback Effect on DSPP with Network Disruptions

In Chapters 3 and 4, we focused on dynamic shortest path problems assuming that the disruptions in the neighboring arcs do not propagate. Though, in every day life, due to the limited capacity of the roads, we observe that when the road in front of us (downstream road) is blocked, the queue propagates backwards to the road we are travelling (upstream road). In traffic engineering this is called as spillback effect. In this chapter, we investigate the impact of considering or ignoring the spillback phenomenon on the quality of routing decisions. We consider the optimal, the online, the offline and two-arcs-lookahead hybrid routing approach used in Chapter 1 with the spillback effect.

5.1. Introduction

In traffic networks, travelers experience disruptions due to accidents, road closures and road bottlenecks. These dynamic disruptions in traffic networks cause a drastic increase in travel times and decrease the probability of being on-time at the destination as stated in Chapter 1 and 2. Dynamic routing algorithms are emerging to reduce the impact of these disruptions by taking into account the stochastic and dynamic nature of the travel times. However, in case of a disruption, an efficient routing model also requires a good description of the congestion dynamics. One of these dynamics is the spillback effect, a congestion propagation to an upstream road. When there is a congestion at a route, due to the limited capacity, the queue at this route spills back to the upstream roads in time. In this phenomenon, a queue on a

downstream road affects the possible output rate of the upstream roads.

In the literature, several studies have been done to analyze the impact of spillback effect on travel time variability (Gentile et al. 2007, Knoop et al. 2007, Knoop 2009, Osorio et al. 2011). These studies show that the travel time distributions and the patterns obtained with and without modeling the spillback phenomenon are considerably different, especially when the congestion level is high. For instance, Knoop (2009) shows that when spillback effects are not considered, the delay increases significantly. The high impact of the spillback phenomenon in highly disrupted networks provides an opportunity to analyze the effect of the spillback in dynamic routing decisions.

Furthermore, several studies focus on the effect of spillback modeling in adaptive routing. Knoop et al. (2008) investigates the value of considering the spillback information in fixed route choice and adaptive route choice models via a simulation model. They show that the effect of not considering the spillback information is higher in fixed route choice models. In adaptive routing as the drivers choose alternative routes, the spillback effect decreases due to lower densities at downstream arcs. In Huang and Gao (2012) and Huang (2012) the effect of spillback is considered implicitly by modeling travel times as a multivariate normal distribution with random coefficient of correlations. In these studies, it is shown that the correlations between arc travel times decrease with temporal and spatial distances. When arc dependency is not taken into account, travelers underestimate the risk of vulnerable arcs and hence delays increase.

In this chapter, we analyze the influence of considering or ignoring the spillback effect on the quality of routing decisions for dynamic shortest path problems. For this, we model the dynamic shortest path problem with travel-time-dependent stochastic disruptions (see Chapter 2) as a discrete-time, finite Markov Decision Process (MDP). We incorporate the spillback effect into the MDP formulation. To analyze the effect of the spillback, we also formulate the MDP ignoring the spillback effect. Comparing both results shows how much delay can be avoided by using the spillback information.

We model the spillback effect by using the simplified Kinematic Wave Model (KWM) (Lighthill and Whitham 1955, Newell 1993)). We find an approximate value for the shockwave speed which is the rate of decrease in speed of the vehicles in the upstream roads. Then, we integrate the rate of decrease in speed to the state transition probability functions in the MDP formulation. We choose to use KWM as the queue dynamics are realistic and the computation is efficient (Knoop 2009).

This chapter analyzes the effect of considering or ignoring the spillback information also for different routing algorithms. First, we provide the optimal algorithms considering the spillback model or ignoring it. However, for increasing network size and disruption levels, the computational complexity increases causing the MDP to become computationally intractable. Therefore, to reduce the state-space, we use a hybrid dynamic programming algorithm using the detailed disruption and spillback

information for a limited part of the network only (Chapter 3). We also provide online and static routing algorithms mostly used in practice. We use a test bed of different network structures with different levels of disruption rates and spillback rates. Then, we compare the solution quality and the computational time of different algorithms in case of the spillback effect in the network.

The main contributions of this chapter are as follows:

1. We model the dynamic shortest path problem as a discrete-time, finite Markov Decision Process. We model the spillback effect by using the well known Kinematic Wave Model. We explicitly integrate the spillback effect into the MDP formulation by modifying the state transition function. Using this framework, we model a stochastic dynamic programming approach using dynamic disruption and spillback information for a limited part of the network. We also model an online and an offline routing algorithm that are mostly used in practice.
2. We analyze the value of considering the spillback information on the dynamic routing decisions by comparing the algorithms considering the spillback effect and ignoring the spillback effect. Numerical results show that considering the spillback effect in the dynamic routing decisions significantly improves the solution quality for the networks with higher number of vulnerable arcs. Moreover, the hybrid dynamic programming approach with the disruption and the spillback information for the limited part of the network, significantly reduces the computational time while providing on average significantly higher solution quality than the full information model that ignores the spillback effect.

The structure of the chapter is as follows. Section 2 provides the details of the spillback model and Section 3 describes our modeling framework based on dynamic programming. Section 4 discusses the offline, online and dynamic programming algorithms within this framework in detail. In Section 5, the experimental design, the numerical results and the important insights are discussed. In Section 6, we conclude the chapter by providing an overview of the results and future directions.

5.2. Modeling the Spillback Effect

In disrupted networks with limited capacities, spillbacks occur when the growing queues at the downstream arcs block the departures from the upstream arc. Spillback is the phenomenon that a queue on a downstream arc affects the possible output volume of the upstream arc or arcs connected to it. In the traffic theory literature, several analytical models are used to analyze the spillback dynamics: finite capacity queueing models (Jain and Smith 1997, Van Woensel and Vandaele 2007, Osorio and Bierlaire 2009, Osorio et al. 2011) and kinematic wave model (Ziliaskopoulos 2000, Skabardonis and Geroliminis 2005, Gentile et al. 2007, Knoop et al. 2007, Hoogendoorn et al. 2008, Knoop et al. 2008, Knoop 2009).

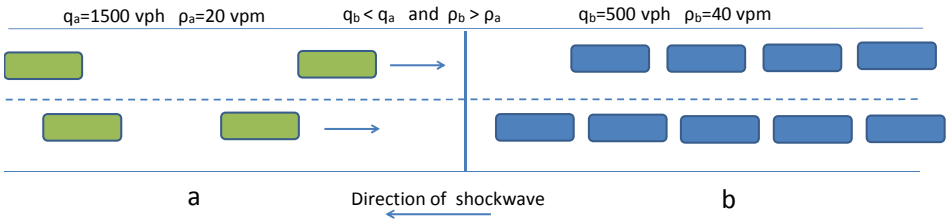


Figure 5.1 Shockwave in traffic (q_a and q_b represent flow of vehicles per hour (vph) on roads "a" and "b"; q_a and q_b represent density of vehicles per metersquare (vpm))

In this chapter, we choose to use the kinematic wave model (KWM) as it is widely accepted in the literature for modeling the spillback effect where the evolution of queues are modeled in a realistic manner and it is computationally efficient and simple (Knoop et al. 2007, Knoop 2009, Qian et al. 2012). Furthermore, we formulate the problem in terms of a Markov Decision Process (MDP) where we use average travel times as inputs to the model. So, KWM is more suitable for our model as it is a first-order-model using average traffic variables such as average flow and average speed on a road.

KWM is based on a wave phenomenon where it is applied to highway traffic flows (Lighthill and Whitham 1955, Newell 1993). Daganzo (1994) and Daganzo (1995) also explain the kinematic wave theory for traffic flow as a discretized cell transmission model which can be used to predict the evolution of traffic over time and space. The theory states that the traffic states are separated by boundaries. The speed at which these boundaries propagate can be computed using the shock wave speed. The shock wave speed is the rate of change in the speed of vehicles at the upstream arcs. Figure 5.1 illustrates that at the downstream of the road ("part b") there is a higher density of vehicles due to a disruption. Due to the limited capacity of the road and the outflow of the upstream road ("part a"), the density will move to the upstream road creating a shockwave between the boundary of the roads. This shockwave speed moves in the opposite direction of the traffic slowing down the vehicles at the upstream. We consider this type of spillback effect where the shockwave occurs between two arcs.

To use the shock wave speed theory, it is assumed that the number of vehicles is conserved and the flow of vehicles depends on the density in an arc. In this thesis, we consider the shockwave speed depending on the realizations of flow and density variables observed at stage k where we make routing decisions. The rate of change in speed, ω_k at stage k, is formulated by the ratio of the difference between the flows of vehicles at these arcs to the difference between the densities of the two arcs given in Equation (5.1):

$$\omega_k = (q_{dk} - q_{uk})/(\rho_{dk} - \rho_{uk}). \tag{5.1}$$

q_{ik} is the vehicle flow at arc i observed at stage k (where u is the upstream arc and d is the downstream arc) and ρ_{ik} is the density of arc i observed at stage k. We only make routing decisions decisions at the nodes and we approximate this equation using only the travel time and the distance, which are the inputs to the MDP formulation. For simplicity, we use first-order-models based on average values and assume that the flow of vehicles at non-congested state is the same for all arcs. Considering the fundamentals of traffic theory (Daganzo 2006), we use the equation below: (v_{ik}: speed at arc i at stage k, v_{fi}: free flow speed at arc i, L_i: length of the arc i, t_{ik}: the observed travel time to travel arc i at stage k and t_{fi}: travel time during the free flow speed at arc i)

$$\rho_{ik} = \frac{q_{ik}}{v_{ik}}. \tag{5.2}$$

When we replace variable ρ in Equation (5.1) with the Equation (5.2), we obtain the rate of change in the speed of the upstream arc as:

$$\omega_k = \frac{(q_{dk} - q_{uk})}{(\frac{q_{dk}}{v_{dk}} - \frac{q_{uk}}{v_{uk}})}. \tag{5.3}$$

The above formulation can be easily calculated with traffic data. However, in this chapter the input to the model is the average travel time or average speeds for each arc. Thus, we investigate relating the traffic flow to the average speed and to the average travel time. It is intuitive that increasing congestion on an arc results in a decrease in the average speed on that arc. When we consider a congestion model using a discrete Markov process with an arc capacity, the rate of change in the traffic flow on an arc is defined as the ratio between the speed of the arc at that stage and the free flow speed: $\frac{v_{ik}}{v_{fi}}$ (Jain and Smith 1997, Wang et al. 2010).

By using the fundamentals of the traffic theory (Daganzo 1994, Rakha and Zhang 2005)), the average travel time observed at stage k for an arc i during the congestion can be computed as the ratio of the length of the arc, L_i and the average speed during traveling on the arc, v_{ik}:

$$v_{ik} = \frac{L_i}{t_{ik}}. \tag{5.4}$$

Using these equations, Equation (5.3) is approximated to: (t_{fd} and t_{fu}: travel time during the free flow speed at the downstream and the upstream arc respectively; L_d: length of the downstream arc)

$$\omega_k \approxeq \frac{(\frac{t_{fd}}{t_{dk}} - \frac{t_{fu}}{t_{uk}}) * aL_d}{(a * t_{fd} - t_{fu})}. \tag{5.5}$$

"a" is the ratio of the distance of the upstream to the downstream arc ($\frac{L_u}{L_d}$). So when $a \geq 1$, the spillback effect is higher at the upstream arc because the queue at the shorter and congested downstream arc spill to the upstream arc at a higher rate.

The rate of change in travel time for the upstream arc (γ_{rk}) is then the ratio of the travel time observed at stage k to the new travel time with modified speed due to the spillback effect becomes:

$$\gamma_{rk} \approxeq \frac{t_{dk} * (a * t_{fd} - t_{fu})}{t_{dk} * (a * t_{fd} - t_{fu}) + (t_{fd} * t_{uk} - t_{fu} * t_{dk})}. \tag{5.6}$$

Osorio et al. (2011) argues that KWM captures average deterministic traffic conditions. Our problem is stochastic and transient. So, in our model, we use KWM to determine the shock wave speed for each disruption state. Then, we incorporate the rate of change in travel time due to the spillback to our probability transition matrix in the MDP formulation. In this way, we modify the spillback model such that it is state dependent and transient.

To model the arc dependency and the approximate effect of propagation from downstream arcs, we use a linear relationship between the one-step-transition function for the upstream arcs at stage k. We consider spillback phenomenon as an effect that increases the probability of having disruptions. Because of this, we incorporate the spillback effect in the probability transition function.

We assume that the downstream arcs, affecting the travel time of an upstream arc, i.e. r, are limited to the arcs that are two-arcs ahead of arc r which is denoted by the neighborhood vector: $Zone(r)$. Note that in Huang (2012), it is shown that the correlations between arc travel times decrease with temporal and spatial distances.

We also define a constant α to control the rate of the propagation which can change from network to network. We calculate the ratio of the increase in unit-time-transition probability for the vulnerable arc r, (β_{rk}) by rate of the change in the travel time of upstream arc calculated using Equation (5.7) (if there is a decrease in speed, otherwise $\beta_{rk} = 0$):

$$\beta_{rk} = max(0, \gamma_{rk}). \tag{5.7}$$

Let $p^r_{u,u'}(k)$ denote the one-stage disruption transition probability between any two disruption levels for vulnerable arc r, $p^r_{u,u'}(k) = P\{\hat{D}_{k+1}(r) = u' | \hat{D}_k(r) = u\}$.

Transition rate to higher disruption scenarios is modeled as:
$\lambda^r_{u,u'}(k) = p^r_{u,u'}(k)(1 + \alpha * \sum_{r' \in Zone(r)} \beta_{r'k})$.
Repair rate to lower disruption scenarios is modeled as:
$\mu^r_{u,u'}(k) = p^r_{u,u'}(k)/(1 + \alpha * \sum_{r' \in Zone(r)} \beta_{r'k})$.

The Markov Process is uniformizable if there exists a constant, δ, such that for disruption state u given the time at stage k, we have $(\sum_{u';u'>u} \lambda^r_{u,u'}(k) + \sum_{u';u'<u} \mu^r_{u,u'}(k)) \leq \delta$. The uniformization constant is chosen as follows: in each disruption state u, we choose δ such that it is equal to the maximum jump rate out of disruption state u. For each disruption state u where the total jump rate is smaller than δ, we add dummy transitions such that the rates out of a disruption state sums

up to δ (Puterman 2009, Blok 2011). This process has the expected transition times as $1/\delta$ for all u'. Then we can define the related discrete-time process with transition probabilities for going into disruption: $p'^{r}_{u,u'}(k)$:

$$p'^{r}_{u,u'}(k) = \begin{cases} \lambda^{r}_{u,u'}(k)/\delta, & \text{if } u' > u, \\ \mu^{r}_{u,u'}(k)/\delta, & \text{if } u' < u, \\ 1 - (\sum_{u';u'>u} \lambda^{r}_{u,u'}(k) + \sum_{u';u'<u} \mu^{r}_{u,u'}(k))/\delta, & \text{otherwise.} \end{cases} \tag{5.8}$$

5.3. Model Formulation- Markov Decision Process

We model the Dynamic Shortest Path Problem with the disruptions in the network as a discrete-time, finite Markov Decision Process (MDP). Consider a traffic network represented by the directed graph $G(N, A, A_v)$ where the set N represents nodes (or intersection of roads), A the set of directed arcs (arcs) and A_v the set of vulnerable arcs, potentially in disruption ($A_v \subseteq A$). The number of vulnerable arcs is R: $R = |A_v|$. Each of these vulnerable arcs has a known probability of going into a disruption and a known probability of recovery from the disruption.

In this paper, we assume that the travel times are affected by spillback phenomena for the travelers who are already traveling on the arc. The spillback effect is caused by the backward propagation of the congestion at the downstream arcs towards the upstream arcs due to limited capacity of the arcs. We model the spillback effect by the Kinematic Wave Theory and we integrate the spillback effect into the transition function of the MDP.

The travel time on an arc follows a discrete distribution (predictable from the historical data) given the disruption level at each arc. We assume that the actual travel time for the current arc can change due the spillback effect from the downstream arcs. We only know after traveling which actual travel time has been realized. We learn the information about the disruption status for all the vulnerable arcs when we arrive at a node. At each node, we make a decision to which node to travel next. Our objective is to travel from the initial node to the destination node with the minimum total expected travel time. We derive the optimal routing decision by the MDP formulation with finite number of stages where stage k represents the number of nodes that have been visited so far from the start node. The end of horizon, i.e. K, is reached by arriving to the destination node. The state including the destination node is the goal state which is absorbing and zero cost. All the other states have positive cost and free of dead locks. The total number of stages, i.e. K, is unknown a priori but finite as the process terminates by arriving at the goal state.

In this section, we formulate the problem as an MDP.

States

The state of the system at stage k, S_k, is represented by two components:

- The current node at stage k, $i_k \in N$.
- The disruption status information at stage k is denoted by $\hat{D}_k$ which gives the disruption level realizations for all vulnerable arcs. Each vulnerable arc can take any value from the disruption level vector U^r. For each arc there can be M_r different types of disruption levels: $\hat{D}_k(r) \in U^r$: $U^r = \{u^1, u^2, ...u^{M_r}\}$. Note that at each stage, we use a realization of the disruption vector, $\hat{D}_k$ and D_k represents the random variable for disruption vector.

The state of the system at stage k is then: $S_k = (i_k, \hat{D}_k)$. The final state, S_K, is reached by arriving to the destination node.

Actions

Our action, x_k, is the next node to travel given the state S_k. We note that each action x_k is an element of the set of all possible actions $\mathcal{X}(S_k)$ which is the neighbor set of the current node. So, the node in the next stage is actually the action decided at the previous state:

$$x_k = i_{k+1}. \tag{5.9}$$

The Exogenous Information

The disruption status of the vulnerable arcs change as we proceed to the next stage. The exogenous information consists of the realization of the disruption status of all the vulnerable arcs. Let $\hat{D}_{k+1}$ denote the disruption status realization that becomes known when stage $k+1$ is reached:

$$W_{k+1} = \hat{D}_{k+1}. \tag{5.10}$$

The State Transition Function

At stage k, if the system is in state S_k, we make a decision x_k and then observe the new exogenous information W_{k+1}. The system transition occurs to a new state S_{k+1} according to the transition function: S_{k+1}= $S^M(S_k, x_k, W_{k+1})$
The state transition involves the following transition functions:

$$i_{k+1} = x_k, \tag{5.11}$$

$$D_{k+1} = \hat{D}_{k+1}. \tag{5.12}$$

The Cost Function

The cost of traveling from current node at stage k, i.e. i_k, to the next node, x_k, given $\hat{D}_k$ is denoted by $C(S_k, x_k) = t_{i_k,x_k}(\hat{D}_k)$. In traffic, in general the travel time on the

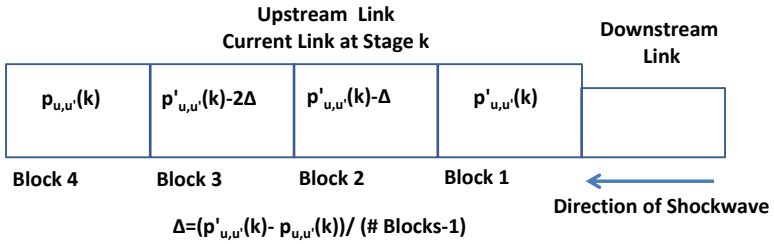


Figure 5.2 The cost function considering the spillback effect on the current arc

current arc changes due to the spillback effect. The sections of the current arc located at different distances from the downstream arc are affected from the spillback at a different rate. To model this, we divide the current arc in blocks of time units. So, that for each block, we compute the relative spillback effect and related increase in travel time. In our model, changes in travel time depend on how the disruption at the vulnerable arcs in $Zone(r)$ spills back to the current arc (Figure 5.2). We model this behavior as follows: we divide the current arc into B blocks. Then, we compute the rate of increase in unit transition probabilities after considering the spillback effect. For each block, we add the weighted effect of the spillback such that the spillback effect increases as the block is located nearer to the congested downstream road. As the number of blocks increases, the estimated travel time will be more accurate as the travel time increase will be much slower through the end of the upstream link.

Let $p'^{r}_{u,u'}(k)$ denote the unit-time-transition probability between any two disruption levels for vulnerable arc r after considering the spillback effect from the downstream arcs at its zone. The immediate traveling cost is also dependent on the disruption status of its zone due to the spillback effect:
$C(S_k, x_k) = C(S_k, x_k|\hat{D}_k(Zone(r)))$.

The cost function becomes:

$$C(S_k, x_k|\hat{D}_k(Zone(r))) = \sum_{j=1}^{B} \sum_{u'=1}^{u'=M_r} (p'^{r}_{u,u'}(k) - (j-1) * \Delta) * t_{i_k,x_k}(\hat{D}_{k+1}(r) = u')/B, \tag{5.13}$$

$$\Delta = (p'^{r}_{u,u'}(k) - p^{r}_{u,u'}(k))/(B-1). \tag{5.14}$$

In this way, we decrease gradually the transition probability considering the spillback effect through the beginning of the upstream arc. In these equations the Δ is the difference between unit transition probability for the arc r considering the spillback effect ($p'^{r}_{u,u'}(k)$) and not considering at all ($p^{r}_{u,u'}(k)$) relative to the number of blocks.

As the block is near to the downstream arc, we multiply the travel time with a higher transition probability which is controlled by Δ. The block very next to the downstream arc (B=1) has the transition probability considering the spillback rate ($p'^{r}_{u,u'}(k)$). The next block has a lower transition rate by $p'^{r}_{u,u'}(k) - \Delta$. This goes until the last block (beginning of the upstream arc) has the transition probability $p^{r}_{u,u'}$. Then, we calculate the expected travel time for each block. Note that the travel time on each block is computed by dividing the total travel time on the arc to the total number of blocks given the disruption realization.

For simplification, the rate of travel time increase in the blocks are only calculated depending on the disruption state of the downstream at stage k. Here, we do not calculate what will happen to the downstream arcs when we reach the next block in the current arc.

The Transition Probability for Travel-Time Dependency

The disruption status vector transits from $\hat{D}_k$ to $\hat{D}_{k+1}$ according to a Markovian transition matrix. We define the transition matrix for a vulnerable arc r from stage k to $k+1$ as $\Theta^r(k|S_k, x_k, \hat{D}_k(Zone(r)))$. This transition matrix is travel-time-dependent: the probability of being in the disruption status of the next stage D_{k+1}, depends on the travel time between the current node at stage k and the next node at stage $k+1$ given the disruption status at stage k and the ones in the relevant zone. As its is discussed in the cost function, we include the effect of spillback on the current arc. The travel time for the current arc after calculating the effect of spillback is denoted as $C(S_k, x_k|\hat{D}_k(Zone(r)))$.

$p'^{r}_{u,u'}(k)$ denotes the one-stage disruption transition probability between any two disruption levels for vulnerable arc r after considering the spillback effect:
$p'^{r}_{u,u'}(k) = P\{\hat{D}_{k+1}(r) = u' | \hat{D}_k(r) = u\}$.

Let $\Phi^r(k|S_k, x_k, \hat{D}_k(Zone(r)))$ represent a unit-time transition matrix considering the spillback effect.

$$\Phi^r(k|S_k, x_k, \hat{D}_k(Zone(r))) = \begin{bmatrix} p'^{r}_{u^1,u'^1}(k) & . & p'^{r}_{u^1,u'^{M^r}}(k) \\ p'^{r}_{u^2,u'^1}(k) & . & p'^{r}_{u^2,u'^{M^r}}(k) \\ & . & \\ p'^{r}_{u^{M^r},u'^1}(k) & . & p'^{r}_{u^{M^r},u'^{M^r}}(k) \end{bmatrix} \tag{5.15}$$

$\Theta^r(k|S_k, x_k, \hat{D}_k(Zone(r)))$ is the transition matrix for a vulnerable arc r from a disruption level realization, $\hat{D}_k(r) = u$, to any disruption realization at stage $(k+1)$, $\hat{D}_{k+1}(r)$:

$$\Theta^{r}(k|S_k, x_k, \hat{D}_k(Zone(r))) = \sum_{t=1}^{t=t_{max}} P(t_{i_k,x_k}(\hat{D}_k|\hat{D}_k(Zone(r)) = t)\Phi^r(k|S_k, x_k, \hat{D}_k(Zone(r)))^t. \tag{5.16}$$

Here t_{max} is determined by the maximum possible travel time of an arc given the disruption state and the spillback effect from its zone. Note that t is an integer.

The probability of having the new state S_{k+1} given S_k is then calculated as ($\Theta^r_{u,u'}(k|S_k, x_k, \hat{D}_k(Zone(r)))$ indicates the specific row and column index in the matrix $\Theta^r(k|S_k, x_k, \hat{D}_k(Zone(r)))$):

$$P(S_{k+1}|S_k) = \prod_{r=1}^{R} \Theta^r_{u,u'}(k|S_k, x_k, \hat{D}_k(Zone(r))). \tag{5.17}$$

The Objective Function

The objective is to minimize the expected total travel time from the initial state until the final state:

$$\min_{\pi \in \Pi} \mathbb{E} \quad \sum_{k=1}^{K} C(S_k, \mathcal{X}^\pi(S_k)), \tag{5.18}$$

where π denotes a policy or decision rule and Π denotes the set of all policies.

Bellman Equations

The optimization problem in Equation (5.18) can be solved using the Bellman Equations:

$$V_k(S_k) = \min_{x_k \in \mathcal{X}_k} C(S_k, x_k|\hat{D}_k(Zone(r))) + \sum_{S_{k+1}} P(S_{k+1}|S_k) V_{k+1}(S_{k+1}), \tag{5.19a}$$

$$V_K(S_K) = 0. \tag{5.19b}$$

The solution to the MDP formulation gives the optimal solution (Opt_S). We refer the reader the optimal algorithm used in Chapter 3.

To investigate the effect of not using spillback in our routing decisions, we also solve the MDP without including the spillback effect. The optimal solution without considering the effect of the spillback is denoted as Opt_{NS}. The cost function becomes: $C(S_k, x_k) = t_{i,x_k}(\hat{D}_k)$. Furthermore, we replace the transition matrix for the vulnerable arc r in Equation (5.16), as:

$$\Theta^r(k|S_k, x_k) = \begin{bmatrix} p^r_{u^1,u^1} & p^r_{u^1,u^2} & \cdots & p^r_{u^1,u^{M_r}} \\ p^r_{u^2,u^1} & p^r_{u^2,u^2} & \cdots & p^r_{u^2,u^{M_r}} \\ \cdot & & & \\ \cdot & & & \\ p^r_{u^{M_r},u^1} & p^r_{u^{M_r},u^2} & \cdots & p^r_{u^{M_r},u^{M_r}} \end{bmatrix}^{C(S_k,x_k)} . \tag{5.20}$$

5.4. Routing Algorithms

In this section, we describe the algorithms used for the routing decisions with or without considering the spillback effect.

5.4.1 Dynamic Programming with Two-arcs Ahead Look Policy with Spillback Effect $(DP(2,H))$

The number of state variables increases exponentially with the number of disruption levels. Because of this, the MDP faces the curses of dimensionality. To solve large scale problems with many disruption levels, the dynamic programming approach for solving the Bellman's equations becomes computationally intractable. To reduce the computational time, we use the travel-time-dependent probability distributions for a limited part of the network. For the vulnerable arcs that are far from the current node, it is intuitive to assume that we experience the long run averages without the spillback effect. We use a Dynamic Programming Approach with Two-Arcs Ahead Policy, $(DP(2,H))$ developed in Chapter 3.

$(DP(2,H))$ considers limited online information for each stage. The intuition behind this is that due to the structure of the disruptions (where the probability of experiencing the disruption decreases within time), by the time we arrive to the further arcs, we will experience the time-invariant probabilities. Therefore, we eliminate the computational burden to calculate the transition probabilities for all vulnerable arcs.

In this policy, we have online information for two-arcs ahead from the decision node. Therefore, the state space of the hybrid policy for any current node i_k is modified as $S_k = (i_k, \hat{D}_k^{i_k})$ where $\hat{D}_k^{i_k} = \{u_k^{r_{i_k 1}}, u_k^{r_{i_k 2}}, .., u_k^{r_{i_k R_{i_k}}}\}$ with r_{i_k} which is the vector of the vulnerable arcs that are two-arc-ahead neighborhood of the node i_k and $R_{i_k} = |r_{i_k}|$, $R_{i_k} \leq R$.

Given the state is S_k, a transition is made to the next decision state, $S_{k+1} = (x_k, \hat{D}_{k+1}^{x_k})$. For the $(DP(2,H))$ policy, we use the travel-time-dependent state transition matrix for the vulnerable arcs for which we have online information. The probability distribution of being in state S_{k+1} from the state S_k given travel-time-dependent transition matrix is the same as the one given in Equation (5.17) except we only consider the limited part of the transition matrix where only the vulnerable arcs that are at most two-arcs ahead from i_k are included. For the rest of the arcs, we calculate the memoryless distributions. Note that we incorporate the spillback effect into the model by considering the transition probabilities given in (5.16).

For the arcs that are beyond the two-arcs ahead neighborhood of i_k, we use the probability distributions with travel-time-independent memoryless probability. The one-step transition probability for the r^{th} vulnerable arc is denoted as $p_{u,u'}^r = P\{\hat{D}_{k+1}(r) = u' | \hat{D}_k(r) = u\}$. Please note that r^{th} vulnerable arc is an element of the vector r_i. For the formulation, we use memoryless transition probabilities for the

vulnerable arcs at each node i as:

$$\Theta^r(k|S_k, x_k) = \left[p^r_{u,u'1} \quad p^r_{u,u'2} \quad \cdots \quad p^r_{u,u'M^r} \cdot \right] \tag{5.21}$$

Then, the probability distribution of transition into state S_{k+1} with memoryless probabilities is:

$$P(S_{k+1}|S_k) = \prod_{r=1}^{R} \Theta^r_{u,u'}(k|S_k, x_k). \tag{5.22}$$

5.4.2 Expected Shortest Path (ESP)

In practice, if there is no real-time information available, only historical information is used to determine the expected shortest path from the start node to the destination node. As we use steady-state probabilities for the calculation of the expected travel time, we do not involve the effect of spillback in our routing decisions. The output of this strategy is a single offline route from the start node to the destination node.

The steady-state probability to be in state u^r for any vulnerable arc r is denoted by $P(u^r)$. Please note that $u^r \in U_r$. The expected value of the arc r for traveling from node i_k to node x_k is denoted by $\bar{t}_{i_k,x_k}$:

$$\bar{t}_{i_k,x_k} = \sum_{u^r} P(u^r) t_{i_k,x_k}(u^r). \tag{5.23}$$

For finding the minimum expected shortest path from x_k to the destination node n, $\tilde{V}_{x_k}$, the Bellman Equations in Equation (5.19) modified into:

$$\tilde{V}_{i_k} = \min_{x_k \in I, x_k \in Neighbor(i_k)} \bar{t}_{i_k,x_k} + \tilde{V}_{i_{k+1}}, \tag{5.24}$$

$$\tilde{V}_{i_K} = 0. \tag{5.25}$$

The immediate travel time in Equation (5.19) is replaced by the expected travel times which makes the shortest path problem deterministic. Here, we backward recursion algorithm where we do not consider any probability transitions.

5.4.3 Online Routing Policy ($Online$)

In practice, routing algorithms mostly use real-time information and historical information. For representing these algorithms, we develop an online policy for the dynamic shortest path problems. At each node, we have the complete online information available for the neighbor arcs of the current node i_k. For the arcs that

are not in this zone, no online information is considered and therefore, the expected values are used based on historical information.

The steady-state probability to be in state u^r for any vulnerable arc r is denoted by $P(u^r)$, with $u^r \in U_r$. The expected value of the arc r for traveling from node i_k to node x_k is denoted by $\bar{t}_{i_k,x_k}$ and computed by using the Equation (5.23).

At each decision node i, online information of the one-arc-ahead arcs of the current node plus the expected values of the arcs until the destination node that are located further from the one-arc-ahead-neighborhood are considered to determine x_t. The online route can be found by solving the following equation:

$$V(S_k, x_k) = \min_{x_k \in I, x_k \in Neighbor(i_k)} C(S_k, x_k) + \tilde{V}_{x_k}. \tag{5.26}$$

For finding the minimum expected shortest path from x_k to the destination node n, $\tilde{V}_{x_k}$, we apply a backward recursion to the following equations:

$$\tilde{V}_{x_k} = \min_{x_{k+1} \in I, x_{k+1} \in Neighbor(x_k)} \bar{t}_{x_k,x_{k+1}} + \tilde{V}_{x_{k+1}}, \quad \text{where} \quad \tilde{V}_n = 0. \tag{5.27}$$

5.5. Experimental Design

The experimental design in this chapter is similar to the network generation process in Chapter 4. We generate different network instances using the same dimensions used in Chapter 4, except we now have an additional spillback rate as a network dimension. The dimensions considered are as follows:

Network size: The network size consists of small, medium and large networks with 16, 36 and 64 nodes respectively. The network is designed such that the origin and destination nodes are situated in the top-left corner and bottom-right corner respectively. With this structure, we prevent evaluating unnecessary nodes far from the shortest path. Clearly, this does not limit the applicability of our results, but merely reduces the number of unnecessary calculations to be evaluated.

Spillback rate: The parameter, α that changes the effect of spillback based on the instance type. We use low and high spillback rates for each network. After preliminary experiments for the generated networks, the low spillback rate is set to $\alpha = 1$ and high spillback rate is set to $\alpha = 15$ to reflect a significant difference between low and high spillback rate.

Disruption level: A vulnerable arc can have 2 and 3 different levels of disruptions. We set the expected travel time for each vulnerable arc the same regardless of the disruption level. For this, we adjust the steady state probabilities accordingly.

Network vulnerability: Vulnerable arcs are randomly assigned to the network with an iterative process. At each iteration, a shortest-path is found by the current vulnerable arcs list and a new vulnerable arc is assigned on the path of the current

shortest path. This continues until we reach the total number of vulnerable arcs. We have instances with low and high percentages of numbers of vulnerable arcs. 20% (low vulnerability) or up to 80% (high vulnerability) of the least-cost arcs in the network are labeled to be vulnerable.

Disruption rate: Define low disruption rate by a low probability of having disruptions to be between $[0.2 - 0.5)$,and high disruption rate by a high probability in the range $[0.5 - 0.8)$.

Travel times: Travel time of each arc for the non-disrupted level is randomly selected from a discrete uniform distribution $U(1, 10)$. The steady state probability of the non-disrupted level for each vulnerable arc is randomly selected based on the disruption rate. If there is a disruption, the travel time for the non-disrupted level is multiplied by a scalar depending on the disruption level.

We define "instance type" as the instance that has specific network properties. For instance, small size network with low number of vulnerable arcs with low disruption probability is a unique instance type. In this chapter, we generate 24 different instance types with the relevant properties as seen in Table 5.1. For each of the instance type, we randomly generate 25 instances with random travel times and random locations for vulnerable arcs along the actual shortest path.

Table 5.1 Summary of instance types

Number of Nodes	Number of vulnerable arcs		Disruption Rate		Number of disruption levels
	Low	High	Low	High	
16	3	5, 7	[0.2 - 0.5)	[0.5 - 0.8)	2, 3
36	3	5, 7	[0.2 - 0.5)	[0.5 - 0.8)	2, 3
64	3	5, 7	[0.2 - 0.5)	[0.5 - 0.8)	2, 3

For the evaluation of the algorithms, we use an exact evaluation. In the exact evaluation, for each algorithm, we obtain routing policies considering each possible state and with/without the spillback effect. Then, with these pre-determined policies for each instance, we compute the exact value function using the Equation (5.19) considering that there is a spillback effect in the network. This value gives the expected cost of the algorithm considering all possible states.

Furthermore, to evaluate the reliability of each algorithm, we calculate the variance of travel times among different disruption scenarios. The variance is calculated as follows (Here, S_0 is the disruption state at time 0 at the origin):

$$\begin{aligned}\sigma^2 = &\sum_{S_0} \text{probability}(S_0) * (\text{Cost Alternative Policy}(S_0))^2 \\ &- [\sum_{S_0} (\text{Cost Alternative Policy}(S_0) * \text{probability}(S_0))]^2\end{aligned} \tag{5.28}$$

For each instance, the percentage cost difference(gap) relative to the optimal policy for each routing policy is calculated as follows:

$$\Delta(\%) = \frac{(E[\text{Alternative Policy}] + \sigma(\text{Alternative Policy})) - (E[\text{Optimal Policy}] + \sigma(\text{Optimal Policy}))}{(E[\text{Optimal Policy}] + \sigma(\text{Optimal Policy}))} * 100 \tag{5.29}$$

The algorithms presented in this paper are programmed in Java. The computational results in this section are obtained by using IntelCore Duo 2.8 Ghz Processor with 3.46 GB RAM.

5.6. Numerical Results

For each instance type, we compute the average of the 25 instances. Then, we aggregate these results according to these network dimensions: the disruption rate, the network size, the vulnerability and the spillback rate. In the numerical results, the aggregated results for the expected travel time, the variance, the percentage gap and the computational time are shown for the network size, the vulnerability and the spillback rate. Note that for computing $C(S_k, x_k|\hat{D}_k(Zone(r)))$, we fix the number of blocks (B) to 4 to observe the effect of the spillback rate more intensely.

Effect of the Network Size

To investigate the effect of considering the spillback information on different network sizes, we compare the expected travel time and the variance of the routing algorithms for small, medium and large networks with 16, 36 and 64 nodes respectively. On small size networks, as the alternatives are limited, the effect of ignoring the spillback effect and using the limited information is higher. For instance, Table 5.3 shows that the percentage gap of the optimal algorithm without the spillback effect (Opt_{NS}) is on average 7.035% worse than the optimal algorithm with the spillback effect (Opt_S) under low disruption rate for small networks. This difference shows that the delay caused by not considering the spillback information in our routing decisions is higher for the small networks.

The performance of $DP(2, H)$ is better than the Opt_{NS}, but still 2.236% worse than Opt_S. This gap is the value of considering only a limited part of the network. Figure 5.3 shows that as the network size increases, the gap and the variance decrease. Figure 5.3 shows that the performance difference between the $DP(2, H)$ and the Opt_{NS} and the Expected Shortest Path algorithms also decreases with the increase in the network size. This shows that as the vulnerable arcs become scattered, the effect of spillback rate in routing decisions decreases compared to the smaller networks. On the other hand, the performance of the Online algorithm gets worse as the network size increases. This is because considering the online but not spillback information for high number of arcs leads to higher cost on average. The variance of total travel time decreases for all algorithms as there are more alternatives in large networks. This way, in case of a high effect disruption an alternative lower cost decision is made.

When we compare the Opt_{NS} with the Online and the Expected Shortest Path algorithm, we observe that using the full information by ignoring the spillback effect gives similar performances as static and online algorithms both in small and large size networks. This indicates that considering the spillback information has on average higher impact on the quality of the routing decisions than considering detailed disruption information for the whole network.

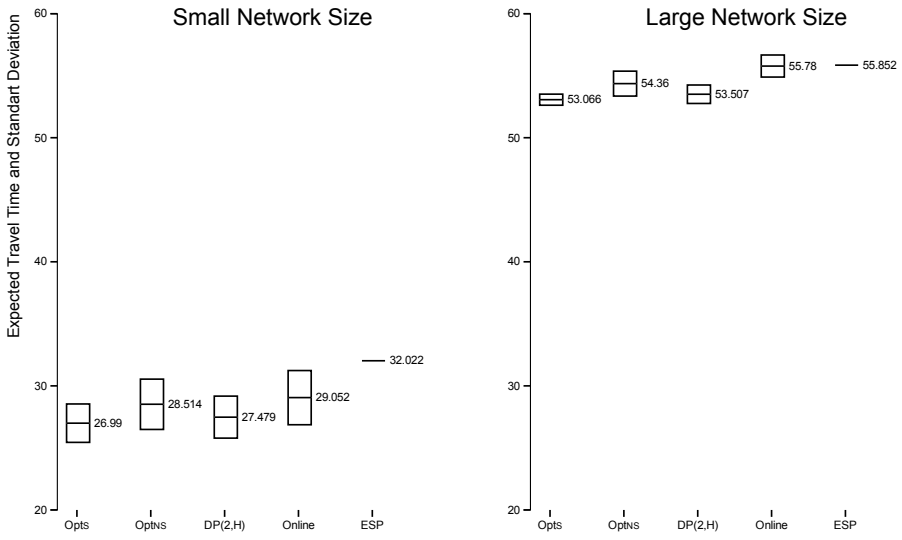


Figure 5.3 The expected value and the variance of the algorithms evaluated with the spillback effect for different network sizes

Table 5.2 The performance of the algorithms evaluated with the spillback effect for different network sizes

		Low Disruption Rate					High Disruption Rate				
Network Size	**Performance**	Opt_S	Opt_{NS}	$DP(2,H)$	$Online$	ESP	Opt_S	Opt_{NS}	$DP(2,H)$	$Online$	ESP
	Expected Value	26.990	28.514	27.479	29.052	32.022	29.158	30.270	29.207	30.910	32.156
	Variance	2.395	4.125	2.878	4.771	0.111	2.101	6.689	3.389	6.427	0.861
	Δ (%)	n.a.	7.035	2.236	9.456	13.376	n.a.	7.346	1.438	9.271	8.091
Small	**Cpu(sec)**	296.819	141.630	0.119	0.001	0.001	297.593	145.830	0.118	0.000	0.000
	Expected Value	39.635	40.772	40.185	41.871	43.269	41.595	42.035	41.803	42.211	43.580
	Variance	0.233	0.928	0.646	1.085	0.034	0.144	0.692	0.574	0.868	0.007
	Δ (%)	n.a.	4.029	2.169	6.964	8.310	n.a.	2.127	1.397	2.783	4.028
Medium	**Cpu(sec)**	1551.290	775.658	0.225	0.003	0.001	1579.452	775.503	0.201	0.001	0.001
	Expected Value	53.066	54.360	53.507	55.780	55.852	54.546	55.334	54.714	55.203	56.106
	Variance	0.199	1.011	0.551	0.796	0.004	0.180	2.122	0.670	0.642	0.002
	Δ (%)	n.a.	3.462	1.377	5.905	4.493	n.a.	3.310	1.023	1.880	2.137
Large	**Cpu(sec)**	4541.214	1453.135	0.523	0.008	0.005	4481.012	1181.773	0.752	0.005	0.004

Table 5.3 The performance of the algorithms evaluated with the spillback effect for low and high vulnerable networks

		Low Disruption Rate					High Disruption Rate				
Vulnerability	**Performance**	Opt_S	Opt_{NS}	$DP(2,H)$	$Online$	ESP	Opt_S	Opt_{NS}	$DP(2,H)$	$Online$	ESP
	Expected Value	38.364	39.503	38.745	39.994	41.281	39.744	40.296	39.883	40.376	41.546
	Variance	0.582	1.327	0.808	1.132	0.011	0.693	2.564	1.172	2.021	0.002
	Δ (%)	n.a.	3.905	1.322	4.936	5.771	n.a.	3.255	0.958	3.011	2.499
Low	**Cpu(sec)**	22.785	3.338	0.089	0.000	0.001	22.091	3.454	0.082	0.000	0.001
	Expected Value	40.919	42.850	41.543	43.727	45.337	43.115	44.046	43.258	44.373	45.549
	Variance	1.182	2.484	1.725	2.941	0.075	0.885	3.570	1.793	3.062	0.482
	Δ (%)	n.a.	5.759	2.022	8.178	8.581	n.a.	4.268	1.230	4.694	4.965
High	**Cpu(sec)**	3534.434	2981.343	0.811	0.006	0.003	3517.527	2166.090	0.923	0.003	0.002

Table 5.4 The performance of the algorithms evaluated with the different rates of spillback effect

		Low Disruption Rate					High Disruption Rate				
Spillback Rate	**Performance**	Opt_S	Opt_{NS}	$DP(2,H)$	$Online$	ESP	Opt_S	Opt_{NS}	$DP(2,H)$	$Online$	ESP
	Expected Value	39.615	40.967	40.164	41.972	43.639	41.569	42.332	41.744	42.623	43.860
	Variance	0.893	2.049	1.391	2.196	0.099	0.814	3.168	1.577	2.930	0.059
	Δ (%)	n.a.	4.532	1.933	7.136	8.367	n.a.	3.862	1.244	4.386	3.841
Low	**Cpu(sec)**	2125.778	722.740	0.410	0.004	0.002	2084.807	724.695	0.397	0.001	0.001
	Expected Value	40.179	41.464	40.683	42.496	43.790	41.963	42.761	42.072	42.926	43.927
	Variance	0.992	1.993	1.326	2.238	0.000	0.803	3.167	1.511	2.362	0.521
	Δ (%)	n.a	4.129	1.600	6.841	6.364	n.a.	3.922	1.032	3.742	4.175
High	**Cpu(sec)**	2133.770	857.542	0.434	0.004	0.003	2153.898	677.376	0.422	0.003	0.002

Effect of the Disruption Rate

Tables 5.2, 5.3 and 5.4 show that as the disruption rate increases, the gap between the algorithms in terms of the expected travel time decreases as they choose the similar routes which are generally the risk aversive ones. As the spillback rate increases, the transition probabilities to a higher disruption level increases at a higher rate. Therefore, the algorithms considering the spillback effect in their routing decisions choose the routes with lower expected travel times. Similarly, as the disruption probabilities are higher and vulnerable arcs are consecutive each other, the algorithms ignoring the spillback effect also choose alternative routes with lower expected travel times.

Though, when we consider the variance, however, due to higher transition probabilities the variance of the routes for each algorithm increases compared to the lower disruption rate case.

Effect of Network Vulnerability

The expected travel time and the variance increases as there are more vulnerable arcs in the network. The impact of taking into account the spillback effect also increases with the number of vulnerable arcs. For instance, Table 5.3 shows that the performance gap between the Opt_S and the Opt_{NS} increases from 3.905% to 5.759% from low vulnerable to highly vulnerable networks with low disruption rate. This result is intuitive because as there are more vulnerable arcs consecutive to each other, the transition rates and so the travel time increase due to the spillback effect. Furthermore, the realized travel times on the current arc at stage $k+1$ is more different than the observed travel times at stage k due to the higher impact of the spillback effect. Therefore, the algorithms using the spillback information choose risk aversive routes. Figure 5.4 shows that the online algorithms performance on highly vulnerable networks is much worse than the performance on the low vulnerable networks. The intuition behind this is that the online algorithm uses the observed travel times for the current arc and expected travel times for the rest of the vulnerable arcs. When the observed value is much more different from the realized value due to the spillback effect, the solution quality decreases significantly.

The percentage gap beween the Opt_{NS} and the $DP(2, H)$ algorithm with the spillback effect increases as the number of vulnerable arcs increases (Table 5.3 and Figure 5.4). This shows that the value of using the spillback effect is more effective than the value of using full network information for highly vulnerable networks.

For all of the routing algorithms (especially for Opt_{NS} and Online) the variance increases with higher vulnerability. This shows that the route choice and its impacts differ significantly among different network states.

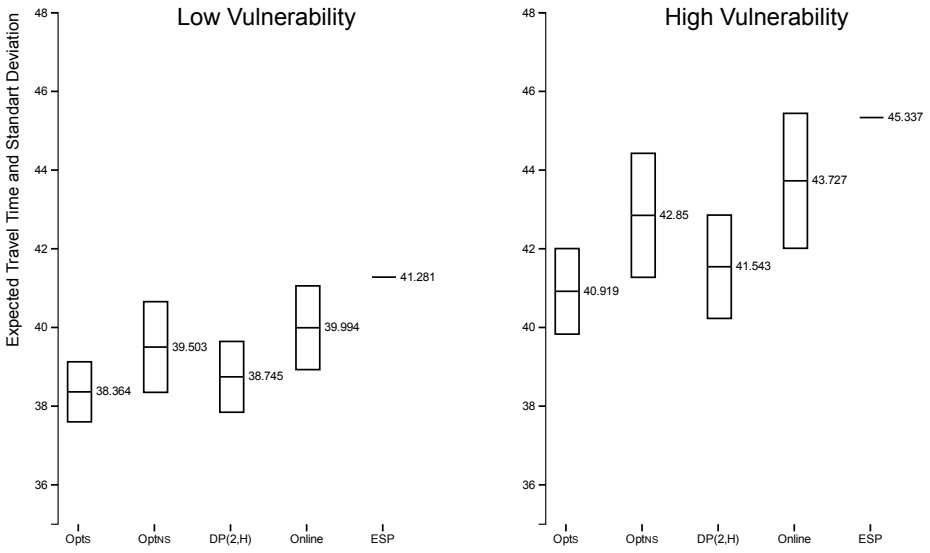


Figure 5.4 The expected value and the variance of the algorithms evaluated with the spillback effect for low and high network vulnerability

Effect of Spillback Rate

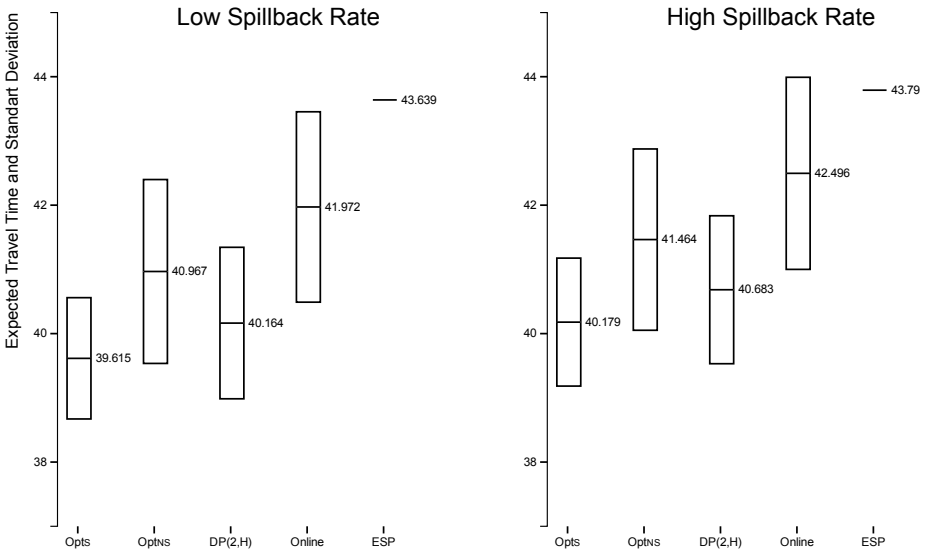
As the spillback rate increases, the gap between algorithms using and ignoring the spillback effect increases slightly (Figure 5.5). This is because the algorithms considering spillback choose risk aversive routes as the transition rates changes at a higher rate. This increases the expected travel time compared to low spillback rate. Note that the rate of increase is higher for the Online algorithm because it does not consider the spillback information.

Also, the gap between the optimal solution and the solution from the $DP(2, H)$ algorithm with the spillback decreases as both of the algorithms choose the risk aversive routes in case of higher spillback rate (Table 4).

Computational Time

The computational time for the optimal algorithms via the MDP is increasing exponentially with the increase in the number of vulnerable arcs and the disruption levels (Table 5.3 and Table 5.5). The $DP(2, H)$, on the other hand, is less affected from the state space explosion as it considers limited part of the network while providing good quality of solutions. The expected shortest path and online algorithms are the fastest, but the performance of the routing decisions are much lower than the other algorithms.

Figure 5.5 The expected value and the variance of the algorithms evaluated with the spillback effect for different spillback rates



5.7. Conclusions

In this paper, we consider dynamic shortest path problems with travel-time-dependent network disruptions and spillback effect. The spillback effect is a congestion propagation to an upstream arc. We analyze the effect of ignoring and considering the spillback effect in the dynamic routing decisions with network disruptions. We model the stochastic dynamic routing problem with travel-time-dependent stochastic disruptions as a discrete-time finite Markov Decision Process (MDP). We incorporate the spillback effect into the MDP formulation. The spillback effect is modeled with the kinematic wave model which is an accepted model in traffic theory. To analyze the effect of spillback, we also formulate the MDP ignoring the spillback effect. However, as the size of the network and the disruption levels increase, the computational complexity increases causing the MDP become computationally intractable. Therefore, to reduce the computational time, we consider a hybrid dynamic programming algorithm using the detailed dynamic and stochastic traffic information for the limited part of the network (Chapter 3). We also provide online and static algorithms mostly used in practice for the routing decisions. These algorithms do not include the spillback effect in their routing decisions.

We use a test bed of different network structures with different levels of disruption rates and spillback rates. Then, we compare the solution quality and the computational time of different strategies in case of the spillback effect in the network. The numerical results show that considering the spillback effect has the highest impact on the solution quality when the network has higher number of vulnerable arcs. Moreover, the hybrid dynamic programming approach with the disruption and the spillback information for the limited part of the network, significantly reduces the computational time while providing higher solution quality than the full information model that ignores the spillback effect.

Future research involves the analysis of the spillback effect by using other traffic models such as queueing theory to investigate the impact of the traffic models on the routing decisions. The extension of the spillback effect model by using time-dependent and second-order models will reflect the real dynamics even better.

Chapter 6

Single VRPSD: Approximate Dynamic Programming

In Chapters 3-5, we focused on developing dynamic routing policies for dynamic shortest path problems. In Chapter 6, we extend our analysis on dynamic routing with a dynamic vehicle routing problem. In this problem, we consider a single vehicle delivery, where there is a set of customers to visit and a depot where we return at the end of the day. The demands of customers are stochastic and realized upon arrival. In this problem, the vehicle has a limited capacity and a failure is observed when the remaining capacity is not enough to satisfy the customer demand. Therefore, the vehicle returns to the depot for replenishment. In the literature, this problem is denoted as vehicle routing problem with stochastic demands (VRPSD) (see Chapter 1). One of the interesting properties of this problem is that, it can be handled by both static and dynamic routing approaches. In this chapter, we tackle the problem with a dynamic approach where we develop dynamic policies for which customer to visit next and whether to return to depot to replenish depending on the real-time state of the current customer visited and the remaining capacity of the vehicle.

6.1. Introduction

Consider a vehicle starting at a central bank and filling up ATMs at different places. For security reasons, it is not allowed to carry a large amount of money. Consequently, the vehicle is forced to make several short tours during its operating period (e.g., a working day) going back and forth to the central bank. Moreover, the needed cash in the ATMs is not known beforehand. Other similar examples of this described problem

family in real-life are: beer distribution to retail outlets, the re-supply of baked goods at food stores, replenishment of liquid gas at research laboratories, stocking of vending machines (Yang et al. 2000), local deposit collection from bank branches, less-than-truckload package collection, garbage collection, home heating oil delivery, and forklift routing (Ak and Erera 2007).

The above described problem is related to the well-known Vehicle Routing Problem (VRP). The standard deterministic VRP is described extensively in the literature (Laporte 2007). In contrast, this chapter studies a single vehicle routing problem where stochastic demand is incurred (denoted as the VRPSD), very similar to Secomandi (2001). In general, this problem is similar to the standard vehicle routing problem where the aim is to construct a set of shortest routes for a fleet of fixed capacity. In the stochastic case, however, each customer has a given and known demand distribution and the actual demand realization is unknown until the vehicle arrives at the customer, when the customer's actual demand is observed. In the VRPSD, the vehicle may be unable to satisfy the actual customer's demand realization when visiting the customer (denoted as a *failure*). As such, the vehicle needs to return to the depot for a refill and return back to the partially served customer. In general, the vehicle serves every customer once unless a *failure* occurs, in which case a detour-to-depot is executed.

In this chapter, we formulate the VRPSD using a stochastic dynamic programming model. Dynamic programming (DP) provides an elegant framework to model stochastic optimization problems. However, DP faces the well-known three curses of dimensionality (states, outcomes, and decisions) and cannot deal with practical size problems. In addition, the single vehicle routing problem with stochastic demands is a difficult and computationally demanding problem. Over the past years, computing power has increased dramatically, giving a sound basis to efficiently handle stochastic and dynamic vehicle routing problems. Our chapter employs an Approximate Dynamic Programming (ADP) strategy. ADP emerges as an efficient and effective tool in solving large scale stochastic optimization problems, combining the flexibility of simulation with the intelligence of optimization. It is a powerful approach to model and solve problems which are large, complex and with stochastic and/or dynamic elements (Powell 2007). Referring to Pillac et al. (2012), ADP successfully solves large-scale freight transport and fleet management problems while coping with the scalability problems of DP (Godfrey and Powell 2002, Powell et al. 2002, Powell and Van Roy 2004, Simão et al. 2009).

In this chapter, we develop ADP algorithms based on Value Function Approximations (VFA) with lookup table representation. We first design a standard VFA algorithm by using the ADP framework. To achieve good computational performance and solution quality, several adaptations are needed. Using post-decision state variables in ADP allows making decisions without having to compute the expectation. However, for the VRPSD, post-decision state variables omit important information about the current state and the decision. Therefore, we consider a Q-learning algorithm with lookup

table representation in which we store the state-decision pairs and their values (i.e., Q-factors). However, the size of a standard lookup table increases exponentially (as it depends on both the state and decision). For this reason, we improve the standard Q-learning algorithm with bounded lookup tables and efficient maintenance strategies. We also design effective exploration/exploitation strategies such that we obtain higher quality solutions with lower computational time, as compared to the rollout algorithm in Secomandi (2001). We denote this improved VFA algorithm as VFA$^+$.

The contribution of this chapter to the literature is as follows. First, we formulate the vehicle routing problem with stochastic demands using a unified stochastic dynamic programming modeling framework (Powell 2007, 2011, Powell et al. 2012), which allows for flexible extensions of the problem in real-life applications. Second, we design efficient ADP algorithms that allow us to solve large scale problems in reasonable time. The standard VFA with lookup table representation is improved using a Q-learning algorithm with bounded lookup tables and efficient maintenance. Our algorithm comparisons on test instances from the literature (Secomandi 2001, Solomon 1987) show that, for small size test instances, the VFA$^+$ algorithm on average covers more than 50% of the performance gap between the Rollout and the optimal solution; for large size test instances, VFA$^+$ consistently outperforms the Rollout algorithm with better solution quality and less computational time. The significant reduction in computational time enables solving larger scale instances, which is important for real-life decision making. Further, we analyze the effect of the depot location on the relative performances of the algorithms. Last, this chapter provides important insights on applying ADP to deal with stochastic and combinatorial problems such as VRPSD, using bounded lookup tables with efficient maintenance and exploration-and-exploitation strategies.

The chapter is structured as follows. The literature review is given in Section 6.2. The problem formulation is presented in Section 6.3, and the ADP algorithms are described in Section 6.4. The experiment design and numerical results are given in Sections 6.5 and 6.6. Section 6.7 concludes this chapter.

6.2. Literature Review

Stochastic vehicle routing problems are characterized by some random elements in their problem definition (Gendreau et al. 1996a). In the literature, researchers consider stochastic demands (Bertsimas 1992, Dror et al. 1993), stochastic customers (Bent and Van Hentenryck 2004b), stochastic demand and customers (Gendreau et al. 1995, 1996b) and stochastic travel times (Laporte et al. 1992, Kenyon and Morton 2003)). Gendreau et al. (1996a) review the literature on stochastic VRPs and their different flavours.

There are a number of papers closely related to this chapter such as Secomandi (2000, 2001), Novoa and Storer (2009), Goodson et al. (2013) dealing with the

VRPSD, considering detour-to-depot schemes and allowing for early replenishment (see Chapter 1 for a detailed review).

This chapter can also be situated in the family of re-optimization algorithms for the VRP with stochastic demands. Dror et al. (1989, 1993) are the early papers that introduce the re-optimization strategies. They optimally re-sequence the unvisited customers whenever a vehicle arrives at a customer and observes the demand. Secomandi and Margot (2009) also considers a VRPSD under re-optimization. They formulate the problem in terms of a finite horizon MDP for the single vehicle case. They develop a partial re-optimization methodology to compute suboptimal re-optimization policies for the problem. In this methodology, they select a set of states for the MDP by using two heuristics: the partitioning heuristic and the sliding heuristic. They compute an optimal policy on this restricted set of states by a backward dynamic programming. Pillac et al. (2011) also considers the same problem with a re-optimization approach. They adapt a multiple scenario approach where pool of scenarios are maintained based on customer demand realizations. The scenarios are optimized during the travel.

In the VRPSD literature, there are a number of papers dealing with developing an optimal restocking policy with a predefined customer sequence. Yang et al. (2000) study strategies of planning preventive returns to the depot at strategic points along the vehicle routes. They prove that for each customer, there exists a threshold number such that the optimal decision is to continue to next customer if the remaining load is greater than or equal to the threshold number or otherwise to return to the depot for replenishment.

Tatarakis and Minis (2009) study the multi-product delivery routing with stochastic demands. They develop a dynamic programming algorithm to solve a compartmentalized case of multi-product delivery to derive the optimal policy in a reasonable amount of time. Minis and Tatarakis (2011) extend the problem to a pickup and delivery case of the VRPSD. They provide an algorithm to determine the minimum expected routing cost and a policy to make the optimal decisions including the detour-to-depot decisions for the stock replenishment. Recently, Pandelis et al. (2012) prove the optimal structure of the same problem for any positive number of multiple products.

The VRPSD is also studied with additional constraints. Erera et al. (2010) consider VRP with stochastic demands and constraints on the travel time durations of the tours. The authors define and study various restocking detour policies in the paper. Lei et al. (2011) study the VRP problem with stochastic customers with time windows. The problem is modeled using stochastic programming with recourse and the solution strategy is proposed as a large neighborhood search heuristic.

In this chapter, we model and solve the VRPSD using an ADP framework. Powell (2007, 2011) provide a comprehensive introduction to the basic ideas of ADP and address key algorithmic issues when designing ADP algorithms. For the dynamic VRP

problems, a unified framework is presented in Powell et al. (2012) where various polices including the ADP approach are explained.

6.3. Problem Description and Model Formulation

We study a single vehicle routing problem with stochastic customer demands (VRPSD). On an undirected graph $G = (V, E)$, $V = \{0, \ldots, N\}$ is the vertex set and E is the edge set. Vertex 0 denotes the depot, whereas vertices $i = 1, \ldots, N$ denote the customers to be served. A nonnegative distance d_{ij} is associated with each edge $(i, j) \in E$, representing the travel distance (or cost, time, etc.) between vertices i and j.

A single vehicle with full capacity Q starts from the depot, serves all the customers to perform only deliveries (or only pick-ups), and returns to the depot at the end of the tour. Each customer $i \in V$ is associated with a stochastic demand D_i, the true value of which is revealed upon the arrival of the vehicle at the customer. If the vehicle does not have sufficient capacity to serve a customer (a "failure" occurs), it partially serves the customer, returns to the depot to replenish, and comes back to the customer to fulfill the remaining demand. The vehicle then continues its tour to the next customer or the depot (at the end of the tour). The objective is to minimize the expected total travel distance. We assume that the depot has plenty quantity of the commodity and the maximum possible demand of each customer is smaller than the vehicle capacity.

If early replenishment is allowed, the vehicle can return to the depot to replenish before encountering a failure. In the offline planning version of the problem, the vehicle always follows a predetermined order to visit the customers, while in the online planning version, the vehicle is allowed to re-route (re-optimize) after serving each customer. In this chapter, our focus is on the online planning problem with early replenishment (as in Secomandi (2000, 2001)).

Next, we formulate the problem as a stochastic dynamic program, using the notation as described in Powell (2007, 2011).

The problem is divided into $t = 0, 1, \ldots, N, N + 1$ stages. $t = 0$ represents the start of the tour at the depot, $t = N + 1$ represents the end of the tour back to the depot, and $t = 1, \ldots, N$ represents the number of customers that have been visited during the tour.

State

The state variable is defined as:

$$S_t = (i_t, l_t, J_t), \quad t = \{0, 1, \ldots, N, N+1\}, \tag{6.1}$$

where
i_t: the current customer being served (or the depot when $t = 0$ and $t = N+1$);
l_t: the current capacity in the vehicle *after* serving the current customer ($0 \leq l_t \leq Q$);
J_t: the vector $(j_{t,1}, \ldots, j_{t,N})$ that represents the customers' service status: $j_{t,i} = 1$, if customer i has already been served; $j_{t,i} = 0$, otherwise.

Therefore, when the vehicle starts at the depot, the initial state is $S_0 = (0, Q, 0, \ldots, 0)$, and when the vehicle returns to the depot after serving all the customers, the final state becomes $S_{N+1} = (N+1, l_{N+1}, 1, \ldots, 1)$. Note that $l_{N+1} = l_N$ is the vehicle's remaining capacity after serving the last customer.

Decision Variables

After serving the current customer, two types of decisions are to be made: which customer to serve next and whether to return to the depot before visiting the next customer. The decision variables are defined as:

$$x_t = (i_{t+1}, r_t), \quad t = \{0, 1, \ldots, N\}, \tag{6.2}$$

where
i_{t+1}: the next customer to be served;
r_t: $r_t = 1$ indicates returning to the depot before visiting the next customer; $r_t = 0$ otherwise.

Further, we define $X_t^\pi(S_t)$ as the *decision function* that determines decision x_t at stage t under *policy* π, given state S_t. Each $\pi \in \Pi$ refers to a different policy and Π denotes the set of all implementable policies.

Exogenous Information

The customer demand $D_{i_{t+1}}$ has a customer specific discrete distribution. The actual customer demand $\hat{D}_{i_{t+1}}$ is only revealed after the vehicle arrives at customer i_{t+1}.

$$W_{t+1} = \hat{D}_{i_{t+1}}, \quad t = \{0, 1, \ldots, N-1\}. \tag{6.3}$$

State Transition Function

Given the current state $S_t = (i_t, l_t, J_t)$, the decision $x_t = (i_{t+1}, r_t)$, and the exogenous information $W_{t+1} = \hat{D}_{i_{t+1}}$, the state transition function is defined as, for

$t = \{0, 1, \ldots, N\}$,

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}) = \begin{cases} (i_{t+1}, Q - \hat{D}_{i_{t+1}}, J_{t+1}), & \text{if } r_t = 1, \\ (i_{t+1}, l_t - \hat{D}_{i_{t+1}}, J_{t+1}), & \text{if } r_t = 0 \text{ and } l_t \geq \hat{D}_{i_{t+1}}, \\ (i_{t+1}, l_t + Q - \hat{D}_{i_{t+1}}, J_{t+1}), & \text{if } r_t = 0 \text{ and } l_t < \hat{D}_{i_{t+1}}, \end{cases} \tag{6.4}$$

where the service status vector J_{t+1} is updated as: $j_{t+1,i} = 1$, if $i = i_{t+1}$; $j_{t+1,i} = j_{t,i}$, otherwise. That is, the service status of customer i_{t+1} is changed to 1 (being served) and the service statuses of other customers remain unchanged. Note the letter "M" in the first equation of Equation (6.4) represents "model" as in Powell (2007, 2011).

If the vehicle returns to the depot to replenish after serving customer t ($r_t = 1$), it arrives at the next customer i_{t+1} with full capacity Q. Therefore, the capacity after serving customer i_{t+1} becomes $Q - \hat{D}_{i_{t+1}}$. If the vehicle travels to the next customer $i_{i_{t+1}}$ without returning to the depot ($r_t = 0$), it arrives at customer i_{t+1} with capacity l_t. If l_t is sufficient to serve the realized demand $\hat{D}_{i_{t+1}}$, l_{t+1} becomes $l_t - \hat{D}_{i_{t+1}}$; otherwise, the vehicle encounters a failure and needs to replenish at the depot to satisfy demand $\hat{D}_{i_{t+1}}$, thus l_{t+1} becomes $l_t + Q - \hat{D}_{i_{t+1}}$.

Cost Function

The vehicle's actual travel distance or cost depends on both the decision and realized demand at the next customer $W_{t+1} = \hat{D}_{i_{t+1}}$. Therefore, the (expected) cost function $c_t(S_t, x_t)$ can be decomposed into a deterministic and a stochastic parts, as below.

$$c_t(S_t, x_t) = C_t(S_t, x_t) + E[\Delta C_{t+1}(S_t, x_t, W_{t+1})], \tag{6.5}$$

where

$$C_t(S_t, x_t) = \begin{cases} d_{i_t,0} + d_{0,i_{t+1}}, & \text{if } r_t = 1, \\ d_{i_t,i_{t+1}}, & \text{if } r_t = 0, \end{cases} \tag{6.6}$$

and

$$\Delta C_{t+1}(S_t, x_t, W_{t+1}) = \begin{cases} 0 & \text{if } r_t = 1, \\ 0, & \text{if } r_t = 0 \text{ and } l_t \geq \hat{D}_{i_{t+1}}, \\ d_{i_{t+1},0} + d_{0,i_{t+1}}, & \text{if } r_t = 0 \text{ and } l_t < \hat{D}_{i_{t+1}}. \end{cases} \tag{6.7}$$

The calculations of (6.6) and (6.7) follow the same logic as in the state transition function (6.4).

Objective Function

The objective of the stochastic dynamic program is to find the optimal policy $\pi \in \Pi$ to minimize the expected total cost (travel distance) to serve all the customers, that

is,

$$\min_{\pi \in \Pi} \sum_{t=0}^{N} c_t(S_t, X_t^{\pi}(S_t)), \tag{6.8}$$

where $x_t = X_t^{\pi}(S_t)$ is the decision made according to the decision function $X_t^{\pi}(S_t)$ under policy π, given the current state S_t. Note that the expectation is embedded in the calculation of the cost function $c_t(S_t, x_t)$.

6.4. Approximate Dynamic Programming

If the state, decision, and outcome spaces are finite discrete, the stochastic dynamic program (6.8) can be solved recursively using Bellman's equations,

$$V_t(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \mathbb{E}[V_{t+1}(S_{t+1})]). \tag{6.9}$$

The value function $V_t(S_t)$ specifies the value of being in a state S_t, in which $C_t(S_t, x_t)$ accounts for the immediate cost associated with the current state S_t and decision x_t, while the value function $V_{t+1}(S_{t+1}) = V_{t+1}(S^M(S_t, x_t, W_{t+1}))$ evaluates the future impact of the decision x_t under the realized exogenous information W_{t+1}.

To overcome the three curses of dimensionality (states, decisions, and outcomes) associated with the classical DP approach, in *approximate dynamic programming* (ADP), we replace the exact value function $V_{t+1}(\cdot)$ in Equation (6.9) with an approximation $\bar{V}_{t+1}(\cdot)$ as in Equation (6.10). Instead of the exact evaluation of $V_{t+1}(\cdot)$ often in a backward manner, $\bar{V}_{t+1}(\cdot)$ can be evaluated via step-forward simulation, by integrating a variety of rich classes of stochastic optimization and simulation methodologies.

$$\bar{V}_t(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \mathbb{E}[\bar{V}_{t+1}(S_{t+1})]). \tag{6.10}$$

While the approximate value function $\bar{V}_{t+1}(\cdot)$ can take a variety of forms (such as weighted sum of basis functions, piecewise linear functions, regression models, neural networks), the lookup table representation is a generic model-free form that is often used when the value function structure can hardly be clearly defined, which is the case of the VRPSD under study. In this section, we first introduce a generic value function approximation (VFA) with lookup table representation, and then describe an improved version (VFA$^+$), which addresses the problem characteristics of the VRPSD.

6.4.1 Value Function Approximation Algorithm (VFA)

Algorithm 4 depicts a generic value function approximation approach with lookup table representation. In Step 0a, we use the Rollout algorithm (Secomandi 2001) as

Algorithm 4 A generic VFA approach with lookup table representation.

Step 0. Initialization.
  Step 0a. Initialize $\bar{V}_t^0(S_t)$ for all states S_t.
  Step 0b. Choose an initial state S_0^1.
  Stap 0c. Set the iteration counter $n = 1$.
Step 1. Choose a sample path ω^n.
Step 2. For $t = 0, 1, \ldots, N$, do:
  Step 2a. If *exploitation*, solve
    $\hat{v}_t^n = \min_{x_t \in \mathcal{X}_t}(C_t(S_t, x_t) + \mathbb{E}[\bar{V}_{t+1}^{n-1}(S_{t+1})])$, $\quad (*)$
  and let x_t^n be the solution.
  If *exploration*, randomly choose a solution $x_t^n \in \mathcal{X}_t$.
  Step 2b. Update $\bar{V}_t^n(S_t)$ using
    $$\bar{V}_t^n(S_t) = \begin{cases} (1-\alpha_{n-1})\bar{V}_t^{n-1}(S_t^n) + \alpha_{n-1}\hat{v}_t^n, & \text{if } S_t = S_t^n, \\ \bar{V}_t^{n-1}(S_t), & \text{otherwise.} \end{cases}$$
  Step 2c. State transition.
    $S_{t+1}^n = S^M(S_t^n, x_t^n, W_{t+1}(\omega^n))$.
Step 3. Let $n = n + 1$. If $n \leq \mathbb{N}$, go to step 1.
  Note that $\mathbb{N}$ denotes the pre-set maximum number of iterations.
Step 4. Output the value function, $\{\bar{V}_t^{\mathbb{N}}(S_t^x)\}_{t=0}^{N-1}$.

the heuristic to initialize the state values.

Striking a good balance between exploration and exploitation remains an important and cutting-edge research question in ADP and other related research areas such as simulation optimization and machine learning. In our VFA algorithm, we use the fixed exploration rate strategy. That is, with probability ρ (for example, 0.10), we explore the impact of a randomly selected decision; otherwise, we stick to the optimal decision based on the current value function (Step 2a). In Section 6.4.2, we describe an improvement on the exploration and exploitation strategy using preventive returns and restocking.

In step 2b, we use the exponential smoothing function to update the value function approximation $\bar{V}_t(S_t)$ with the observed value $\hat{v}_t(S_t)$. That is,

$$\bar{V}_t^n(S_t) = (1-\alpha_{n-1})\bar{V}_t^{n-1}(S_t^n) + \alpha_{n-1}\hat{v}_t^n, \tag{6.11}$$

where α_{n-1} is the stepsize.

For calculating α_{n-1}, we apply the *Bias-adjusted Kalman Filter* (BAKF) stepsize rule

(George and Powell 2006, Powell 2007), which is given by

$$\alpha_{n-1} = 1 - \frac{(\bar{\sigma}^2)^n}{(1+\bar{\lambda}^{n-1})(\bar{\sigma}^2)^n + (\bar{\beta}^n)^2}. \tag{6.12}$$

$(\bar{\sigma}^2)^n$ denotes the estimate of the variance of the value function $\bar{V}_t^n(S_t^n)$ and $\bar{\beta}^n$ denotes the estimate of bias due to smoothing a nonstationary data series. The BAKF stepsize rule adaptively balances the estimate of the noise $(\bar{\sigma}^2)^n$ and the estimate of the bias $\bar{\beta}^n$ that is attributable to the transient nature of the data in the ADP solution process. We refer to George and Powell (2006) and Section 6.5.3 in Powell (2007) for more details.

In Equation (6.10), the approximate value function $\bar{V}_t(S_t)$ is associated with the *pre-decision state* S_t. Solving for Equation $(*)$ in Step 2a requires the calculation of the expected value of $\bar{V}_{t+1}(S_{t+1})$ within the $\min$ operator, which is computationally demanding. To improve the computational efficiency in ADP, Powell (2007) introduces the notion of *post-decision state* $S_t^x = S^{M,x}(S_t, x_t)$, which captures the state of the system immediately after the decision making, but before new information arrives. With the approximate value function around post-decision state $\bar{V}_t^{n-1}(S_t^x)$, we can solve for $\hat{v}_t^n(S_t)$ in Step 2a with

$$\hat{v}_t^n(S_t) = \min_{x_t \in X_t} (C_t(S_t, x_t) + \bar{V}_t^{n-1}(S_t^x)), \tag{6.13}$$

which avoids the expectation within the $\min$ operator, but normally requires more effort in estimating $\bar{V}_t^{n-1}(S_t^x)$.

In VRPSD, the post-decision state omits certain critical information. For example, besides knowing the next customer to visit, the actual cost or travel distance depends on both the realized demand of the next customer $\hat{D}_{i_{t+1}}$ and the current capacity l_t, which can vary significantly (refer to equation (6.7)). Further, the tradeoff between traveling directly to the next customer or detour-to-depot also depends on the relative distances between the current customer and the next customer or depot, respectively. Consequently, we somehow lose the "memoryless" property in VRPSD. Our numerical experiments also show that the approximate value function around post-decision state does not work very well, which confirms our observation.

Q-learning is another algorithm that has certain similarities to DP using the value function around post-decision states. In Q-learning, a Q-factor, $Q_t(S_t, x_t)$, stores the value of a state-decision pair and it captures the value of being in a state and taking a particular decision (Bertsekas and Tsitsiklis 1995, Sutton and Barto 1998, Powell 2007, Bertsekas 2012). However, a potential problem with Q-learning is that the size of the lookup table increases exponentially because it depends on both the state and the decision. In the next section, we describe how to better utilize the lookup tables with Q-factors via efficient maintenance and exploration/exploitation strategies in our improved algorithm (VFA$^+$).

6.4.2 Improved Value Function Approximation Algorithm (VFA$^+$)

The value function approximation (VFA) with lookup table representation as described in Algorithm 4 is a generic ADP approach. As previously mentioned, an alternative way to store the value function is to use Q-factors. Q-factors, $Q_t(S_t, x_t)$, store the value of a state-decision pair in the lookup table. The state-decision pair at stage t is: $(S_t, x_t) = ((i_t, l_t, J_t), (i_{t+1}, r_t))$. We should note that the decision consists of the next customer to visit at stage $t+1$ and whether to return to the depot before visiting the next customer.

For the VRPSD, we improve the computational performance and the solution quality of the standard VFA algorithm with Q-learning by considering the approximate values of the Q-factors. We define $\bar{Q}_t^n(S_t, x_t)$ as the approximate value of $Q_t(S_t, x_t)$ after n iterations. We use a double pass algorithm to update $\bar{Q}_t(S_t, x_t)$, as shown in Algorithm 5. At each iteration, we first find a customer sequence based on the values of the state-decision pairs (Q-factors) from the past iterations in the forward pass. Then, we update the Q-factors using the realized cost in the backward pass.

Algorithm 5 Q-learning approach for the VFA^+ algorithm with double pass

Step 0. Initialization.
  Step 0a. Initialize $\bar{Q}_t^0(S_t, x_t)$ for all states S_t and decisions $x_t \in \mathcal{X}_t$.
  Step 0b. Choose an initial state (S_0^1).
  Stap 0c. Set the iteration counter $n = 1$.
Step 1. Choose a sample path ω^n.
Step 2. (Forward pass) For $t = 0, 1, \ldots, N$, do:
  Step 2a. If *exploitation*, find
    $$x_t^n = \underset{x_t \in \mathcal{X}_t}{\operatorname{argmin}} (\bar{Q}_t^{n-1}(S_t, x_t) + SD(S_t, S_t^n)),$$
    If *exploration*, choose a solution $x_t^n \in \mathcal{X}_t$
    based on the exploration-and-exploitation strategies described.
  Step 2b. Compute the next state $S_{t+1}^n = S^M(S_t^n, x_t^n, W_{t+1}(\omega^n))$.
Step 3. (Backward pass) Set $\hat{q}_{N+1}^n = 0$ and do for all $t = N, N-1, \ldots, 0$:
  Step 3a. Calculate:
    $$\hat{q}_t^n = C_t(S_t^n, x_t^n) + \hat{q}_{t+1}^n$$
  Step 3b. Update $\bar{Q}_t^n(S_t, x_t^n)$ using
    $$\bar{Q}_t^n(S_t^n, x_t^n) = (1 - \alpha_{n-1})\bar{Q}_t^{n-1}(S_t^n, x_t^n) + \alpha_{n-1}\hat{q}_t^n.$$
Step 4. Let $n = n + 1$. If $n \leq \mathbb{N}$, go to step 1.
  Note that $\mathbb{N}$ denotes the pre-set maximum number of iterations.
Step 5. Return the Q-factors, $\{\bar{Q}_t^{\mathbb{N}}\}_{t=0}^{N-1}$.

As the size of the lookup table with Q-factors grows exponentially with both state and decision, we limit the number of stored Q-factors in the lookup table (*bounded* lookup table). At each iteration, we mostly visit the state-decision pairs that are already in the lookup table. Consequently, the values of the state-decision pairs (Q-factors) in the lookup table are more frequently updated, and therefore more accurate.

In Step 2a of Algorithm 5, one important notion is the *state difference*, $SD(S_t, S'_t)$. Due to the limited size of the bounded lookup table, we may frequently come into states that are not contained in the lookup table. In this case, we look at the most similar states. To determine these most similar states, we define the state difference SD between two states with identical i_t value as (where c_1 is a constant):

$$SD(S_t, S'_t) = \sum |j_t - j'_t| + c_1 \times |l_t - l'_t| \tag{6.14}$$

As the number of states-decision pairs grows with the vehicle capacity and with the number of customers, then, we consider a bounded lookup table with a subset of state-decision pairs. When we arrive at a state-decision pair that is not contained in the bounded lookup table, we identify the "nearest" state-decision pair according to Equation (6.14). Accordingly, we update the value of the nearest state-decision pair, already in the bounded lookup table with the minimum state difference, instead of the state-decision pair that is not in the bounded lookup table.

Further, the exponential growth in the state and decision space in the VRPSD forces us to find a good balance between exploration and exploitation. In VFA$^+$, the exploitation is obtained by focusing on a limited number of state-decision pairs, such that good cost estimates can be found by frequent visits to these pairs. The exploration is obtained by using a variety of randomized heuristics for our travel decisions.

Further, we improve the performance in terms of solution quality and computational time by considering the following algorithmic strategies:

1. Use different initialization heuristics;
2. Organize and maintain the bounded lookup table;
3. Explore and exploit using preventive returns and restocking.

We give more details on the algorithmic strategies below.

Use Different Initialization Heuristics

We use a mix of three simple initialization heuristics. The first one is based on the *cyclic tours* that Secomandi (2001) derives in his a priori approach. By considering all the possible customers being visited first in the route, we obtain Q-factor values associated with each customer visited at stage t, i.e., i_t. The second heuristic is a *randomized nearest neighbor* heuristic, where early replenishments are only done when this gives an immediate advantage. In the third heuristic, we use a variation of the *cone covering* method, introduced by Fisher and Jaikumar (1981) and then applied by Fan et al. (2006) to a similar problem. The advantage of the third heuristic lies in that it considers both the geographic location and the expected demand of each customer, thus generates routes with approximately the same expected total demand.

For the initialization of the Q-factors, we first cluster the customers into a few groups (depending on the expected number of replenishments). Then, we apply the nearest neighbor heuristic to the clusters to create one tour per group. Finally, we create one tour for all customers by applying a savings algorithm to the customers linked to the depot, until there are no intermediate visits to the depot in the tour. For the resulting customer sequence, we determine the optimal replenishment visits similar to Secomandi (2001). In the clustering, we apply randomization to get a larger solution set. For each of the heuristics, we find 600 sample paths to create an initial set of state-decision pairs and their values (Q-factors) in the lookup table. For state-decision pairs that are visited multiple times, we take the minimum of the Q-factors.

Organize and Maintain the Bounded Lookup Table

We denote the set of state-decision pairs that have the same i_t as an "i_t set", i.e., $\{(S_t, x_t)|S_t = (i_t, \cdot, \cdot)\}$, and let $|i_t|$ denote its size. That is, an i_t set consists of all possible combinations of the state-decision variables: (i_t, l_t, J_t) and the decision: (i_{t+1}, r_t) with the same i_t. We provide a detailed illustration of i_t sets in the Appendix. To control the exponential growth of the lookup table, we limit the number of stored Q-factors in each i_t set to a *maximum* (denoted as $|i_t|_{max}$).

We use a three-level pruning to maintain the bounded lookup table. Pruning Procedure I examines and maintains the size of each i_t set at each iteration, while Pruning Procedures II and III provide additional maintenance and value update of the i_t set every fixed number of iterations.

Pruning Procedure I:

When the size of an i_t set reaches $|i_t|_{max}$, we perform the pruning procedure I (Algorithm 6). We remove the state-decision pairs that have been visited only once. In addition, we remove the state-decision pairs that have a value higher than the average value and also a number of visits less than the average number of visits in the i_t set. Usually, about half of the state-decision pairs is removed in this procedure. Note that, apart from the Q-factor, we also record the number of visits to each state-decision pair, $n'(S_t, x_t)$, which is used in Pruning Procedure I as well as in Equation (6.15) of the value updating procedure.

Based on our numerical evaluation, we set $|i_t|_{max}$ as 150 state-decision pairs for instances with less than 20 customers and 250 state-decision pairs for larger instances. Higher $|i_t|_{max}$ values lead not only to longer computational time but also to poorer results, because the most relevant Q-factors will be updated less often.

Pruning Procedures II and III:

Different from Pruning Procedure I, Pruning Procedures II and III limit *the number of potential next customers* in the i_t set. These two procedures are implemented every fixed number of iterations.

In Pruning Procedure II (Algorithm 7), we look at the decision on the next customer to

Algorithm 6 Pruning Procedure I

Step 1. Check the size of the i_t set.
Step 2. Keep the state-decision pairs in the i_t set, if $|i_t| < |i_t|_{max}$.
Step 3. If the size of the i_t set reaches its maximum ($|i_t| \geq |i_t|_{max}$), remove the state-decision pairs
from the i_t set under the following criteria:
Step 3a. If their value is greater than the average Q-factor value of the i_t set, that is,
$Q(S_t, x_t) \geq (\sum_{\{(S_t,x_t)|S_t=(i_t,\cdot,\cdot)\}} Q(S_t, x_t))/|i_t|$;
and if also their number of visits is less than the average number of visits of the i_t set, that is,
$n'(S_t, x_t) \leq (\sum_{\{(S_t,x_t)|S_t=(i_t,\cdot,\cdot)\}} n'(S_t, x_t))/|i_t|$.

visit (i_{t+1}) among the state-decision pairs in the i_t set. For each i_{t+1}, we calculate the average Q-factor value of the state-decision pairs with $x_t = (i_{t+1}, \cdot)$, denoted as $\bar{Q}_{i_{t+1}}$. We only keep the state-decision pairs whose $\bar{Q}_{i_{t+1}}$ are among the best k_t potential next customers. The number k_t depends on the stage t, and decreases during the ADP, to gradually focus more on the best decisions on the potential next customer to visit.

Algorithm 7 Pruning Procedures II

Step 1. For each i_{t+1} of the state-decision pairs in the i_t set, calculate the average Q-factor
$\bar{Q}_{i_{t+1}} = (\sum_{\{(S_t,x_t)|S_t=(i_t,\cdot,\cdot),x_t=(i_{t+1},\cdot)\}} Q(S_t, x_t))/n(i_t, i_{t+1})$,
where $n(i_t, i_{t+1})$ denotes the number of state-decision pairs in the i_t set with the same i_{t+1}.
Step 2. Sort the customers, i_{t+1}, according to an ascending order of $\bar{Q}_{i_{t+1}}$.
Step 3. Select the first k_t customers and remove all the state-decision pairs (S_t, x_t) whose
next customer, i_{t+1}, are not among these k_t customers.

Pruning Procedure III uses the double pass DP approach, and at the same time *updates the Q-factors*. In this procedure, we update the Q-factors in the backward pass while pruning the lookup table in the forward pass.

When we have a sample path, we only update the values of the state-decision pairs that we actually visit as in Algorithm 5. To obtain better estimates, we also update the values of the state-decision pairs that use a part of the sample path. To achieve this, we perform an update process every fixed number of iterations by using a backward DP algorithm for all state-decision pairs in the bounded lookup table. For the update process, we start from the last stage and compute the minimum expected cost for each state-decision pair in the bounded lookup table. The values of the state-decision pairs, including the ones using a part of the sample path, are updated using the

minimum expected cost, and recursively, the updated values are used for the update of the Q-factors in previous stages.

After the backward DP, we remove the state-decision pairs with the same next customer, i_{t+1} that never give the minimum expected value, $Q_{min}(S_t)$, in the backward DP procedure. The update and the pruning procedure is described in Algorithm 8.

Algorithm 8 Pruning Procedures III

Step 1. (Backward DP) For all $t = N, N-1, \ldots, 0$:
Step 1a. Find the minimum state value among the possible next customers:
$Q_{min}(S_t) =$
$\min_{i_{t+1}}[\mathbb{E}[(C_t(S_t, x_t)] + \min_{x_{t+1} \in \mathcal{X}_{t+1}}(SD(S_{t+1}, S'_{t+1}) + Q(S_{t+1}, x_{t+1}))]$.
Step 1b. Update all $Q_t(S_t, x_t)$ in the lookup table with the decision x_t using:
$Q_t(S_t, x_t) = (1 - \alpha_{n-1})Q_t(S_t, x_t) + \alpha_{n-1}Q_{min}(S_t)$.
Step 2. (Forward DP) Pruning: do for all $t = 0, 1, \ldots, N+1$:
Remove the state-decision pairs with the same i_{t+1} that are never qualified for $Q_{min}(S_t)$.

Updating Procedure:

In every sample path simulation, we update the value of the visited state-decision pairs using the harmonic stepsize rule.

$$\alpha_{n-1} = \frac{a}{a + n'(S_t, x_t) - 1}, \tag{6.15}$$

where a is a constant. Note that the stepsize α_{n-1} depends on the number of visits to the state-decision pair, $n'(S_t, x_t)$, rather than the iteration counter n.

In VFA$^+$, the value of a depends on the decision. If all decisions taken after the visit of a state-decision pair in the sample path are optimal (exploitation decisions), we assign a high value to a. If however after the visit of a state-decision pair, we take an exploration decision, we either assign a low value to a (in case of improvement) or set a equal to zero (in case of non-improvement). In this way, we avoid that the state-decision pair becomes unattractive by not considering the best route afterwards.

Exploration and Exploitation using Preventive Returns and Restocking

In each step of the ADP algorithm, we may select the best decision based on the current Q-factors in the bounded lookup table (exploitation), or we may select an alternative decision in order to discover potentially better decisions (exploration). In VFA+, the exploitation options are:

- the decision with the best Q-factor value in the lookup table (and perfect match for remaining customers and capacity),

- the decision with the lowest sum of the Q-factor and the *state difference* in the lookup table.

The exploration options are:

- the decision with the second best Q-factor value (as this is the most promising alternative),
- the decision where the next customer is randomly selected from the lookup table, combined with a randomized early replenishment decision r_t,
- the decision where the next customer, i_{t+1} is randomly selected from the set of m_t nearest (in distance) unvisited customers, combined with a randomized early replenishment decision r_t. The value of m_t may be different for different stages.

The options above are considered with different probabilities. Note that once we have applied an exploration decision, the further decisions are preferably exploitation decisions, in order to obtain good cost estimates for this explorative decision.

At the end of the tour, when the vehicle capacity gets scarcer, it makes sense to return to the depot for early replenishment if the vehicle is close to the depot. Using the demand probability distribution of the non-visited customers, we determine the minimum expected number of remaining depot visits, both with and without returning to the depot. If the difference between the two values exceeds 0.9, we first return to the depot and then serve the remaining customers.

Parameter Settings

The algorithmic strategies in VFA$^+$ described above are all designed to improve the computational performance, but they also create a large number of parameter settings, such as probabilities in the sample path selection, updating parameters, parameters in the maintenance of the bounded lookup table, etc.

We conduct a number of preliminary tests to set the parameter settings to be used. Based on our preliminary computational evaluation, we fix some of the parameter settings and limit the possibilities for others to two or three options. For instance, the total duration of the ADP is fixed on $250,000$ iterations; Pruning Procedures II and III are performed every $10,000$ iterations; the value a in the harmonic stepsize rule, Equation (6.15) is set to 0 (exploration without improvement), 1 (exploration with improvement), or 5 (exploitation). We use this setting for all instances and report the results in the chapter.

6.5. Experimental Design

In this section, we describe the test instances used in the numerical experiments and our methodology to evaluate the algorithms.

6.5.1 Test Instances

We use two sources of instance generation in the literature, from Secomandi (2001) and Solomon (1987). In both sets of test instances, we consider delivery to customers from the depot. We use the test instances of Secomandi (2001) to compare our value function approximation algorithms (VFA and VFA^{+}) with the Rollout algorithm in Secomandi (2001). We also test the algorithms with the Solomon instances which is a standard reference in the VRP literature. Two different sets of Solomon instances are generated to evaluate the effect of the depot location, which is either at the center or at the corner.

The first set of instances are based on Secomandi (2001). The instances are generated with different number of customers. Specifically, the instances with 5 to 19 customers are denoted as the "small size instances," and the instances with $20, 30, \ldots, 60$ customers are denoted as the "large size instances." The test instances also differ in the values of the expected fill rate $\bar{f} = \sum_{i=1}^{N} E(D_i)/Q$. $\bar{f}' \equiv max\{0, \bar{f} - 1\}$ can be viewed as the expected number of route failures, and $\bar{f}'$ is in the set $\{0.75, 1.25, 1.75\}$. Therefore, there are 3 variants for each small size and large size instances. The values of Q for all possible $(\bar{f}', N)$ pairs are computed by rounding $3N/\bar{f}$ to the nearest integer. For each instance, the customer demands are divided into low, medium, and high categories, following three discrete uniform probability distributions. Every customer is assigned to one of these demand categories with equal probability $(1/3)$. For the small size instances ($N < 20$), the demand categories are $U(1,3), U(2,4), U(3,5)$, and for the large size instances ($N \geq 20$), they are $U(1,5), U(6,10), U(11,15)$. The depot is fixed at the corner. For each $(\bar{f}', N)$ pair, 10 replications are generated for each of the small size instances and 5 replications are generated for each of the large size instances. We refer to these instances as the "Secomandi instances" in the rest of the chapter.

The Solomon instances are based on the RC instances ($RC101$ to $RC105$ and $RC201$ to $RC205$) from Solomon (1987). As the number of the customers and the demand distributions in Solomon (1987) are not comparable to the Secomandi instances, we modify the Solomon instances in two ways: the demand distribution and the customer selection. The demand distributions are modified as follows. Originally, the customer demands in the Solomon instances are between 0 and 40. We denote the demand types in the Solomon instances between the intervals $(0, 10]$, $(10, 20]$, $(20, 30]$, $(30, 40]$ as 1, 2, 3, 4, respectively. We then assign the demand distributions $U(0,4), U(2,6), U(4,8)$ and $U(6,10)$ to the four demand types respectively. The customer selection process is based on the customer ready times. The customers are ordered based on their ready times without using their time windows. We pick the

first N customers to construct our instances. For the small size instances, we select the first 5 to 15 customers. The vehicle capacity Q is then set to $[8N/1.75]$. After the modification, we generate two instance sets according to the location of the depot: Solomon A, where the depot is located at the center, and Solomon B, where the depot is located at the corner. For Solomon A instances, the depot is located at $(40, 50)$, which is approximately at the center of the customers. For Solomon B instances, we swap the depot located at $(40, 50)$ and the customer located at $(5, 5)$. We generate 10 replications for each of the Solomon instances.

6.5.2 Evaluation Methodology

We study three algorithms for the VRPSD: the Rollout algorithm, the VFA, and the VFA$^+$. The solution quality is evaluated by policy simulation and the solution time is measured by the central processing unit (CPU) time. We evaluate the policy of each algorithm using simulation with a sample size of 2000 and report the mean of the evaluated objective values and solution times.

In the simulation, the same set of random seeds are used such that the different algorithms use the same demand samples. We report the improvements of VFA and VFA$^+$ relative to the solution from the Rollout algorithm. For small size instances, we find the optimal objective values by solving a standard backward MDP using Equation (6.9). We also report the optimal objective values and their improvements compared to the Rollout algorithm. Note that we choose the Rollout algorithm as the benchmark because the optimal policy can only be obtained for small size instances.

6.6. Numerical Results

This section provides the numerical results and analysis. All evaluations are run on a computer with an Intel Xeon CPU X7560 (2.27GHz) and 63.9GB RAM. The programming language is Java.

Algorithm Comparison: Solution Quality

We first compare the solutions quality of the algorithms for the VRPSD. Specifically, we compare the evaluated objective values of the Rollout algorithm, the VFA, the VFA$^+$, as well as the optimal values (for small size instances). Further, we also provide the improvements of the latter three algorithms relative to the Rollout algorithm.

Table 6.1 summarizes the results for the Secomandi instances for both small and large size instances. The entries are the averages of all variants of the fill rate and demand distribution for each instance size. When we consider the instances with $N \leq 15$, the optimal algorithm performs on average 3.78% better than the Rollout algorithm. On the same instances with $N \leq 15$, we see that both the VFA and the VFA$^+$ on average perform better than the Rollout algorithm. For instance, the VFA$^+$

on average improves the solution of the Rollout algorithm by 1.97%, covering the performance gap between the Rollout and the optimal solution by more than 50%.

Table 6.1 also demonstrates that the difference between the solution quality of the VFA$^+$ and VFA is on average not very large for small size instances with $N \leq 15$. However, as the problem size increases from $N > 8$, the performance of the VFA algorithm gets worse whereas the VFA$^+$ algorithm still outperforms the Rollout algorithm.

When the number of customers increases from 16 to 19, obtaining the optimal solution becomes computationally intractable. For these instances, the performance of the VFA gets worse than the Rollout algorithm. The VFA$^+$, however, still outperforms the Rollout algorithm.

For large size instances ($N \geq 20$), both the optimal algorithm and the VFA become computationally intractable. The VFA$^+$ algorithm outperforms the Rollout algorithm and the percentage improvement of the VFA$^+$ algorithm increases from 1.87% (small size instances) to 2.97% (large size instances).

Table 6.1 Overview of the performance of the Rollout, VFA, VFA$^+$ and the Optimal Algorithm on Secomandi instances

Secomandi N	Rollout Value	VFA Value	VFA$^+$ Value	Opt Value	VFA Imprv.*%	VFA$^+$ Imprv.*%	Opt Imprv.*%
5	5.51	5.37	5.55	5.34	2.41%	-0.75%	3.07%
6	5.40	5.28	5.29	5.21	2.24%	2.08%	3.55%
7	5.27	5.13	5.13	5.07	2.76%	2.67%	3.90%
8	5.54	5.44	5.40	5.31	1.82%	2.57%	4.12%
9	5.36	5.29	5.25	5.16	1.36%	2.10%	3.62%
10	5.44	5.35	5.33	5.26	1.59%	2.09%	3.41%
11	6.08	6.01	5.97	5.84	1.10%	1.88%	3.90%
12	6.27	6.20	6.11	5.95	1.26%	2.56%	5.10%
13	6.20	6.10	6.11	6.00	1.56%	1.42%	3.19%
14	6.21	6.14	6.08	6.01	1.22%	2.08%	3.29%
15	6.22	6.10	6.04	5.95	1.92%	2.91%	4.31%
Average	5.77	5.67	5.66	5.56	1.73%	1.97%	3.78%
16	6.64	6.61	6.51		0.40%	1.89%	
17	6.36	6.41	6.28		-0.71%	1.27%	
18	6.23	6.27	6.12		-0.54%	1.89%	
19	6.66	6.72	6.57		-0.98%	1.36%	
Average (Small size instances)	5.96	5.89	5.85		1.10%	1.87%	
20	6.86	6.73	6.61		1.81%	3.64%	
30	7.51	7.52	7.21		-0.10%	4.04%	
40	8.14		7.98			1.98%	
50	8.49		8.29			2.29%	
60	9.07		8.79			3.09%	
Average (Large size instances)	8.01		7.78			2.97%	
Grand Average	6.47		6.33			2.21%	

* Percentage cost improvement relative to the Rollout algorithm.

Tables 6.2 and 6.3 present the results of the Solomon A and the Solomon B instances with the number of customers: $N \leq 15$. The results represented are the average values for each instance size. The left part of the table shows the results when the depot is approximately at the center of the customers (Solomon A) and the right part of the table presents the results when the depot is approximately at the corner (Solomon B). We use these two sets of instances to analyze the effect of the depot location on the performance of the algorithms.

Table 6.2 Overview of the performance of the Rollout, VFA, VFA$^+$ and the Optimal Algorithm on Solomon A instances

	Solomon A (The Depot at the Center)						
	Rollout	VFA	VFA$^+$	Opt	VFA	VFA$^+$	Opt
N	Value	Value	Value	Value	Imprv.*%	Imprv.*%	Imprv.*%
5	1.423	1.406	1.391	1.387	1.23%	2.31%	2.57%
6	1.645	1.652	1.623	1.612	-0.44%	1.35%	2.01%
7	1.854	1.849	1.862	1.836	0.22%	-0.44%	0.96%
8	2.090	2.079	2.106	2.065	0.54%	-0.75%	1.21%
9	2.290	2.292	2.285	2.257	-0.08%	0.20%	1.43%
10	2.358	2.350	2.336	2.323	0.32%	0.92%	1.47%
11	2.502	2.513	2.487	2.468	-0.41%	0.60%	1.38%
12	2.662	2.673	2.642	2.633	-0.39%	0.77%	1.09%
13	2.714	2.717	2.693	2.673	-0.10%	0.80%	1.51%
14	2.905	2.921	2.892	2.874	-0.56%	0.44%	1.04%
15	3.014	3.017	2.981	2.954	-0.12%	1.09%	1.99%
Average	2.314	2.315	2.300	2.280	-0.05%	0.63%	1.47%

* Percentage cost improvement relative to the Rollout algorithm.

Table 6.3 Overview of the performance of the Rollout, VFA, VFA$^+$ and the Optimal Algorithm on Solomon b instances

	Solomon B (The Depot at the Corner)						
	Rollout	VFA	VFA$^+$	Opt	VFA	VFA$^+$	Opt
N	Value	Value	Value	Value	Imprv.*%	Imprv.*%	Imprv.*%
5	2.811	2.799	2.791	2.779	0.43%	0.74%	1.14%
6	2.817	2.776	2.775	2.766	1.46%	1.49%	1.81%
7	2.789	2.718	2.712	2.699	2.56%	2.76%	3.22%
8	3.047	2.911	2.935	2.890	4.48%	3.70%	5.16%
9	3.146	3.031	3.060	3.007	3.63%	2.73%	4.42%
10	3.086	3.054	3.038	2.996	1.04%	1.58%	2.92%
11	3.261	3.230	3.217	3.169	0.95%	1.36%	2.83%
12	3.387	3.369	3.334	3.300	0.53%	1.57%	2.55%
13	3.358	3.335	3.331	3.283	0.67%	0.79%	2.22%
14	3.605	3.558	3.590	3.489	1.31%	0.43%	3.22%
15	3.673	3.607	3.627	3.519	1.78%	1.25%	4.18%
Average	3.180	3.126	3.128	3.082	1.69%	1.64%	3.09%

* Percentage cost improvement relative to the Rollout algorithm.

The results for Solomon A instances show that when the depot is at the center, the relative performance of the VFA, the VFA$^+$ algorithm and the optimal algorithm is

on average not much different from the performance of the Rollout algorithm. This suggests that the Rollout algorithm performs good and also the VFA algorithms and the optimal algorithm behaves similarly with the central depot.

When we look at Solomon B instances, the results are similar to the results of the Secomandi instances with $N \leq 15$. Both the VFA and the VFA$^+$ algorithm improve the solution of the Rollout algorithm by covering on average more than 50% of the performance gap between the Rollout and the optimal solution. Intuitively, moving the depot from the center to the corner increases the average customer-depot distance. Therefore, when the depot is at the corner, the penalty of a failure is higher as we have to travel on average longer distance back to the depot. This indicates that in value function algorithms, the decisions for which customer to go next and for the early replenishment are made efficiently such that the overall cost decreases.

Algorithm Comparison: Computational Times

Figure 6.1 shows the normalized computational times (in logarithmic scale) for the different algorithms in solving the Secomandi instances. The normalized time is calculated as the computational time relative to solving the instances with $N = 5$ customers. Figure 6.1 shows that the MDP solution time increases at a much higher rate than other algorithms. The VFA$^+$ has the slowest rate of increase as the number of the customers increases. This shows that the VFA$^+$ algorithm reduces the computational time significantly when compared to the optimal algorithm, the Rollout algorithm and the VFA while providing good quality solutions by using bounded lookup tables with efficient maintenance.

In Figure 6.1, there is a sudden jump up in the computational time of the VFA and Rollout algorithms from the small size (≤ 19 customers) to the large size instances (≥ 20 customers). This is because, in the experimental design of the large size instances, we use demand categories with a wider range that increases the number of the state-decision pairs (see Section 6.5.1). However, due to the use of bounded lookup tables with efficient maintenance, the VFA$^+$ algorithm successfully mitigates this increase in the state-decision space (see "state difference" explanation in Section 6.4.2).

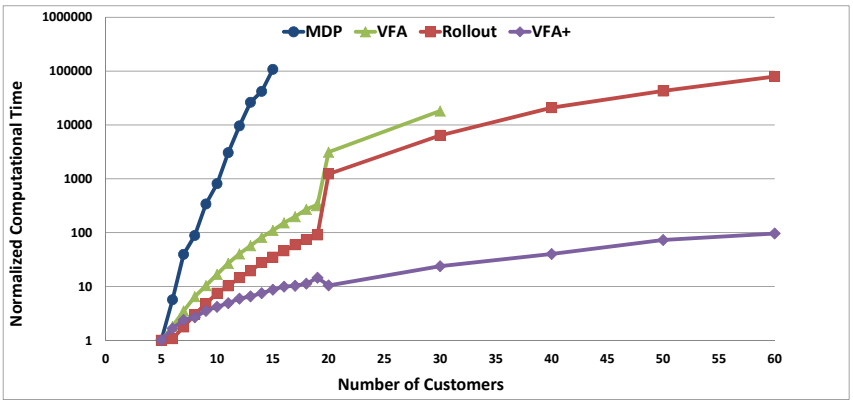


Figure 6.1 Computational times for different algorithms based on the Secomandi instances

6.7. Conclusions

This chapter deals with the single vehicle routing problem with stochastic demands (VRPSD). The VRPSD is a difficult stochastic combinatorial optimization problem that becomes intractable for large size instances. In this chapter, we formulate a multi-stage stochastic dynamic programming model and implement Approximate Dynamic Programming (ADP) algorithms to overcome the curses of dimensionality for the VRPSD. The ADP algorithms are based on Value Function Approximations (VFA) with a lookup table representation. The standard VFA is improved for VRPSD with a Q-learning algorithm with bounded lookup tables and efficient maintenance (VFA$^+$), as well as exploration-and-exploitation strategies using preventive returns and restocking.

We validate and benchmark our proposed algorithms using test instances in the literature. The VFA$^+$ algorithm obtains good quality solutions with shorter computational time, especially for large size instances. For small size instances where the optimal solutions are available, VFA$^+$ improves the Rollout algorithm by covering on average more than 50% of the performance gap between the Rollout and the optimal solutions. For large size instances, VFA$^+$ outperforms both VFA and the Rollout algorithm. This demonstrates the effectiveness in the algorithm design of VFA$^+$.

In Approximate Dynamic Programming, the use of post-decision states helps to capture the state of the system immediately after the decision making but before the new (exogenous) information arrives. It also helps to avoid the expectation within the min or max operator. However, in a stochastic combinatorial optimization problem such as VRPSD, we need both the state and the decision information to evaluate the impact of the decision, where the practice of post-decision states appears to be inappropriate. Therefore, we improve the ADP algorithm with a Q-learning algorithm where we store the values of state-decision pairs, i.e., Q-factors. It however suffers more from the curses of dimensionality. We design bounded lookup tables with efficient maintenance to overcome this. Further, we design exploration-and-exploitation strategies using preventive returns for VRPSD. The combination of the above algorithmic strategies appear to play an important role in making better routing and restocking decisions in VRPSD. This chapter provides an exploratory algorithmic research on the application of Q-learning algorithms with bounded lookup table and efficient maintenance as well as exploration-and-exploitation strategies in dealing with difficult stochastic and combinatory problems. More in-depth research is called for along this line.

6.A. Appendix

A The illustration for state-decision pairs and i_t sets

In Figure 6.1, an illustration for the state-decision pairs and i_t sets is given for a small single vehicle routing problem. In this example, there are only 3 customers, the capacity of the vehicle is 4 units and the demand of each customer follows a discrete uniform distribution U(1,2). Here, we only show the branch from the customer 1 at state 1 until the termination state which ends at the depot. The i_t sets are grouped according to the current customer.

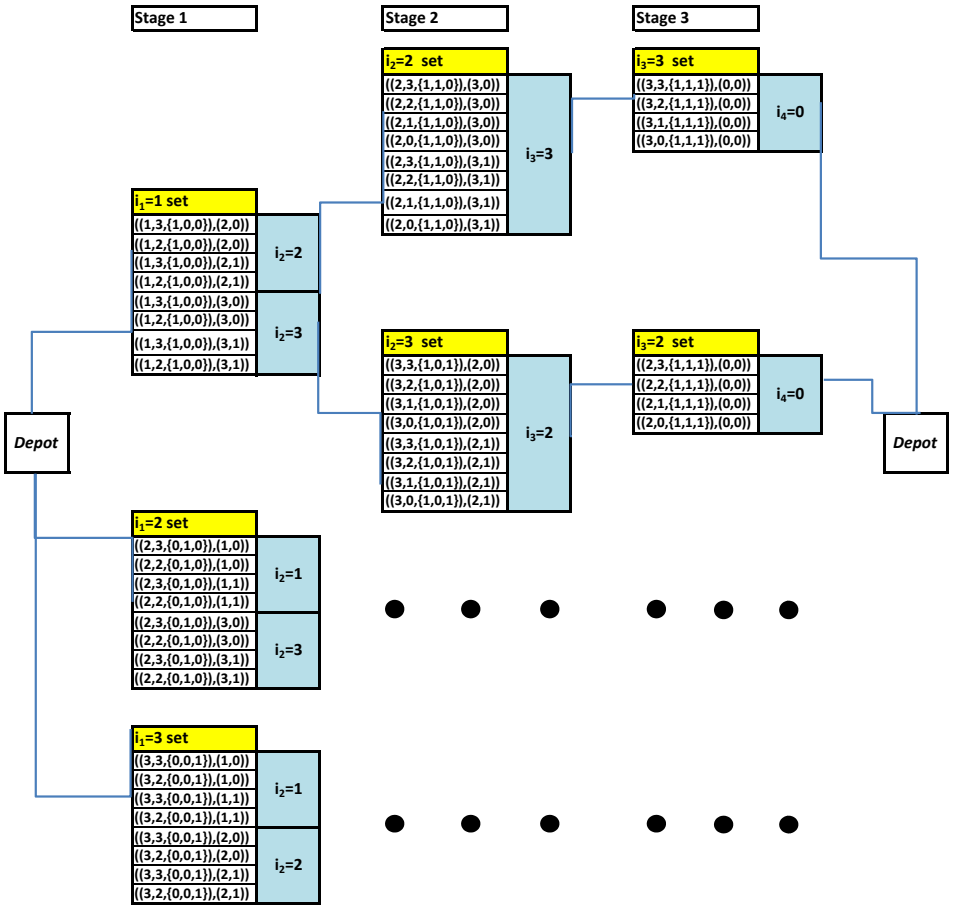


Figure 6.1 The illustration for state-decision pairs and i_t sets

Chapter 7

Conclusions

This chapter presents the conclusions of the research presented in this thesis. First, we provide the results from the research in Section 7.1. Section 7.2 concludes this chapter with directions for future research.

7.1. Results

The traffic conditions and customer profiles in transportation networks are dynamic and stochastic where travel times are non-stationary due to random traffic disruptions and customer demands are uncertain. Today's information system technology provides the planners a wide range of information including real-time information. This creates an opportunity for researchers to develop algorithms using dynamic information. Providing routing algorithms that are simple, fast, accurate and feasible for real-life situations becomes crucial as the traffic networks become more dynamic and congested. In the coming future, the dynamic routing algorithms will be frequently used with the advances in in-car navigation systems. According to Van De Weijer and Rutten (2012), travelers tend to follow information from in-car navigation systems rather than central signals. Therefore, the implications from the thesis play an important role for determining the value of using real-time and probabilistic information to be used in-car navigation systems. The research presented in this thesis aims to develop fast and efficient dynamic routing algorithms both for dynamic shortest path problems with network disruptions and vehicle routing problems with stochastic demands. In this way, with the advancements of traffic information predictions, these dynamic routing algorithms can be implemented in navigation technologies such as TomTom.

In the thesis, our objective is to minimize the total expected travel time. This choice is due to the high value of travel time for both passenger and freight transportation. We provide the experimental results by discussing the percentage decrease in travel

time. For freight transportation, this indicates reduction in fuel costs, less overtime work for employees due to uncertainty and higher service levels. For passenger transportation, it indicates lower fuel costs and improvement in societal effects such as less stress due to traffic. However, in real-life, the implications of the travel time reduction is, however more. According to the traffic studies (Hansen 2001, Koopmans and Kroes 2003) time spent in traffic congestion causes external costs such as air pollution and accidents which are more costly than the fuel costs, etc. Therefore, the magnitude of savings for unit travel time will be more if we also consider societal and environmental costs.

As discussed in Chapter 1, we aim to develop dynamic routing policies that are accurate, simple and fast. First of all, the numerical results show that the algorithms provide good quality solutions with small percentage gaps from the optimal solutions. Furthermore, considering Markov Decision Process (MDP) provides a simple framework for real-time usage by developing routing policies depending on the network state. The travelers observe the network states and choose the relevant policy. The dynamic routing policies developed in this thesis are fast and can be computed in feasible amount of time to be used in practice. The computational time differs from less than a second to less than a minute for small networks with low network states (hundred states) to large networks with higher network states (million states). Following, we shortly summarize our findings.

In Chapter 3, we analyzed the influence of using different levels of traffic information on the performance of routing policies in case of network disruptions. We compared the performance of the current known heuristics such as the offline and the online policies, with the hybrid routing policies. By testing different policies using different levels of stages, we analyzed the effect of having a sufficient level of online and offline information about the network for different networks. Moreover, we gained insights into the network conditions in which it is better to use what type of policies. We found that the usage of only limited online and predictive information leads to a significant decrease in computation time compared to the policies that use higher levels of information. We observed that the performance of hybrid policies only deteriorates marginally compared to the optimal solution. The average percentage gap between the optimal solution and the solution from the best hybrid dynamic algorithm is at most 1.38% with more that 100% computational time reductions. This shows that for practice, we do not need to collect real-time information for the complete network to obtain good performance. This result is beneficial when availability and reliability of online information are low and the cost of retrieving this information is high. Furthermore, we observed that under more complex networks, the more detailed information about the disruption's state becomes effective and cost efficient even if the network knowledge is limited with few stages with online and probabilistic information for the hybrid policies. Compared to traditional routing algorithms that do not consider the disruptions, the hybrid algorithms provides at least 30% cost reductions.

In Chapter 4, we extended the dynamic shortest path problems presented in Chapter 3 for larger networks with many disruption states. For large scale networks with many levels of disruptions, obtaining the optimal solution faces the curses of dimensionality, i.e., states, outcome, and decisions. Therefore, we used an Approximate Dynamic Programming (ADP) algorithm which is a well-known approximation approach. The ADP algorithms used in this chapter are based on the value function approximations using a lookup table representation. First, we employed a standard value function approximation algorithm with various algorithmic design variations for updating the state values with efficient exploration/ exploitation strategies. Then, we improved the value function approximation algorithm by developing a hybrid ADP with a deterministic lookahead policy and value function approximations using a clustering approach. We also used the hybrid algorithms developed in Chapter 3 as a benchmark. We developed a test bed of networks to evaluate the efficiency (both in computational time and solution quality) of our algorithms. In our numerical analysis, we showed that the hybrid ADP algorithms with the clustering approach outperforms the standard ADP algorithms. Furthermore, we observed that considering a shorter horizon for the deterministic lookahead policy improves the solution quality. The hybrid ADP algorithms outperforms the hybrid dynamic algorithm as the network size gets larger and the disruption rate gets higher. The computational time of the hybrid ADP algorithms shows the slowest rate of increase with respect to the exponential increases in the state space. The computational time of the benchmark heuristic increases at a higher rate when the number of disruption levels increases. Although the benchmark heuristic has relatively good solution quality, for large scale networks with many disruption levels the hybrid ADP algorithms with the clustering approach becomes more attractive with lower computational time and high solution quality for practice.

In Chapters 3 and 4, we assumed that there is no propagation of disruptions and there is no link-dependency. In Chapter 5, we removed this constraint and considered the effect of backward propagation of disruptions in downstream roads to upstream roads due to limited capacity of roads. We denote this as the spillback effect. The spillback effect is a congestion propagation to an upstream link. We analyze the effect of ignoring and considering the spillback effect in the dynamic routing decisions with network disruptions. We incorporated the spillback effect into the MDP formulation. However, as the size of the network and the disruption levels increase, the computational complexity increases causing the MDP become computationally intractable. Therefore, to reduce the computational time, we used a hybrid dynamic programming algorithm developed in Chapter 3. We also provided online and static algorithms mostly used in practice for the routing decisions. We used a test bed of different network structures with different levels of disruption rates and spillback rates. The numerical results show that considering the spillback effect has the highest impact on the solution quality when the network has higher number of vulnerable links. Moreover, the hybrid dynamic programming approach with the disruption and the spillback information for the limited part of the network significantly reduces the

computational time while providing higher solution quality than the full information model that ignores the spillback effect.

In Chapter 6, we dealt with the single vehicle routing problem with stochastic demands (VRPSD). The VRPSD is a difficult stochastic combinatorial optimization problem that becomes intractable for large size instances. In this chapter, we formulated a stochastic dynamic programming model and implement Approximate Dynamic Programming (ADP) algorithms to overcome the curses of dimensionality for the VRPSD. The ADP algorithms are based on Value Function Approximations (VFA) with a lookup table representation. The standard VFA was extended and improved for the single VRP with stochastic demands (VFA$^+$) with a Q-learning algorithm where we store the values of state-decision pairs, i.e., Q-factors. It however suffers more from the curses of dimensionality. We designed bounded lookup tables with efficient maintenance to overcome this. Further, we design exploration-and-exploitation strategies using preventive returns for VRPSD. The combination of the above algorithmic strategies appear to play an important role in making better routing and restocking decisions in VRPSD. We validated and benchmarked our proposed algorithms using test instances in the literature. The VFA$^+$ algorithm obtains good quality solutions with shorter computational time, especially for large size instances. For small size instances where the optimal solutions are available, VFA$^+$ improves the Rollout algorithm by covering on average more than 50% of the performance gap between the Rollout and the optimal solutions. For large size instances, VFA$^+$ outperforms both VFA and the Rollout algorithm. This demonstrates the effectiveness in the algorithm design of VFA$^+$.

7.2. Future Research Directions

The research presented here can be extended in several directions.

In Chapters 3-5, we focus on time-invariant and travel-time-dependent traffic information. The dynamic shortest path algorithms presented can be extended by considering time-dependent travel times. This can be done by adding a time dimension in the network state definition. By this way, one can also incorporate the effect of daily fluctuations explicitly into the model. Furthermore, in Chapters 3 and 4 we assumed only travel-time-dependency based on the travel time of the link being traversed. The models can be extended with whole network dependency. For instance, when the weather conditions are extreme, all of the links in the network are affected.

The input information processing and availability are crucial for the dynamic shortest path problems. In this thesis, we generate random networks that represent highway road networks in the Netherlands with wide variety of network properties. However, for the future research performing the routing algorithms with real-life data is also essential. Fortunately, there are new projects such as Daipex (Dutch Institute for Advanced Logistics 2013) that will create the opportunity to develop real-time traffic

data processing mechanism. By this way, the logistics information can be retrieved in real-time. In addition to this, the effect of receiving delayed real-time information and low quality of information can be investigated.

In the dynamic shortest path problems, we focused on minimizing the cost of a single traveler. However, in terms of system equilibrium, considering travelers as a system is a way to improve the overall traffic. For instance, when the routing softwares suggest most of the travelers to travel to the same road, then also that road will become congested. This research can be further extended for traffic equilibrium.

The results for the dynamic shortest path problems in Chapters 3-5 provide implications for improving the passenger transportation. For the future research, the dynamic routing algorithms can be implemented to freight transportation by developing dynamic vehicle routing algorithms with dynamic and stochastic travel times. The approximate dynamic programming algorithms and limited lookahead policies can be extended to the vehicle routing problems.

Chapter 6 can be extended by considering multiple vehicle case. This will increase the state-space compared to the single vehicle case. It is interesting to investigate the performance of the Rollout and ADP algorithms with Q-learning with this problem setting. Furthermore, the developed Q-learning algorithm can also be implemented for the dynamic vehicle routing algorithms with dynamic travel times.

Throughout the thesis, we used total expected travel time as the performance measure. The performance measure may also include travel-time reliability and also societal costs such as reducing CO2 emissions. Furthermore, in dynamic routing algorithms, the preference of the travelers such as not visiting a specific intersection or having a priority in visiting the customers can be considered as a future direction.

Bibliography

Ak, A., A. L. Erera. 2007. A paired-vehicle recourse strategy for the vehicle-routing problem with stochastic demands. *Transportation Science* **41**(2) 222–237.

Attanasio, A., J. Bregman, G. Ghiani, E. Manni. 2007. Real-time fleet management at ecourier ltd. *Dynamic Fleet Management*. Springer, 219–238.

Bander, J. L., C. C. White. 2002. A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost. *Transportation Science* **36**(2) 218–230.

Barceló, J., H. Grzybowska, S. Pardo. 2007. Vehicle routing and scheduling models, simulation and city logistics. *Dynamic Fleet Management*. Springer, 163–195.

Barry, J. L., L. P. Kaelbling, T. Lozano-Pérez. 2011. Deth: approximate hierarchical solution of large markov decision processes. *Proceedings of the twenty-second international joint conference on artificial intelligence*, vol. 3. 1928–1935.

Ben-Akiva, M., M. Bierlaire, J. Bottom, H. Koutsopoulos, R. Mishalani. 1997. Development of a route guidance generation system for real-time application. *Proceedings of the IFAC Transportation Systems 97 Conference, China*.

Bent, R., P. Van Hentenryck. 2004a. Regrets only, online stochastic optimization under time constraints. *Proceedings of the National Conference on Artificial Intelligence*. 501–506.

Bent, R. W., P. Van Hentenryck. 2004b. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research* **52**(6) 977–987.

Bertsekas, D. P. 2012. *Dynamic programming and optimal control: Approximate dynamic programming*, vol. 2. 4th ed. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D. P., J. N. Tsitsiklis. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research* **16** 580–595.

Bertsekas, D. P., J. N. Tsitsiklis. 1995. Neuro-dynamic programming: an overview. *Proceedings of the 34th IEEE Conference on Decision and Control*, vol. 1. 560–564.

Bertsekas, D. P., H. Yu. 2010. Q-learning and enhanced policy iteration in discounted dynamic programming. Tech. rep., Lab. for Information and Decision Systems Report 2831, MIT.

Bertsimas, D. J. 1992. A vehicle routing problem with stochastic demand. *Operations Research* **40**(3) 574–585.

Blok, H. 2011. Markov decision processes with unbounded transition rates: structural properties of the relative value function. Master's thesis, Utrecht University.

Bock, S. 2010. Real-time control of freight forwarder transportation networks by integrating multimodal transport chains. *European Journal of Operational Research* **200**(3) 733–746.

Bonet, B. 2007. On the speed of convergence of value iteration on stochastic shortest-path problems. *Mathematics of Operations Research* **32**(2) 365–373.

Bottom, J., M. Ben-Akiva, M. Bierlaire, I. Chabini, H. Koutsopoulos, Q. Yang. 1999. Investigation of route guidance generation issues by simulation with dynamit. *Proceedings of 14th International Symposium on Transportation and Traffic Theory, Jerusalem*.

Boutilier, C., R. Dearden, M. Goldszmidt. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* **121**(12) 49–107.

Brown, G. G., C. J. Ellis, G. W. Graves, D. Ronen. 1987. Real-time, wide area dispatch of mobil tank trucks. *Interfaces* **17**(1) 107–120.

CE Delft. 2013. Transport economics- external costs of transport in Europe. www.cedelft.eu/publicatie/external_costs_of_transport_in_europe.

Chen, X., J. Hu, X. Hu. 2009. A new model for path planning with interval data. *Computers and Operations Research* **36** 1893–1899.

Chen, Z.-L., H. Xu. 2006. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science* **40**(1) 74–88.

Cheung, R. K. 1998. Iterative methods for dynamic stochastic shortest path problems. *Naval Research Logistics (NRL)* **45**(8) 769–789.

Christiansen, C. H., J. Lysgaard. 2007. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters* **35**(6) 773–781.

Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin, F. Semet. 2002. A guide to vehicle routing heuristics. *Journal of the Operational Research society* 512–522.

Cortés, C. E., A. Núnez, D. Sáez. 2008. Hybrid adaptive predictive control for a dynamic pickup and delivery problem including traffic congestion. *International Journal of Adaptive Control and Signal Processing* **22**(2) 103–123.

Daganzo, C. F. 1994. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological* **28**(4) 269–287.

Daganzo, C. F. 1995. The cell transmission model, part ii: network traffic. *Transportation Research Part B: Methodological* **29**(2) 79–93.

Daganzo, C. F. 2006. In traffic flow, cellular automata= kinematic waves. *Transportation Research Part B: Methodological* **40**(5) 396–403.

Dantzig, G. B., J. H. Ramser. 1959. The truck dispatching problem. *Management science* **6**(1) 80–91.

Davies, C., P. Lingras. 2003. Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks. *European Journal of Operational Research* **144**(1) 27–38.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* **1**(1) 269–271.

Donati, A. V., R. Montemanni, L. M. Gambardella, A. E. Rizzoli. 2003. Integration of a robust shortest path algorithm with a time dependent vehicle routing model and applications. 2–31.

Dror, M., G. Laporte, F. V. Louveaux. 1993. Vehicle routing with stochastic demands and restricted failures. *Mathematical Methods of Operations Research* **37**(3) 273–283.

Dror, M., G. Laporte, P. Trudeau. 1989. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science* **23**(3) 166–176.

Dutch Institute for Advanced Logistics. 2013. Dinalog R&D projects- Daipex. http://www.dinalog.nl/en/projects/.

Ehmke, J. F., S. Meisel, S. Engelmann, D. C. Mattfeld. 2009. Data chain management for planning in city logistics. *International Journal of Data Mining, Modelling and Management* **1**(4) 335–356.

Ehmke, J. F., S. Meisel, D. C. Mattfeld. 2010. Floating car data based analysis of urban travel times for the provision of traffic quality. *Traffic data collection and its standardization*. Springer, 129–149.

Eksioglu, B., A. V. Vural, A. Reisman. 2009. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering* **57**(4) 1472–1483.

Eliasson, J. 2004. Car drivers valuations of travel time variability, unexpected delays and queue driving. European Transport Conference.

Erera, A. L., J. C. Morales, M. Savelsbergh. 2010. The vehicle routing problem with stochastic demand and duration constraints. *Transportation Science* **44**(4) 474–492.

European Environment Agency. 2013. Indicators and fact sheets about transport demand in Europe. www.eea.europa.eu/data-and-maps/indicators.

Fan, J., X. Wang, H. Ning. 2006. A multiple vehicles routing problem algorithm with stochastic demand. *The Sixth World Congress on Intelligent Control and Automation*, vol. 1. 1688–1692.

Ferris, M. C., A. Ruszczynski. 2000. Robust path choice in networks with failures. *Networks* **35** 200–218.

Ferrucci, F. 2013. Introduction to tour planning: Vehicle routing and related problems. *Proactive Dynamic Vehicle Routing*. Springer, 15–79.

Field, A. 2009. *Discovering Statistics Using SPSS*. Sage Publications Limited.

Fisher, M. L., R. Jaikumar. 1981. A generalized assignment heuristic for vehicle routing. *Networks* **11**(2) 109–124.

Fleischmann, B., S. Gnutzmann, E. Sandvoß. 2004. Dynamic vehicle routing based on online traffic information. *Transportation science* **38**(4) 420–433.

Fu, L. 1996. Real-time vehicle routing and scheduling in dynamic and stochastic traffic networks. Tech. rep., University of Waterloo.

Fu, L. 2001. An adaptive routing algorithm for in-vehicle route guidance systems with real-time information. *Transportation Research Part B: Methodological* **35**(8) 749–765.

Fu, L., L. R. Rilett. 1998. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research Part B: Methodological* **32**(7) 499–516.

Gao, S., I. Chabini. 2006. Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B: Methodological* **40**(2) 93–122.

Gao, S., H. Huang. 2012. Real-time traveler information for optimal adaptive routing in stochastic time-dependent networks. *Transportation Research Part C: Emerging Technologies* **21**(1) 196–213.

Gendreau, M., F. Guertin, J.-Y. Potvin, E. Taillard. 1999. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science* **33**(4) 381–390.

Gendreau, M., G. Laporte, R. Séguin. 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science* **29**(2) 143–155.

Gendreau, M., G. Laporte, R. Séguin. 1996a. Stochastic vehicle routing. *European Journal of Operational Research* **88**(1) 3–12.

Gendreau, M., G. Laporte, R. Séguin. 1996b. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research* **44**(3) 469–477.

Gendreau, M., J.-Y. Potvin. 1998. *Dynamic vehicle routing and dispatching*. Springer, US.

Gentile, G., L. Meschini, N. Papola. 2007. Spillback congestion in dynamic traffic assignment: a macroscopic flow model with time-varying bottlenecks. *Transportation Research Part B: Methodological* **41**(10) 1114–1138.

George, A., W. B. Powell. 2006. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning* **65** 167–198.

Ghiani, G., F. Guerriero, G. Laporte, R. Musmanno. 2003. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research* **151**(1) 1–11.

Givan, R., T. Dean, M. Greig. 2003. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence* **147**(1) 163–223.

Godfrey, G. A., W. B. Powell. 2002. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science* **36**(1) 21–39.

Goodson, J. C., J. W. Ohlmann, B. W. Thomas. 2012. Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand. *European Journal of Operational Research* **217**(2) 312–323.

Goodson, J. C., J. W. Ohlmann, B. W. Thomas. 2013. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research* **61**(1) 138–154.

Güner, A. R., A. Murat, R. B. Chinnam. 2012. Dynamic routing under recurrent and non-recurrent congestion using real-time its information. *Computers and Operations Research* **39**(2) 358–373.

Haghani, A., S. Yang. 2007. Real-time emergency response fleet deployment: Concepts, systems, simulation & case studies. *Dynamic Fleet Management*. Springer, 133–162.

Hall, R. W. 1986. The fastest path through a network with random time-dependent travel times. *Transportation Science* **20**(3) 182–188.

Hall, R. W. 1993. Non-recurrent congestion: How big is the problem? are traveler information systems the solution? *Transportation Research Part C: Emerging Technologies* **1**(1) 89–103.

Hansen, I. 2001. Determination and evaluation of traffic congestion costs. *EJTIR* **1**(1) 61–72.

Helbing, D., M. Treiber, A. Kesting, M. Schnhof. 2009. Theoretical vs. empirical classification and prediction of congested traffic states. *The European Physical Journal B* **69** 583–598.

Hoogendoorn, S. P., H. van Lint, V. L. Knoop. 2008. Macroscopic modeling framework unifying kinematic wave modeling and three-phase traffic theory. *Transportation Research Record: Journal of the Transportation Research Board* **2088**(1) 102–108.

Huang, H. 2012. Real-time information and correlations for optimal routing in stochastic networks. Ph.D. thesis, University of Massachusetts.

Huang, H., S. Gao. 2012. Real-time traveler information for optimal adaptive routing in stochastic time-dependent networks. *Transportation Research Part C: Emerging Technologies* **21**(1) 196–213.

IBM. 2013. Smarter traffic. URL `http://www.ibm.com/smarterplanet/us/en`.

Ichoua, S., M. Gendreau, J.-Y. Potvin. 2000. Diversion issues in real-time vehicle dispatching. *Transportation Science* **34**(4) 426–438.

Ichoua, S., M. Gendreau, J.-Y. Potvin. 2006. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science* **40**(2) 211–225.

Ichoua, S., M. Gendreau, J.-Y. Potvin. 2007. Planned route optimization for real-time vehicle routing. *Dynamic Fleet Management*. Springer, 1–18.

Immers, B., J. Stada, I. Yperman, A. Bleukx. 2004. Towards robust road network structures. *Slovak Journal of civil engineering* **12**(4) 10–17.

Inrix. 2012. Traffic scorecard. URL `http://scorecard.inrix.com/scorecard/`.

Jaillet, P., M. R. Wagner. 2008. Online vehicle routing problems: A survey. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 221–237.

Jain, R., M. J. Smith. 1997. Modeling vehicular traffic flow using M/G/C/C state dependent queueing models. *Transportation Science* **31**(4) 324–336.

Kenyon, A. S., D. P. Morton. 2003. Stochastic vehicle routing with random travel times. *Transportation Science* **37**(1) 69–82.

Kerner, B. S. 2004. Three-phase traffic theory and highway capacity. *Physica A: Statistical Mechanics and its Applications* **333** 379–440.

Kim, K.-E., T. Dean. 2002. Solving factored mdps with large action space using algebraic decision diagrams. *PRICAI 2002: Trends in Artificial Intelligence* 47–56.

Kim, S., M. E. Lewis, C. C. I. White. 2005a. State space reduction for nonstationary stochastic shortest path problems with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems* **6**(3) 273–284.

Kim, S., M. E. Lewis, I. White, C. C. 2005b. Optimal vehicle routing with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems* **6**(2) 178–188.

Knoop, V., H. van Zuylen, S. Hoogendoorn. 2008. The influence of spillback modelling when assessing consequences of blockings in a road network. *EJTIR* **4**(8) 287–300.

Knoop, V. L. 2009. Road incidents and network dynamics: Effects on driving behaviour and traffic congestion. Ph.D. thesis, Delft University of Technology, TRAIL Research School.

Knoop, V. L., S. P. Hoogendoorn, H. J. Van Zuylen. 2007. Quantification of the impact of spillback modeling in assessing network reliability. *Transportation Research Board 86th Annual Meeting*. 07-0859.

Koopmans, C., E. Kroes. 2003. Estimation of congestion costs in the Netherlands. *Proceedings of the European Transport Conference (ETC), France*.

Laporte, G. 2007. What you should know about the vehicle routing problem. *Naval Research Logistics* **54**(8) 811–819.

Laporte, G., F. Louveaux, H. Mercure. 1992. The vehicle routing problem with stochastic travel times. *Transportation Science* **26**(3) 161–170.

Laporte, G., F. V. Louveaux, L. Van Hamme. 2002. An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research* **50**(3) 415–423.

Larsen, A., O. B. Madsen. 2000. The dynamic vehicle routing problem. Ph.D. thesis, Technical University of Denmark, Department of Transport, Logistics & ITS.

Lei, H., G. Laporte, B. Guo. 2011. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research* **38**(12) 1775–1783.

Li, J.-Q., P. B. Mirchandani, D. Borenstein. 2009. A lagrangian heuristic for the real-time vehicle rescheduling problem. *Transportation Research Part E: Logistics and Transportation Review* **45**(3) 419–433.

Lighthill, M. J., G. B. Whitham. 1955. On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* **229**(1178) 317–345.

Lorini, S., J.-Y. Potvin, N. Zufferey. 2011. Online vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research* **38**(7) 1086–1090.

Lu, X., S. Gao, E. Ben-Elia. 2011. Information impacts on route choice and learning behavior in a congested network. *Transportation Research Record: Journal of the Transportation Research Board* **2243**(1) 89–98.

Medina, C. 2010. Assessment of non-recurrent congestion on dutch motorways. Master's thesis, Delft University of Technology, Trail Research School.

Miete, O. 2011. Gaining new insights regarding traffic congestion, by explicitly considering the variability in traffic. Master's thesis, Delft University of Technology, Trail Research School.

Miller-Hooks, E., H. S. Mahmassani. 2000. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science* **34**(2) 198–215.

Minis, I., A. Tatarakis. 2011. Stochastic single vehicle routing problem with delivery and pick up anda predefined customer sequence. *European Journal of Operational Research* **213**(1) 37–51.

Montemanni, R., L. M. Gambardella, A. E. Rizzoli, A. V. Donati. 2005. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization* **10**(4) 327–343.

Newell, G. F. 1993. A simplified theory of kinematic waves in highway traffic, part i: General theory. *Transportation Research Part B: Methodological* **27**(4) 281–287.

Novoa, C., R. Storer. 2009. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research* **196**(2) 509–515.

Novoa, C. M. 2005. Static and dynamic approaches for solving the vehicle routing problem with stochastic demands. Ph.D. thesis, Northwestern University.

Osorio, C., M. Bierlaire. 2009. A surrogate model for traffic optimization of congested networks: an analytic queueing network approach. *Report TRANSP-OR* **90825** 1–23.

Osorio, C., G. Flötteröd, M. Bierlaire. 2011. Dynamic network loading: A stochastic differentiable model that derives link state distributions. *Procedia-Social and Behavioral Sciences* **17** 364–381.

Pandelis, D. G., E. G. Kyriakidis, T. D. Dimitrakos. 2012. Single vehicle routing problems with a predefined customer sequence, compartmentalized load and stochastic demands. *European Journal of Operational Research* **217**(2) 324–332.

Pillac, V. 2012. Dynamic vehicle routing: solution methods and computational tools. Ph.D. thesis, Ecole des Mines de Nantes.

Pillac, V., M. Gendreau, C. Guéret, A. L. Medaglia. 2012. A review of dynamic vehicle routing problems. *European Journal of Operational Research* **225**(1) 1–11.

Pillac, V., C. Guéret, A. Medaglia, et al. 2011. A dynamic approach for the vehicle routing problem with stochastic demands. *Roadef 2011*.

Polychronopoulos, G. H., J. N. Tsitsiklis. 1996. Stochastic shortest path problems with recourse. *Networks* **27**(2) 133–143.

Potvin, J.-Y., Y. Xu, I. Benyahia. 2006. Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research* **33**(4) 1129–1137.

Powell, W. B. 2007. *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley and Sons, Inc., New York.

Powell, W. B. 2011. *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley and Sons, Inc., New York.

Powell, W. B., P. Jaillet, A. Odoni. 1995. Chapter 3 stochastic and dynamic networks and routing. C. M. M.O. Ball, T.L. Magnanti, G. Nemhauser, eds., *Network Routing*, *Handbooks in Operations Research and Management Science*, vol. 8. Elsevier, 141 – 295.

Powell, W. B., J. A. Shapiro, H. P. Simo. 2002. An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science* **36**(2) 231–249.

Powell, W. B., H. P. Simao, B. Bouzaiene-Ayari. 2012. Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal of Transportation and Logistics* **1**(3) 237–284.

Powell, W. B., B. Van Roy. 2004. Approximate dynamic programming for high dimensional resource allocation problems. *Handbook of learning and approximate dynamic programming* 261–280.

Psaraftis, H. N. 1988. Dynamic vehicle routing problems. *Vehicle routing: Methods and studies* **16** 223–248.

Psaraftis, H. N. 1995. Dynamic vehicle routing: Status and prospects. *annals of Operations Research* **61**(1) 143–164.

Psaraftis, H. N., J. N. Tsitsiklis. 1993. Dynamic shortest paths in acyclic networks with markovian arc costs. *Operations Research* **41**(1) 91–101.

Puterman, M. L. 2009. *Markov decision processes: discrete stochastic dynamic programming*, vol. 414. John Wiley and Sons.

Qian, Z. S., W. Shen, H. Zhang. 2012. System-optimal dynamic traffic assignment with and without queue spillback: Its path-based formulation and solution via approximate path marginal cost. *Transportation research part B: methodological* **46**(7) 874–893.

Rakha, H., W. Zhang. 2005. Consistency of shock-wave and queuing theory procedures for analysis of roadway bottlenecks. *84th Annual Meeting of the Transportation Research Board, Washington, DC*.

Regio Delft Project. 2013. Traffic data in the regiolab-delft area. http://www.regiolab-delft.nl/.

Rehborn, H., S. L. Klenov, J. Palmer. 2011. Common traffic congestion features studied in usa, uk, and germany based on kerner's three-phase traffic theory. *Intelligent Vehicles Symposium (IV), IEEE*. 19–24.

Ross, S. M. 1999. *Introduction to Probability and Statistics for Engineers and Scientists, Second Edition*. Academic Press.

Schrank, D., B. Eisele, T. Lomax. 2012. Texas transportation 2012 urban mobility report. Tech. rep., Texas A&M Transportation Institute.

Schreuder, M., G. Tamminga, M. Kraan. 2008. Vulnerability of a national road network. *Transportation Research Board 87th Annual Meeting*. 08-1504.

Secomandi, N. 2000. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research* **27**(11-12) 1201–1225.

Secomandi, N. 2001. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research* **49**(5) 796–802.

Secomandi, N., F. Margot. 2009. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research* **57**(1) 214–230.

Sever, D., N. Dellaert, T. Van Woensel, T. De Kok. 2013. Dynamic shortest path problems: Hybrid routing policies considering network disruptions. *Computers & Operations Research* **40**(12) 2852–2863.

Simão, H. P., J. Day, A. P. George, T. Gifford, J. Nienow, W. B. Powell. 2009. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science* **43**(2) 178–197.

Skabardonis, A., N. Geroliminis. 2005. Real-time estimation of travel times on signalized arterials. *Proceedings of the 16th International Symposium on Transportation and Traffic Theory*. 387–406.

Snelder. 2010. Designing robust road networks. Ph.D. thesis, Delft University of Technology, Trail Research School.

Snelder, M., H. Van Zuylen, L. Immers. 2012. A framework for robustness analysis of road networks for short term variations in supply. *Transportation Research Part A: Policy and Practice* **46**(5) 828–842.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* **35**(2) 254–265.

Suson, A. 2010. Dynamic routing using ant-based control. Master's thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology.

Sutton, R., A. Barto. 1998. *Reinforcement learning*. The MIT Press, Cambridge, MA.

Taniguchi, E., H. Shimamoto. 2004. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C: Emerging Technologies* **12**(3) 235–250.

Tatarakis, A., I. Minis. 2009. Stochastic single vehicle routing with a predefined customer sequence and multiple depot returns. *European Journal of Operational Research* **197**(2) 557–571.

Thangiah, S. R., A. Fergany, S. Awan. 2007. Real-time split-delivery pickup and delivery time window problems with transfers. *Central European Journal of Operations Research* **15**(4) 329–349.

Thomas, B., I. White, C. C. 2007. The dynamic shortest path problem with anticipation. *European Journal of Operational Research* **176**(2) 836–854.

TomTom. 2013. Route planner. URL `http://www.routes.tomtom.com`.

Van De Weijer, C. J. T., B. J. C. M. Rutten. 2012. How navigation centric traffic management is rapidly changing. *19th ITS World Congress*. Austria.

Van Woensel, T., N. Vandaele. 2007. Modeling traffic flows with queueing models: a review. *Asia-Pacific Journal of Operational Research* **24**(04) 435–461.

Wald, A. 1945. Statistical decision functions which minimize the maximum risk. *The Annals of Mathematics* **46**(2) 265–280.

Waller, S. T., A. K. Ziliaskopoulos. 2002. On the online shortest path problem with limited arc cost dependencies. *Networks* **40**(4) 216–227.

Wang, H., K. Rudy, J. Li, D. Ni. 2010. Calculation of traffic flow breakdown probability to optimize link throughput. *Applied Mathematical Modelling* **34**(11) 3376–3389.

Yang, W. H., K. Mathur, R. H. Ballou. 2000. Stochastic vehicle routing problem with restocking. *Transportation Science* **34**(1) 99–112.

Ziliaskopoulos, A. K. 2000. A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation science* **34**(1) 37–49.

Summary

Routing in Stochastic Networks

In real-life, road networks are congested due to random events such as accidents, travel times depend upon the moment one leaves the distribution center, demand varies, etc. Neither traffic conditions nor customer demands are known with certainty but are rather dynamic and stochastic. They are dynamic because traffic conditions as well as supply and demand fluctuate over space and time dimensions. These dynamics occur due to stochastic events such as road blockages from accidents, weather conditions, uncertain demand, vehicle breakdowns, traveler choices, etc. Transportation networks are characterized by uncertainty of traffic conditions and/or customer demands on the networks. These stochastic and fluctuating elements on transportation networks lead to stochastic travel times and so uncertain travel costs for transportation companies and drivers.

Traditionally, most routing algorithms and software consider networks in a static way with deterministic or stochastic information, i.e. the route is determined before going en-route with all model inputs known with certainty or the inputs are random variables that follow certain probability distributions based on historical characteristics, respectively. Obviously, the world is dynamic and stochastic and does not fit into a deterministic and static straitjacket. Any schedule built on these unrealistic assumptions results in higher transportation costs.

Today, for transportation companies the market is becoming competitive with higher customer service expectations and drivers want to be on time at their destinations in this dynamic and stochastic environment. Fortunately, the advances in information technologies provide travelers and dispatchers with real-time information on the location of the vehicles, traffic network conditions and customer demand realizations. These present operations researchers an opportunity to tackle dynamic and stochastic conditions on transportation networks to provide travelers and transportation companies with low transportation costs via dynamic routing. Considering high quality and frequently updated real-time information, planners can replan and reroute travelers and vehicles during traveling. In other words, dynamic routing allows planners to change their routing decisions during traveling as more accurate information becomes

available to handle fluctuating demands and travel times.

Dynamic routing models can capture real-life dynamics more realistically. According to recent studies, dynamic decision making by considering uncertainty and real-time information lead to significant cost savings. However, these savings are at the expense of computational complexity as the routing problem becomes combinatorial. For practical purposes, we need fast and simple dynamic routing algorithms that provide the travelers with high quality routing decisions. In this thesis, balancing the trade-off between computational time and solution quality is of particular importance because fast, simple and accurate routing decisions prevent drivers wasting time for retrieving decisions with lower transportation costs and higher customer service levels for transportation companies.

This thesis addresses routing problems in networks where the state of the network changes regarding to stochastic and dynamic events such as traffic accidents, congestions due to bottlenecks and random customer demands, etc. We limit our scope to networks where the stochasticity arises from either the travel time changing due to traffic disruptions (event that increase the travel time significantly) or stochastic customer demands that are realized upon arrival. We do not focus on unpredictable disruptions on roads and new customer arrivals.

By considering the predictability of these events, we propose a multi-stage stochastic dynamic programming model based on a discrete-time, finite MDP formulation. In our models, we couple real-time network information with probabilistic information on the stochastic elements to improve the decision making process. In a real-time setting, the network information (e.g. travel times, customer demands, disruption realizations and current location) is realized during the execution of the routing process. Before the departure, we develop fast and efficient algorithms to obtain stationary routing policies based on network state realizations. As the traveler collects information on the realizations during travel, the relevant policy is selected. In other words, given the stochastic information, we develop dynamic decisions depending on the realizations of the stochastic elements to minimize the total expected travel time.

We solve the problem with a Dynamic Programming (DP) approach. For the routing decisions, the DP approach provides a practical framework to find routing decisions for multiple stages. However, for large-scale networks, finding the optimal solution suffers from the curse of dimensionality. Throughout the thesis, we provide approximations to reduce the computation time significantly while providing high quality solutions.

In Chapters 3-5, we focus on dynamic shortest path problems in stochastic networks where the uncertainty comes from travel time. In these chapters, we develop dynamic routing policies and we compare various offline and online algorithms as well. In the last chapter, we deal with vehicle routing problem where we have stochastic customer demands. In Chapter 2, we describe the traffic networks that we focus on for dynamic shortest path problems. In this chapter, we explain how we implement the dynamic

algorithms from input information retrieval to output generation. This chapter is a prologue to Chapters 3-5.

In Chapter 3, we consider dynamic networks where there are stochastic disruptions due to accidents, and bottlenecks that cause congestions that lead to significantly higher travel time. For developing routing policies, we both consider real-time traffic information and stochastic information. Furthermore, we also consider the probability distribution of the duration of a congestion caused by a disruption. In this chapter, we analyze the effect of considering different types of information of different parts of the network on quality of routing decisions and the computation time. We develop a framework based on a DP in which we formulate and evaluate various online and offline routing policies in the literature. Next to this, we develop computationally efficient hybrid lookahead policies considering both real-time and historical probabilistic information. To test the efficiency of different routing policies, we develop a test bed of networks based on a number of characteristics such as network size, network vulnerability and disruption rate. We analyze the results in terms of routes, solution quality and calculation times. Our results show that a significant part of the cost reduction can be obtained by considering only a limited part of the network in detail at online level.

In Chapter 4, we consider the same dynamic shortest path problem considered in Chapter 3. However, in this chapter we extend the problem for larger network sizes with multiple levels of disruptions. We model this as a discrete-time, finite MDP. For large-scale networks with many levels of disruptions, the formulation suffers from the curse of dimensionality. To mitigate the curses of dimensionality, we apply Approximate Dynamic Programming (ADP) algorithm. In this algorithm, instead of computing exact value functions (computationally challenging), we consider value function approximations. We develop a hybrid ADP algorithm with efficient value function approximations based on a clustering approach. In this hybrid algorithm, we combine a deterministic lookahead policy with a value function approximation. We provide various algorithmic design variations for the ADP where multiple initial solutions and various update methods are used with efficient exploration/exploitation strategies. We show insights about the performance of these variations based on different network structures. Furthermore, we develop a test bed of networks to evaluate the efficiency of our approximation with the standard ADP algorithm and the hybrid routing policy based on stochastic lookahead algorithm (developed in Chapter 3). The results show that the hybrid ADP algorithm reduces the computational time significantly. Furthermore, the hybrid ADP outperforms the standard ADP algorithm. We also show that the hybrid ADP algorithm outperforms the stochastic lookahead algorithm in large size networks with many disruption levels.

In Chapter 5, we consider another dynamism in traffic networks which is the propagation effect of disruptions on the dynamic shortest path problems. When a disruption occurs at a certain link, due to the limited capacity, the effect of the disruption propagates to upstream links. We denote this as the spillback effect. To

have better routing decisions, we should capture the dynamics of spatial and temporal correlations. In this chapter, we model the dynamic shortest path problem with stochastic disruptions as a discrete-time, finite MDP. To reduce the state-space, we only use real-time information for a limited part of the network (Hybrid routing policy from Chapter 3). We analyze the effect of considering and not considering the spillback effect in our routing decisions. Numerical results show that considering the spillback effect in the dynamic routing decisions significantly improves the solution quality for the networks with higher number of vulnerable arcs. Moreover, the hybrid routing policy with the disruption and the spillback information for the limited part of the network, significantly reduces the computational time while providing on average significantly higher solution quality than the full information model that ignores the spillback effect.

Chapter 6 deals with the vehicle routing problem with stochastic demands, considering a single vehicle. In this problem, the actual demand realization is unknown until we visit a customer. We build a stochastic dynamic programming model and implement an ADP algorithm to overcome the curses of dimensionality. The ADP algorithms are based on the Value Function Approximations (VFA) with a lookup table. The standard VFA with lookup table is extended and improved for the DVRP with stochastic demands. The improved VFA algorithm reduces the computation time significantly with good quality of solutions. A significant reduction of computational time enables us to conduct systematic larger scale numerical experiments, important for real-life decision making. Several test instances found in the literature are used to validate and benchmark our obtained results.

Curriculum Vitae

Derya Sever was born on May 4, 1985 in Ankara, Turkey. She received her BSc degree in Industrial Engineering in 2007 at Bilkent University, Ankara. In her study, she mainly focused on hub-location problems and distribution system design for retailers. In 2009, she obtained her MSc degree in Operations Management and Logistics at Eindhoven University of Technology in Eindhoven, The Netherlands. Her master project was on designing a decision support system for return acceptance and replenishment policy, conducted under the supervision of Gudrun Kiesmüller and Simme Douwe Flapper.

In 2009, she started her PhD research on routing problems considering uncertainties on transportation networks under the supervision of Tom van Woensel, Nico Dellaert and Ton de Kok. She carried out part of her research for 3 months at the Department of Industrial Engineering, Tsinghua University in Beijing, China, under the supervision of Lei Zhao. On January 21, 2014 Derya defends her PhD thesis at Eindhoven University of Technology.