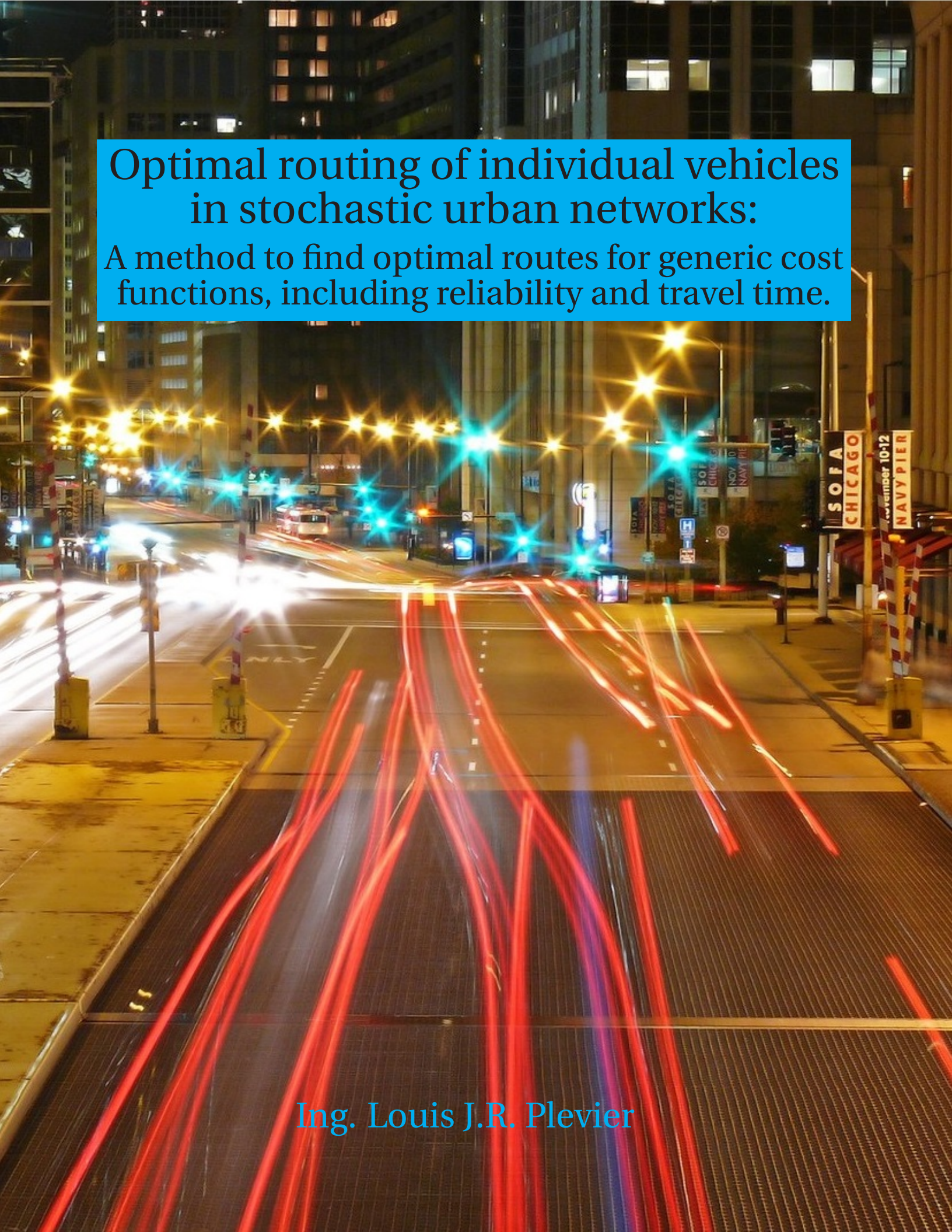

Optimal routing of individual vehicles in stochastic urban networks:

A method to find optimal routes for generic cost functions, including reliability and travel time.

Ing. Louis J.R. Plevier

Optimal routing of individual vehicles in stochastic urban networks:

A method to find optimal routes for generic cost functions, including reliability and travel time.

by

Ing. Louis J.R. Plevier

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday January 14, 2021 at 10:00 AM.

Student number:	4514971	
Thesis committee:	Dr.ir. A. Hegyi,	TU Delft, Chair assessment committee
	Dr. ir. A.M. Salomons,	TU Delft
	Dr. M. Snelder,	TU Delft
	Dr. A. Dabiri,	TU Delft

TU Delft

Preface

Before you lies my thesis “Optimal routing of individual vehicles in stochastic urban networks”, in which two new routing algorithms are designed that are capable of finding the optimal routes while using generic cost functions. It combines my endeavor of creating an optimal traffic system with my interest in traffic lights. It has been written as the last part of my curriculum to be awarded the Master of Science degree at the TU Delft.

I would like to thank the members of my thesis committee. And especially my daily supervisors, Andreas Hegyi and Maria Salomons, for their excellent guidance and support during this process. From the beginning where we discussed my interests to find an enthralling research, to the end in which frequent feedback and revisions greatly contributed to the quality of this thesis. Next to their professional contributions they were also very understanding of my medical situation and slow progress during these times. Without this understanding and flexiblity I would not have been able to continue working on my thesis during this time.

I would also like to thank family and friends for their support. For both motivating me to work on my thesis and also for some distraction when I was working to much on my thesis. I would like to thank Nico in particular for reading my entire thesis for spelling mistakes and structure suggestions.

Ing. Louis J.R. Plevier
Bergen op Zoom, December 2020

Summary

Introduction

Current day routing systems use predicted travel times to calculate optimal routes. Some routing algorithms even use predicted traffic lights states to optimize these routes and to improve the expected travel time. But all these methods optimize the expected travel time and do not take reliability into account. Reliability is a major service indicator and with non-flexible departure or arrival constraints, the value of reliability can increase up to three times the value of time (Markovich, Concas, & Kolpakov, 2009).

To use a measure of reliability, not the expected travel time, but the travel time distribution should be known. Since reliability is not uniquely defined, an algorithm is designed with an individual cost function, that gives users the freedom to use their own definition of reliability. Since knowing the traffic light states at intersections could improve the travel time and reliability, this should be used in the algorithm as well. Therefore the main goal of this research is:

Designing an urban routing algorithm, that incorporates predicted traffic light states, where the routing is based on the probability distribution of the travel time and an individual cost function that expresses the tradeoff between travel time and reliability.

For this research only urban areas with traffic lights are considered, other intersections in this network are not taken into account. The traffic lights are assumed to be traffic lights with a changing cycle time. If this algorithm works for these traffic lights it will also work for fixed time traffic lights and most other type of traffic lights, since these have less degrees of freedom.

Routing algorithms

Most routing algorithms used today do not take predicted traffic light states into account. There are different methods developed in researches, but only a few could work with probabilistically known signal timings or delays. Since traffic lights with a changing cycle time are considered in this research, the signal timings are probabilistic. The ARSC method by B. Yang and Miller-Hooks (2004) can route traffic in a time-dependent stochastic network using discrete time steps and take probabilistically known signal timings or delays into account. Therefore this algorithm is used as the base of the newly developed algorithm. This algorithm saves the route with the lowest expected travel time at intermediate nodes and does this by searching for the optimal route backwards, from the destination node towards the departure node and from the latest timestep in the prediction horizon to the current time. This eases the calculation of travel times around the traffic light, but waiting times at the traffic lights cannot be taken into account correctly. But even though the waiting times at the traffic light are not taken into account correctly, the probability on a traffic light being green is taken into account. And since the probability on a traffic light being green can differ per arrival time at that traffic light, the optimal route can differ per arrival time at intermediate nodes. Therefore the optimal route is a set of route choices depending on the arrival time at the intermediate nodes, which is called a routeplan. These routeplans are calculated pre-trip for all algorithms in this research and can deal with delay since the next link to take is dependent on the arrival time at that node. In the case a vehicle wants to update their routeplan the algorithm needs to be run again completely.

But it does not support any form of reliability, which is why the ARSCR (Adaptive Routing with Signal Controls and Reliability) algorithm is created, which calculates the travel time distribution at intermediate nodes. To take reliability into account a user dependent objective function is used, which has the travel time distributions as input. With these travel time distributions and objective function, the optimal routeplan could be determined at every node in the network and the optimal routeplan according to the objective function can be found. But not every objective function works with the ARSCR algorithm. Since the optimal routeplan is determined at all intermediate nodes, the optimal routeplan cannot be dependent on data that will be added at the next node in the calculations. Since the optimal routeplan is dependent on the objective function, not all objective functions can be used. Some objective functions cannot have their routeplan calculated at intermediate nodes because the reliability measure used, need to have the travel time distribution of the entire routeplan before they can be compared. If an optimal routeplan is selected at intermediate nodes, this might result in sub optimal routeplans for the complete routeplan. For this situation a new algorithm is created, the Forwards Probability Propagation (FPP) algorithm.

The FPP algorithm calculates the entire travel time distribution for every routeplan before calculating the objective function, instead of at intermediate nodes as well. Therefore all objective functions can be used, but the calculation becomes more complex. Since this algorithm is calculated forwards, from the departure to the destination node, and from the current time until the destination is reached, waiting times at the traffic lights can be taken into account correctly.

Traffic light state prediction

To predict the traffic light states, a method was developed that predicts traffic light states a few minutes ahead, resulting in a probability distribution, has a reasonable complexity and uses only historic or SPaT (Signal Phase And Timing) data. The most useful method that was found created a list of traffic light states and calculated the probabilities of transition between these states (Bodenheimer, Brauer, Eckhoff, & German, 2014; Dabiri & Hegyi, 2018). If the current state is known, then the probability on all future states could be calculated.

This method is used in this research as well, and since some traffic lights can have alternative realisations (give a different direction green if some directions have no traffic), the definition of a state was expanded. A traffic light state is a combination of all the directions that *have* green at that moment and not the directions that *can* be green at that moment. This will increase the number of states, but also increases the information saved in a state. This is used to calculate the probability of a traffic light being green for every direction and every timestep in the algorithm. This is directly used in the ARSCR algorithm, but for the FPP algorithm the data is adjusted to give the waiting time distribution at the traffic light depending on the arrival time at the traffic light. This uses more calculations, but offers more realistic predictions of the waiting times at a traffic light.

Link travel time prediction

There are three main models that are used in urban link travel time prediction: the queueing model, the queue discharge model and the platoon dispersion model. All these models depend on each other and mainly on the position in the queue. Since the position in the queue is not known in the routing algorithms described before, these models cannot be used with these algorithms directly. Therefore a link travel time distribution per link will be used, which is assumed to be independent of the position in the queue or time waited at the traffic light. This distribution should be made available to the routing algorithm, since current routing algorithms already take (realtime) link travel times into account, this is usually already known.

Evaluation of the algorithm

Proof of concept

In the evaluation the FPP algorithm is verified with a simulation. This is done to check if the travel time distributions, predicted by the FPP algorithm, were realized in a simulation and are thus calculated correctly. This is done by first calculating the optimal distribution with the FPP algorithm on a small network with 2 intersections with traffic lights and 4 links. This resulted in an optimal routeplan and a travel time distribution of this routeplan. This travel time distribution, calculated by the FPP algorithm, should also be realized if this routeplan is traveled repeatedly by vehicles. Therefore this is also simulated on the same network with the optimal routeplan according to the objective function, resulting in a travel time distribution. Both travel time distributions match and as a result the FPP algorithm calculates the travel time distributions correctly.

Improvement in objective values

In the second part of the evaluation the FPP algorithm is compared with a standard routing algorithm and different objective functions. This is done to calculate the improvements in objective values compared with a standard algorithm and different objective functions, which shows the effect of the FPP algorithm. Since the user can enter the objective function themselves, four objective functions are used in this evaluation. The evaluation is done by calculating the FPP algorithm for a specific network, considering all possible starting traffic light states. This results in numerous travel time distributions, and with the objective function, in numerous objective values. The mean of these values is used to compare the results of the different objective functions and compare it with a standard algorithm that does not take the traffic light state or link travel time distribution into account. This standard algorithm only takes the average delay and average link travel time into account. This part of the evaluation does not simulate individual vehicles since in the first part of the evaluation it is shown that for enough runs, the travel time distribution of those runs will become equal to the travel time distribution given by the FPP algorithm.

Results

The results show an improvement in the expected travel time (if that was used as objective function) of 1.22 seconds (on an expected travel time of 18.22 seconds expected by the standard algorithm). This is possible since the FPP algorithm is more flexible, since it has a routeplan dependent on the predicted arrival times at the intermediate nodes. In the small evaluation network used, the optimal routeplan according to the FPP algorithm had a different route depending on the predicted arrival times at intermediate nodes in 19-56% of the cases. In all other cases the routeplan consist of the same route independent on the arrival time at the intermediate nodes. This percentage is dependent on the objective function and the network, since a very small network is used this might increase in larger networks. The other objective functions took reliability into account and these mainly showed that they optimized for their functions, but that they all had different values since they take reliability into account differently. Sometimes the mean travel time might be low, but the probability to actually arrive on that mean travel time is low as well since there is high probability of being earlier and a high probability of being later. Thus even when the algorithm works, finding an optimal function to take reliability into account is hard since people's preferences differ and different functions can accommodate different aspects of those preferences.

Complexity

Besides the aforementioned results, the computation time of the algorithm also became clear in the evaluation. Since a vehicle can choose a different next link at every node, for every timestep it can arrive at that node, the number of calculations increase quickly. Therefore a simulation resolution can be used that increases the timestep of the algorithm and therefore decreases the complexity. But since this simultaneously decreases the performance of the results, further research into alternative methods is needed. For instance an algorithm that preselects which links to take into consideration, or an optimization that reduces the number of timesteps that have to be calculated at intermediate nodes. This can be done for instance if it is highly unlikely that one small change in the routeplan could improve the objective value from bad to good.

Conclusion and discussion

The FPP algorithm has an improved performance compared with standard algorithms. The potential is shown in this research, but the computation time is the main drawback. Therefore this should be improved before it can be implemented in real time. Besides the suggestions mentioned before, this could also be done by using this algorithm for the next one or two traffic lights and a conventional routing algorithm, or average waiting times and link travel times, for the rest of the network. Larger networks can be used with this method and the performance might not drop much, since the quality of the prediction decreases over time anyways.

If this algorithm is built into a navigation system, the user interface should be made user friendly with some suggested objective functions, explanation on how those objective functions take reliability into account and clear visualization of the route to take. This might seem trivial, but since the routeplan is dependent on the arrival time at intermediate nodes, it cannot always be displayed a long time in advance. Since switching lanes last moment can be unsafe, some restrictions or a warning should be build into the system.

Contents

1

Introduction

Finding the best route is something we aim for every time we travel somewhere. Because of new technological developments like GPS, navigations systems, connected vehicles and smart infrastructure there is more information available for the road user. Therefore the traffic flow and delay are known more precisely and can be predicted increasingly more accurate. Also the infrastructure itself is becoming connected with the vehicles, to give them warnings about traffic jams or road works (Sjoberg, Andres, Buburuzan, & Brakemeier, 2017) and traffic lights are sending details about the current state or time to green/red (Iglesias, Isasi, Larburu, Martinez, & Molinete, 2008). The vehicles themselves become more capable to visualize this information or improve this information by sending data themselves. A research from Bansal and Kockelman (2017) expected that in 2030 98% of the light U.S. consumers vehicles are connected because of U.S. regulation. In a future with all these connected vehicles, connected traffic lights and more precise predictions, a higher reliability of travel times could be realized. To assist towards this goal, the current and expected traffic lights state could already be incorporated into a navigation system to improve travel time reliability.

1.1. Research motivation

While most navigation systems are already using historical and real time data for travel time predictions, for travel time reliability in urban areas only historical data is commonly used (Kaparias, Bell, & Belzner, 2008; Kaparias & Bell, 2010; S. Yang & Wu, 2019). But reliability is next to travel time and level of service, a major indicator of service quality for travelers (S. Yang & Wu, 2019). This is taken into account in the Value of Reliability (VOR). With empirical estimates of the VOR, varied from 0.55 to 3.22 times the value of time savings (Devarasetty, Burris, & Shaw, 2012), reliability is an important factor to take into account (H. X. Liu, Recker, & Chen, 2004). The difference in the estimations of the VOR can partly be explained by the type of traveler and trip. If there are non-flexible departure or arrival constraints, the VOR can increase up to three times that of the travel time (Markovich et al., 2009). Business travelers in general also have a higher VOR than VOT (Kouwenhoven et al., 2014). For some truckers being on time includes not being too early, since early deliveries could result in extra holding costs (Urban, 2009), therefore a system that gives a more reliable route could be more useful than the fastest route.

Traffic light state prediction

In urban areas the travel time reliability is dependent on the speed variation at road segments and the delay variability at the traffic lights. In the current situation travel times or delays are widely communicated to the drivers but traffic light states are often not real-time communicated. Especially in urban areas intersections create an uncertain travel time, since encountering multiple red traffic lights on the route might increase the travel time by minutes (Hong-En Lin, 2005). Waiting times at intersections depend on many factors such as the amount of traffic already waiting, conflicting traffic and when the traffic light turns green. There has been some research done on predicting when the traffic light turns green with fixed time traffic lights (Koukoumidis, Peh, & Martonosi, 2011) and actuated traffic lights (Chai, Zhang, Ghosal, & Chuah, 2017; Protschky, Wiesner, & Feit, 2014; Protschky, Feld, & Wälischmiller, 2015). Some governments are making the current traffic light color available online (Apple et al., 2011) or the expected time to green (*Talking Traffic Gebruikstoepassingen*, n.d.). With this information the navigation systems could be improved to incorporate the real time traffic light states and time to green.

There are three main approaches of predicting future traffic light states (Protschky et al., 2014). The first approach is by predicting the complete future traffic state. This is based on complete knowledge of the current state

to create a model of all the traffic and the functioning of the traffic lights. Then the predicted future state can be calculated in a simulation environment. The second approach are statistical prediction models. From historical data new prediction models can be derived. Compared to the first approach, complete knowledge of the traffic state is not required, making it more scalable to larger networks. In a approach using a Markov Decision Process by Dabiri and Hegyi (2018) only the current state of the traffic light and historical data is required to predict the future state. These first two methods could be combined to create a third, hybrid approach. Statistical methods could be combined with detector information or public transport times. This could create more realistic predictions since some intersections are dependent on for instance high frequent (prioritized) public transport. For a navigation system especially the second and third approach are favored since it should be scalable to a network and usually in some parts of a network complete knowledge is not available.

Reliable routing

In current research traffic light states are already taken into account to improve travel time in the network (Apple et al., 2011) and this could result in shorter travel times (Protschky et al., 2015), lower delay and a decrease in fluctuations of the speed in the network (Chai et al., 2017). A decrease in fluctuations of the speed in the network might already be an improvement for reliability since larger fluctuations might result in less reliable routes. But the routing approach itself can also be optimized to improve the reliability. The future traffic light states can be predicted and with this information taken into account, a different route might be more reliable. This can happen if a traffic light has a high expectancy of being green at arrival or being able to facilitate enough green to make sure the vehicle can pass the first green phase. This could all be incorporated into one routing system that could calculate routes and their reliability dependent on the current traffic and traffic light state and use a measure of reliability.

The new approach

The main feature of this new approach is that not the route with the lowest expected travel time will be found, but a trade-off between reliability and travel time can be used. Therefore not the lowest expected travel time will be optimal, but the reliability of that travel time will have an impact as well. Reliability is not a single formula that can be used, but a term with different expectations for different people and situations. Some travellers find a route reliable if they have a high probability of arriving before a certain time at their destination. Other travellers prefer a reliable route as long as they have a high probability of arriving close to the expected arrival time. While others want to have a high probability of arriving in a certain time window, since they cannot be late and early. Since these are just a few examples of how reliability can be taken into account, a general method should be found to take reliability into account. This will be done by using an individual cost function, in which the user can give a function that will take reliability into account. Therefore every user can create a personalised function, specific for a certain (type of) trip. Since the new approach will optimize the route depending on traffic light states, this research will exclusively uses a network of urban roads with traffic lights and no unsignalized intersections. Some traffic lights do have a fixed cycle length and structure, which makes predicting easier, but most traffic lights do not. In this research traffic lights with a changing cycle time are researched, but if this principle works on these traffic lights, it will also work on traffic lights without changing cycle times or fixed time traffic lights, since these are a special case of traffic lights with a changing cycle time. If for instance the amount of traffic from each direction is exactly the same for some time, some traffic lights might have exactly the same cycle time or even the same timing for some cycles.

The data that is assumed to be available is SPaT (Signal Phase And Timing) data from the traffic lights. This gives information on the current state of the traffic light and sometimes about the future states. The historic SPaT data should also be available to base the predictions on. The traffic and delay on the links between the traffic lights should also be taken into account. Data from gps providers like TomTom or loop detectors around the traffic lights can be used, as well as new methods like the number of vehicles entering or leaving a link based on the connected vehicles passing by. Even though this information is not available right now, with an increasing amount of connected vehicles and traffic lights, it is probable that in the future traffic lights can communicate with most of the traffic around the traffic light and therefore can accurately estimate these numbers. For disruptions in the middle of the link a mobile phone connection (like LTS) or VANET might be required to share this information (Saiáns-Vázquez et al., 2018). In this research disruptions are not taken into account since it is still unknown how this could be predicted and how quickly traffic can react to it. Concluding, a new approach for a route with a trade-off between reliability and travel time, while taking current traffic lights states into account, will be researched.

1.2. Research goal and questions

Navigation systems are widely used these days but will not let you select a route based on reliability, instead of the shortest travel time, while taking the current and expected future states of traffic lights into account. If a driver has to be at a certain place on-time, or before a specific time, it could be more useful to calculate a reliable route to their destination instead of the shortest route. Therefore the following research goal is described:

Designing an urban routing algorithm, that incorporates predicted traffic light states, where the routing is based on the probability distribution of the travel time and an individual cost function that expresses the trade-off between travel time and reliability.

Research questions

For this research goal multiple research questions should be answered, along with the development of the routing algorithm.

- *What are the existing approaches to predict traffic light states and what is the used input, the accuracy, the complexity, the type of traffic lights on which it can be used and how can it be incorporated in the routing algorithm?*
- *What are the existing approaches to predict travel times on links?*
- *What are the existing routing algorithms and what is the used input, the complexity and how can it incorporate a probability distribution of the total travel time and an individual cost function?*
- *How does the routing algorithm perform in a simulation, compared to a standard algorithm that searches for the lowest expected travel time?*

1.3. Research approach

To clarify how the research questions are connected together, the steps taken in this research are further explained in this section with a compact scheme showing the main elements afterwards.

Traffic light state prediction

The first step is to do literature research to review the current state of the art of topics close to this research. Current methods of future traffic light state prediction and their accuracy will be researched. The aim of looking into these methods is to assess the most relevant ones. The best (parts of) prediction methods will be selected based on multiple criteria. The prediction method has to work with traffic lights with a changing cycle time since traffic lights with fixed timing are not common anymore. It should also be able to predict the future traffic light states accurately and preferably result in a probability distribution. This distribution can be used as input for the routing algorithm, while other inputs can be used as well, they should be adjusted to have values for multiple arrival moments. The prediction method should be scalable to multiple traffic lights since it is used on a network and should therefore not be to complex. Statistical methods are preferred since they require less computational power. How the accuracy develops over time and if this drops significantly should also be taken into account. Some methods might have a high accuracy for a time span of a minute but might have a low accuracy for 10 minutes in advance, while short term predictions are usually better than long term predictions, they should both be sufficient to create an efficient algorithm. The data that is assumed to be available are the current and historic SPaT data from the traffic lights, this contains the traffic light state and timing of the traffic light. Data like the current color of all traffic lights is included as different elements such as the time every direction is already in this state. The structure of the traffic light is not included in this data and if necessary its availability form the local authorities is assumed.

Travel time prediction

The future travel times between intersections should also be known. If the travel times on those links are calculated with the local maximum speed this could go wrong when there is some delay on the link. Therefore different prediction methods and their accuracy will be reviewed in this part of the literature research. These methods might have a lower accuracy in urban areas. The preferred method is also chosen based on the accuracy in short and medium term prediction, the input used and the complexity, since it should be scalable. Currently available input can be used like loop detectors at traffic lights and gps data (from for instance TomTom). But also new data sources like the amount of traffic entering and leaving the link could be used. This data will be more available in the near future, with the growing number of connected vehicles. Unexpected disruptions like incidents or vehicle breakdowns are not taken into account in this research.

Routing algorithms

The future traffic light states and travel times on the links are input for the new routing algorithm. To create this new routing algorithm the current routing algorithms are researched with specific attention towards time dependent algorithms and algorithms which take intersections or traffic lights into account. This is required since the predicted delay at the intersection is dependent on the arrival time at the intersection. Also the different directions at an intersection might have a different delay and this should be taken into account. Routing algorithms that calculate a probability of arriving on time or have a probability distribution should also be researched since the new routing algorithm should take the reliability into account. After the literature research the new routing algorithm will be developed which should be able to calculate the probability distributions of different routes while using the predicted traffic light states and link travel times. Since the travel time itself should be taken into account next to the reliability, an objective function with the reliability of arriving on time as the travel time itself will be used. There is no special feature for rerouting in the algorithm. But since the route is based on predicted states of the traffic and traffic lights, more information is known every time step. Therefore the program should be able to run quickly and recalculate the entire process on certain key moments or continuously.

Evaluation

The algorithm will be verified to show that it works by simulating the travel time distribution generated by the algorithm. If multiple runs of a simulation results in the same travel time distribution, the algorithm calculates the correct travel time distribution and therefore works. The effect of the algorithm will be evaluated as well, which is done by a simulation in Python. An urban network with traffic lights will be build and input from the traffic light prediction and travel time prediction modules will be gathered. To check the effectiveness of the algorithm the travel times and reliability of vehicles with and without the algorithm will be compared. This will be tested for multiple objective functions.

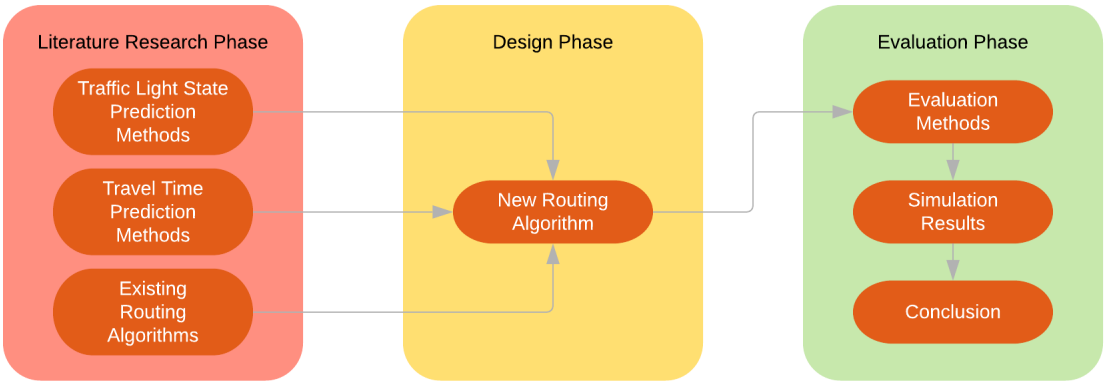

Figure 1.1: Research approach.

1.4. Thesis outline

Section 2 will start with several definitions to understand the functioning of traffic light controllers. Afterwards the current literature on traffic light state prediction, travel time prediction and routing algorithms will be reviewed. In Section 3 the new routing algorithm will be designed and for the input of this algorithm the traffic light state prediction algorithm will be designed as well. After which the input from the link travel time prediction will be explained. In Section 4 the algorithm will be evaluated. First the algorithm is validated with a simulation, after which the effectiveness of the algorithm is evaluated for different objective functions. The last section is Section 5, with the conclusion and discussion of this thesis.

2

Literature review

For the new routing algorithm three elements should be studied. Firstly, the future traffic light states are taken into account and therefore traffic light state prediction methods are required. The current literature on this topic will be discussed in Section 2.2. Secondly, the future travel time between the intersections should be known and therefore travel time prediction has been researched in Section 2.3. Lastly, Section 2.4 is dedicated to routing algorithms to find complete algorithms or elements of it that could be used. All these elements will be combined together in the new routing algorithm which is the topic of Section 3.

2.1. Definitions

To understand the functioning of traffic light controllers several definitions should be known. There are multiple signalgroups in each traffic light controller. At least one per side of the intersection, but usually more for seperate directions (eg. from east towards south) and different ones for bicycles, pedestrians or separated public transport. These signalgroups are combined into phases in which a few non-conflicting signalgroups are given green at the same time. The (signal)cycle are all these phases after each other until the first phase reoccurs again. The cycle time is the time from the beginning of the first phase until it's re-occurrence. For pre-timed traffic lights the cycle time and duration of green is always the same. But adaptive, actuatued and dynamic traffic lights take the traffic situation into account and adjust the signal timing (and phasing) accordingly.

The traffic light state could be defined in different ways. First is the most simple way and is the current state of all the colors of the traffic lights on the intersection. This is exactly the same information you could get by just looking at the traffic lights and does not give any information on the internal state of the traffic light controller. The internal state of the traffic light controller is taken into account in the second definition. This information could contain in which state every signalgroup is, such as fixed green or extension green and information when these phases started, the earliest possibility to end this phase and the latest possibility to end this phase. Sometimes data about the next signalgroups that are going to get green is included, but this is not necessarily the case with all type of traffic light controllers since this is not always known or communicated a few seconds before. Since the algorithm designed in this thesis needs to be applicable to as much traffic light controllers as possible this information will not be used. Therefore the definition of "traffic light state" in this research will be: The current colour of the light of all the signalgroups.

The information of the traffic light state is send by many traffic lights via SPaT (Signal Phase And Timing) data to traffic control centers or vehicles. SPaT data contains information about the traffic light state. The minimum data send is the current state of the traffic light signals, but sometimes expected times to green, red or the next signal group to receive green is also communicated. Since these predictions are short term, the accuracy is unknown and these predictions are not available for all traffic lights, these will not be used. Often this SPaT data is saved to a database which can be used to gain historic data of the intersection.

2.2. Current traffic light state prediction methods

There have been some research on predicting future signal states of traffic lights, in short traffic light state prediction. Most methods are statistical approaches sometimes added with detector or public transportation information. In this section the related work will be reviewed in perspective to this research. The research that focused on the prediction of the state of fixed time traffic lights cannot be used in this research because they simply follow the same

timing schedule every cycle and the traffic lights from this research will not. One fixed time method is different since it is based on busses crossing intersections (Fayazi, Vahidi, Mahler, & Winckler, 2015). This could predict the green times with sporadically a bus crossing the intersection and updating location approximately every 200 meters. This did show that even with limited information, some prediction could be done and that public transport can assist in it.

Koukoumidis et al. (2011) predicted future traffic light states by mounting iPhones on dashboards of various cars. This predicted the future traffic light states of fixed-time traffic lights and some adaptive traffic lights. The adapative traffic lights for which it predicted the states used GLIDE which is based on the widespread SCATS system. Therefore the cycletime and allocation of green to each direction can be changed by the central system, resulting in a cycletime and green time that should be estimated. This is done by support vector regression which uses prior signal transition history. This predicts when a traffic controller is going to switch to the next phase, depending on the length of the previous phases. The regression method allows for larger scale predictions and was in a test in Singapore able to predict with a mean absolute error of 4.1 seconds four phases ahead (four phases are about 187 seconds in this case). The phaselength (and thus cyclelength as well) in SCATS are set after each cycle and this might therefore react somewhat different compared to for instance actuated traffic lights that do not have a set green time, but completely relies on detector data. Therefore the error might increase significantly when the traffic lights have more freedom compared to the SCATS system. This system also predicts the exact moment of switching and does not give a probability (distribution) of actually switching at that moment.

Apple et al. (2011) combined routing and traffic light state prediction together for coordinated traffic lights. They derived stochastic models for each group of traffic lights and calculated a probability distribution on the delay time to cross the intersection dependent on the arrival time. These models were different for uncoordinated traffic lights or traffic lights with protected-permissive left turns and used historic information to determine probabilities. Some phases in these traffic lights have fixed green times but for the phases with varying green times, the level of demand, the historic green times and historic waiting times are used to create a stochastic model. From this model the likelihood of arriving during a green period could be calculated and a probability distribution of the delay time if a vehicle does not arrive during a green period. A more precise description than previously mentioned is not given since the paper focussed more on routing, but this paper shows that routing using predicted traffic lights states of actuated traffic lights will actually lead to different routes.

Another method is calculating the average green and red time and knowing the current state of the traffic light (Mahler & Vahidi, 2012). If this is known for all directions, you can estimate at what time the traffic light will be green. In this paper the method is also based on the assumption that the cycle time stays the same, which is not the case for all traffic lights. In an stochastic dynamic programming approach by Dabiri and Hegyi (2018) a comparable method was introduced. A Descrete-time Markov Chain is used, which uses a probability matrix of the transition between traffic light phases. This probability matrix can be extracted from historic SPaT data from the traffic light and contains the chance of switching to the next phase for every second the traffic light is already in that phase. The benefit of this system is that the next state of the system is only dependent on the current state and independent of the previous state of the system. The time a direction already is green is part of the state since this is required to calculate the probability of switching. In the paper from Dabiri and Hegyi (2018) this is done to give speed advice to cyclist to catch green and adjust their speed. This is a statistical method that has a low complexity and can therefore run realtime and network wide. This is mainly because the transition probability matrices are created once and do not have to be calculated every time the routing algorithm is run. These matrices might be updated frequently but this does not directly interfere with the prediction process. This method can also deal with traffic lights that do not use a fixed cycle time. The accuracy on the long term might decrease because directions can be skipped if there is no traffic, which will decrease the cycle time by seconds. Over the course of multiple cycles the effect of this uncertainty might increase, but this method can at least take this into account by skipping that direction, while the previous mentioned methods could not. During the rush hours this effect is minimal since traffic is usually coming from every direction.

In another approach with probability distributions, comparable to the one above, Protschky, Feit, and Linnhoff-Popien (2014) created a method to work on a realistic network level. The research aimed at a 95% reliability of the data and if it was not this accurate, the data would not be send out. The used data was limited to historic data of about an hour with an update every 5 minutes. This was then added to a prediction vector to create a probability distribution to predict the future traffic light signal states. After five minutes the accuracy was checked for the previous five minutes and dependent on this result some weighted new data was added to the prediction vector. With this weighted new data the algorithm tried to keep the reliability at 95%, while having the highest possible availability. While the feedback loop increased the reliability, the availability of these predictions were around 60%. This means that for 40% of the time some traffic lights could not be predicted with this method.

In a different approach by Bodenheimer et al. (2014) the traffic light state is predicted with graphs. The traffic light controller is visualized by a controller graph that shows every possible signal combination. Therefore each node is a static state in which certain traffic directions have green. The edges are the transitions between these states. They contain the clearance and amber times for the specific transition. A prediction graph is then created with these transitions as nodes and offset times and probabilities of changing towards a different state as edges. These probabilities are again based on historic data. Therefore this is a very comparable method compared to the Markov Chain approach by Dabiri and Hegyi (2018). They both list all possible traffic light states and calculate the probability of it switching to a next one depending on how long it already have been in this state. Unfortunately Bodenheimer et al. (2014) only looked a maximum of 30 seconds ahead since it was made for a GLOSA system as well. Therefore they also included the detectors since they give a lot of information for short term. The order of the transition can be predicted more precisely if it is known how much traffic there is at certain points. For public transport that passes multiple traffic lights the added value might increase, since it is already known that the bus will be at the next intersections far ahead. The method that is used for public transport is mainly the same as before, but the probability of changing to a next state is calculated separately if the detector is triggered and if it is not triggered. Green could be delayed for a certain direction to give way to a prioritized bus or green could be extended to ensure that the bus can pas the intersection. This creates another variable but still uses the same method, since it results in a different prediction graph that is treated the same way. In a follow up research, Bodenheimer, Eckhoff, and German (2015) revealed that, the assumption that traffic slowly changes and basing the predictions on the last cycles, does not yield the optimal results. Basing the predictions on historic situations of the same type was better, eg. a Monday evening peak or Thursday morning during the holiday. Furthermore, basing the predictions on too much information might create inaccurate predictions since the current situation might have changed. Therefore they based their predictions on four historic periods (e.g. 4 Monday morning peaks) since the accuracy hardly increased when taking a higher number of considered periods into account.

2.3. Current link travel time prediction methods

The future travel times on the link between intersections should also be known. If the link travel times are calculated with the local maximum speed this could go wrong when there is some delay on the link. The vehicles on the link can experience different delays which should be taken into account. Unexpected delays like traffic accidents or planned delays like roadworks are not taken into account in this research. Therefore three main elements of link travel time are researched: a queueing model, a queue discharge model and a platoon dispersion model. The queueing model will be used for the vehicles that drive towards a red traffic light and have to stop in the queue. When the traffic light turns green the queue discharge model will explain the departure of the vehicles and their behaviour. The platoon dispersion model will then be used to determine the amount of dispersion before the traffic arrives at the next traffic light. These models depend on each other and should be combined to result in one value for the link travel times. Since those models are dependent on each other a single model that results in the link travel times can be used as well as long as it takes the urban characteristics into account.

2.3.1. The queueing model

The queueing model calculates how many vehicles there are in the queue for every direction. If a road has multiple lanes or separate turning lanes, these should be taken into account as well. If the traffic lights are modelled with separate turning lanes, the model should be able to deal with traffic that cannot reach the turning lanes due to queues on the main lanes.

There are two main types of queueing models, vertical and horizontal queueing models. Vertical queueing models stack the vehicles on top of each other at the stop line, creating a vertical queue. When the traffic line turns green the queue will dissolve one vehicle at a time. Spillback and blocking of access to turning lanes cannot be taken into account in these models. Horizontal models also take the length of the vehicles and the distance between them into account. These models can take spillback into account. Cell and link transmission models also take the shockwave effects of the vehicles braking and accelerating into account (Y. Liu, Guo, & Wang, 2018). The vertical queue and horizontal queues without shockwave theories in them underestimates the effect of congestion in networks with spillover. While the more complicated horizontal models might return better results since it can take shockwave effects and spillback into account, the computational costs increase as well (Agarwal, Lämmel, & Nagel, 2018). And part of the shockwave effects are also incorporated in the queue discharge model. This model describes namely how the traffic behaves when the traffic light turns green. Therefore it should be considered if it is worth the added computational costs compared to the benefits and the possibility of combining it with the queue discharge model or replace it.

2.3.2. The queue discharge model

Traffic that is waiting at the traffic light will have to accelerate to their desired speed when the traffic light turns green. Since this starts slowly, the first cars drive slowely over the stop line while cars further in the queue drive faster past the stop line. To model this behaviour a queue discharge model is used. In this research this model gets input from the queueing model and gives input towards the platoon dispersion model.

The queue discharge model has two parts, the queue discharge rate, while traffic is driving past the stop line at their desired speed, and an acceleration loss part, when traffic is driving over the stop line not at their desired speed. The queue discharge rate differs depending on many factors like the width of the road, weather, presence of parked cars, amount of busses and heavy vehicles, the direction traffic is turning, etc. (Wilson, 2006). Therefore every direction of a traffic light has their own queue discharge rate. Since average values differ from 1800 to about 2100 vehicles per hour these should be measured on the road (Akçelik & Besley, 2002).

The acceleration loss part is harder to determine and parameters for functions to calculate this part can differ considerably at each intersection (Akçelik & Besley, 2002). A simpler approach is to calculate how many vehicles pass in the first N seconds and then calculate how many seconds an average vehicle takes to cross the intersection. After N seconds the queue discharge rate will be used for the remaining vehicles (Wilson, 2006). The question still remains how many vehicles or seconds should be counted before using the queue discharge rate. It was assumed that after four vehicles the discharge rate would stabalize but recent research shows that it takes longer than 4 cars (Dey, Nandal, & Kalyan, 2013; Lin & Thomas, 2005). Even though this value might keep increasing for at least 17 cars as Lin and Thomas (2005) suggests, the biggest difference is in the first 4 to 6 vehicles.

2.3.3. Platoon dispersion model

A platoon dispersion model is required since not all drivers want to drive at the same speed. Also drivers want different headways and therefore the platoon will disperse as it moves away from the stop line.

In the 1950s Pacey created a platoon dispersion model based on normally distributed vehicular speeds and a possibility for all verhicles to overtake (Grace & Potts, 1964). Since drivers have a different speed, the platoon will slowly disperse. The results from Pacey's model was a continuous function and to facilitate applications that required a discrete model, Robertson created one. This model from Robertson is found effective in under-saturated flow conditions and can therefore be used in traffic signal optimization. This model is used in many applications that are currently used like: TRANSYT, SCOOT, SATURN and TRAFLO (Paul, Mitra, & Maitra, 2016).

Robertsons platoon dispersion model is writen as follows:

$$q_t^d = F_n \cdot q_{t-T} + (1 - F_n) \cdot q_{t-n}^d \tag{2.1}$$

Where q_t^d is the arrival rate of traffic at the downstream traffic signal. q_{t-T} is the flow rate at the upstream traffic signal at time t-T. T is the time lag, the number of seconds that it takes for a vehicle to arrive at the traffic light downstream after getting green at the upstream traffic light. This is visualized in Figure 2.1. q_{t-n}^d are the arrivals at the downstream traffic signal during the previous time step. F_n is a smoothing factor that uses a weighted combination of arrivals during the previous time step and the departures from the upstream intersection T seconds ago to calculate the downstream arrivals.

Formula 2.2 shows that this smoothing factor is dependent on the average link travel time from the upstream traffic light to the downstream traffic light (T_a) and a factor α and β. These factors α and β will be explained in the next section.

$$F_n = \frac{1}{1 + T_a \alpha \beta} \tag{2.2}$$

In formula 2.3 the factor T is further explained. It extends further than the average link travel time from traffic light to the next one, since this factor is multiplied with β. Therefore these factors α and β have a significant impact on the model.

$$T = \beta T_a \tag{2.3}$$

Robertsons platoon dispersion model requires two parameters as input: the dispersion factor α and a travel time factor β. The results of the model are highly dependent on these factors and therefore there is much research on optimizing these values under different circumstances. Some researches studied specific situations while others tried to link external factors like width of the road, number of lanes or saturation grade to these parameters. Some results from these researches are that when the number of lanes increased, the optimal value of α drops (Bie, Liu, Ma, & Wang, 2013). While if the traffic volume and thus density increases, the value of platoon dispersion factor α increases up to a maximum at half of the capacity. If the volumes increases more, the amount of dispersion drops until a minimum at the capacity (Baass & Lefevre, 1987; Manar & Baass, 1996). In a different research Yu (2000)

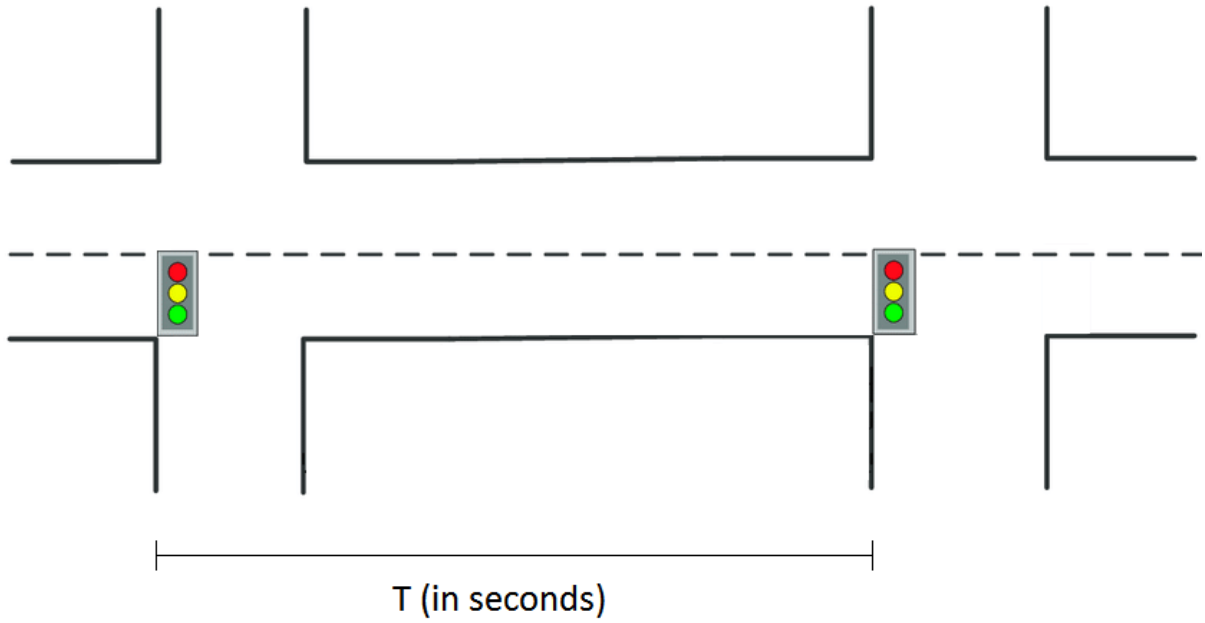

Figure 2.1: Visualization of T.

found that the parameters α and β could be determined depending on the link travel time and standard deviation. Since other factors like the number of lanes and saturation grade influences the link travel time and the standard deviation, these other factors are taken into account as well. Since the link travel time and standard deviation change over time these values should be updated over time. This is a realistic situation since the density varies over time and this is also proven to influence the parameters (Baass & Lefevre, 1987; Manar & Baass, 1996). The link travel time and standard deviation are needed for this method, but since more and more vehicles are connected, it will be easier to obtain these values in the future. Travel time information can be collected from navigation systems like Google maps, Waze and TomTom to calculate the link travel times and standard deviation.

2.4. Current routing algorithms

The last part that needs to be researched are existing routing algorithms. These routing algorithms (or parts of) can be used as a basis for the new routing algorithm.

Dijkstra et al. (1959) was one of the pioneers of routing problems and created a shortest path algorithm. In short the Dijkstra algorithm calculates the distance to all adjacent nodes from the start node and saves the total distance traveled to that node. Then it does the same for the node with the shortest distance travelled. If a node already has a travel distance (via a different route), the lowest distance will be saved. With a backtracking algorithm the shortest route can be determined when the final node is reached. This works under the condition that the travel distance, travel time or costs on the arcs are time-independent and deterministic. Hart, Nilsson, and Raphael (1968) developed the A* strategy, another approach in solving minimum cost paths. A* uses the estimated cost to complete the path to select the next node. Therefore A* is considered to be more efficient compared to the Dijkstra or similar algorithms (Zeng & Church, 2009). Since the estimated costs to complete the path can be estimated by for instance the euclidean distance to the final node, A* is preferred to the Dijkstra algorithm.

In networks where the travel time are both time-dependent and stochastic, Hall (1986) states that the optimal route does not only depend on the nodes and links but also on the time a vehicle arrives at these nodes. Therefore the optimal route is not a simple path, but a set of time-adaptive decision rules, in which the optimal next path is defined for each node as a function of arrival time at that node. This can give more efficient and reliable results since the traveller gains knowledge of the network before the next decision. The traveller knows for instance the experienced travel time on the links it already traversed. The Dijkstra and A* algorithms work on time-independent stochastic networks as well, but not on networks that are time-dependent (Eiger, Mirchandani, & Soroush, 1985). But these time-independent shortest path algorithms (like Dijkstra and A*) can be modified for (discrete) time-dependent shortest path problems as well by using a time-space expansion representation(Chabini, 1997). This adds an entire dimension to these algorithms and for it to work the link travel times should satisfy the FIFO (first in

first out) condition. In short this means that there cannot be any overtaking since the first vehicle to enter the link should leave the link first. But because of this rule some other limitation should be in place as well, such as only forwards labelling algorithms could be used, it calculates the shortest path for one specific (departure)time instance and it could only be used for fastest path problems and not minimum costs (Chabini, 1997). A dynamic programming approach to this problem is presented based on the concept of decreasing order of time (DOT) (Chabini, 1998). This approach calculates the expected times from each node, to the final node, through any successor nodes, per time instance. This is done for the latest time instance within the time period first and the results are saved as the adaptive expected times. This is then repeated for every time step until the first time interval and if a faster route is found the adaptive expected time is adjusted. Since this requires to calculate values for all time steps, a different approach was presented by E. D. Miller-Hooks and Mahmassani (2000) that used expected travel times and started from the departure time instead of the other way around. This lowered the complexity and for dense networks, such as data networks, the calculation time was lowered. But for sparse networks, such as transportation networks, this approach was outperformed by the DOT approach (E. Miller-Hooks, 2001).

Another search method is AO*, a heuristic search technique that is significantly more efficient compared to dynamic programming if lower bounds in the network are available (Bander & White, 2002). These lower bounds could be chosen as the euclidean distance but if these are too low, the extra complexity of the AO* technique might not improve compared to a more simple dynamic programming approach. Therefore the lower bounds should be chosen as high as possible. This heuristic approach will always return an optimal solution, if one exists, if the lower bounds are chosen below the true distance/costs (Gao & Chabini, 2006). But this method does not take time-dependent situations into account and this should be added before it could be used in this research.

These algorithms do not take the additional delay due to intersections or traffic lights into account. There have been some adjustments to algorithms to facilitate these delays. One of the approaches is to expand each node to multiple nodes and arcs to represent every turning movement with an individual delay (Protschky et al., 2015). In this way simple algorithms can be used but this is not efficient. Ziliaskopoulos and Mahmassani (1996) modified a label correcting algorithm to calculate the shortest path in a computationally efficient way. They added the waiting time (or turning delay) at the intersection directly to the link after the intersection. This worked efficient but did not take any time depended element into consideration. Y.-L. Chen and Yang (2000) added a time constraint to a shortest path algorithm which was efficient but did not take clearance times or capacity of an intersection into account and based the algorithm on pre-timed traffic lights. B. Yang and Miller-Hooks (2004) developed the penalty approach which added traffic light dependent delay to the label correcting algorithm from E. Miller-Hooks (2001). But they assumed the traffic light timing to be known as well and realised this does not hold for probabilistically known signal timings or uncertain delays. Therefore they adjusted the algorithm again to the ARSC (Adaptive Routing with Signal Controls) algorithm. This algorithm creates time steps to make the time discrete. The algorithm then saves for every node and time step the expected travel time to the destination. It does this from the destination node towards the departure node. This takes the waiting times for a red light and a link travel time distribution into account. The expected travel time to the destination node is calculated per arrival time at the current node, by adding up all the probabilities on the different travel times. For example, there is a 40% chance to have a travel time of 8 seconds, a 35% chance to travel 6 seconds and a 25% chance on traveling 3 seconds. This leads to an expected time of (0.4*8+0.35*6+0.25*3=) 6.05 seconds. This is done for all time steps and nodes and leads to the path with the lowest expected travel time while taking all traffic lights and delays into account. Therefore this does not result in a probability distribution, but it does take probabilistically known traffic lights into account while other researches did not. The traffic signal timings are estimated by a two-state continuous time Markov chain. This method is based on a chance that the state is changed to the other state which is independent from the previous state. For actuated traffic lights this makes sense since most actuated traffic lights do not adjust their behaviour based on vehicles that already passed the traffic light. There are only two states in this Markov chain while it is not uncommon that traffic lights have more states. Therefore a different traffic light prediction model, as presented in Section 2.2, could be used. They also assume that during a certain period (like weekday evening peek) the chances of transition stay the same (B. Yang & Miller-Hooks, 2004).

2.5. Summary

The main findings of this chapter are summarized below:

2.2 A traffic light state prediction method was researched that could predict traffic lights with a changing cycle time for a few minutes ahead, results in a probability distribution, has a reasonable complexity and uses only historic or SPaT data. The methods by Dabiri and Hegyi (2018) and Bodenheimer et al. (2014) seemed usefull since they were applicable to traffic lights with a changing cycle time, resulted in a probability distribution,

seemed not too complex and used available data. These methods are based on creating a list of traffic light states and calculating the probabilities of all transitions between these states (depending on how long they are already in that state). Both researches only used their method to predict shortly ahead for GLOSA and not further ahead for routing purposes. These methods both used historic data as well and Bodenheimer et al. (2015) found that they should use historic data separated per time period and from not too long ago (about four periods, e.g. weekend mornings). Detectors might be used to increase reliability although this would not improve longer time ahead prediction.

2.3 To make a prediction about the link travel time vehicles are going to encounter in this research, three separate models can be integrated into one. In urban areas most delays can be calculated by the queueing model, queue discharge model and the queue dispersion model. Therefore these models, or a model that takes these urban aspects of delay into account, can be used to predict the link travel times.

2.3.1 There are two main types of queueing models, horizontal and vertical. The vertical queueing models cannot take spillback into account while the horizontal ones can. Some horizontal queueing models also take shockwave effects into account. While horizontal models can take spillback into account and some can take shockwave effects into account they do not underestimate the delay, but the calculation time does increase.

2.3.2 There are two parts in the queue discharge model. The queue discharge rate, how many vehicles per hour cross the stop line while they drive their desired speed, and the acceleration loss part, how quickly the vehicles pass the stop line while they are not driving their desired speed. The acceleration loss part stops after about 4 to 6 vehicles since the majority vehicles are already driving their desired speed at that point. Both values depend on many factors and should therefore be locally optimized or calculated on actual vehicles.

2.3.3 There are different platoon dispersion models but the model from Robertson is often used. This model needs some parameters to function optimally and there are a lot of factors influencing these parameters. Yu (2000) found that the parameters could be determined depending on the link travel time and standard deviation. These factors could be extracted from the simulation model and estimated in real life application of this system via navigation systems or other ITS.

2.4 There are many different routing algorithms, but just a few that take traffic lights into account. Traffic light delay can be incorporated in routing algorithms by expanding nodes and using simple algorithms although this is an inefficient method. Other algorithms were more efficient but they all used pre-timed traffic lights and did not support probabilistically known signal timings or delays. The ARSC method by B. Yang and Miller-Hooks (2004) can route traffic in a time-dependent stochastic network using discrete time steps and take probabilistically known signal timings or delays into account. The traffic light prediction method from this algorithm can only use 2 states and should therefore be adjusted or replaced by one researched in Section 2.2.

3

Algorithm design

In this section the new routing algorithm will be designed and in Figure 3.1 the different subsections and their relations are visualized. The goal of the routing algorithm will be to find an optimal route dependent on an objective function given the travel times and their corresponding probability. Therefore a new routing algorithm has to be designed to not only find the expected fastest path but also take the individual objective function into account. In Section 3.1 the design process is explained as well as the objective function which is required for this new routing algorithm. In Section 3.2 the predicted traffic light states will be calculated and transformed into the correct format for the routing algorithm. This will result in a probability distribution of possible traffic light states, which can be used to create a probability of being green and a probability distribution of the waiting time at the traffic light. In Section 3.3 the predicted link travel times are discussed to feed into the routing algorithm which is the last required input for the algorithm to function. In this section mulitple formulas are presented with different variables and therefore a list of variables is added in Appendix A. .

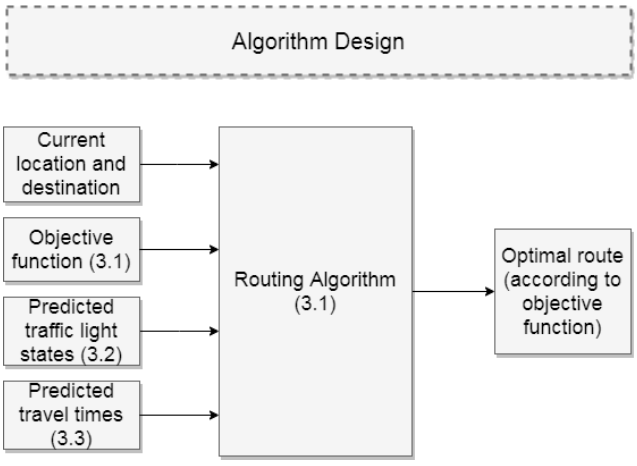

Figure 3.1: Visualization of the algorithm.

3.1. Routing algorithms

In the previous chapter different routing methods are described but the ARSC method (B. Yang & Miller-Hooks, 2004) is the only method that took probabilistically known signal timings into account. This method resulted in a path with the lowest expected travel time to the destination. This route was dependent on the arrival time of every node, since the optimal next link could be different if the vehicle arrives at an intermediate node at a different time, since the predicted traffic lights states could be different. Since this differs from the standard definition of a route, which is a set of links and nodes that will bring you to your destination, a different term is introduced. A route in which the next link to take is dependent on the arrival time at the intermediate nodes is from here on called a routeplan, which is a set of route choices for each node depending on the arrival time at that node. This concept holds for all algorithms described in this section, it gives the optimal routeplan, which is dependent on the actual

arrival time at the nodes. This is calculated when the algorithm is run and not updated when a vehicle arrives at the intermediate nodes. The vehicle simply checks from the result of the algorithm what link to take next when it arrives at an intermediate node at that time.

The lowest expected travel time to the destination is a single value and cannot be used to determine the probability of arriving on time or any other objective function that depends on the probability distribution of the arrival time. Since none of the other researched algorithms could do this as well, this needs to be added to the chosen method. Therefore the ARSC method is explained first, the objective function secondly and lastly the adjustments to this algorithm to facilitate a probability distribution are explained.

3.1.1. The ARSC algorithm

The ARSC algorithm tries to minimize the expected travel time by finding the optimal routeplan trough the network while taking probabilistically known traffic light states into account. The ARSC algorithm is a so-called backwards label correcting algorithm, and a label is a value that is stored at intermediate steps in the algorithm. In this algorithm the labels are stored at the route choice points in the network since at those points the algorithm picks the optimal routeplan to follow. Those route choice points are at the stop line of the traffic lights, just before the node. The probability of a traffic light being green might differ depending on the link a vehicle is approaching the node from (and the time a vehicle arrives at the node). Therefore a label is not stored at the node, but at the stop line right before the node and a different label will be used for every node depending on the link that node is approached from. The values that are saved at the label are the expected travel time to the final destination node. Since link travel times and probabilities of a green traffic light are dependent on the time vehicles arrive at those points, the expected travel time from any label to the final destination node varies per arrival time at that label. Therefore the expected travel time that is saved to the label, is saved for its corresponding arrival time at that label.

Since the algorithm is a backwards label correcting algorithm, the labels are calculated backwards, from the destination node towards the departure node and also backwards in time, from a set maximum calculation time. An expected travel time is calculated at the labels and therefore one value should be the result, which is a combination of the expected travel time if the traffic light is green, and if it is red. If the traffic light is red, the vehicle have to wait at least one time step, and therefore the expected travel time from the same label, but one time step later, could be used. This has already been calculated since all labels are calculated backwards from the latest time considered. If the traffic light is green, the link travel time from the current node to the next node should be used, plus the label that is stored just before the next node (for the arrival time at that label). This label is also already calculated since the labels are also calculated backwards from the destination towards the departure node. Then these two values can be used to calculate the expected travel time to the final destination for the current label and current time. This shows that for every action in the algorithm, just the part between a label and its successor label is calculated and then the expected travel time to the destination node is saved to its label. Since the expected travel time to the final destination is already saved in the successor label, instead of calculating the entire routeplan until the destination node, just the part between the current label and the succes label needs to be calculated.

The label at the destination node is set to 0 for all time steps since this is the destination. But for the first calculation there might also be a probability of the traffic light being red. Since this is the first calculation and no other labels (except at the destination node) are set, this calculation cannot be done, because the label one time step later is not calculated. To make sure the algorithm can still run, all traffic lights are set to green for a certain time before the last considered time in the algorithm. This makes sure that the first calculations can be done, but also that all routeplans that depart from the departure node at the last/lowest time step calculated, lead to an expected travel time. Otherwise for some small probabilities a travel time to the destination has not been calculated yet, since there is a probability that some traffic lights are still red.

Finally, the last part of the backwards label correcting algorithm, is the correcting. This means that if a better value is found for a label, this label is corrected and the new value is saved. This can happen when taking a different link, results in a lower expected travel time to the final destination node. Since this label only stores the lowest expected travel time, another label, the pointer label, is used to store the optimal routeplan. In this label the optimal next link to take is stored so that the routeplan with the lowest expected travel time can be found.

Notation

Now that the general working of the algorithm is explained, it is explained in more detail with the correct notation. The notation of the label is explained first, and this is denoted as, $\lambda_i^h(t)$. This gives the expected travel time to the final destination, given that the vehicle is at the stop line before node i at time t and approached node i from node h, which is the left picture of a vehicle in Figure 3.2. In this figure the notation of the nodes is visualized as well, with i being the current node, h being the node a vehicle approached node i from and j being the next node. The

notation for node j is required since a vehicle will drive one node ahead, if the traffic light is green, and the label saved at that node will be used to calculate the label at the previous node. Therefore this label is denoted as $\lambda_j^i(t)$ since it is one node ahead. Notice that being at node i means being just before the stopline at node i while coming from node h. Therefore when approaching node i form a different direction, it is saved as a different label.

The link travel time distribution between node i and j should also be known since after crossing the intersection the vehicle will drive towards the stopline of the next node. The link travel time between node i and j, when getting a green light at node i at time t, is given in $\tau_{ij}^k(t)$. k is an index to denote each possible link travel time from node i to node j at time t. This also contains acceleration loss at the intersection since the part that takes traffic lights into account, only account for a probability of being green and not the delay while accelerating or decelerating. In Figure 3.2 $\tau_{ij}^k(t)$ is visualized as the time it takes to drive the length of the arrow.

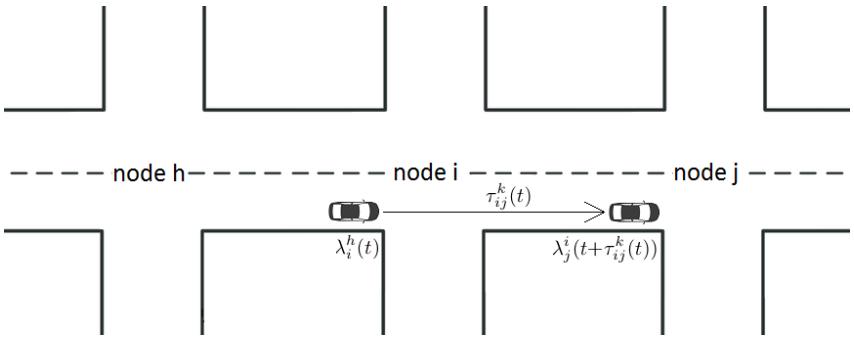

Figure 3.2: Visualization of the nodes, with $\lambda_i^h(t)$ being the travel time to the destination while at node i, while coming from node h at time t and $\tau_{ij}^k(t)$ being the link travel time from node i to node j while getting a green light at node i at time t. k is an index to denote each possible link travel time.

The network is searched backwards from the destination node towards the departure node. This is done systematically and the order is shown in Figure 3.3. For each combination of nodes the expected travel time to the destination node will be calculated. The exact formula for this calculation is given later in this section. In this figure, node 1 is the destination node and therefore the labels at this node $\lambda_1^2(t)$ and $\lambda_1^3(t)$ will be set to 0 for all t. Then step 1a can be calculated and this is calculating the expected travel time from node 2, going to the destination node, while coming from node 4 ($\lambda_2^4(t)$). Then the other upstream node of node 2 can be picked, node 5, to calculate step 1b. Then the next node i is picked and the same steps are repeated. Once all upstream nodes of node j are calculated the first node i becomes node j and this repeats itself until it reaches the departure node. As shown in the figure, all vehicles drive from left to right in the network and looping back is not allowed. The calculation of the labels could deal with looping routes, but this will result in the a higher expected travel time and therefore routes with a complete loop are never optimal for the lowest expected arrival time. But the part of the algorithm that selects the next node cannot deal with looping routes. This is because it selects the upstream nodes of the current node, and if for instance node 2, 4, 7 and 5 are upstream nodes of each other, the algorithm will loop forever. Therefore looping routes cannot be used without adjusting the algorithm.

The label correcting part from the name means that if a lower value is found for the expected travel time ($\lambda_i^h(t)$), the previous value is overwritten. Therefore every time, the expected travel time from a node to the destination has been calculated, it is compared with the already known shortest expected travel time from that point to the destination. If the new value is lower, it is saved to the node, otherwise it is discarded. This can for instance occur at step 2b and 3a in Figure 3.3. In both these cases the expected travel time is saved towards $\lambda_i^h = \lambda_5^7$. But since the route differs afterwards (2b goes via 2, and 3a goes via 3) the expected travel time can be different. Therefore only the shortest expected travel time will be saved. If 2b saved an expected travel time of 7 minutes and in step 3a the expected travel time is calculated as 6 minutes, route 3a will overwrite route 2b and route 2b will be discarded.

Besides the network being searched backwards from the destination node to the departure node , it is also searched backwards in time. A time step (t) is used to make time discrete, which can be chosen by the user. As a starting point, a maximum time should be given, which is denoted in the maximum number of time steps, S. Therefore the algorithm calculates backwards from time step S to time step 0. So if the algorithm is run at 8 o'clock exactly, with time steps of 1 second, $t = 0$ is at 8 o'clock exactly while $t = 120$ is at 8:02. The maximum number of time steps, S, should be picked manually and should be dependent on the size of the network, the waiting times at the traffic lights and the link travel times. It is important that it is picked high enough that the expected travel times of most routeplans are calculated completely, while picking it too high will result in unnecessary extra calculations for the algorithm.

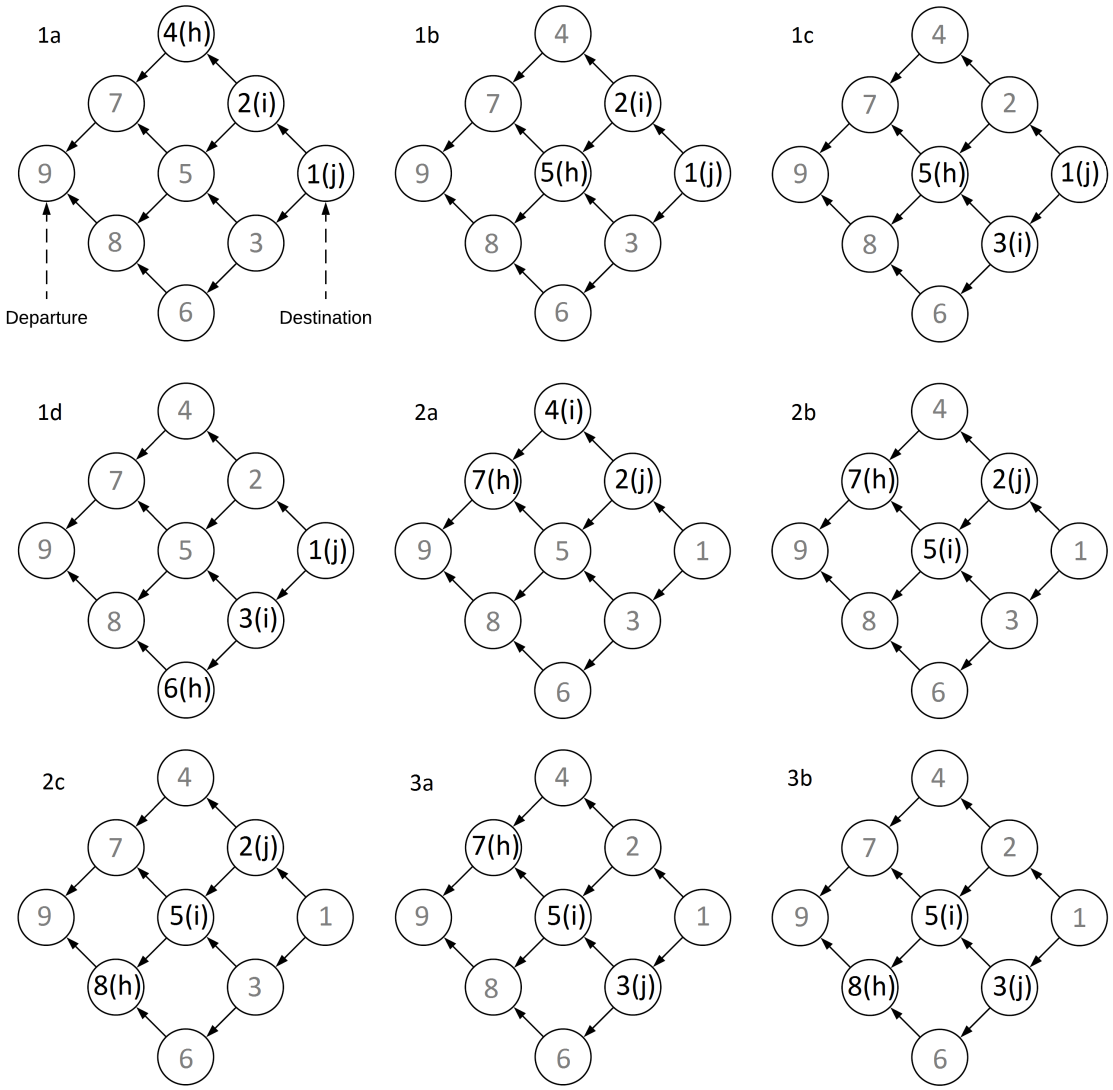

Figure 3.3: Visualization of the order of searched nodes, note that the vehicles drive in the direction opposite to the arrows.

B. Yang and Miller-Hooks (2004) does not specify how they picked the maximum number of time steps, but they do specify the need of a maximum number. With some testing an upper boundary can be picked that (almost) always gives the desired results. This can be for instance the 99.5^{th} percentile of the travel time distribution from departure to destination node, which will lead to incomplete routeplans in 0.5% of the routeplans that depart from the departure node at $t = 0$. The maximum number of time steps could be increased to very large numbers, but there will always be some routeplans that are not calculated completely yet. This is because the states of the traffic lights are probabilistic and therefore a certain probability of a red traffic light is (almost) always present. And since the algorithm does not take a maximum waiting time at a traffic light into account, there is a small probability of waiting very long for a traffic light. This happens if there is a probability of 0.7 to have a red light. And the next time step there is again a probability of 0.7 of a red light, so the probability to to wait 2 time steps is $(0.7 \cdot 0.7 =)$ 0.49. This will lead to ever decreasing probabilities of an ever increasing travel time. To make sure that most routeplans are complete, the traffic lights are all set to green a specific time before the longest possible travel time considered (S). Adding up all the maximum link travel times could be a value for this time since this is the only delay left in the system after setting all traffic lights to green. But since the vehicle is often already somewhere in the system this value might be too high and a lower value might be more efficient.

The last notation that needs to be introduced is given by $Z_j^{hi}(t)$, which is a value between 0 and 1. This gives the probability of the traffic light being green, while coming from the link between node h and i and going towards node j. This value differs per time step a vehicle arrives at the traffic light and therefore t denotes the arrival time

at the traffic light. If the traffic light is green there might still be a delay since there could still be traffic waiting for the traffic light but this is not taken into account with this value. Also extra delay when the traffic light is red (for instance due to traffic slowly accelerating when turning green) is not taken into account. The delay is calculated as the number of seconds that the traffic light is not green. If a more elaborate model of local traffic is used, the traffic light states and link travel time distributions might be dependent on each other which might result in more precise predictions. But for this research these are not dependent on each other, and therefore the link travel time distribution and the traffic light probabilities are assumed independent of each other.

The main formula

Now that most variables are defined, the main formula from the ARSC algorithm will be explained. With this formula the label $\lambda_i^h(t)$ will be calculated. But since it is a label correcting algorithm, the calculated value might be replaced with a new value. Therefore the main formula will be calculated with a temporary label, $\eta_{ij}^h(t)$, and afterwards the lowest temporary label will be saved to $\lambda_i^h(t)$. An extra node, node j, is used in defining $\eta_{ij}^h(t)$. This is because η will be calculated for all different nodes j, which are all nodes you can reach from i, and the lowest expected travel time will become λ. This is shown in Equation 3.1, with $D(i,h)$ being the set of downstream nodes of node i, that can lead to the destination, and can be reached when the vehicle is coming from from node h.

$$\lambda_i^h(t) = \min_{j \in D(i,h)} \eta_{ij}^h(t) \tag{3.1}$$

For every η the main formula will be calculated to find the expected travel time. In this formula $\tau_{ij}^k(t)$ represents the link travel time with k being an index to denote each possible link travel time from node i to node j at time t. This formula consists of two parts, one for if the traffic light is green and one for if it is not. Since an expected value is calculated, both are used and multiplied with $Z_j^{hi}(t)$ for the part where the traffic light is green or $1 - Z_j^{hi}(t)$ for the part where it is red. The first part of the equation that deals with a green traffic light starts with $Z_j^{hi}(t)$ up to the plus sign just before $1 - Z_j^{hi}(t)$, where the second part starts. The first part of this formula takes the link travel time to the next node, $\tau_{ij}^k(t)$, and adds the label at the next node, $\lambda_j^i(t + \tau_{ij}^k(t))$, which contains the expected travel time from that point to the destination node. This is taken at the time $t + \tau_{ij}^k(t)$ since the link travel time it takes to get to the next node should be taken into account as well. This is added together and multiplied by the probability of this link travel time happening, $p_{ij}^k(t)$, and then added up for all possible link travel times, k. The number of different link travel times there are on each link is denoted as $k_{ij}^{max}(t)$, this is dependent on the link since each link can have its own link travel time distribution, and the time the link is entered, since the travel time distribution could be different from moment to moment. Once this is multiplied with the probability of the traffic light being green, $Z_j^{hi}(t)$, it gives the first part of the new expected travel time. The second part is the part when the traffic light is red. Instead of finding the moment when it turns green, it just takes the current label one time step later and adds a time step before using it in this calculation. Since queueing effects are not taken into account in the original algorithm it does not matter when the vehicle arrived at the node since all vehicles will depart at the same time when the traffic light turns green. This results in the following main formula of the ARSC algorithm:

$$\eta_{ij}^h(t) = \overbrace{Z_j^{hi}(t) \sum_{k=1}^{k_{ij}^{max}(t)} \left\{ \left(\tau_{ij}^k(t) + \lambda_j^i(t + \tau_{ij}^k(t)) \right) \cdot p_{ij}^k(t) \right\}}^{1} + \overbrace{\left(1 - Z_j^{hi}(t)\right) \cdot \left(\lambda_i^h(t+1) + 1\right)}^{2}. \tag{3.2}$$

This equation will lead to a new calculated value for the expected travel time, $\eta_{ij}^h(t)$, and the lowest $\eta_{ij}^h(t)$ will be saved as the current label, $\lambda_i^h(t)$. This have to be done for all nodes from to destination node, up to the departure node, and also for all time steps from the maximum number of time steps, S, to 0. Therefore the order as explained before, in Figure 3.3, is kept, but for each combination of nodes this is calculated from time step S to time step 0, before this is calculated for the next combination of nodes. This will result in some values for t being larger than S. For instance if $t = S$ for a calculation and $t + \tau(t)$ is calculated. These calculations are discarded, since the traffic lights should be set to green with a probability of 100%, on a specific time, in a way that every routeplan that starts at $t = 0$, will reach the destination node before time step S. Those discarded values will therefore not be used in the final routeplan.

To keep track of the optimal path another label is introduced, the pointer label, $\pi_i^h(t)$. This label saves for node i coming from node h, the optimal next node j, for which Equation 3.1 holds. This is therefore the next node to go to j, while being at node i, coming from node h at time t, with the lowest expected travel time to the destination node.

The formula for this is given in 3.3. If there are multiple next nodes j for which $\eta_{ij}^h(t)$ is minimal, only the one with the lowest number for j will be saved to $\pi_i^h(t)$ since just one node can be saved.

$$\pi_i^h(t) = \arg\min_j(\eta_{ij}^h(t)) \mid j \in D(i,h) \tag{3.3}$$

Once the algorithm has calculated trough all the nodes, the optimal next node to travel to is saved to this pointer label for every label and every time step. Therefore the optimal routeplan can be traced and if a disturbance occurs the route can be adjusted since the optimal next node is saved for every arrival time at every label. But it does not give the reliability of arriving on time. The lowest expected travel time might be a 50% chance of arriving in 10 minutes and a 50% chance of arriving in 20 minutes, while some might prefer the option with a 50% chance on 15 minutes and a 50% chance on 16 minutes since this is more reliable but leads to a higher expected travel time. Therefore the reliability should be taken into account as well, which will be done in Section 3.1.3.

3.1.2. The objective function

The reliability can be taken into account in many different ways and calculated with different formulas (Z. Chen & Fan, 2019). The exact definition of some of those methods are explained later on in this section but an algorithm that can be used to calculate the reliability of a routeplan requires in almost all cases a travel time distribution.

Reliability can mean something different for every user, some travellers want a high probability of arriving at their destination before a certain time, some travellers want a high probability of arriving in a certain time slot (thus also not too early), some travellers want to arrive close to their predicted arrival time, etc. To deal with all these different definitions of reliability and the different ways of calculating the reliability, an objective function is introduced. In this objective function, the user of the algorithm can create their own function to take a reliability measure into account. This gives the user the freedom to create functions however they like, as long as it is based on the travel time distribution. The travel time distribution is a good input since it gives all possible arrival times with their probability of occurring. This can then be used to calculate reliability measures, for instance the percentage of arrivals before a certain time. This travel time distribution is given by $\Lambda_i^h(t)$ and the probability on a travel time λ is given by $Pr\big(\Lambda_i^h(t) = \lambda\big)$. $\Lambda_i^h(t)$ is the travel time distribution to the destination while being at node i, coming from node h at time t. $Pr\big(\Lambda_i^h(t) = \lambda\big)$ gives the probability of a travel time to the desination, λ, happening. With this distribution most measures of reliability can be calculated.

Conditions for the ARSC algorithm

But if this objective function will be implemented into the original ARSC algorithm, some problems may arise. Since the original ARSC algorithm is a backwards algorithm that determines and saves the optimal next node from each node at every timestep, calculation of the optimal next node to take should be possible from every node, before the entire distribution of the complete routeplan is known. This leads to the condition when an adaptation of the original ARSC algorithm can be used:

Condition 1 *The optimal routeplan from a label to the destination node, at a specific time, should be independent of data that will be added at every upstream label.*

The upstream label is mentioned since the algorithm is calculated backwards and therefore the upstream label has more information. All functions that fulfil the aforementioned condition are called type 1 functions, while all functions that do not, are called type 2 functions.

The original ARSC formula, with the lowest expected arrival time, met this condition. But some measures of reliability do not. An example of this is the standard deviation, as shown in Equation 3.4. Equation 3.5 shows how to calculate the mean, which is used in Equation 3.4. This gives a good indication of the variance in the travel times and is therefore often used as a reliability measurement. The main condition does not hold, since the optimal routeplan of a downstream node can be dependent on data added upstream.

$$O_i^h(t) = \sqrt{\sum_{\lambda \in \Lambda_i^h(t)} Pr\big(\Lambda_i^h(t) = \lambda\big) \cdot \left(\lambda - \overline{\lambda}\right)^2} \tag{3.4}$$

$$\overline{\lambda} = \sum_{\lambda \in \Lambda_i^h(t)} Pr\big(\Lambda_i^h(t) = \lambda\big) \cdot \lambda. \tag{3.5}$$

To show that the main condition does not hold, the following example is given, as visualized in Figure 3.4. Two different routeplans can be taken, via node B and C to the destination or via node B and D to the destination. But

when a vehicle is departing from node A, the link travel time to B can be 11 or 12 seconds. This leads to two different arrival times at node B, which can lead to different travel times to the destination node. In a normal situation there would be more different travel times but this is kept simple for the example. The probability on a each link travel time is 50%. On the right side of the figure, the probability on each travel time from node B to the destination node is given. Multiple values represent the travel time distribution here, since variations in link travel times or waiting times at the traffic light occur. For simplicity just two values per route are given and the probability on both values is 50% as well for all routes.

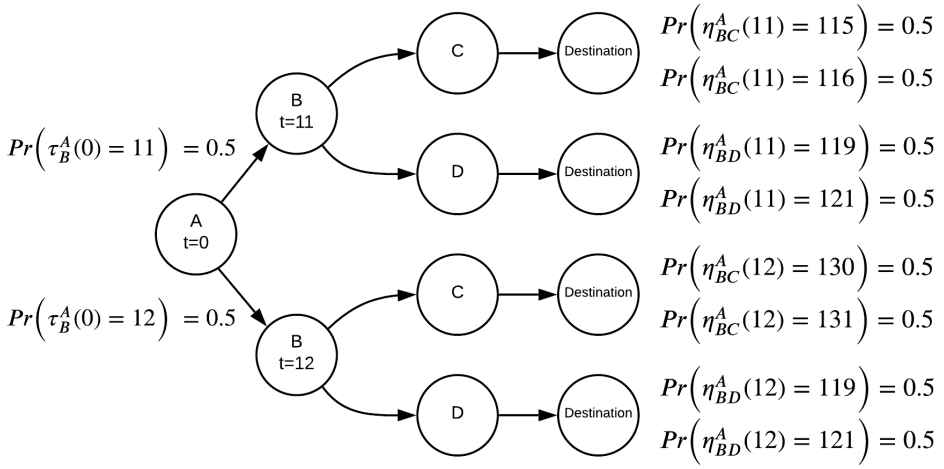

Figure 3.4: Illustration of the network used in the clarification of why the standard deviation is a type 2 function.

At this point the optimal routeplan can be calculated in this network, according to the lowest standard deviation objective function, given in Equation 3.4. When working backwards, node B to node C and D is calculated before node A is taken into account. Since the algorithm also calculates backwards in time, node B at time $t = 12$ is calculated first. The standard deviation of the travel time to the destination node, while arriving at node B at $t = 12$ and taking the route via node C, is $\sqrt{(0.5 \cdot (130 - 130.5)^2) + (0.5 \cdot (131 - 130.5)^2)} = 0.5$. For the route via node D, while arriving at node B at $t = 12$, the standard deviation is 1. Therefore the route via node C has a lower standard deviation and this route is therefore better. If node B is reached at $t = 11$, the standard deviation, while taking the route via node C, is 0.5 as well. The standard deviation of the route via node D is 1. Therefore, when arriving at node B at $t = 11$, route C is chosen as well.

If this objective function would satisfy the condition given before, the optimal routeplan should not change when it is calculated from an upstream node with more data available. To check this, the optimal routeplan is searched from node A. Since the optimal routeplan can be dependent on the arrival time at node B, the optimal routeplan might be to travel via C when arriving at node B at $t = 11$ and via D when arriving at node B at $t = 12$. Therefore there are four possible combinations. 1: Going via node C, while arriving at node B at $t = 11$ and going via node D when arriving at node B at $t = 12$. 2: Going via node D while at node B at $t = 11$ and going via node C while at node B at $t = 12$. 3: Going via node C both times. 4. Going via node D both times. The standard deviation of these routeplans is given in Table 3.1. This is calculated by using the objective function, the travel times and the probability on these travel times. The probability on each travel time is in this case 0.25 since there was a probability of 0.5 of the link travel time between A and B and a probability of 0.5 of having one of the two travel times from node B to the destination.

This shows that the optimal routeplan, when calculated from node A, is to go via node D when arriving at B at $t = 11$ and going via node D when arriving at node B at $t = 12$. Therefore the standard deviation does not satisfy the condition, because when optimizing at node B the optimal route is C, when arriving at node B at $t = 11$, and node C as well, when arriving at node B at $t = 12$. Since this routeplan is different compared to the routeplan when node A is taken into account, the optimal routeplan does change with data added at an upstream label. Therefore this is an example of a type 2 function, that does not meet the condition.

Table 3.1: The standard deviation of the routeplans. Calculated with Equation 3.4 and the data from Figure 3.4.

Routeplan	Travel times from node A to the destination	STD
At B at $t = 11$: C At B at $t = 12$: C	126, 127, 141, 142	7.52
At B at $t = 11$: C At B at $t = 12$: D	126, 127, 130, 132	2.38
At B at $t = 11$: D At B at $t = 12$: C	130, 132, 141, 142	5.31
At B at $t = 11$: D At B at $t = 12$: D	130, 132, 130, 132	1

The special case for type 1 functions

To clarify when a function is a type 1 function, a special case is found which is always a type 1 function. This is explained by the following example, which uses the frequency of congestion as the objective function. This is the frequency of trips that exceeds a threshold (travel time) value (Z. Chen & Fan, 2019). This could be interpreted as the probability of exceeding a certain travel time. This is common practice for trucks that need to arrive before a certain time but is also applicable to leisure trips where people do not want to arrive too late. To create an objective function that facilitates this, a threshold target value is introduced and given in the function as Ω. This target could be any positive number of time steps. The objective formula that facilitates the calculation of this should add up all the probabilities of having a travel time higher than the target, since it is determining the probability of exceeding the target travel time. This results in the following example of an objective function:

$$O_i^h(t) = Pr(\Lambda_i^h(t) > \Omega). \tag{3.6}$$

This is a function of type 1, since the condition given before holds. This is because only the probability of arriving after a target time is passed along upstream. If for example a vehicle has a 30% chance of a green traffic light and a 70% of a red traffic light. The objective value can be calculated by 0.3 times the objective value if the traffic light is green, plus 0.7 times the objective value for if the traffic light is red, just as how it is done with the ARSC formula. Therefore these can just be added up, while being multiplied with their corresponding probability. This is comparable to the lowest expected travel time in the original ARSC formula, but now the expected chance of arriving after the target number of time steps is passed along. Since the algorithm is a backwards algorithm, the travel times to the destination could be known from the beginning, but the probabilities on those arrival times cannot. This leads to a specific case of the condition given before:

Condition 2 *If an objective value of an upstream label, can be calculated by adding up the probabilities of arriving at a downstream label, at a specific time, times the corresponding objective value, than it is a type 1 function.*

This can be written down mathematically in the following Equation:

$$O_i^h(t) = \sum_{j \in D(i,h)} \sum_{k}^{k_{ij}^{max}} p_{ij}^k(t) \cdot O_j^i\left(t + \tau_{ij}^k(t)\right) \tag{3.7}$$

All objective values should be able to be calculated with this equation, if the condition of the special case holds. Therefore if the general, special or both conditions apply to a function, it is a type 1 function.

Mathematical functions in the objective function

To show that also other mathematical functions can be used in the objective function as well, but that a subtle difference might change the type, the following example is given. Companies try to reduce their costs as much as possible, and arriving too late (or too early) can lead to additional costs. If for instance the costs of arriving too late or too early increases quadratic, this can be incorporated in the objective function. In the following equation this is shown by squaring, the possible arrival times minus the target arrival time (set again as Ω), before multiplying them with the probability of occurring:

$$O_i^h(t) = \sum_{\lambda \in \Lambda_i^h(t)} \Big(\big(\lambda - \Omega\big)^2 \cdot Pr\big(\Lambda_i^h(t) = \lambda\big) \Big). \tag{3.8}$$

This is a type 1 function since this formula satisfies the main condition to be a type one function, and also satisfies the condition of the specific case. All values can be passed along with the probability of occurring since it is based on every travel time separately. From every travel time separately the target travel time will be subtracted and this will be squared, afterwards this is multiplied with the probability of occurring. If for instance the mean would be used, instead of every travel time separately, the optimal routeplan would become dependent on data added upstream and therefore it would become a type 2 function. This might seem unexpected since the mean on its own is a type 1 function. But this is the case since a lower mean is not always better, the mean will try to become as close as possible to the target value. Thus a slightly higher travel time than the target, when for instance the traffic light is red, might be beneficial if a slightly lower travel time is encountered when the traffic light is green. The mean of those 2 values might be closer to the target compared to a value exactly equal to the target value and a higher value combined. An example of the formula of this type 2 function is shown hereafter:

$$O_i^h(t) = \Big(\sum_{\lambda \in \Lambda_i^h(t)} \Big\{ \lambda \cdot Pr\big(\Lambda_i^h(t) = \lambda\big) \Big\} - \Omega \Big)^2 \tag{3.9}$$

This is not a type 1 function since having a higher arrival time might be beneficial if data added upstream also adds a probability of a lower arrival time. Therefore the optimal routeplan is dependent on data from upstream labels and this is a type 2 function.

3.1.3. The ARSCR algorithm

Now that the objective function and the 2 types of functions are explained, it is clear that an adjustment of the ARSC algorithm can only be used in certain cases. If an objective function is type 1, the ARSC algorithm can be expanded to facilitate the usage of an objective function. This expansion of the ARSC algorithm into the ARSCR (Adaptive Routing with Signal Controls and Reliability) algorithm, will be explained in this section.

The concept of the algorithm will stay the same as the ARSC algorithm, and therefore this is also a backwards label correcting algorithm. But to facilitate the usage of reliability in the ARSCR algorithm, a few adjustments have to be made. First of all, not just the expected travel time should be saved at the labels, but the travel time distribution, containing multiple travel times and their corresponding probabilities, as well. Secondly, the optimal routeplan should not be the lowest expected travel time, but the optimal routeplan according to the objective function, that takes the travel time distribution as input. Lastly the main formula should be adjusted to calculate the probability distribution instead of the lowest expected travel time.

Adjusting the labels

Since a travel time distribution should be saved at the labels, the original labels should be adjusted. In the original algorithm the label, $\lambda_i^h(t)$, contains the travel time to the destination node. This is sufficient for the lowest expected travel time, but a distribution cannot be stored. Therefore the variable is changed into $\Lambda_i^h(t)$ which contains the travel time distribution to the destination node, while being at node i at time t and coming from node h. But for the distribution, the probability on that travel time should be stored as well. Therefore, $Pr\big(\Lambda_i^h(t)\big)$ is added, which stores the probability on the travel times. These probabilities, in combination with the travel times, give a probability distribution of possible travel times to the destination node, while being at node i at time t and coming from node h. Since η is still used, η, should be adjusted in the same way as the label, λ. This results in $H_{ij}^h(t)$ and $Pr\big(H_{ij}^h(t)\big)$ which gives the probability distribution of the temporary labels.

Now that the labels are adjusted, the optimal routeplan can be saved depending on the objective function and travel time distribution. The ARSC algorithm, as described in Section 3.1.1, calculates the temporary label, $\eta_{ij}^h(t)$, which is the expected travel time, first and only saves it to the label, $\lambda_i^h(t)$, if the temporary label is lower. But since this does not take any kind of reliability into account the algorithm is adjusted to use the travel time distribution instead of the expected travel time. To also facilitate the usage of an individual cost function, as given in the goal of this research in Section 1.2, the objective functions is used to check if the objective value is improved. This is done by checking which of the objective values of the different $H_{ij}^h(t)$, compared with all possible j, is best. j can be all the downstream nodes of i, so when the optimal j is found for a minimal objective value of $H_{ij}^h(t)$, the optimal downstream node to travel to is found. Therefore Equation 3.1 is changed into Equation 3.10, which searches for the distribution with the lowest objective function. If a higher objective function is better, the maximum should

be searched for instead of the minimum, or the objective function could be multiplied with -1. Since the objective function changes from user to user, a general notation is used. In this notation the objective function of node i while coming from node h at time t, $O_i^h(t)$, is dependent on the temporary travel time distribution, formed by $H_{ij}^h(t)$ and $Pr\left(H_{ij}^h(t)\right)$.

$$\Lambda_i^h(t) = H_{ij}^h(t) \;\Big|\; j = \underset{j \in D(i,h)}{\arg\min}\; O_i^h\left(H_{ij}^h(t), Pr\left(H_{ij}^h(t)\right)\right) \tag{3.10}$$

The new main formula

The main formula also needs to be adjusted to work with the travel time distributions. Since the travel time distributions should be calculated and not the expected travel time, entire distributions should be passed along. Therefore the probabilities on the different travel times are calculated. The probability on a specific travel time, λ, from the temporary label, $H_{ij}^h(t)$, is denoted as $Pr\left(H_{ij}^h(t) = \lambda\right)$. This probability is dependent on multiple variables. First of all the probability on a specific link travel time, $p_{ij}^k(t)$. Every link travel time has a probability of occurring and this should be taken into account. Secondly the probability of a green traffic light, $Z_j^{hi}(t)$, (or red traffic light, $1 - Z_j^{hi}(t)$,) should be known. This is required since the probability of the travel time to the destination is calculated differently for when the traffic light is green, and when it is red. When it is green, the probability of a green light, the probability of that specific link travel time and the probability of the travel time from the downstream label to the destination node, is used. And when it is red, the probability of a red traffic light and the current label, but one time step later, is used for calculating the travel time to the destination node. This leads to the last part, the probability of having a travel time to the destination node from the label downstream (if the traffic light is green), $Pr\left(\Lambda_j^i(t)\right)$, or the same label one time step later (if the traffic light is red), $Pr\left(\Lambda_i^h(t+1)\right)$.

Now that all notations are explained, the formula itself can be explained. In the formula all probabilities on a certain travel time, $Pr\left(H_{ij}^h(t) = \lambda\right)$, should be calculated. Since λ is the travel time from node i to the destination node, the link travel time between node i and j and the travel time from node j to the destination should be taken into account. Therefore different probabilities are added up, since the same travel time from node i to the destination node can be the result of a combination of different link travel times and travel times to the destination node at the downstream label j. Therefore, for the case the traffic light is green, the probabilities are added up for all cases in which the link travel time and travel time at the downstream label added up are equal to this specific travel time λ. Therefore the probability of a certain link travel time occurring, $p_{ij}^k(t)$, is multiplied with the probability that the travel time from the downstream node, to the destination node, is equal to λ minus the link travel time, $Pr\left(\Lambda_j^i(t+\tau_{ij}^k(t)) = \lambda - \tau_{ij}^k(t)\right)$. If this is added up for all possible link travel times and multiplied with the probability of the traffic light being green, it results in the first part of equation 3.11.

For a red traffic light the current label, at one time step later, should be equal to the travel time λ minus 1, since the vehicle will wait one timestep at the traffic light before using the travel time from that label. This should be multiplied with the probability on a red traffic light to get the second part of the equation. If both those probabilities are added up the total probability of a specific travel time from the current node to the destination node is known, and this results in the following formula:

$$Pr\left(H_{ij}^h(t) = \lambda\right) = \overbrace{Z_j^{hi}(t) \cdot \sum_{k=0}^{k_{ij}^{max}} p_{ij}^k(t) \cdot Pr\left(\Lambda_j^i(t+\tau_{ij}^k(t)) = \lambda - \tau_{ij}^k(t)\right)}^{1} + \overbrace{\left(1 - Z_j^{hi}(t)\right) \cdot Pr\left(\Lambda_i^h(t+1) = \lambda - 1\right)}^{2}. \tag{3.11}$$

To get the full travel time probability distribution this equation needs to be calculated for all possible values of λ. This can simply be noted as $\lambda \in H_{ij}^h(t)$, but for clarity the complete set $H_{ij}^h(t)$ is explained. A range of values between a minimum and a maximum possible travel time from this label to the destination can be part of $H_{ij}^h(t)$. These vales are dependent on the minimum (or maximum, respectively) link travel time and minimum/maximum travel time from the downstream node to the destination node. Besides this value, the travel time from the node one second later is used if the traffic light is red, therefore the minimum/maximum of this should also be taken into account as well. This results in the set of all numbers between $\min\left\{\min_{k \in k_{ij}^{max}}\left(\tau_{ij}^k(t) + \min \Lambda_j^i(t+\tau_{ij}^k(t))\right), \min \Lambda_i^h(t+1) + 1\right\}$ and

$$\max\Big\{\max_{k\in k_{ij}^{max}}\Big(\tau_{ij}^{k}(t)+\max\Lambda_{j}^{i}(t+\tau_{ij}^{k}(t))\Big),\max\Lambda_{i}^{h}(t+1)+1\Big\}.$$

Once Equation 3.11 is calculated for all λ and for all j that are downstream nodes of i and reachable from node h, the optimal routeplan for this part can be known. To determine this, the objective function is used to determine the optimal travel time distribution $Pr(\Lambda_i^h(t))$, which will be equal to the optimal travel time distribution from all different temporary travel time distributions $Pr\Big(H_{ij}^h(t)=\lambda\Big)$. For this a formula is introduced, that saves the temporary travel time probability label to the travel time probability label, if the objective function is better:

$$Pr(\Lambda_i^h(t))=Pr(H_{ij}^h(t))\;\Big|\;j=\underset{j\in D(i,h)}{\operatorname{argmin}}\;O_i^h\Big(H_{ij}^h(t),Pr\big(H_{ij}^h(t)\big)\Big). \tag{3.12}$$

If this is repeated for all nodes in the network, as described in 3.1.1, it will result in the optimal, time dependent routeplan to the destination.

3.1.4. The FPP algorithm

Since the ARSCR algorithm cannot be used with type 2 functions, another algorithm is designed that can be used with all functions. This new algorithm no longer tries to know the optimal routeplan at every intermediate node, since the optimal routeplan might not be known at intermediate nodes. Therefore a new algorithm is designed that calculates the probability distribution for every possible routeplan and compares these with other routeplans, once those are completely calculated as well. The new algorithm is a forwards algorithm, and therefore only the exact time steps a vehicle has a probability (greater than 0) of arriving at a next node are taken into account for calculating the arrival times at the nodes after that node. This skips calculations at some time steps at some nodes. But since all routeplans are searched completely and the optimal routeplan cannot be known at intermediate nodes, the algorithm might become slow in a network. Therefore an algorithm that makes a preselection of links that should be calculated in the network is advised. Since the algorithm is changed from a backwards to a forwards algorithm and the probability is propagated forwards for all routeplans, the new algorithm is called the Forward Probability Propagation (FPP) algorithm.

There are three main differences between the ARSCR algorithm and the FPP algorithm. In the first place, the FPP algorithm can use all functions as objective function and not just the type 1 functions. Secondly, the FPP algorithm is a forwards algorithm and can therefore incorporate waiting times at the traffic light more accurately, this will be explained in the next subsection. And lastly, the FPP algorithm calculates all routeplans completely and does not know optimal routeplans at intermediate nodes. Therefore it calculates all routeplans separately and compares the objective values of the routeplans once they have been calculated completely.

The required input

The algorithm will use the traffic light prediction module and link travel time prediction module as input. For the link travel time, $\tau_{ij}^k(t)$ and p_{ij}^k are calculated in the same way as in the ARSCR algorithm, but the notation is different to create a consistent notation in this section. The set of all possible link travel times will be noted as $K_{ij}(t)$. The probability on link travel time k is then: $Pr(K_{ij}(t)=k)$. Therefore $K_{ij}(t)$ is now the set of all possible link travel times from node i to node j when departing at time t, the calculation of this is explained in Section 3.3.

For the traffic light prediction model, using a forwards algorithm, allows for an improved inclusion of the prediction of the traffic signal. In the ARSC(R) algorithm the probability of green is calculated for a signal group and time. This does not take into account how long a vehicle is already waiting for a red traffic light. This is not possible because it is a backwards algorithm and therefore it is not known, how long the vehicle is already waiting for a red traffic light. Also the probability of being at that node, is a combined result of arriving at different moments. With the forward FPP algorithm it is possible to know how long a vehicle is already waiting, since it knows when a vehicle arrives at the node and can therefore predict waiting times depending on this arrival time. This results in more precise predictions. Another benefit is that the maximum cycle time can be taken into account correctly and therefore vehicles cannot wait longer than that time. To incorporate this the traffic light prediction module need to be adjusted to return a distribution of the waiting time at the traffic light, dependent on the arrival time of the vehicle at that traffic light. This is explained in Section 3.2 and results in $L_{hij}(t)$ being the set of all possible waiting times when arriving at time t at the traffic light at node i while coming from node h and going to node j. The probability on waiting time l is then: $Pr(L_{hij}(t)=l)$. The waiting times are the number of timesteps the traffic light is red, before it turns green. Therefore this is dependent on the moment of arrival (t).

The concept of the FPP algorithm

The concept of the new algorithm is to calculate the travel time distribution to the destination node, depending on the routeplan taken. This routeplan is location and time dependent, since a different next link could be taken at a node if a vehicle arrives at that node at a different time. Since the calculations are done forwards, from the departure node and departure time to the destination node, some nodes, which have a probability of zero of occurring, can be skipped. The probability of arriving at a next node at a certain time is dependent on the waiting time at the traffic light and the link travel time. Since those are given in a distribution, the arrival time at the nodes is a distribution as well. Therefore, for every node, the travel time distribution at the upstream node, the waiting time distribution at the upstream traffic light and the link travel time distribution are added up. This will (often) result in a more spread out distribution. With this method all nodes are calculated from the departure node up to the destination node, which will result in a travel time distribution at the destination node, on which the objective function can be used. Once the objective value has been calculated for all routeplans, the optimal routeplan is known.

Since all combinations of possible next links, depending on the arrival time at the node before that link, have to be calculated, the set of all routeplans is defined as R, with $r \in R$ being all possible routeplans. To define the next node to travel to from any point in the network a new function is created:

$$j = J^r(h, i, t). \tag{3.13}$$

This function returns the next node to travel to j, depending on the node the vehicle is, i, at this time t, while coming from node h. This function is different for every routeplan r and can give a different next link for every routeplan. If a vehicle will arrive a timestep later, the variable t has changed and therefore the next node to drive to, j, can be different. Where the number of different options as next nodes is usually limited to a few (3 for a four way intersection), the number of different arrival times at those nodes, t, can take many different values. But those values should always be higher than or equal to the minimum travel time to that node, t_{min}, and lower than or equal to the maximum travel time to that node, t_{max}. These values can be calculated as the minimum (respectively maximum) waiting time plus the minimum/maximum link travel time of all upstream links. Since the probability of a vehicle being at a specific node at the minimum or maximum time might be zero, a more elaborate selection method might be used that only uses the arrival times at that node which have a probability of occurring. But taking more values into account is not a problem since the algorithm will simply assign them a probability of 0.

Now the routeplans are explained, the new formula can be introduced. For this formula, the new variable $X^r_{i,j}$ is introduced, which gives the travel time distribution from the departure node to node j and arriving at node j from node i, while following routeplan r. The r makes sure that this is calculated separately for every routeplan. The probability on a travel time λ is given by $Pr(X^r_{i,j} = \lambda)$. This probability is set to 1 for $Pr(X^r_{f,h} = 0)$, given that h is the departure node, f is the predecessor node of h (required since the vehicle is on the link before that node and not on that node exactly) and the vehicle arrived at the departure node at time $t = 0$.

To calculate the probability on the travel time, $Pr(X^r_{i,j} = \lambda)$, the probability distribution should be known. To calculate this distribution the travel time distribution at the upstream node should be known as well, $Pr(X^r_{h,i})$. This can be visualized using the example in Figure 3.5, in which $Pr(X^r_{i_1,j_1} = \lambda)$ can only be calculated if $Pr(X^r_{h,i_1})$ is known. The next part that needs to be known is the travel time from i_1 to j_1 which consists of a waiting time from the set $L_{hi_1j_1}(t)$ and a link travel time from the set $K_{i_1j_1}(t)$. This can be a combination of different waiting times and link travel times, as visualized by the solid lines in the enlargement in Figure 3.5. In this figure, when the vehicle is at i_1 at time t, an arrival at j_1 at time t+2 can be reached trough a waiting time of 0 seconds and a link travel time of 2 seconds, and with a waiting time of 1 second and a link travel time of 1 second. Therefore a certain total travel time from link i_1 to link j_2 can be a combination of a waiting time and a link travel time. Different combinations can lead to the same total travel time and they should therefore be added up.

But besides the different combinations of waiting times and link travel times that could lead to the same arrival time at node j_1 in figure 3.5, a different arrival time at the node upstream (i_1) could still result in the same arrival time at node j_1. This is shown in the figure with the dotted double lines. These show that a vehicle that arrives at node i_1 at $t+1$ can arrive at $t+2$ at j_1. Since the solid line shows that a vehicle who arrives at time t can also arrive at time $t+2$ at j_1, the arrival times at the upstream node should be taken into account as well. Therefore not only a combination of waiting time and link travel time that leads to a certain travel time is sought, but a combination with the arrival time at the upstream node, the waiting time and the link travel time.

And this goes even one step further. Since the FPP algorithm is not calculated for a route, but a routeplan, there could be multiple nodes from which the current node is approached. This is shown in Figure 3.6, which shows that if node j_2 is used to reach the destination node D, node i_1 and i_2 could both be the predecessor nodes of node j_2. Therefore all travel times should also be calculated via all possible nodes.

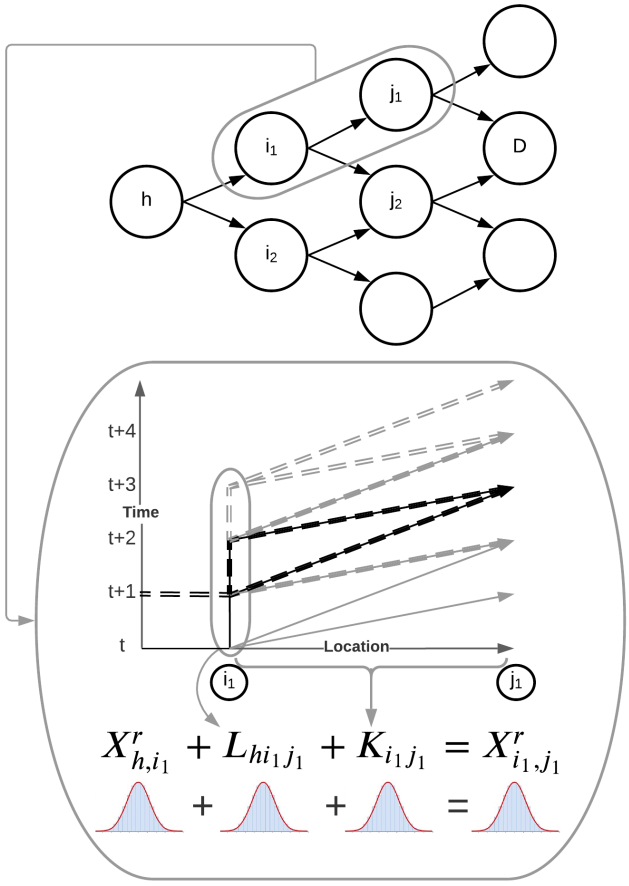

Figure 3.5: An example network with the nodes i_1 and j_1 enlarged, including the link between these nodes. In the enlargement the solid lines give possible waiting and link travel times when a vehicle arrives at i_1 at t and the dashed lines give the possible waiting and link travel times if a vehicle arrives at $t+1$.

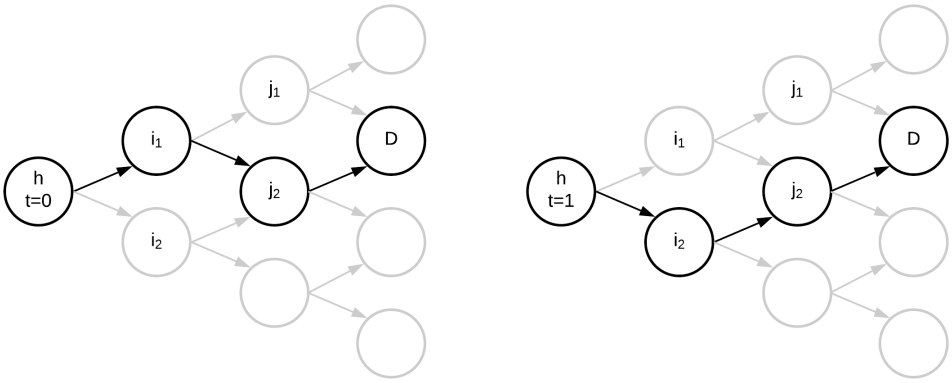

Figure 3.6: An example network that shows that if a vehicle is at node h at time $t = 0$ a different route can be part of the routeplan compared to when the vehicle is there at $t = 1$. Since the travel time from node h to the successor nodes can vary, vehicles can arrive at node j_2 at the same time from different nodes, because of a different link used before, as part of the routeplan.

This leads to three main elements of the new formula to calculate the travel time distribution at a node. First of all the waiting times and link travel times should be combined. Secondly the arrival time at the upstream node should be taken into account. And lastly all possible upstream nodes should be taken into account.

The new formula

Now that the concept of the algorithm is clear, the formula that is used will be explained. The formula can be used to calculate the probability on a certain travel time λ at a the current node j, $Pr(X^r_{i,j} = \lambda)$. Since there are multiple possible travel times, $Pr(X^r_{i,j} = \lambda)$ should be calculated for multiple values of λ. How to select all possible values for λ will be discussed after the formula is explained. To calculate this probability the three main elements described before should be taken into account. Firstly the waiting time and link travel time. The probability on a combination of the waiting time l and link travel time k can be calculated by $Pr(L_{hij}(t) = l) \cdot Pr(K_{ij}(t) = k)$. Secondly the probability of being at the upstream node i at time t should be taken into account, $Pr(X^r_{h,i} = t)$. Now the probability on a travel time λ can be calculated by multiplying the probability of being at the upstream node at time t, the probability on waiting time l and the probability on link travel time k. Since the probability on travel time λ is sought, t, l and k should add up to λ. Therefore the probability on the link travel time is not calculated as $Pr(K_{ij}(t) = k)$, but as $Pr(K_{ij}(t) = \lambda - t - l)$, since $\lambda - t - l$ should be equal to k.

The third and last part are all the predecessor nodes that need to be taken into account. The probability on a travel time to node i is given by $Pr(X^r_{h,i})$, thus the predecessor node h should be taken into account as well. Therefore this should also be added up over all possible nodes h, that are predecessor nodes of node i and from where node j is reachable (this might not be te case when there are for instance restricted turns). This is given in H_{ij}, which is the set of all nodes h which can be used to travel to node i and from node i to node j as well. Now the nodes h are known, the possible arrival times at node i can be known. Since these times are also used in equation 3.13, this formula can be used to identify the possible arrival times at node i. h, i and j are known and therefore the equation can be checked for all possible arrival times that satisfy the equation. With this method the set T^r_{hi} is created that contains all possible arrival times at node i, while coming from node h and following routeplan r.

The calculation of the probability should therefore be added up over three things. The first is the summation over all possible nodes h, that can be used to travel to node i and then to node j, which is given in H_{ij}. Secondly it should be added up over all possible arrival times at node i, while coming from node h, which are given in T^r_{hi}. The last part is adding up over all possible waiting times l at node i at time t while coming from node h and going to node j, which is the set $L_{hij}(t)$. The formula does not need to be added up over all possible link travel times k since they can be calculated as $k = \lambda - t - l$. Once this is all calculated the probability on a travel time of λ to node j, while arriving from node i is known, $Pr(X^r_{i,j} = \lambda)$.

This is shown mathematically in the following equation:

$$Pr(X^r_{i,j} = \lambda) = \sum_{h \in H_{ij}} \sum_{t \in T^r_{hi}} \sum_{l \in L_{hij}(t)} Pr(X^r_{h,i} = t) \cdot Pr(L_{hij}(t) = l) \cdot Pr(K_{ij}(t) = \lambda - t - l) \tag{3.14}$$

If this equation is repeated for all λ, the travel time distribution is known. This should then be repeated for the entire network and for every routeplan in this network.

The starting point of the algorithm is known and the probability on a travel time of 0 to the departure node is 1, $Pr(X^r_{f,h} = 0) = 1$, for all routeplans r assuming that h is the departure node and f the predecessor node of h. From this point on the next node to calculate can be found with Equation 3.13. Variables h and i from the departure node are known and the first t is 0, thus the next j is known. This can be done for $\lambda \in \mathbb{Z}$, with $\mathbb{Z}$ being all integer numbers, since the probabilities will be zero for timesteps that do not occur in the calculation. But a minimum and maximum value is preferred which could be $\min_{t \in T^r_{hi}} + \min_{l \in L_{hij}(t)} + \min_{k \in K_{ij}(t)}$ and $\max_{t \in T^r_{hi}} + \max_{l \in L_{hij}(t)} + \max_{k \in K_{ij}(t)}$ respectively. This is the minimum/maximum travel time from the departure node to node i, while coming from node h, plus the minimum/maximum waiting time at node i, plus the minimum/maximum link travel time to node j. Since there might be different nodes h, the minimum of all those nodes should be picked, just as the maximum of all those nodes for the maximum. Once the probability, $Pr(X^r_{i,j} = \lambda)$, is calculated for all those different values of λ, the next node should be picked to be calculated. This can be done again with Equation 3.13, since the node refered to as i will become node h and j will become node i. Then j can be calculated again, but since there might be multiple arrival times t at that node, there might be different nodes j. Therefore the first node j is selected to be calculated and all other nodes j that need to be calculated are added to the search list. Once this calculation is done, the next node j will be used from the search list to calculate the probability distribution and so on. Once all nodes j have been calculated, Formula 3.13 can be used to calculate the next nodes j again, and this process can be repeated until the destination node is reached.

The result will be a probability distribution for every routeplan. When al these distributions are known the objective value can be calculated by using the distributions as input for the objective function. This result in one optimal routeplan which should be taken. This routeplan includes the scenarios for specific arrival times at the next traffic light and which link to take there dependent on the arrival time.

3.2. Traffic light state prediction algorithm

In the previous section the routing algorithms are explained and a traffic light state prediction method is required for these algorithms. This traffic light prediction method should result in $Z_j^{hi}(t)$ for the ARSCR algorithm and in $Pr(L_{hij}(t))$ for the FPP algorithm. $Z_j^{hi}(t)$ gives the probability of a green light at node i, coming from node h and going to node j at time t. $L_{hij}(t)$ is the set of all possible waiting times when arriving at the traffic light at node i at time t and going to node j while coming from node h. The probability on a waiting time is given in $Pr(L_{hij}(t) = l)$. For the traffic light state prediction used by the ARSCR algorithm, the probability on every possible state of the traffic light in the future is calculated and afterwards the states that result in the requested direction having green are added up to know the probability of having green for that direction. For the FPP algorithm, the traffic light state prediction gives the probability on a waiting time, therefore the FPP algorithm should known the predicted traffic light states in the future as well and then for every state it could be in, at a certain time, the possible waiting times are calculated. Therefore the method for the ARSCR algorithm is explained first since almost everything of that part is used by the FPP algorithm as well. Since the complexity is discussed for each algorithm as well, this is discussed first before the adjustments for the FPP algorithm are explained.

Main part for both algorithms

Literature on traffic light state prediction was reviewed and discussed in Section 2.2. The method that was looked for could predict traffic lights with a changing cycle time for a few minutes ahead, results in a probability distribution, has a reasonable complexity and uses only historic or SPaT data. The methods by Bodenheimer et al. (2014) and Dabiri and Hegyi (2018) are the most useful methods since they are applicable to traffic lights with a changing cycle time, result in a probability distribution, use available data and are not to complex. These methods are based on creating a set of traffic light states and calculating the probabilities of all transitions between these states (depending on how long they are already in that state). Both researches only used their method to predict shortly ahead (e.g. 30 seconds) for GLOSA and not further ahead (e.g. 10 minutes) for routing purposes. These methods both used historic data and Bodenheimer et al. (2015) found that they should use historic data separated per time period and from not too long ago (about four periods, e.g. weekend mornings). The method that is explained in this section is based on these researches.

It is key for both methods to know the probability of transitions between different states of the traffic light. Therefore the state of the traffic light needs to be defined first. Dabiri and Hegyi (2018) defined a state for their method as

$$S = (B_b, n_1^b, n_2^b), \tag{3.15}$$

in which B_b is the current block, the directions that have green at that moment, with $b \in 1, ..., N$ being the current block selected from N possible blocks. In this case a block is a unique combination of directions that are green at the same time. While in some other literature blocks are directions that *can* be green at the same time, the directions *have* to be green at the same time for this definition. Therefore if a minor directions can get green, since some other direction does not have any traffic, a different block will be active since that minor direction is also included in the block while the direction without traffic (and thus a red light) is excluded. n_1^b is the number of seconds the first direction from that block already is green and n_2^b is the number of seconds the second direction from that block already is green. These values are selected from 1 to the maximum number of seconds that direction can be green, resulting in $n_1^b \in (0, n_{1,max}^b)$ and $n_2^b \in (0, n_{2,max}^b)$.

In the paper of Dabiri and Hegyi (2018) there are only two directions that can have green at the same time. But on larger intersections with more dedicated turning lanes that number is usually larger. Therefore the state should contain more values to facilitate a timer for all the directions that can be green at the same time (given by m).

$$S = (B_b, n_1^b, n_2^b, ..., n_m^b) \tag{3.16}$$

Now that the state is defined, the probability of transitioning into a next state should be determined. To do this Bodenheimer et al. (2014) creates a controller graph which is then reconstructed into a transition graph. This graph shows all the possible changes in states of the traffic light. Historic data is then used to add the probabilities to each transition. The time since the transition is also tracked to calculate the probabilities, since the time the signal is in a certain state influences the probability of changing into the next state. But since it only tracks the time it is in this exact state previous changes in state are neglected. Normally a traffic light does not consider previous changes but with actuated traffic lights a minor direction that has no more traffic could turn red and a different direction might get an extra time green in this cycle. As long as the main direction keeps having traffic and the maximum green time has not been reached the traffic light will stay in this new state. Therefore it is not only relevant exactly how long the traffic light is already in this state, but also how long all the separate directions already have green. The method

from Dabiri and Hegyi (2018) does take this into account by adding a timer for every direction that is green. This is basically the same method as Bodenheimer et al. (2014) uses, but with more states since the states are dependent on the timers of all directions that are green and not just one timer. Since the method of Dabiri and Hegyi (2018) keeps timers for every direction, this method is preferred.

To calculate the future state of the traffic light, some defintions need to be introduced. $\mathscr{S}$ is the set of all possible states for this traffic light, with this set, states that has a longer green time than the maximum green time or a combination of directions that have green that cannot happen, can be excluded from calculation. $\mathbb{P}$ returns the probability of transitioning from the state S_t toward S_{t+1}. Besides these new notations, the definition of a state should be made time step dependent. Therefore Equation 3.17 is introduced, that makes the state dependent on time step t.

$$S_t = \left(B_b(t), n_1^b(t), n_2^b(t), ..., n_m^b(t)\right) \tag{3.17}$$

Now the probability on a future state, given the current state, $(P_{ss'})$ can be calculated by the next equation:

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s], \forall s, s' \in \mathscr{S} \tag{3.18}$$

This calculates the probability on a future state s', given the current state s. To do this, the probability is calculated on the future state s' being the state a time step later, S_{t+1}, when the current state s is S_t. This is done for all states s given that the next state exists ($s^{'} \in \mathscr{S}$). With this equation the probability on every future state is calculated and it is independent on the past states of the traffic light. Therefore it displays the Markov property of the process. The historic data can be gathered from historic SPaT data while selecting on the current state, next state, the timers for how long the traffic light is green and the active time period (e.g. Friday afternoon peak).

Now that the probability of transition between states is known. The probability on all states for all time steps can be calculated. Since different states can transition into the same next state, the notation for a state is adjusted into s_i, with $i \in [1,2,...,I]$, instead of writing down the complete state with the block and timers for every state. I is dependent on the amount of blocks and amount of different combination the timers can have in each block. Therefore this is different per traffic light. For future state the notation is changed into s'_j with $j \in [1,2,...,I]$.

The probability on different states is given in the matrix $Y(t, s_i)$. In this matrix the probability on the state s_i at time t is given. For the first time step there is only one value, since the traffic light is 100% certain to be in just one state. Therefore Y will look like this: $Y(0, s_i) = [0,0,1,0,0,...]$ assuming that the traffic light is in the state s_3 now.

For every next time step the probabilities can be calculated by multiplying the probabilities of the previous time step with the probability of transitioning from that state to a future state $(P_{s_i s'_j})$.

$$Y(t+1, s'_j) = \sum_{i=1}^{I} Y(t, s_i) \cdot P_{s_i s'_j} \tag{3.19}$$

Specific part for the ARSC(R) algorithm

This can be repeated to know the probabilities for all states in the future. But for the ARSC(R) algorithm, the probability of being green for a certain signalgroup should be known $(Z^{SG}(t))$. Therefore we need to find all the states that give green to a certain signalgroup and add up all the probabilities. This is done by checking for every state if the signalgroup is green, and if so, adding the probability of that state to the probability of being green. This is shown in Equation 3.20. It states that the probability only should be added up if $SG \in s_i$, this makes sure that the signalgroup is in the block of the current state and therefore the correct direction is green.

$$Z^{SG}(t) = \sum_{i=1}^{I} Y(t, s_i) | SG \in s_i \tag{3.20}$$

This results in a probability of being green for signalgroup SG at time t $(Z^{SG}(t))$. In the ARSCR algorithm $Z_j^{hi}(t)$ is requested, but since the signalgroup (SG) uniquely matches the corresponding nodes (h, i, j), this is can be replaced. Then $Z_j^{hi}(t)$ is finally known and can be used in the routing algorithm.

Complexity

These calculations need to be repeated for every state of the traffic light separately for the time steps in the prediction horizon. The last calculation, to calculate the probability for a direction begin green, needs to be done for all direction within the states separately. But once this is done for all direction this could be saved into matrices and the results can be retrieved right away. This is because there are a limited set of possible states of the traffic light

and the probabilities of changing to a new state, do not change since they are based on historic data. Therefore this could be calculated for all states, so that the routing algorithm can run more quickly since it does not have to do these calculations every time again. But in addition to time complexity in algorithms, memory requirement is also relevant. Therefore creating these large matrices might not be the most efficient way after all. This is dependent on the number of states the traffic light has, which is different for every traffic light since it is dependent on the type of traffic light, total cycle time, alternative realisations and other flexibility improving measures. Therefore if these calculations should be done real time or before running the algorithm should be determined for every situation separately.

There are three points in the algorithm described before, where a matrix could be made from the data processed up until that point. The first point is when the historic data is processed and $P_{s_i s'_j}$ is known, but nothing has been done with the processed historic data. This matrix is 2 dimensional with a size of the number of states squared (I^2). This is a relatively small matrix but there are still some computations left before the data can be used in the algorithm. The second option is after $Y(t, s_i)$ is calculated. This gives the probability of a certain state (s_i) at a certain time (t) given a certain start state at $t = 0$. Therefore the dimensions of this matrix are larger and three dimensional. For every state at $t = 0$, the matrix $Y(t, s_i)$ should be stored, which is a matrix with the size of the number of states (I) times the prediction horizon (T), $I \cdot T$. Since this has to be stored for every possible (starting) state, multiple matrices need to be stored which results in the following space complexity: $I \cdot I \cdot T$, which is the number of states squared, times the number of time steps. This matrix is therefore times T larger, which is a large difference if the prediction time is long or the time steps are small. A third option is to save it to the matrix, after $Z^{SG}(t)$ is calculated. Which is the probability of a certain signalgroup being green at time t. Since this is dependent on the state of the traffic light at $t = 0$, the size of this matrix is three dimensional as well, the number of states times the total number of signalgroups (SG^{max}) times the prediction horizon, $I \cdot SG^{max} \cdot T$. But the number of signalgroups is always smaller than the number of states since the state can change every time step. Therefore this third point in the algorithm is more efficient compared to the second point, since more calculations are done and there is a smaller table with data. But the first and third point are hard to compare. The third point is more time efficient, but the space complexity depends on the number of signalgroups, time steps and states. The first point had a complexity of I^2 and the third point a complexity of $I \cdot SG^{max} \cdot T$. This leads to the statement that if $SG^{max} \cdot T$ is smaller than I, the third option is better. This is true, but the other way around it does not necessary hold since the space complexity might be improved, but the time complexity will be decreased. Therefore where the data should be stored, should be assessed per situation under the local circumstances.

Specific part for the FPP algorithm

Besides this traffic light state prediction algorithm for the ARSCR algorithm, a slightly different one is required for the FPP algorithm. This is because the FPP algorithm calculates the probability on a waiting time at a traffic light dependent on the state when a vehicle arrives at the traffic light. This is will lead to $Pr(L_{hij}(t) = l)$ which is the probability of having red for $l-1$ time steps and green for the l^{th} time step for the signalgroup coming from node h and going to node j. t stands for the time step the vehicle arrives at the node (i) and l gives the time steps it has to wait before having a green light. This results in a distribution of the number of time steps l with the corresponding probability. The beginning of the algorithm is the same as described before, since if a vehicle arrives at a node at time t, all possible states of the traffic light at that moment and corresponding probability should still be known. This results in $Y(t, s_i)$ which are the probabilities on all states of the traffic lights.

The step afterwards is different since the time until green is predicted and not the total probability on a green traffic light. Therefore the probability on states in which the signalgroup is green are added up to $Pr(L_{hij}(t) = l)$, with the corresponding waiting time l. For the first timestep (when arriving at the traffic light) l will be zero since the traffic light might already be green. But if a state of the traffic light is not green, it will be checked again one timestep later if it is green. Since oversaturated conditions are not taken into account, the probability of having green a second time is not taken into consideration and if a traffic light state gives the direction green, this probability should not be taken into account in the next calculations since we are not looking for the when it is getting green again. This is stored in $X(t, s_i)$, which is set equal to $Y(t, s_i)$ for the arrival time at the traffic light (t) but is different for every step thereafter. i in s_i is an index since there are multiple possible states. j is an index for s'_j which are all possible future states. The next steps are not taken by increasing the time step t but the waiting time l. But this should only be calculated if the current state is not green, since otherwise the vehicle already drove away and is no longer waiting. Therefore $SG \notin s_i$ is added, so that the probability on future states are only calculated from directions that are not green for SG. This leads to Equation 3.21

$$X(t+l+1, s'_j) = \sum_{i=1}^{I} \left(X(t+l, s_i) \cdot P_{s_i s'_j} \right) \Big| SG \notin s_i. \tag{3.21}$$

This should be calculated for all next possible states j and when this is done for the next possible waiting time l, until all states have been green. Now that the probability on all future states are known, while only calculating these if the traffic light is not green, the probability on a waiting time can be calculated. For this, Equation 3.22 is used, which adds up all the probabilities on the states that give the signalgroup green. This is almost the same as for the ARSCR algorithm, only in this equation the time is not dependent on t but on the waiting time l. This is calculated for all waiting times, and $\sum_{l \in L_{hij}(t)} Pr(L_{hij}(t) = l) = 1$ since the probability of having a waiting time between 0 and the maximum waiting time should always be 1.

$$Pr(L_{hij}(t) = l) = \sum_{i=1}^{I} X(t+l, s_i) \; \Big| SG \in s_i \tag{3.22}$$

This has to be done for every time step a vehicle can arrive at a traffic light separately which increases computational complexity compared to the ARSCR part of the traffic light prediction module. The space complexity is not that large since only data is stored for every time step, possible waiting time and signalgroup. Therefore the space complexity is $T \cdot L \cdot SG_{max}$. This is a lower space complexity compared to the last part of the ARSCR algorithm, if the maximum number of waiting times is smaller than the number of states, $L < T$. Since this is further in the calculations as well, calculating up to this point is preferred. But this part could run real time as well or just once before the algorithm is ran, as explained before in the previous subsection. Since calculation complexity is also relevant, ending the calculations of this algorithm at this point or at a different point should be chosen depending on the situation and network.

3.3. Link travel time prediction algorithm

Another input for the routing algorithms is the predicted link travel time. This consists of two parts, the estimated link travel time (distribution), $\tau_{ij}^k(t)$, and the probability of that link travel time occurring, $p_{ij}^k(t)$. In the previous section, literature research on this topic has been done. In urban areas the link travel time is mainly dependent on the queueing model, the queue discharge model and the platoon dispersion model. Other delays like road works or incidents are not taken into account in this research.

The ARSCR algorithm proposed in section 3.1.3 do not use a queueing model and only supports travel time estimations via the link travel time distribution, $\tau_{ij}^k(t)$. But since this distribution is only dependent on the time the link is entered, queueing effects cannot easily be incorporated. This is because once a vehicle enters the queue, it's position in the queue or how long the vehicle is already in the queue is not longer tracked. Therefore no queueing or queue discharge model can be used. Some acceleration loss might be incorporated in the link travel time distribution, but vehicles who can pass the traffic light without stopping might not encounter those delays. Another part of the link travel time prediction is the platoon dispersion model. For this model it is relevant in what position in the queue the vehicle is (or at least in what position it enters the next link) and can therefore also not be implemented. This is a part in the routing algorithm that could be improved since adding those models might improve performance of the link travel time prediction. But adding these might also decrease the speed of the routing algorithm since the complexity will increase.

Waiting time is incorporated in the FPP algorithm but the position in the queue is not known. Only the number of time steps the vehicle is in front of a red light is known. This information could be used to improve the link travel time prediction, since if a vehicle did not have to wait at all, it probably does not have to stop and therefore has a lower acceleration loss and thus queue discharge time. Or if for instance a vehicle had to wait almost the cycle time, it is probably in front of the queue, since the traffic light was green just before arriving at the stop line. But when entering the queue somewhere in between it is hard to predict the position in the queue and therefore a link travel time might not improve. This is because in an urban network, arrivals at a traffic light are usually not uniformly distributed but somewhat dependent on intersections and traffic lights upstream. And when a platoon arrives it is hard to determine if the vehicle is in the front or the back of the platoon, while this has influence over the next link travel time. Therefore the waiting time at the traffic light is not used to influence the link travel time prediction model. Besides that, the added part of the link travel time prediction model should be evaluated separately to make sure it works properly and to make sure it does not interfere with the results of the routing algorithms. Since the routing algorithms are the main research goal, these models are not implemented in this research and alternative link travel time prediction methods are used.

On some roads average and actual link travel times are known via loop detectors in the road and made available by the local authorities. But on many roads this data is not available. Navigation service providers (like TomTom and Google Maps) do have data on average travel time on every link and sometimes even real time data. These link travel times indirectly incorporate the before mentioned three models since it gives the travel time experienced by travellers in those urban areas. If this data is going to be used, a new model should be created to predict the travel time distribution depending on the time a vehicle enters the link and (for instance) the number of vehicles already on the link. Since this research designs a new routing algorithm it can be assumed that it will be implemented by one of those companies who have this data already available. Another method of gaining these data is by connected vehicles. If these vehicles sent data to the traffic lights, queue and link travel time estimations could be made using this data. Therefore in the remainder of the research this data is assumed to be available and in the simulation this data is made available as well.

3.4. Summary

In this chapter two new routing algorithms including the traffic light prediction model were designed and the link travel time prediction model was further explained.

3.1.1 The ARSC algorithm from B. Yang and Miller-Hooks (2004) is a backwards label correcting algorithm that saves intermediate steps to labels. These are calculated from the destination node towards the departure node and from the latest expected travel time until the current time. The label in the original algorithm saves the lowest expected travel time and can therefore not be used in combination with reliability. But it does save an optimal route for the lowest expected travel time. This route is dependent on the arrival time at intermediate nodes since the traffic lights can be in different states when arriving at a different moment and therefore the optimal next link might be different. For all algorithms described from here on this concept holds, and a route in which the next link to take is dependent on the arrival time at intermediate nodes, is from here on called a routeplan.

3.1.2 To use reliability in the new routing algorithm, the label as described in the ARSC algorithm is adjusted into an distribution that can be used to calculate an objective function. This objective function can be entered by the user and can be complex, but to work with an adaptation of the ARSC algorithm it has to satisfy the following condition: The optimal routeplan from a node to the destination node, at a specific time, should be independent of data that will be added at every upstream node. If the objective function does not satisfy that condition the new FPP algorithm can be used that can have all functions as objective function.

3.1.3 The ARSCR algorithm uses the method of the original ARSC algorithm but saves a travel time distribution at the labels instead of the expected travel time. This requires some adaptations since the travel time distribution have to be passed along. Therefore $Pr\left(\Lambda_i^h(t) = \lambda\right)$, is introduced that saves the probability on a travel time λ. At every label the optimal next node to travel to is determined by using this travel time distribution and the objective function. The lowest value of the objective function of all possible routeplans to travel to, is the optimal one. The benefit of this algorithm is that at every node the optimal routeplan to the destination can be calculated and this algorithm is therefore more suitable for larger networks compared to the FPP algorithm. If delay is encountered when the vehicle is driving along the routeplan, the algorithm already knows the optimal routeplan as well since it is saved per label for every time step. Recalculating is therefore not required (but it might improve the routeplan since prediction less far ahead is more accurate).

3.1.4 The FPP algorithm works forwards, and saves the entire distribution for every single routeplan before calculating the objective function. This requires more calculation power, but all objective functions can be used. This algorithm can get slow for larger networks and therefore a separate algorithm can be used to select the important links for which the distribution should be calculated. Since this algorithm works forwards the waiting time at the traffic light can be incorporated in this algorithm and therefore vehicles cannot wait longer than the cycle time. In the ARSCR algorithm a vehicle could wait infinitely long with ever decreasing probabilities. Therefore the FPP algorithm deals more precisely with the traffic light predictions.

3.2 The traffic light model is explained from the historic data until the required input for the new routing algorithms. From the historic data the probabilities of changing from a state to a specific next state are calculated while keeping track of the time every direction is already green for the best predictions. This is done once, but the following steps have to be recalculated for every new state. To get the probability of having green for a specific direction the probabilities on every state of the traffic light are calculated per node and per time

step. Afterwards all the probabilities of different states in which the requested direction is green, are added up for the required input for the ARSCR algorithm. For the FPP algorithm the last step also predicts how long a the vehicle have to wait, after the vehicle is predicted to arrive at the traffic light, and creates a distribution of when the traffic light becomes green (the waiting time), dependent on the arrival time at that traffic light.

3.3 To facilitate the use of queueing models, queue discharge models or platoon dispersion models the position in the queue should be known. Since the ARSCR algorithm does not keep track of this, these models cannot easily be implemented without adjusting the algorithm. For the FPP algorithm the waiting time at the traffic light is taken into account but the position in the queue is not known as well. Besides this waiting time, which is the time the traffic light is red, the acceleration loss is not taken into account. This could improve the link travel time prediction, but negative consequences for the routing algorithm might arise and since the routing algorithm is the main research goal the algorithm is not adjusted. Therefore a link travel time distribution per link will be used, which should be made available to the routing algorithm.

4

Evaluation of the algorithm

In this section the evaluation of the routing algorithms will be discussed. In Section 4.1 the algorithm will be verified with a simulation to show that the travel time distribution from the FPP algorithm will be realized when simulated. This proves that the algorithm works and the calculated travel time distribution represents the possible travel times. In section 4.2 the improvement in objective value, compared with a standard algorithm, is shown and comparisons between different objective functions are shown as well. The algorithm is not compared in travel time, since reliability cannot be measured in travel time alone. Therefore the objective functions are used, and an improvement in objective value is therefore better. This shows that the algorithm minimizes the objective value of other objective functions and improves the expected travel time (if you use that as objective function). The complexity of the algorithm with possible solutions and their downsides is discussed as well.

The ARSCR algorithm will not be evaluated, just as the traffic light prediction module. Since the ARSCR and traffic light prediction module are both similar to the original algorithms it is based on, it is more useful to evaluate the completely new FPP algorithm.

4.1. Proof of concept

This proof of concept will show that the algorithm is working by verifying it with a simulation. Therefore the FPP algorithm will be run on a network, with signalized intersections, which will result in the travel time distribution of the optimal routeplan. To check if this distribution is a good representation of the actual travel times a vehicle could encounter, the network is also simulated, with a vehicle following the routeplan suggested by the FPP algorithm. This will result in a travel time for this vehicle, and if this is simulated multiple times, this will result in a travel time distribution over all simulation runs. The travel time distribution of the simulated vehicles should match the travel time distribution calculated by the FPP algorithm for the algorithm to work. Because if the simulated travel time distribution is the same as the FPP travel time distribution, the algorithm calculates the travel times distribution correctly and the objective function can be used to determine the optimal travel time distribution, and thus the optimal routeplan.

In this section the network, Section 4.1.1, the link travel time, Section 4.1.2, and the traffic light prediction module, Section 4.1.3, will be explained first. Then the calculation of the optimal routeplan and travel time distribution by the FPP algorithm will be discussed, Section 4.1.4. And finally the simulation and the results will be discussed, Section 4.1.5 and Section 4.1.6.

4.1.1. The network

The network that is used in this simulation is a simple network with traffic lights at node A and B and two links between the nodes, 1 and 2, and 3 and 4. This network is shown in Figure 4.1. The vehicle starts at the stop line before the traffic light at node A. It starts at link 0, which results in going to link 1 is taking a left turn, and going to link 2 is going straight on. Since these directions have different waiting time distributions at the traffic light, it is relevant that the vehicle is coming from this direction. The exact waiting time distributions will be explained in the next subsection. From there it goes to link 1 or 2 and then via node B to link 3 or 4. Looping back is not allowed in this network. The destination is at node C and there is no traffic light or any delay present at the destination node. Therefore it does not matter if a vehicle arrives from link 3 or 4, only the link travel time of that link is counted to get to the destination.

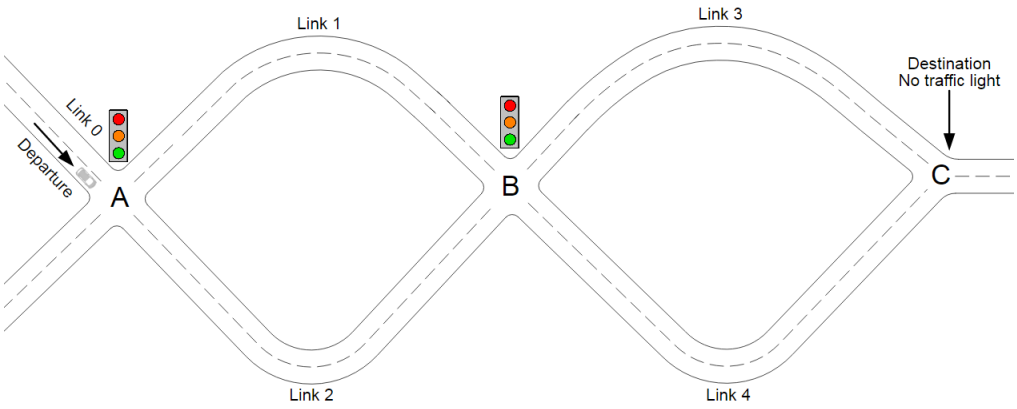

Figure 4.1: Network for the evaluation of the algorithm. Note that all roads have 1 lane in this picture for clarity, but there are three lanes per approach at the intersections, one for each direction.

4.1.2. The link travel time

The travel time on the links is given in distributions that are based on Gamma distributions. This is a fitting distribution for local traffic with undersaturated conditions, since most vehicles will have a travel time slightly longer than the free flow travel time while almost none will have a shorter travel time (Nie, Wu, Dillenburg, & Nelson, 2012). For saturated conditions a different link travel time distribution would be more fitting, but in this research undersaturated conditions were assumed. A minimum travel time on a link is also present, since a vehicle will always take at least a certain amount of time to cross a link. The exact distributions are given in Figure 4.2. This shows that a vehicle drives at least 10 seconds on the link and a maximum of 16 seconds. The travel time distribution is the same for every link in this evaluation. The algorithm works with a different distribution per link and can even have different distributions per arrival time, but this is not used in this evaluation.

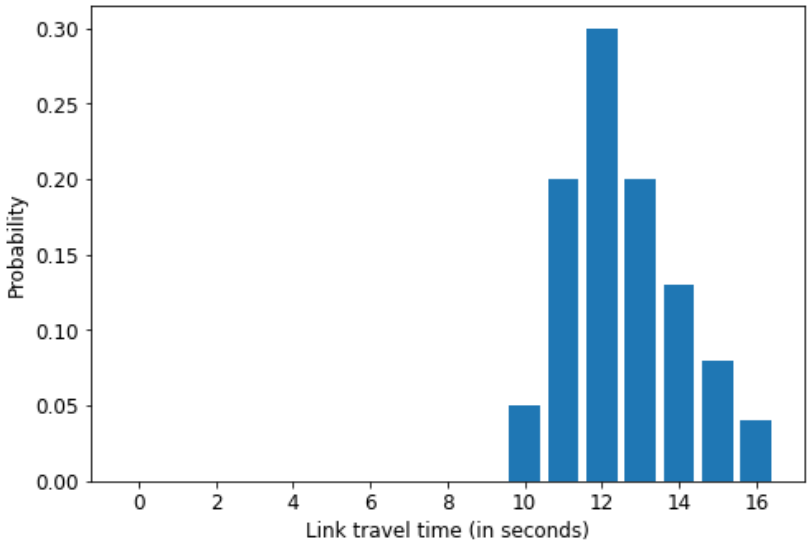

Figure 4.2: The link travel time distribution used in the evaluation.

4.1.3. The traffic light prediction module

The waiting times at the traffic light are calculated based on green time distributions given per phase. In this evaluation both intersections of the network (Figure 4.1) have four roads coming together and each road will have three possible signalgroups, one for each direction. There are four phases in which non-conflicting signalgroups can receive green, as shown in Figure 4.3, which are given green in the numerical order (1-2-3-4-1). The signalgroups are numbered clockwise starting from the direction east. In phase 1 and 3 vehicles that drive straight on or make a right turn can drive, while in phase 2 and 4 the left turns are facilitated. Since the right turns can get green on those left turns as well, these are having green in two phases while the other directions only get green in one phase. The

maximum green time of the phases differs as well with 5 seconds for phases 2 and 4 and 8 seconds for phases 1 and 3. This is a very short maximum green time for a traffic light, but this is done to keep the calculational complexity low. The calculation complexity and the effects on the performance are discussed in Section 4.2.3. These values can be used for the proof of concept since the concept does not change.

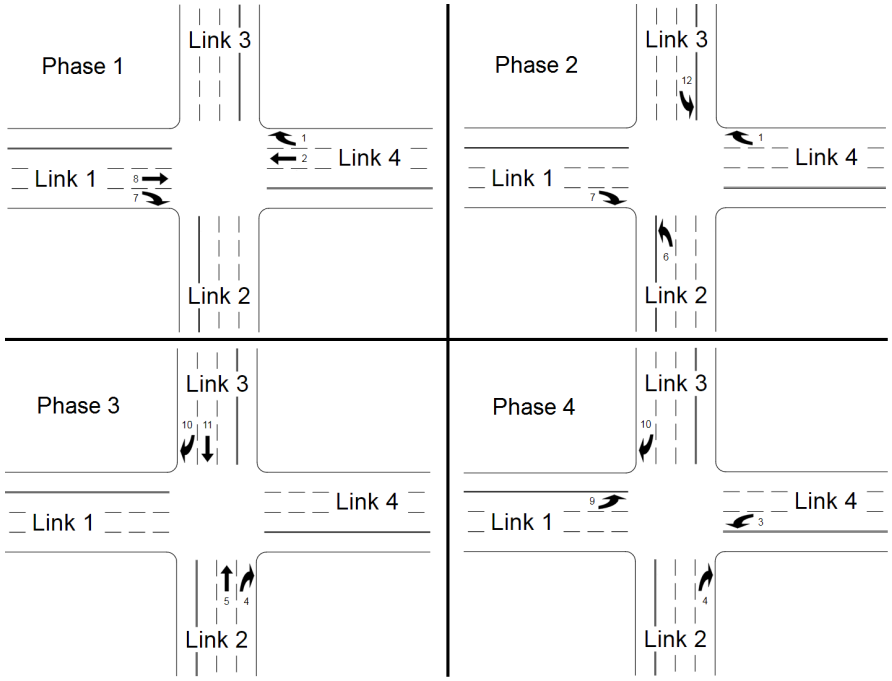

Figure 4.3: The four phases of the signal control. The phases and directions are shown for traffic light B, but are the same for both traffic lights.

The complete green time distribution is shown in Table 4.1. This shows the probability on a green time when a phase starts, thus if phase 1 just started there is a 10% chance on a green time of 3 seconds. For both traffic lights the signalgroups and the timing of the traffic light is the same. If a green time of 0 seconds is realized, the phase is skipped since no traffic is at those signal groups at that moment. With these probabilities on green times, a prediction for the waiting time at the traffic light is made with the traffic light prediction module. How this is done is described in Section 3.2. The starting state (starting phase and time in this phase) of a traffic light will stay the same for this part of the evaluation. This is necessary since a different starting state of the traffic light will result in different travel times to the destination and therefore a different travel time distribution. In Section 4.2 the algorithm is evaluated with changing traffic lights states. The starting state of the first traffic light is 2 seconds in phase 4 and for the second traffic light 5 seconds in phase 2.

Table 4.1: The probability on a green times of a phase when that phase starts, thus the probability on a green time of 3 seconds is 10% for phase 1 and 3.

Green time (in seconds)	0	1	2	3	4	5	6	7	8
Probability phase 1 & 3	0.1	0.04	0.07	0.1	0.15	0.22	0.15	0.1	0.07
Probability phase 2 & 4	0.1	0.125	0.2	0.25	0.2	0.125	0	0	0

4.1.4. The travel time distribution calculated by the FPP algorithm

Now that the network, link travel time distribution and waiting time distribution are known, the FPP algorithm can be used to calculate the travel time distribution of all possible routeplans. This will result in a lot of travel time distributions. To compare one of these distributions with a simulation, one distribution should be selected. This could be done at random, but since finding an optimal routeplan is the main goal of this research, an optimal routeplan is used. This is done with the mean plus the standard deviation of the travel time distribution as objective function and this takes the travel time and reliability into account. This function is explained in the next section as

Equation 4.5, since it is used there as well. For this part it is only relevant which routeplan is taken, which is to take link 1, and when arriving at the second node between 10 and 16 seconds, to take link 4, and when arriving between 17 and 20 seconds, to take link 3. This is shown in Table 4.2. For arrivals before 10 and after 20 seconds no next link is given, since these travel times are not possible given the input distributions. The switch after 16 seconds makes sense, since the traffic light at node B has a high probability of being in phase 1 when arriving at the traffic light before 16 seconds. After phase one is over, the next green light will be in phase 4, which contains the left turn to take link 3. Therefore the optimal routeplan switches to link 3 when arriving after 16 seconds.

Table 4.2: The routeplan that is being evaluated.

Travel time to node B (in seconds)	10	11	12	13	14	15	16	17	18	19	20
Chosen next link	4	4	4	4	4	4	4	3	3	3	3

The travel time distribution, as predicted by the calculation of the FPP algorithm, is shown in Figure 4.4. This shows a lower probability of arriving at 31 seconds, which can be explained at the higher probability of waiting for a red traffic light at the second intersection when node B is reached at 17 seconds or later. If the link travel time of the link after node B (in this case link 3) is added as well, vehicles will have a lower probability of arriving after (17 + 10 =) 27 seconds to (17 + 20 =) 37 seconds. After some time the probability of the traffic light being green increases again and the probability of arrival times around 37 seconds is slightly higher.

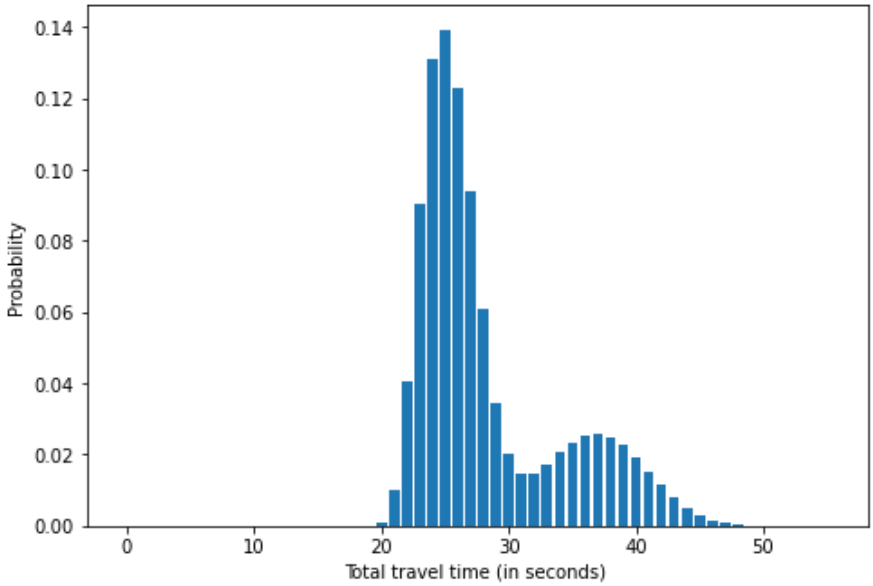

Figure 4.4: The probability on the total travel times to the destination.

4.1.5. The simulation

To check if this travel time distribution is correct, a vehicle is simulated trough this network with the above mentioned link travel time distribution and green time distribution and starting states for the traffic lights. With these distributions as input, one value for the link travel time on the first link can be picked, and one for the second, while taking the corresponding probabilities into account. The routeplan that is followed is described before as well, and is link 1, followed by link 4, when arriving at node B between 10 and 16 seconds, and link 3, when arriving between 17 and 20 seconds. For the traffic lights, the green times will be randomly picked (while taking the probabilities into account) and this will be repeated until the traffic light is simulated long enough for the vehicle to pass. So, for the second traffic light the first state is that the traffic light is in phase 2 for already 5 seconds (this is given, as stated before). Therefore there is a 100% chance of going to a next phase at the next timestep, since the maximum green time was 5 seconds. Then for phase 3 a value is picked, for instance a green time of 5 seconds, and then for phase 4 a green time is picked, for instance 4 seconds, and then for phase 1, 7 seconds, etc. If a link travel time of 14 seconds is realized, the vehicle arrives at the second node after 14 seconds because it will take link 1 and the first traffic light

is green for that direction. After 14 seconds the second traffic light is in phase 1, since the traffic light is in phase 1 from (1+5+4+1=)11 seconds until (1+5+4+7=)17 seconds. Since link 4 is the next link when arriving after 14 seconds, and this directions gets green in phase 1, the vehicle can drive trough and there is no delay at the traffic light. If there is delay more phases are simulated until the vehicle gets a green light. This is simulated 1000 times to check if the distributions match.

4.1.6. The results

The simulation is run as described above and this resulted in a travel time distribution. This distribution together with the distribution that was calculated by the FPP algorithm is shown in Figure 4.5.

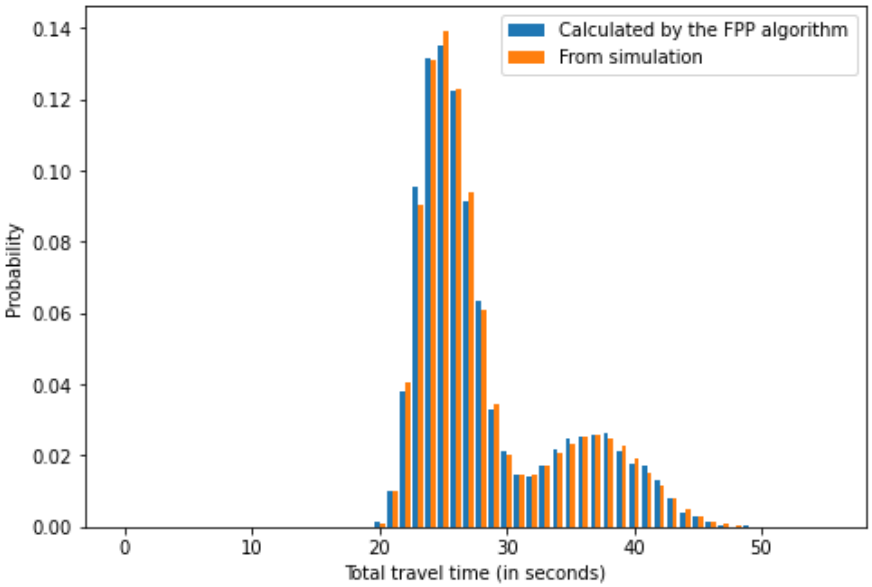

Figure 4.5: The probability on the total travel times to the destination.

This figure shows that the distributions match. There are some small differences but those are inevitable while simulating. Therefore the travel time distribution from the FPP algorithm is the same as from the simulated distribution and the FPP algorithm calculated the correct travel time distribution. This shows that the FPP algorithm works.

4.2. The improvement in objective values

To check the effectiveness of the FPP algorithm, the same scenario as in the previous section will be used, but with four different objective functions which are used to determine the optimal routeplan. Next to these four objective functions for the FPP algorithm, an optimal routeplan is also determined with average waiting times at the traffic light and average link travel times instead of a distribution. This is done to compare the FPP algorithm with standard algorithms that do not take reliability, current traffic light states or link travel times into account. This were most of the algorithms described in Section 2.4. This evaluation will show the change in objective value when using the FPP algorithm compared to the standard algorithm and between the different objective functions used with the FPP algorithm.

Since the previous section shows that the FPP algorithm calculates the correct travel time distribution, there is no need to simulate actual vehicles, since with many simulations the simulated travel time distribution will eventually turn into the distribution calculated by the FPP algorithm. But those distributions do change depending on the starting state (starting phase and number of timesteps in this phase) of the traffic lights, since this will result in different waiting times at the traffic light and might result in a different optimal routeplan. But for every starting state of the traffic light there is just one optimal routeplan (for every objective function) with one travel time distribution. Therefore these distributions can be calculated for every starting state of the traffic light and combined with the probability of that starting state occurring, the mean objective values can be determined. These mean objective values gives the mean of all objective values possible in this specific network. If these values are compared to, for instance the standard algorithm, the mean effect of the algorithm in this network can be known. The change in objective value will give the complete effect on that objective function for this specific network and situation.

4.2.1. Evaluation setup

The network that is used is the same as in Section 4.1, with the same links and nodes, the same link travel times and the same traffic lights. The only difference is that the algorithm will be calculated for all starting traffic light state states and that multiple objective functions are evaluated. Four different objective functions are used and these are explained and shown with the notation given before in Section 3.1.2 of which an overview of the notation can be found in Appendix A.2. The first objective function is the expected travel time (ETT):

$$O_i^h(t) = \sum_{\lambda \in \Lambda_i^h(t)} Pr\left(\Lambda_i^h(t) = \lambda\right) \cdot \lambda. \tag{4.1}$$

This is commonly used in standard algorithms and do not take reliability into account. The second objective function is the 95^{th} percentile travel time, which will give the travel time the vehicle is 95% guaranteed to arrive before:

$$O_i^h(t) = \lambda \mid \{Pr\left(\Lambda_i^h(t) \leq \lambda\right) \geq 0.95\}. \tag{4.2}$$

The objective function will therefore look for the lowest travel time that can be realised with a 95% certainty. There are probably multiple travel times λ that satisfy the above equation, in that case the lowest λ should be picked. The third objective function is the standard deviation (STD), which uses the probabilities on the travel times as weights, and for which a lower deviation is better. This makes sure that travel times with a low probability of occurring have a small impact on the total deviation. The formula for this is given by:

$$O_i^h(t) = \sqrt{\sum_{\lambda \in \Lambda_i^h(t)} Pr\left(\Lambda_i^h(t) = \lambda\right) \cdot \left(\lambda - \overline{\lambda}\right)^2}, \tag{4.3}$$

with the mean noted as $\overline{\lambda}$ and calculated by:

$$\overline{\lambda} = \sum_{\lambda \in \Lambda_i^h(t)} Pr\left(\Lambda_i^h(t) = \lambda\right) \cdot \lambda. \tag{4.4}$$

The last function is the mean plus the standard deviation (mean + STD), in which the mean is used as given in Formula 4.4:

$$O_i^h(t) = \overline{\lambda} + \sqrt{\sum_{\lambda \in \Lambda_i^h(t)} Pr\left(\Lambda_i^h(t) = \lambda\right) \cdot \left(\lambda - \overline{\lambda}\right)^2}. \tag{4.5}$$

This is added since most users would want to keep their travel time low while having a certain (also as low as possible) deviation of the mean travel time. Note that the mean is equal to the expected travel time.

Next to these four objective functions for the FPP algorithm there is also one routeplan calculated without the FPP algorithm for comparison. This is done by using the average waiting time at the traffic light (per direction) and the average link travel time. The current state of the traffic light is not known with this standard algorithm and the preferred routeplan is therefore the same for all starting states of the traffic light. But the actual total travel time distribution, experienced when driving or simulating, is different per state of the traffic light, since the delay at the traffic light is different. Since this is not taken into account in the standard algorithm, the FPP algorithm is expected to result in an improved routeplan. The goal of this standard algorithm is to minimize the expected travel time, as shown in Formula 4.1.

Now that the network and the objective functions are known, the algorithm can be calculated for all possible starting states of the traffic light. This will result in different objective values depending on the objective function and starting state of the traffic light. Since not every starting state of the traffic light has the same probability of occurring, these probabilities should be calculated. Once these are calculated, these can be multiplied with the objective values to create a mean objective value for each objective function for this network.

This can be calculated with the green time distributions given in Table 4.1. This is done in two steps, first the probability on a specific phase should be calculated and secondly the probability of being green for already a specific number of seconds should be calculated. The probability of being in a specific phase can be calculated by calculating how long the green time of every phase on average is. This is the green time, times the probability on that green time, added up for all green times. This leads to an expected green time of 2.7 seconds (=0.1*0+0.125*1+0.2*2+0.25*3+0.2*4+0.125*5) for phase 2 and 4 and 4.34 for phase 1 and 3. Now the probability on being in a certain phase can be calculated since on average phase 2 or 4 is 2.7 seconds green in a cycle of (2.7+2.7+4.34+4.34=)14.08 seconds. Therefore in 19.18% of the cases phase 2 is active, in 19.18% of the cases phase

4 is active, in 30.82% of the cases phase 1 is active and in 30.82% of the cases phase 3 is active. For calculating the probability of being a certain number of seconds green, the probabilities given in Table 4.1 cannot be used directly. This is because if the green time is 2 seconds, the state that the traffic light has been green 1 second did also occur. Therefore the probabilities should be accumulated and corrected accordingly. This is shown in Table 4.3 for phase 2 and 4. In this table the probabilities of green times are given just as in Table 4.1, but the accumulated probabilities are given as well. This is calculated by adding up all probabilities for green times in which the current green time already had green. In the last row these accumulated probabilities are divided by the sum of these accumulated probabilities to get the corrected probabilities. This are the actual probabilities of a certain green time occuring. If these corrected probabilities are multiplied with the probability of the phase occuring, the probability of a state occuring is kown. Thus for the state 3 seconds in phase 2, this is: $0.1918 * 0.1554 = 0.0298$.

Table 4.3: The probability on a state occurring of phase 2 en 4.

Green time (in seconds)	0	1	2	3	4	5	Total
Probability of occurring	0.1	0.125	0.2	0.25	0.2	0.125	1
Accumulated probabilities	1	0.9	0.775	0.575	0.325	0.125	3.7
Corrected probabilities	(1/3.7=0.2703)	0.2432	0.2095	0.1554	0.0878	0.0338	1

4.2.2. Results

Once the afore mentioned calculations are done, the mean objective values can be calculated. This gives a representation on the performance of the algorithm in this network. The calculations are done with a resolution of 2 seconds per timestep, this means that every calculation in the algorithm is done per 2 seconds and not per second. This will be more elaborately explained in Section 4.2.3, which also discusses the effects on the simulation and the algorithm performance.

The results of these calculations are given in Table 4.4. This shows the different objective values for each objective function. It also shows the objective values for the other objective functions, for each optimal routeplan according to an objective function, given in the first column. So the first row contains the objective values for different objective functions, when the routeplan is the optimal routeplan according to the standard algorithm (which optimizes for lowest expected travel time). The second row are the objective values for the different objective functions when the routeplan is optimal according to the ETT objective function, used with the FPP algorithm, etc. The standard algorithm does not have values when optimizing for the other objective functions, since the standard algorithm picks one route independent on the arrival time at intermediate nodes. Therefore a routeplan with a different link taken, depending on the arrival time at that node, cannot be calculated with the standard algorithm. The other way around is possible, since for the other objective functions the same next link for all arrival times at that intermediate node can be used.

Table 4.4: Results of the simulation with in the first colomn the objective function that row is optimized for. Every colomn gives the objective value of the objective function given in the first row. Standard is the standard algorithm evaluated, ETT is the expected travel time objective function (Formula 4.1), 95^{th} is the 95^{th} percentile travel time objective function (Formula 4.2), STD is the standard deviation objective function (Formula 4.3) and Mean + STD is the objective function for the mean plus standard deviation (Formula 4.5).

Optimized for \ Evaluated for	Standard	ETT	95^{th} percentile	STD	Mean + STD
Standard	18.22	18.22	23.48	2.43	20.65
ETT	n/a	17.00	23.08	2.61	19.61
95^{th} percentile	n/a	17.68	22.23	2.15	19.83
STD	n/a	18.32	22.66	2.03	20.36
Mean + STD	n/a	17.13	22.49	2.32	19.45

In the diagonal line the lowest values can be found for each column. This is because it is optimized for that specific objective function. The standard and ETT values are the same, when optimized for the standard algorithm. This makes sense since they are both calculated the same way, and for the same routeplan this value should be the same. When the routeplan is optimized for the ETT objective function, there is an improvement of 1.22 seconds in

expected travel time. This shows that even when the same objective function is used, the algorithm improves the objective value, since a lower value is better, and that taking the states of the traffic lights into account is useful. This is because the state of the traffic light when running the algorithm is taken into account and therefore not the expected waiting times, but the recently predicted waiting times are used. This is more precise and therefore a better routeplan can be found. The routeplan is also dependent on the arrival time at the traffic light and therefore there are more possible combinations compared to the standard algorithm. In 44% of the cases the routeplan differs per arrival time at the traffic light. In those 56% of the cases that the routeplan is the same for all arrival times at the second node, the second link that is taken is always 4. Since in most of these instances the first link taken was 2, this makes sense since link 4 gets green earlier. For the part when link 1 is chosen, traffic light B has a high probability of being in Phase 1 and thus a high probability of being green for link 4.

But the improvement in objective function shows that even for the same objective function, improvements can be expected with this algorithm. And even the standard deviation over all the objective values is improved (and thus lower) for the ETT objective function with the FPP algorithm (2.59) compared to the standard algorithm (3.13). For the other objective functions, it differed per objective function how much the routeplans were dependent on the arrival time at the intermediate nodes. Some objective functions almost always chose the same next link, independent of the arrival time. For the 95^{th} percentile 16% of the cases had different routeplans dependent on the arrival time at the node, 56% for the std and 44% for the mean+std. Even though the 95^{th} percentile uses the freedom of changing routes depending on the arrival time in 16% of the cases, the improvement is still more than a second compared to the standard algorithm, but this algorithm does not optimize for the 95^{th} percentile travel time. Therefore if the standard algorithm did optimize for the 95^{th} percentile travel time, this might change. If the network would be larger, with more options, this might also change, but then the calculation complexity would become a larger issue. All other objective functions, except the standard deviation, also have a better expected travel time compared to the standard algorithm. This might be the case because all these objective functions have a part in them that takes the travel time into account, which they try to lower. And with more options to chose from (since they can vary the route depending on the arrival time at intermediate nodes), the expected travel time can be improved. But this might also happen just in this specific network with these specific green time and link travel time distributions, since the objective functions are different and an improvement is not guaranteed.

To clarify the results and the differences between objective functions, some travel time distributions are shown that are calculated for a specific starting traffic light state for both traffic lights. The standard deviation does not take a low travel time into account, and the expected travel time (ETT) does not take reliability or deviations into account. Therefore the objective function of the STD scores worse when optimized for the ETT and the ETT scores worse when optimizing for the STD. The travel time distribution shown in Figure 4.6 clarifies this. The standard deviation picks a routeplan with a higher mean travel time but with a lower standard deviation compared to all other objective functions. All other objective functions prefer a routeplan with a lower mean travel time in this case. This was expected to happen since the goal of the STD objective function is only to lower the standard deviation, therefore it does not matter if the travel time increases, as long as the deviation in this travel time decreases. For the ETT this is exactly the other way around, the deviation does not matter, only the expected travel time matters. And since only reliability is not taken into account by the ETT objective function, some travel time distribution have multiple tops. This is shown in Figure 4.7. The expected travel time might be lower for the ETT objective function in this distribution, but the probability of having a travel time around the expected travel time is lower compared to the other objective functions in this example. And you are more likely to arrive before or after the expected time, people might think the routeplan was not optimal. In some cases this also happens for the standard algorithm as shown in Figure 4.8. This is due to the fact that traffic lights have a higher probability of being red on certain times and routes and therefore there is a lower probability of arriving at the destination some time later. Adding reliability into the objective function could accommodate the users preference in a certain travel time distribution. If for instance the mean plus standard deviation is used as objective function, the probability of having a travel time around the expected travel time is increased in these two examples. But in Figure 4.9 the 95^{th} percentile travel time has a higher probability of arriving around the expected time, but has a higher expected travel time. This shows that reliability is something that is taken into account in the objective function, but finding a function that weighs the reliability and travel time to a representation of the users preference, could be hard since there are many different situations. Therefore such a system should be flexible enough to accommodate the user's objectives that may change from trip to trip.

4.2.3. Complexity and resolution

The complexity of the FPP algorithm can increase quickly since it will try to find the optimal routeplan, dependent on every possible arrival time at all intermediate nodes. Therefore the complexity will increase with every

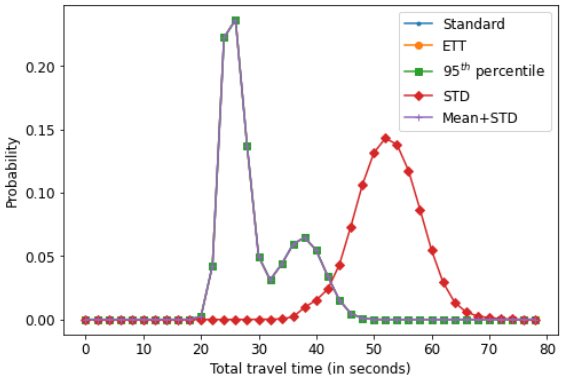

Figure 4.6: The travel time distribution when the routeplans are optimized for the corresponding objective function. The standard deviation is the line with the top on the right side, all other objective functions follow the other line. The start states of the traffic lights were 1 second in phase 1 for the first traffic light and 1 second in phase 2 for the second traffic light.

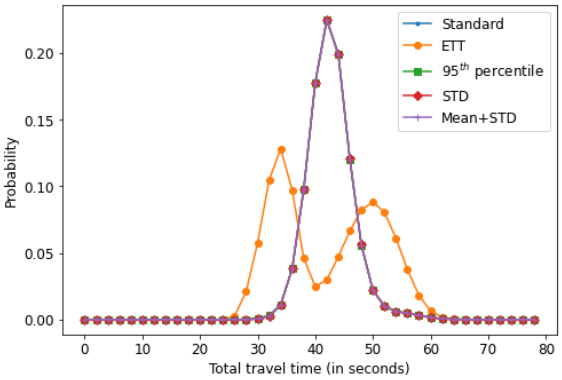

Figure 4.7: The travel time distribution when the routeplans are optimized for the corresponding objective function. The expected travel time is the line with two tops, all other objective functions follow the other line. The start states of the traffic lights were 3 seconds in phase 2 for the first traffic light and 1 second in phase 2 for the second traffic light.

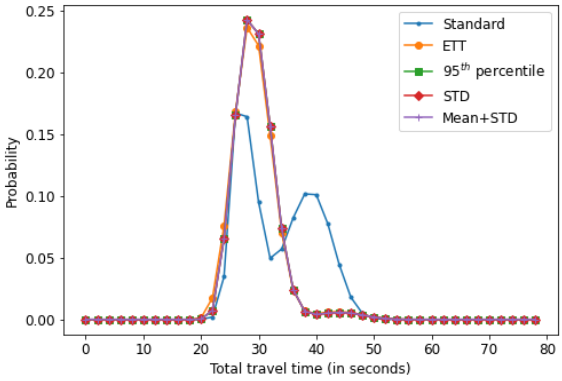

Figure 4.8: The travel time distribution when the routeplans are optimized for the corresponding objective function. The standard algorithm is the line with two tops, all other objective functions follow the other line. The start states of the traffic lights were 3 seconds in phase 4 for the first traffic light and 1 second in phase 2 for the second traffic light.

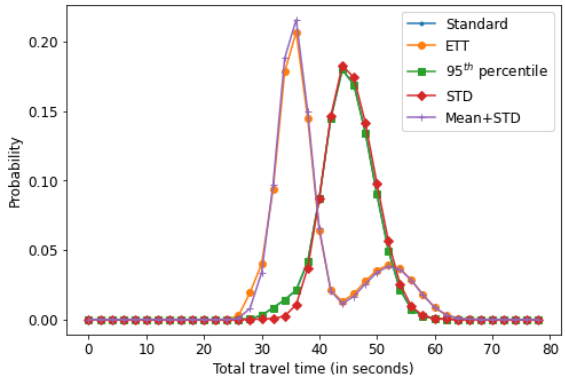

Figure 4.9: The travel time distribution when the routeplans are optimized for the corresponding objective function. The expected travel time and the mean + std are the lines with the top on the left, all other objective functions follow the other line. The start states of the traffic lights were 3 seconds in phase 2 for the first traffic light and 3 seconds in phase 1 for the second traffic light.

researched time step, and at a node with two possible next links, it will increase with 2^t, with t being the number of time steps a vehicle can choose between the next links. Naturally every extra node also increases the complexity and therefore, besides a preselection of nodes, the resolution of simulation is critical for its complexity. The resolution is the ratio between the timestep from the input data (traffic light prediction module and link travel time prediction module) and the timestep that the algorithm uses. This resolution can be used to reduce the calculational complexity. In this case the input data has a timestep of 1 second and the algorithm will be run with a timestep of 2 seconds, thus a resolution of 2.

Increasing the resolution from one simulation step per 2 seconds, to one simulation step per second, will increase the calculation complexity of this part. If there are for instance 16 seconds in which the vehicle can arrive at the intermediate node and decide between the two possible next links, this will lead to $2^{16} = 65,536$ calculations, while with a resolution of two seconds this are just $2^8 = 256$ calculations. If there are 32 timesteps in which a vehicle can arrive at intermediate node, the number of calculations would be more than 4 billion. During the calculations done before, the cpu-time was measured, which is the time the cpu is doing actual calculations and not occupied with other tasks. In a situation with 2^{16} calculations, the cpu time was 1540 cpu seconds, while in a situation with 2^8 calculations, the cpu time was just 12 cpu seconds. Therefore the calculational complexity of the algorithm should be kept in mind when selecting the resolution of the simulation.

In this part of the evaluation a resolution of 2 is used. Thus the FPP algorithm uses timesteps of 2 seconds, while the input uses timesteps of 1 second. The FPP algorithm makes this work by taking the link travel time and waiting

time at the traffic lights per 2 timesteps instead of one. Thus two link travel times or waiting times are taken as one and their probabilities added up. Normally the average of both values will be used as new travel time or waiting time, but since the input from the link travel time and traffic light prediction module is in seconds and not 0.5 seconds, this cannot be done. Therefore this value is rounded up and the travel times might be overestimated. This seemed better than forcing it to round down, with underestimations as result. Of course this is not an ideal situations, but very large calculation times are also not ideal.

Besides the resolution, the complexity is also dependent on the number of timesteps a vehicle can arrive at the intermediate nodes, which is dependent on multiple factors. If the travel time on the links can vary a lot, there are more arrival times at the next node. And for the traffic light, the maximum waiting time is relevant, since a larger maximum waiting time might result in a larger set of possible waiting times. The maximum waiting time can be calculated as the cycletime minus the green time for the phase the direction gets green (assuming every direction gets green once in a cycle). Therefore traffic lights with long cycletimes can have more travel times (including the waiting time) to the next node. But besides the maximum waiting time, a minimum waiting time is relevant as well. For some traffic lights this is 0 since phases can be skipped and thus for some probability a vehicle can always have green. This happens if there is a probability on having no traffic on a certain link, and thus a probability on that phase being skipped. If there is a probability of skipping the phase, for all phases, there is always a small probability of the traffic light turning green straight away when a vehicle approaches it. But for other traffic lights, like fixed time traffic lights or network coordinated traffic lights, a minimum waiting time might be present. Since this minimum waiting time reduces the amount of possible travel times to the next traffic light, this reduces the calculational complexity also for the next traffic light.

When the calculational complexity is reduced by increasing the size of the timesteps, the performance of the FPP algorithm in finding the optimal routeplan might decrease. If for instance the timestep is set on 2 seconds, and the traffic light changes after one second, the FPP algorithm can only react to this by a suggested routeplan for both seconds. Another aspect to consider is that if the objective function tries to optimize the probability of arriving before a certain moment, it might not even be possible to arrive exactly at that moment. If the resolution is 2 seconds, and it is tried to arrive before 55 seconds, the algorithm will determine 56 seconds as too late and 54 seconds as on time, but exactly 55 seconds can never occur, which decreases the performance. If the resolution is increased to multiple timesteps, the expected travel time calculated with the FPP algorithm might become less accurate and at some point, using a standard algorithm (with a better resolution) might result in a better route. The amount of decrease in performance and the effect on the results is dependent on the objective function and network. Therefore increasing the resolution of the simualation is something that should be done carefully and not without considering the specific situation, benefits and downsides.

4.3. Summary

In this section the FPP algorithm is evaluated by showing that the algorithm is working with a verification by simulation. The algorithm is also calculated for a specific case for all possible starting states of the traffic light to calculate the improvements it could make.

4.1 In this subsection it is shown that the algorithm is working, by verifying it with a simulation. The FPP algorithm has been calculated, with an objective function, for a network, which resulted in a travel time distribution, for that specific routeplan and starting state of the traffic lights. A simulation with vehicles driving trough the network has been done as well, which resulted in a travel time for every vehicle, and thus a travel time distribution for all vehicles. These vehicles drove the routeplan, calculated by the FPP algorithm before, to check if the distributions match. These distributions matched and therefore the algorithm calculated the correct distribution. The network that was used had two traffic lights, with 2 links between each traffic light. The link travel time distribution was given, just as the probability on green times for each phase of the traffic lights. Then with the algorithm described in Section 3.2 the waiting time distributions were calculated. Which is all the input required for the algorithm.

4.2 For this part of the evaluation the objective values for four objective functions and a standard algorithm, that only takes the expected values for link travel times and waiting times at the traffic light into account, are calculated with the FPP algorithm. This is done for every starting state of the traffic lights. This standard algorithm allows for comparison of the same objective function, with and without the FPP algorithm. The four objective functions that are calculated with the FPP algorithm are the expected travel time, 95^{th} percentile travel time, the standard deviation and the mean + the standard deviation.

For this analysis individual vehicles do not need to be simulated, since in the section before it is shown that the travel time distribution calculated by the FPP algorithm represents the actual travel time distribution

when simulating. Therefore the FPP algorithm is calculated for every starting state of the traffic lights and for every objective function. Since the mean objective values of all runs are used, these objective value should be multiplied with the probability on the starting states of the traffic lights happening. This results in the mean objective value for this entire network. These values can then be compared to the other objective values.

The results from this calculation showed that the FPP algorithm improved the expected travel time trough the network with 1.22 seconds (on an expected travel time of 18.22 seconds). Therefore even when the same objective function is used, the FPP algorithm will improve the routeplan. This is possible since it makes the route time dependent and thus a different next link could be selected per arrival time at the intermediate nodes. The ETT does this in 44% of all cases. The improvement could be dependent on the network and resolution. For the other objective functions the FPP algorithm optimized the objective function as well and the objective function uses the flexibility in 16% of the cases for the 95^{th} percentile objective function, 56% for the std and 44% for the mean+std. Besides the results of the calculation, it is also important to note that finding an ideal objective function is difficult. The travel time distributions can take different shapes and creating one function that gives the optimal distribution for the preferences of different users is not trivial since there are many ways to take reliability into account.

The complexity of the FPP algorithm can increase quickly and depends on the number of timesteps a vehicle can choose between different links at intermediate nodes. At this point the number of calculations are n^t with n being the number of next possible links at that node and t being the number of timesteps a vehicle can make the choice. This increases quickly with a larger number of timesteps and therefore these can be reduced by using a simulation resolution. This resolution is the difference in timesteps used in the input of the algorithm (link travel times and traffic light data) and the timestep used in the FPP algorithm. If the FPP algorithm uses a timestep of 2 seconds, while the input data uses 1 second, the resolution is 2. This decreases the calculational complexity, but also has influence on the optimal routeplan and thus the performance of the algorithm.

5

Conclusions and discussion

In the first section of this thesis the research goal and questions were formulated. In Section 5.1 the conclusions of the research are given by answering these research questions and explaining how the research goal has been fulfilled. In Section 5.2 improvements to this research and recommendations on further research are given.

5.1. Conclusions

In this research two new algorithms are proposed for routing vehicles trough an urban network with traffic lights while taking reliability into account. While developing these algorithms four research questions were answered which are presented in this section. Afterwards the research goal is repeated and explained why this goal has been met.

What are the existing approaches to predict traffic light states and what is the used input, the accuracy, the complexity, the type of traffic lights on which it can be used and how can it be incorporated in the routing algorithm?

There are many approaches to predict traffic light states, but since traffic lights with a changing cycle time in a larger network were researched, methods that predict the complete actual traffic state could not be used because of calculational complexity. Most of the current approaches in predicting the traffic light state were limited to usages with GLOSA and would therefore only predict for about 30 seconds ahead. Because the predictions will be done for a routing system, 30 seconds is not enough and a method is searched that can be expanded to work for minutes ahead. Two researches used historic data to create transition probabilities between different states of the traffic light. With these transition probabilities, the probability of being in a future state could be calculated once the current state was known. This method was picked as the basis of the traffic light prediction module because it could be expanded to predict further ahead, used available data, was not to calculational complex and could be used on traffic lights with a changing cycle time.

To incorporate this into the algorithm, the probability of a direction being green in the future should be calculated. This is done with the methods from the paper above and resulted in the probability of being green for every direction at every time in the prediction horizon. All of this calculations can be done once before running the algorithm, but since the memory requirement might become too large, some parts of it can also be calculated when running the algorithm. If the routing algorithm is calculated forwards, the waiting time at the traffic light could be calculated instead of the probability on a green traffic light. This results in more precise predictions and makes sure that a vehicle can never wait longer then the cycle time.

What are the existing approaches to predict travel times on links?

Since this research is focused on routing in urban areas, travel times of the links should be known. But the travel times on those links fluctuate because of the interaction between vehicles and the presence of intersections. Three models are relevant to predict the link travel time, namely the queueing model, queue discharge model and platoon dispersion model. For the queueing model it is relevant if a horizontal or vertical queueing model is used, since waiting traffic could block entrance to a turning lane with the horizontal queueing model. With the vertical queueing model vehicles are stacked on top of each other and spillback effects are therefore not taken into account. The queue discharge model is dependent on two factors. The queue discharge rate, which is how many vehicles per hour are crossing the stop line, while driving their desired speed. And the acceleration loss, which is how quickly

vehicles are crossing the stop line just after the traffic light turned green and while they are not driving their desired speed. Those values are best locally measured since these are dependent on a lot of factors that influence driver behaviour. The last model is the platoon dispersion model, this model determines how quickly a platoon disperses after driving away at a green light. Robertsons model is often used (Paul et al., 2016) and has two parameters, which are both influenced by many factors. But most of these factors also have effect on the link travel time and standard deviation and therefore these parameters can be calculated from the link travel time and standard deviation. The position in the queue should be known to use the aforementioned models, because if the position in the queue is not known, the speed and location of the vehicle when driving out of the queue is not known and these are required for the queue discharge model and platoon dispersion model. But since the algorithm that will be used does not support exact positions in the queues, these models cannot be used. Adding these could be an improvement to the algorithms. In this research a link travel time distribution will be used that can be dependent on the time the vehicle enters the link, but the position in the queue is not known and when the traffic lights turns green the vehicle is assumed to be driving to the next link directly.

What are the existing routing algorithms and what is the used input, the complexity and how can it incorporate a probability distribution of the travel time and an individual cost function?

There are just a few algorithms that take traffic lights into account and even less that can deal with probabilistically known traffic light states. The ARSC method by B. Yang and Miller-Hooks (2004) was best suited since it could route traffic in a time dependent stochastic network, take probabilistically known traffic light states into account and stored optimal routes at intermediate nodes, to reduce its calculational complexity. It determines the optimal route while taking the travel time on the links and the probability of the traffic lights being green into account. This probability is dependent on the arrival time at each node, and therefore an optimal route is a set of route choices depending on the arrival time at the node, which is called a routeplan.

Since it only gives the lowest expected travel time, the method should be adjusted to save the travel time distribution and use an individual cost function. Since the ARSC algorithm is a label correcting algorithm that saves the optimal routeplan at intermediate labels, these labels should be changed to include a distribution. Next to that, the optimal routeplan should be determined dependent on the individual cost function. This is done at every intermediate label which results, once all labels are calculated, in an optimal routeplan.

This new algorithm, the ARSCR algorithm, does not work for all objective functions. Since it saves the optimal routeplan at intermediate nodes and works backwards from the destination node towards the departure node, the optimal routeplan should be independent of data that will be added at every upstream node. If this condition is not met, the result might be a suboptimal routeplan, because the new data added at a node upstream, might result in a different optimal routeplan downstream, while this part has already been calculated. Since reliability might be dependent on the entire distribution, another algorithm is created.

This algorithm propagates the probability of having a certain travel time forward, and is therefore called the Forward Probability Propagation (FPP) algorithm. This algorithm calculates the probability distribution of the travel time to the destination, while taking a certain routeplan. A routeplan is not just a series of links which should be used after each other, but also the time at which a vehicle arrives at the intermediate nodes. The time a vehicle arrives at all intermediate nodes is relevant, since the probabilities on the states of the traffic light are dependent on the arrival time at that node. This is done from the departure node towards the destination node and from time t=0 until the complete routeplan is calculated. Every routeplan will result in a different distribution and from all those distributions the optimal routeplan could be determined with the objective function. Since this is calculational complex, a different algorithm that selects certain links to take into consideration might be required. The input for both algorithms is the link travel time prediction and the traffic light prediction model. But for the FPP algorithm, the traffic light prediction model gives the probability on a certain waiting time at the traffic light, instead of the probability of being green. This gives a more precise prediction, but also requires more calculation power. For the ARSCR algorithm this is not possible since it is a backwards algorithm and therefore it is not known at what time the vehicle arrives at the traffic light.

How does the routing algorithm perform in a simulation, compared to a standard algorithm that searches for the lowest expected travel time?

First it is proven that the algorithm is working, by verifying it with a simulation. The FPP algorithm has been run on a network, with an objective function, which resulted in a travel time distribution, for that specific routeplan and starting state of the traffic lights. A simulation with vehicles driving trough the network has been done as well, which resulted in a travel time for every vehicle, and thus a travel time distribution for all vehicles. These vehicles

take the routeplan, calculated by the FPP algorithm before, to check if the distributions match. These distributions matched and therefore the algorithm calculated the correct distribution.

To show the change in performance of the algorithms, the network does not need to be simulated, since in the section before it is proven that the calculated travel time distribution becomes the actual travel time distribution, when simulating. Therefore the FPP algorithm is calculated for every starting state of the traffic lights with every objective function. Since this is done for all traffic light states to create all possible situations, the objective value should be multiplied with the probability on the starting traffic light state, to calculate the mean objective value for this entire network. These values can then be compared to the other objective values.

The results from this calculation showed that the FPP algorithm improved the expected travel time trough the network with 1.22 seconds (on an expected travel time of the standard algorithm of 18.22 seconds). Therefore even when the same objective function is used, the FPP algorithm will improve the routeplan. This is possible since it makes the route time dependent and thus a different next link could be selected per arrival time at the intermediate nodes. The expected travel time objective function does this in 44% of all cases, which might increase with larger networks since there are more possible different routeplans to follow. But these numbers could also be dependent on the network and resolution. Besides the results of the calculation, it is also important to note that finding an ideal objective function is hard. The travel time distributions can take different shapes and creating one function that gives the optimal routeplan for an individual user in all situations is not trivial since there are many ways to take reliability into account.

The complexity of the FPP algorithm can increase quickly and depends on the number of timesteps a vehicle can choose between different possible next links at intermediate nodes. At this point the number of calculations are n^t with n being the number of possible next links at that node and t being the number of timesteps a vehicle can make a choice between those next links. This increases quickly with a larger number of timesteps and therefore these can be reduced by using a simulation resolution. This resolution is the difference in timesteps used in the input of the algorithm (link travel times and traffic light data) and the timestep used in the FPP algorithm. If the FPP algorithm uses a timestep of 2 seconds, while the input data uses 1 second, the resolution is 2. This decreases the calculational complexity, but also has influence on the optimal routeplan and thus the performance of the algorithm.

Now that the research questions are answered the research goal can be reviewed.

Designing an urban routing algorithm, that incorporates predicted traffic light states, where the routing is based on the probability distribution of the travel time and an individual cost function that expresses the tradeoff between travel time and reliability.

By designing the ARSCR and FPP algorithm together with the traffic light prediction module, the research goal is met. The individual cost function is introduced and all cost functions can be used for the FPP algorithm, while only some can be used together with the ARSCR algorithm. But since the ARSCR algorithm is more efficient, using this algorithm might be preferred, if possible. The trade-off between travel time and reliability can be made by every individual user since all functions can be entered as the cost function. This algorithm can even improve performance if the same objective function is used as in other navigation systems, since it is routing vehicles depending on the arrival time at each node.

5.2. Discussion

The performance of the developed ARSCR and FPP algorithms could be improved and computation times reduced. This section elaborates on the enhancements for the algorithms and other further research.

5.2.1. Improving the link travel time prediction

Taking the link travel time correctly into account is one of the aspects of the algorithms that could be improved. Even though these link travel times are assumed to be available, not all aspects are taken into account correctly.

Queueing model

There is no queueing model incorporated into the algorithm and therefore the delay at the traffic light is independent of the queue at the traffic light. This does not only affect the waiting time at the traffic light, a vehicle could also miss the green phase entirely if there is too much traffic. While in this research undersaturated conditions were assumed, on the street there might always occur a situation which leads to oversaturated conditions. And even in an undersaturated condition, spillback can still occur. For instance if enough traffic is waiting to go straight on and just a few vehicles are turning right. If the dedicated turning lane is short, the vehicles might not be able to reach this turning lane. An extensive horizontal model would be best, but a simple vertical model could be expanded with

a maximum number of vehicles going straight on, before the next vehicles cannot turn right anymore. This might not be accurate enough but might be better than not having a queueing model at all. Adding this model requires values for expected arrivals, which might be dependent on the network and other traffic lights in the area. Therefore adding these features might improve the prediction accuracy but could increase computational time as well.

Queue discharge and platoon dispersion model

Since the queueing model is not taken into account in this method, the queue discharge and platoon dispersion model cannot be used. Both models need to have knowledge of the position in the queue, which is not known because no queueing model is used. But if a queueing model is added, as described in the paragraph before, these models can be implemented. This could increase the accuracy of the link travel time prediction but could increase computational times as well.

Combining the link travel time prediction and the traffic light prediction model

If the link travel time prediction model is improved, it could improve the traffic light predictions, since the length of the queue has a direct impact on the green time and thus the state of the traffic light. Therefore, if these models are used, these could be integrated with the traffic light prediction model as well to get the most accuracy out of the increased complexity. This can be done by adding for instance a (bandwidth of) queue length to the state of the traffic light. The method can stay the same since only the definition of a state is changed, but the calculation time for the traffic light prediction model increases. If this is done on a network level for multiple vehicles, part of this could be done system wide to reduce the number of calculations for each vehicle.

5.2.2. Traffic light state prediction

Predicting traffic lights for minutes ahead is a complicated task that can be done in different ways. In this research two different methods are explained, one for the ARSCR algorithm and one for the FPP algorithm. Firstly, it is discussed if and how the traffic light state prediction method for the FPP algorithm could be implemented into the ARSCR algorithm. Secondly, some methods that might reduce the computation time are given.

Improving the traffic light prediction for the ARSCR algorithm

The FPP algorithm is developed to facilitate the usage of all objective functions, and since it was changed to a forwards algorithm it also facilitates an improved traffic light prediction model. The computation of the traffic light prediction model is somewhat more complex compared to the ARSCR counterpart. This is because the waiting time is calculated instead of the probability of being green. But this will increase prediction accuracy since waiting longer than the cycletime has become impossible. If this feature is added to the ARSCR algorithm, this algorithm has to change to a forwards model (or at least for certain parts). This could be done by saving the optimal previous node instead of the optimal next node, but requires quite a change to the algorithm. In the end it might be worth it, since at the moment there is always a small part of the traffic, that is waiting longer than the cycle time. An even smaller part does not reach the destination, if the traffic lights are not turned to green with a probability of 1. Therefore adding a better traffic light prediction model would at least remove these problems. If the algorithm is changed to a forwards model, it might also decrease the amount of paths to calculate, since at every node only the travel time to the next node has to be calculated if there is a probability (greater than zero) of a vehicle being at that moment at that node. With the current algorithm all paths are calculated for all time steps within a certain range.

Predictions converging to averages

The accuracy of predictions will decrease when predicting further ahead and this might converge to the average waiting time at the traffic light during that time period. Therefore it might be relevant to research how long ahead the predictions still add value to the routing program or when averages can be used for quicker calculations. This might result in two parts in the routing algorithm, one that uses the real time traffic light predictions, and one that does not and routes the vehicle trough the end of the network. After crossing a (few) intersection(s) the algorithm might be run again, to use the first algorithm for the oncoming traffic lights as well. This might decrease calculation time while keeping a good accuracy. This should be done before the FPP algorithm could be implemented in the real world, since otherwise it is calculationally too complex. This large calculation complexity is mainly an issue with the FPP algorithm. The ARSCR algorithm does not have to calculate all routeplans completely, since it can determines the optimal routeplan at intermediate nodes and calculates the optimal routeplan therefore much faster in larger networks.

Reducing the number of states

Another way of improving calculation time is by changing the definition of a state to decrease the number of states a traffic light could be in. The states could be made less dependent on timers of minor directions. If for instance a right turn is always able to get green with the traffic going straight on, but has almost no traffic, the timer of this direction might not be relevant for when the next phase starts. But if there is traffic and it turns green, it is noted as a different state compared to when there is no traffic, even though the next phase only starts when the queue on the main direction is dissolved. Another way of reducing the number of states is by assigning certain directions as alternative directions in the state of the traffic light. These directions could receive green when there is less traffic on certain main directions. By doing this, the number of states is reduced since a state is now just a phase, one timer and for instance a percentage that a certain alternative direction can get green. Since the percentage of being green for smaller directions is no longer a timer as well, this single value (percentage that direction can get green) reduces the number of states. All these methods could decrease complexity and might even improve performance, but the effects will be different for most traffic lights.

5.2.3. Routing algorithms

There are two algorithms developed in this research, the ARSCR algorithm and the FPP algorithm. Both have its benefits and downsides and some improvements to both are discussed here.

Limiting the number of routeplans

Another improvement might be to select the set of routeplans to take into consideration more efficiently. Since the calculational complexity is high, methods of reducing this should be searched. Since all routeplans are searched for, smart methods of reducing this number of routeplans could be used. Besides methods that limits the number of links to take into account, methods that limits the number of routeplans that need to be calculated completely can also be used. As described in the paragraph "Predictions converging to averages" before, only using this algorithm for a few traffic lights and using a standard algorithm or averages for the rest reduces the number of routeplans. But sometimes some routeplans do not need to be calculated since an almost identical routeplan is already calculated with a high objective value (assuming a low value is better). The probability that a tiny change improves this from a routeplan with a high objective value to the best routeplan (with the lowest objective value) is very small. If for instance it is known that taking link 4 for all arrival times up until 20 seconds and link 3 afterwards returns a certain objective value and taking link 4 for all arrival times up until 21 seconds and link 3 afterwards returns a higher objective value, it might not be required to calculate taking link 4 for all arrival times up until 22 seconds since this probably increases the objective value again (assuming a lower objective value is better). This should be checked for every few (e.g. 5) timesteps if this still holds, but this could reduce the total number of searched routeplans. Methods like this could be used to decrease calculation time and allow lower resolutions and thus might improve the performance.

Introducing a minimum probability

Since the number of routeplans to calculate is dependent on the number of timesteps a vehicle can arrive at intermediate nodes, this number should be kept as low as possible. This can be done by using a simulation resolution and thus using larger timesteps in the simulation. However, other methods can be used as well. Sometimes there is a very small probability on a certain travel time to that intersection but for this very small travel time all possible combination of routeplans have to be calculated as well. Therefore removing very small probabilities (or adding them to the closest other travel time) might reduce calculation time while minimizing the effects on the performance.

Updating the routeplan when arriving at the traffic light

An improvement for both algorithms might be to incorporate the option to change your routeplan when the states of all traffic lights are known, when the vehicle is at a route decision point. A vehicle does not change its routeplan depending on the actual state of the traffic light once it arrives at that traffic light and still follows the predictions. Doing this might improve performance, although it should be taken into account that a person cannot see the state of the traffic light, just the color of the traffic light, but their navigation system could. An update from the traffic light via SPaT data could be received since some traffic lights are sending them real time, but then the algorithm have to be run again with this data. Since running the algorithm takes some time, the latest update should be a few moments before a vehicle arrives at the decision point. It should also be taken into account that redirecting a route at the last moment might be unsafe, since the persons will have a short time to change lanes.

5.2.4. Algorithm evaluation

The FPP algorithm is evaluated, which might be improved with a better resolution, a longer network and more possible routes to take. But the simple network used, already had a high calculation time and therefore complexity might be reduced first. The evaluation itself could also be extended with a microscopic simulation or using real world data.

Microscopic simulation

Another way of improving the evaluation is by simulating it in a microscopic simulation model since this will simulate other vehicles as well. Therefore effects on the network could be measured, as well as the effect of the other vehicles on the routing program. Even though the evaluation chapter shows an improvement in the objective value, in reality this improvement might be less, since there are more influences than variables used in this algorithm. For instance the link travel time is an uncertainty that could be incorporated more thoroughly and the dynamics between vehicles are not taken into account. In a microscopic simulation the situation that multiple vehicles are using this system can also be simulated. Multiple vehicles could then take the same suggested route which might increase the travel time on that route.

Simulating the ARSCR algorithm

The ARSCR algorithm is not evaluated at all. Since a different traffic light prediction model is used, the effects on the objective function could be different. Therefore it is interesting to see what the effect of the different traffic light prediction module is before adding the more complicated traffic light prediction module (used by the FPP algorithm) to the ARSCR algorithm, as suggested before. The complexity of the algorithm is less since the optimal routeplan can be saved at intermediate nodes. Therefore it might also be interesting to see what the actual difference in calculation time is. With the effects on the objective function and change in calculation time, a trade-off between using the ARSCR and FPP algorithm can be made more substantiated.

Using real world data

Real world data can be used to verify the effects of the algorithm. The traffic light prediction algorithm can be validated by using real world data. Using sensitivity analyses a large dataset can be used to check if a smaller part of that dataset still represents the larger dataset in some way. This also helps to check if the traffic does not change considerably in one time period since this will make predictions less accurate.

Some vehicles could also be equipped with a pilot version of the algorithms and it could be checked how much the benefits are on the road. Of course the system should be quick and accurate enough for this to work and it should show the routeplan to the user in a clear way. Since the complexity should be reduced and a method of dealing with larger networks is required as well, these problems have to be tackled first. Besides these points, the difficulties on implementing these algorithms in practice should be solved as well. These are explained in the next subsection.

5.2.5. Implementation in practice

In this research two new algorithms are designed that calculate the optimal routeplan to the destination. They both give a routeplan, which gives a next link dependent on the arrival time at that node. The algorithm also allows the user to use an objective function to optimize their route so that they can take reliability into account. These two points are the main contributions of this paper and are both not commonly used in routing systems these days. Therefore users need to be informed clearly on this system and the routing system should be designed with an user friendly interface.

Informing the driver

Before one of the algorithms can be used by individual road users, a good way of informing the driver about the system should be created as well. Since the routing is dependent on the arrival time at nodes, the exact route might change if a vehicle arrives slightly later at the node than expected. Therefore this should be communicated clearly to the drives so that they know that the route shown is not definitive, but dependent on the arrival time. It might also be useful to not show a more optimal route if the vehicle is very close to the intersection, since the driver should keep his attention on the road and intersection. If a driver wants to change lanes quickly because the optimal route has changed, dangerous situations might arise.

Objective functions

A few suggested, most used, objective functions should also be set in the system, so that not every user needs to create their own function. But these objective functions are not easy to create or to communicate what it exactly does, since weighing reliability with travel time is still a matter of personal preference depending on the situation. The preferred route can be different depending on a lot of factors, such as: the person, the type of trip, the time of the day and maybe even the mood of the user. Therefore researching what function best represents reliability for certain group of users might be worthwhile but also complicated.

Conclusion

In short, the calculation time of the FPP algorithm needs to be reduced before it can be used in real routing systems. For both algorithms there is further research needed to check if they can be used for just a few traffic lights and use average values for the rest of the network. A clear system of informing the driver about the routeplan and using the objective functions should be researched as well before it can be implemented in routing systems.

References

Agarwal, A., Lämmel, G., & Nagel, K. (2018). Incorporating within link dynamics in an agent-based computationally faster and scalable queue model. *Transportmetrica A: Transport Science*, *14*(5-6), 520-541.

Akçelik, R., & Besley, M. (2002). Queue discharge flow and speed models for signalised intersections. *Transportation and Traffic Theory in the 21st Century, proceedings of the 15th International Symposium on Transportation and Traffic Theory*, 99-118.

Apple, J., Chang, P., Clauson, A., Dixon, H., Fakhoury, H., Ginsberg, M., Keenan, E., Leighton, A., Scavezze, K., & Smith, B. (2011). Green driver: AI in a microcosm. , 1311–1316.

Baass, K. G., & Lefevre, S. (1987). *Analysis of platoon dispersion with respect to traffic volume.*

Bander, J., & White, C. (2002). A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost. *Transportation Science*, *36*, 218-230.

Bansal, P., & Kockelman, K. M. (2017). Forecasting americans' long-term adoption of connected and autonomous vehicle technologies. *Transportation Research Part A: Policy and Practice*, *95*, 49 - 63.

Bie, Y., Liu, Z., Ma, D., & Wang, D. (2013). Calibration of platoon dispersion parameter considering the impact of the number of lanes. *Journal of Transportation Engineering*, *139*(2), 200-207.

Bodenheimer, R., Brauer, A., Eckhoff, D., & German, R. (2014). Enabling GLOSA for adaptive traffic lights. In *2014 ieee vehicular networking conference* (p. 167-174).

Bodenheimer, R., Eckhoff, D., & German, R. (2015). GLOSA for adaptive traffic lights: Methods and evaluation. In *2015 7th international workshop on reliable networks design and modeling* (p. 320-328).

Chabini, I. (1997). A new algorithm for shortest paths in discrete dynamic networks. *IFAC Proceedings Volumes*, *30*(8), 537 - 542.

Chabini, I. (1998). Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation research record*, *1645*(1), 170–175.

Chai, H., Zhang, H., Ghosal, D., & Chuah, C.-N. (2017). Dynamic traffic routing in a network with adaptive signal control. *Transportation Research Part C: Emerging Technologies*, *85*, 64 - 85.

Chen, Y.-L., & Yang, H.-H. (2000). Shortest paths in traffic-light networks. *Transportation Research Part B: Methodological*, *34*(4), 241 - 253.

Chen, Z., & Fan, W. (2019). Data analytics approach for travel time reliability pattern analysis and prediction. *Journal of Modern Transportation*, *27*.

Dabiri, A., & Hegyi, A. (2018). Personalised optimal speed advice to cyclists approaching an intersection with uncertain green time. In *2018 european control conference* (p. 1666-1671).

Devarasetty, P. C., Burris, M., & Shaw, W. D. (2012). The value of travel time and reliability-evidence from a stated preference survey and actual usage. *Transportation Research Part A: Policy and Practice*, *46*(8), 1227 - 1240.

Dey, P. P., Nandal, S., & Kalyan, R. (2013). Queue discharge characteristics at signalised intersections under mixed traffic conditions. *European Transport*, *55*(7), 1–12.

Dijkstra, E. W., et al. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271.

Eiger, A., Mirchandani, P. B., & Soroush, H. (1985). Path preferences and optimal paths in probabilistic networks. *Transportation Science*, *19*(1), 75-84.

Fayazi, S. A., Vahidi, A., Mahler, G., & Winckler, A. (2015). Traffic signal phase and timing estimation from low-frequency transit bus data. *IEEE Transactions on Intelligent Transportation Systems*, *16*(1), 19-28.

Gao, S., & Chabini, I. (2006). Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B: Methodological*, *40*(2), 93 - 122.

Grace, M. J., & Potts, R. B. (1964). A theory of the diffusion of traffic platoons. *Operations Research*, *12*(2), 255-275.

Hall, R. W. (1986). The fastest path through a network with random time-dependent travel times. *Transportation Science*, *20*(3), 182-188.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100-107.

Hong-En Lin, M. A. P. T., Rocco Zito. (2005). A review of travel-time prediction in transport and logistics. *Proceedings of the Eastern Asia Society for Transportation Studies, 5*, 1433-1448.

Iglesias, I., Isasi, L., Larburu, M., Martinez, V., & Molinete, B. (2008). I2V Communication Driving Assistance System: On-Board Traffic Light Assistant. In *2008 ieee 68th vehicular technology conference* (p. 1-5).

Kaparias, I., Bell, M. G., & Belzner, H. (2008). A new measure of travel time reliability for in-vehicle navigation systems. *Journal of Intelligent Transportation Systems*, *12*(4), 202-211.

Kaparias, I., & Bell, M. G. H. (2010). A reliability-based dynamic re-routing algorithm for in-vehicle navigation. In *13th international ieee conference on intelligent transportation systems* (p. 974-979).

Koukoumidis, E., Peh, L.-S., & Martonosi, M. (2011). Signalguru: Leveraging mobile phones for collaborative traffic signal schedule advisory. , 127-140.

Kouwenhoven, M., de Jong, G. C., Koster, P., van den Berg, V. A., Verhoef, E. T., Bates, J., & Warffemius, P. M. (2014). New values of time and reliability in passenger transport in The Netherlands. *Research in Transportation Economics*, *47*, 37 - 49.

Lin, F.-B., & Thomas, D. R. (2005). Headway compression during queue discharge at signalized intersections. *Transportation Research Record*, *1920*(1), 81-85.

Liu, H. X., Recker, W., & Chen, A. (2004). Uncovering the contribution of travel time reliability to dynamic route choice using real-time loop data. *Transportation Research Part A: Policy and Practice*, *38*(6), 435 - 453.

Liu, Y., Guo, J., & Wang, Y. (2018). Vertical and horizontal queue models for oversaturated signal intersections with quasi-real-time reconstruction of deterministic and shockwave queueing profiles using limited mobile sensing data. *Journal of Advanced Transportation*.

Mahler, G., & Vahidi, A. (2012). Reducing idling at red lights based on probabilistic prediction of traffic signal timings. In (p. 6557-6562).

Manar, A., & Baass, K. G. (1996). Traffic platoon dispersion modeling on arterial streets. *Transportation Research Record*, *1566*(1), 49-53.

Markovich, M., Concas, S., & Kolpakov, A. (2009). Synthesis of research on value of time and value of reliability. *National Center for Transit Research, Center for Urban Transportation Research*.

Miller-Hooks, E. (2001). Adaptive least-expected time paths in stochastic, time-varying transportation and data networks. *Networks*, *37*(1), 35-52.

Miller-Hooks, E. D., & Mahmassani, H. S. (2000). Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, *34*(2), 198-215.

Nie, Y. M., Wu, X., Dillenburg, J. F., & Nelson, P. C. (2012). Reliable route guidance: A case study from chicago. *Transportation Research Part A: Policy and Practice*, *46*(2), 403 - 419.

Paul, B., Mitra, S., & Maitra, B. (2016, 01). Calibration of robertson's platoon dispersion model in non-lane based mixed traffic operation. *Transportation in Developing Economies*, *2*(2), 11.

Protschky, V., Feit, S., & Linnhoff-Popien, C. (2014). Extensive traffic light prediction under real-world conditions. In *2014 ieee 80th vehicular technology conference* (p. 1-5).

Protschky, V., Feld, S., & Wälischmiller, M. (2015). Traffic signal adaptive routing. In *2015 ieee 18th international conference on intelligent transportation systems* (pp. 450–456).

Protschky, V., Wiesner, K., & Feit, S. (2014). Adaptive traffic light prediction via kalman filtering. *IEEE Intelligent Vehicles Symposium, Proceedings*, 151-157.

Saiáns-Vázquez, J. V., Ordóñez-Morales, E. F., López-Nores, M., Blanco-Fernández, Y., Bravo-Torres, J. F., Pazos-Arias, J. J., Gil-Solla, A., & Ramos-Cabrer, M. (2018). Intersection intelligence: Supporting urban platooning with virtual traffic lights over virtualized intersection-based routing. *Sensors*, *18*(11).

Sjoberg, K., Andres, P., Buburuzan, T., & Brakemeier, A. (2017). Cooperative intelligent transport systems in europe: Current deployment status and outlook. *IEEE Vehicular Technology Magazine*, *12*(2), 89-97.

Talking traffic gebruikstoepassingen. (n.d.). Retrieved 2019-06-19, from `https://www.talking-traffic.com/nl/thema-s/data-beschikbaar/download/16_673aecd36c485e47fe222a14fc22d8a7`

Urban, T. L. (2009). Establishing delivery guarantee policies. *European Journal of Operational Research*, *196*(3), 959 - 967.

Wilson, A. (2006). *Handboek verkeerslichtenregelingen*. CROW.

Yang, B., & Miller-Hooks, E. (2004). Adaptive routing considering delays due to signal operations. *Transportation Research Part B: Methodological*, *38*(5), 385 - 413.

Yang, S., & Wu, Y.-J. (2019). Chapter 4 - data-driven approaches for estimating travel time reliability. In Y. Wang & Z. Zeng (Eds.), *Data-driven solutions to transportation problems* (p. 81 - 110). Elsevier.

Yu, L. (2000). Calibration of platoon dispersion parameters on the basis of link travel time statistics. *Transportation Research Record*, *1727*(1), 89-94.

Zeng, W., & Church, R. L. (2009). Finding shortest paths on real road networks: the case for A*. *International journal of geographical information science*, *23*(4), 531–543.

Ziliaskopoulos, A. K., & Mahmassani, H. S. (1996). A note on least time path computation considering delays and prohibitions for intersection movements. *Transportation Research Part B: Methodological*, *30*(5), 359 - 367.

A

Variables

A.1. ARSC Algorithm

A list of variables, their domains and their descriptions for the ARSC algorithm.

Variable	Domain	Description
$\eta_i^h(t)$	$[0, S]$	Temporary label that temporarily saves the expected travel time to the destination node. This is the label at node i coming from node h at time t
$\lambda_i^h(t)$	$[0, S]$	Label that saves the expected travel time to the destination. This is the label at node i coming from node h at time t
$\pi_i^h(t)$	$[0, S]$	Pointer label that saves the optimal next node
$\tau_{ij}^k(t)$	$[0, S]$	The predicted link travel time between nodes i and j at time t. k represents an index for the different link travel times possible
d		The destination node
$D(i,h)$		The set of downstream nodes of node i while coming from node h

A.2. ARSCR Algorithm

A list of variables, their domains and their descriptions for the ARSCR algorithm. Duplicate variables with the ARSC algorithm are shown in Appendix A.1.

Variable	Domain	Description
$O_i^h(t)$	$[0, S]$	The objective value calculated by the objective function using the travel time distribution $\Lambda_i^h(t)$ and $Pr\left(\Lambda_i^h(t)\right)$ or $H_{ij}^h(t)$ and $Pr\left(H_{ij}^h(t)\right)$.
$Pr\left(H_{ij}^h(t) = \lambda\right)$	$[0, S]$	The probability on the travel time λ which is part of the set of all travel times to the destination, while departing from node i at time t and coming from node h and going to node j, $H_{ij}^h(t)$.
$Pr\left(\Lambda_i^h(t) = \lambda\right)$	$[0, S]$	The probability on the travel time λ which is part of the set of all travel times to the destination, while departing from node i at time t and coming from node h, $\Lambda_i^h(t)$.

A.3. FPP Algorithm

A list of variables, their domains and their descriptions for the FPP algorithm. Duplicate variables from the ARSC(R) algorithm are listed in Appendix A.1 and Appendix A.2.

Variable	Domain	Description
H_{ij}		All possible nodes h, that can be used to travel to node i and then to node j.
$K_{ij}(t)$		The set of all possible link travel times k when departing from node i at time t and going to node j.
l		Number of time steps the vehicle has to wait for a red traffic light before it turns green.
$L_{hij}(t)$		The set of all possible waiting times l when arriving at the traffic light at node i at time t while coming from node h and going to node j.
$Pr(K_{ij}(t) = k)$	$[0, S]$	Probability on the link travel time k while departing from node i at time t and going towards node j.
$Pr(L_{hij}(t) = l)$	$[0, S]$	Probability of the traffic light at node i being green when arriving from node h and going towards node j at l time steps after the time t.
R		The set of all possible routeplans r.
$X^r_{i,j}$		The travel time distribution from the departure node to node j while arriving at node j from node i and following routeplan r.

A.4. Traffic light prediction module

A list of variables, their domains and their descriptions for the traffic light prediction module.

Variable	Domain	Description
$B_b(t)$	$[0, S]$	The block that is green, which contains all the signalgroups that are green.
$n^b_m(t)$	$[0, S]$	Number of time steps direction m of block b is green.
I		Maximum number of states saved at a time.
l		Number of time steps the vehicle has to wait for a red traffic light before it turns green.
$L_{hij}(t)$	$[0, S]$	The set of all possible waiting times l when arriving at the traffic light at node i at time t while coming from node h and going to node j.
m		Total number of directions in each block.
$P_{s_i s'_j}$		The probability of transfering from state s_i into s_j.
$Pr(L_{hij}(t) = l)$	$[0, S]$	Probability of the traffic light at node i being green when arriving from node h and going towards node j at l time steps after the time t.
$\mathbb{P}$		The probability of transitioning from the state S_t toward S_{t+1}.
s_i	$[0, I]$	The state of the traffic light with index i.
s'_j	$[0, I]$	The next state of the traffic light with index j.
SG	$[0, SG_{max}]$	The signalgroup that is currently being calculated for.
SG_{max}		The number of signalgroups in this intersection.
t		Current time step that the algorithm is calculating the values for.
T		The prediction horizon in time steps.
$Y(t, s_i)$	$t \in [0, T]$ $i \in [0, I]$	The probability of state s_i happening at time step t.
$X(t, s_i)$	$t \in [0, T]$ $i \in [0, I]$	The probability of state s_i happening at time step t, different from Y since this does not take a red traffic light into account once it got green once.
$Z^{hi}_j(t)$	$[0, T]$	Probability of the traffic light at node i being green when arriving from node h and going towards node j at time t, sometimes the h, i and j are replaced by SG.
$Z^l_{hij}(t)$	$[0, T]$	Probability of the traffic light at node i being green when arriving from node h and going towards node j at l time steps after the time t, sometimes the h, i and j are replaced by SG.