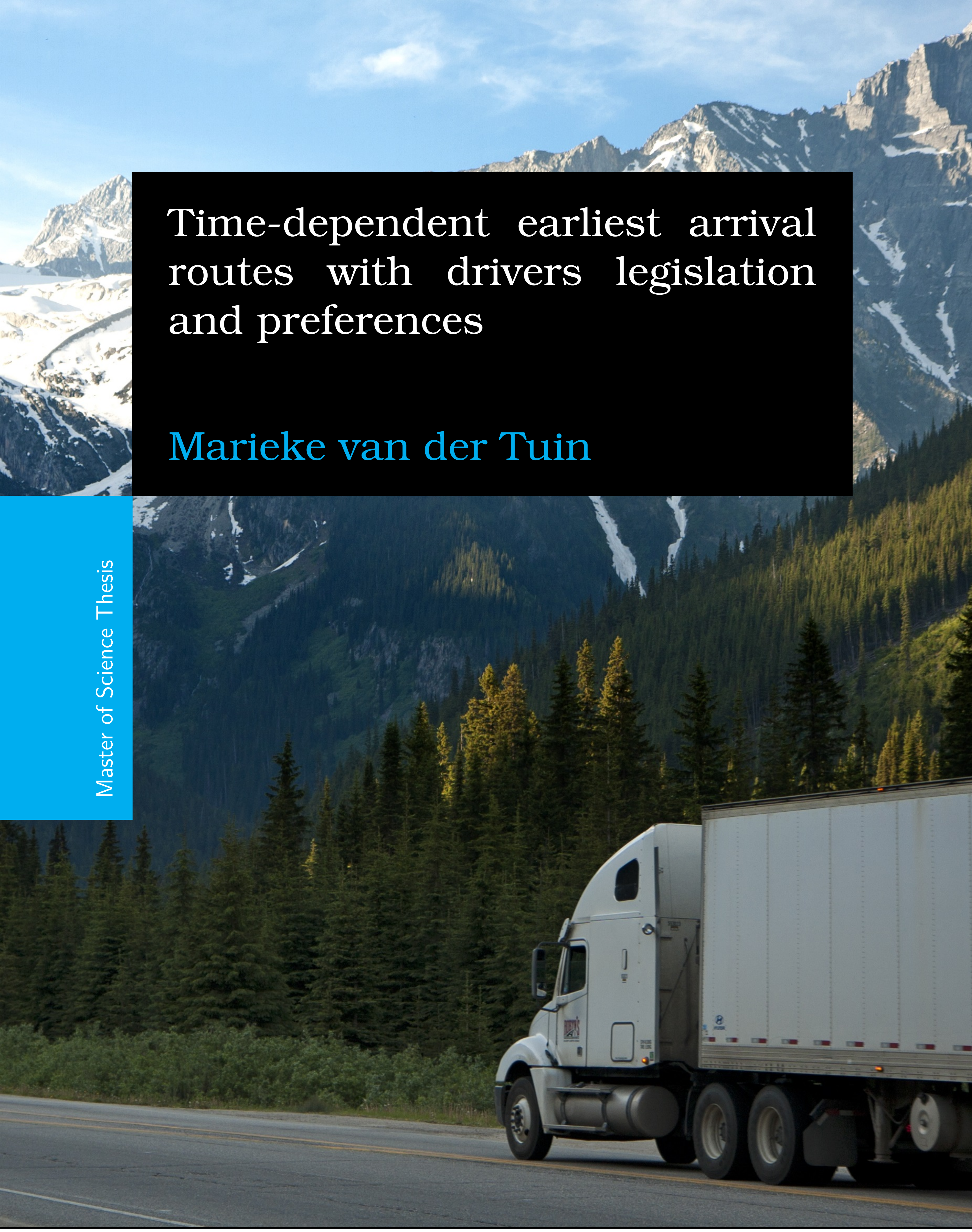


Time-dependent earliest arrival routes with drivers legislation and preferences

Marieke van der Tuin

Master of Science Thesis

Individual Double Degree Programme
Computer Science - Software Technology
Transport, Infrastructue and Logistics

Routes of trucks are highly affected by night and weekend bans in the Alps. Such routes give opportunities for planning breaks optimally during waiting time, as well as incorporating preferences by possibly avoiding toll roads and steep road grades.

Time-dependent earliest arrival routes with drivers legislation and preferences

Master of Science Thesis

For the degree of Master of Science in Computer Science
and Master of Science in Transport, Infrastructure and Logistics
at Delft University of Technology

Marieke van der Tuin

June 22, 2017

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Faculty of Civil Engineering and Geosciences (CEG)
Faculty of Technoloty, Policy and Management (TPM)
·
Delft University of Technology

The work in this thesis was supported by ORTEC. Their cooperation is hereby gratefully acknowledged.

Abstract

Freight transporters use software kits to plan the routes for their trucks. Breaks required by the European drivers legislation are nowadays planned as late as possible. However, in some cases it is beneficial to plan these mandatory breaks during waiting time, such as truck driving bans. In this thesis this problem is addressed by computing earliest arrival routes with optimal break planning. The problem is formulated and optimally solved as a variation of Dijkstra's algorithm. The slow Dijkstra computation times of several seconds per route are improved using time-dependent contraction hierachies, which enable a query time of several milliseconds per route while the solution quality remains good. For two days of driving with a night rest in between, 17% of the analysed routes improves with optimal break scheduling, resulting in an average improvement of 5 hours of driving time.

If the taking of breaks is additionally restricted to parking lots, the influence on the arrival time is on average increased with only 3 minutes. However, 5% of the considered routes are not feasible any more due to absence of truck parking lots along the planned route. Another 15 % of the routes face large changes in roads that should be taken.

The planned routes are all optimized for having the earliest arrival time. However, the freight company's objective also concerns fuel optimized driving or providing reliable arrival times to the customer. This thesis analyses such preferences and combines them into a route planner that incorporates results of stated and revealed choice experiments. It is shown that generally toll avoidance and congestion avoidance have most influence on the route choice and arrival time. On average differences are small, but for some routes this leads to changes of up to 15 % in travel time.

This thesis analyses the problem of route planning from two perspectives: algorithmic and behavioural. Several observations are made: algorithms are not suitable for obtaining the best route instead of the fastest, and the output of behavioural research cannot be used directly in practice to compute preferred routes. Effort could be made in future to integrate both research communities such that algorithms are able to reflect what a freight transporter actually wants.

Preface

In September 2014, I started with the master Software Technology. Although very interesting, I missed the practical application of the algorithms I learned during the courses. Therefore, I decided to follow some additional courses of Transport & Planning at the Civil Engineering faculty. It was John Baggen giving me the idea of not only following some courses, but a complete second master: Transport, Infrastructure and Logistics. By creating an Individual Double Degree programme, I was able to combine the thesis projects into one and complete both masters after three years of studying. It is a pleasure that the TU Delft and the responsible examination committees offered me this challenging opportunity.

The combined thesis project took place at ORTEC. I'm happy that Bas den Heijer and Veit Batz were willing to not only support the algorithmic research, but also the behavioural point of view of route planning. Furthermore, I would like to thank them for teaching me on contraction hierarchies, proofs, efficient programming and other scientifically relevant topics. Working at ORTEC was a great pleasure, especially the walks during lunch time and trips after work.

In addition, I would like to thank my supervisors of the TU Delft, especially for their willingness to deal with the undefined process of a Double Degree thesis: Mathijs de Weerdt, Victor Knoop, Sander van Cranenburgh and Caspar Chorus. This thesis would not be as succesful without their enthusiasm, ideas, feedback and criticism during the frequent meetings we had.

Furthermore, I would like to thank my parents who gave me the possibility to study both masters and always supported me during my studies. Finally, many thanks to my beloved Pim, who was able to cheer me up at the moments I needed it the most.

Marieke van der Tuin
Delft, June 2017

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Freight transporters use software tools to plan the routes for their trucks. Breaks required by the European drivers legislation are nowadays not incorporated in the planning process, but inserted later to reduce computation time. However, in some cases travel time can be reduced by planning these mandatory breaks optimally. This thesis addresses this problem by introducing optimal break planning algorithms. Not only an optimal break location is computed, these are also restricted to parking lots, disallowing breaks at the shoulder.

Besides the suboptimal planning of breaks, routes are currently planned to arrive at the destination as fast as possible. However, truck companies and truck drivers are generally not only interested in travel time, but also in other factors such as toll road avoidance, fuel-efficient driving and the level of congestion. These preferences are analysed and incorporated in route planning algorithms.

1-1 Context

In Europe, lots of goods are transported using trucks. The routes of these trucks are planned using sophisticated software tools. However, a logistic company is not just interested in planning precise routes between a start and destination location. They have a large set of pick-up and delivery tasks to fulfil with a fleet of trucks. A good planning is essential to reduce costs. Creating such a planning is called a Vehicle Routing Problem (VRP). A visualisation of the VRP is shown in Figure 1-1. Computing travel times is one of the components of solving this problem.

To solve a VRP, first a many-to-many query is performed, such that all travel times between all pick-up and delivery locations are known. This is used as an input to determine the best assignment of trucks to orders and the best delivery sequence.

Due to computing routes from and to each of the locations, large running times are not acceptable: with only 1000 locations this results in a million start-destination

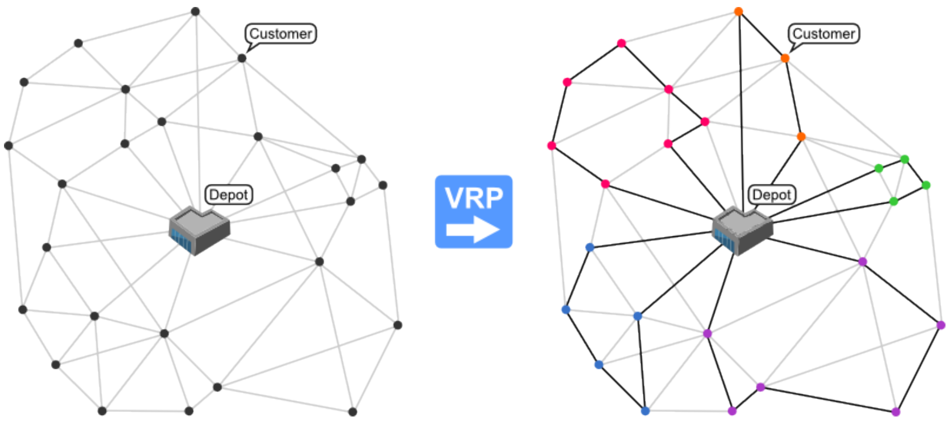


Figure 1-1: Example of a vehicle routing problem instance (left) and solution (right) with 5 available trucks at the depot

queries. With a running time of 1 second per start-destination query, this gives a total computation time of more than 10 days. In the competitive world of freight deliveries and last-minute orders one would only accept solutions being computable within a few hours.

Therefore, lots of effort is made by the algorithmic community to optimize such shortest path queries and create special many-to-many algorithms. The fastest recent algorithms are able to compute an optimal route in a continental size network in significantly less than a microsecond, in contrary to the several seconds that early algorithms (like the well-known Dijkstra's algorithm [19]) need for performing such queries [1].

To speed-up computation times, the problem of computing shortest-paths is simplified by currently used VRP-solving software. Breaks as required by the European Drivers legislation [22] to avoid fatigueness are often not included in route planning: only 22 % of the current research on VRP solvers included drivers legislation in some way [34]. Most of them set constraints, such that breaks are only possible at the location of a customer when already performing a pick-up or delivery task.

Taking breaks at a customer is not possible if long-haul routes are assumed, where a route from one customer to the next takes more than one day of driving. In these cases, a break is automatically inserted as late as possible.

The insertion of a breaks might cause influences in the travel time, mainly due to road bans: specific time periods at which it is not possible to drive with a truck through some countries in Europe. Research on three specific case studies showed that the presence of road bans only cause a 4% increase in total costs, due to the availability of alternative routes and the planning of breaks during such road bans [53]. However, this research assumes that the route planner is able to select the best route out of multiple viable options. If the planner just uses the results of the currently used software, this is not the case. In one of the case studies of the report [53], a suboptimal route choice leads to an increase in costs of 15%! This can be even larger if looking at specifically worse examples. For example, in Germany the roads are blocked for trucks on Sunday between 0:00 and 22:00. If due to the insertion of a mandatory night rest in Germany the country is not left before Sunday 0:00, this implies an additional waiting time – and thus delay – of 22 hours before the truck can drive further on toward the destination.

Besides neglecting breaks, the currently used shortest path algorithms just assume that the earliest arriving route is the best route. However, this is not always the case. A logistics company typically wants to make as much profit as possible. Routing decisions made can reflect such policies of reducing costs. In some cases, it is possible to skip a toll road for the cost of only limited additional travel time. Furthermore, by choosing for less congested routes, it is possible to predict reliable arrival times, which is seen as favourable by the customer. Such route preferences are sometimes incorporated in navigation software, but not in a sophisticated manner. For example, in most navigation systems it is possible to avoid toll roads. However, it is questionable whether one would like to take a detour of 1 hour just to save 1 euro of toll costs.

Preferences of logistics companies and their truck drivers are analysed in multiple experiments. The results of these experiments are currently not used to compute routes that are not just the fastest, but the best. It is also unknown what the influence on the arrival time is of incorporating such preferences.

1-2 Objective and research questions

The previous section showed that the influence of road blocks on total costs are minimal due to the possibilities of choosing alternative routes and efficient planning of breaks. However, it is assumed that a planner is able to select the best route himself. The goal of this thesis is to automate this process, by incorporating optimal break planning in shortest path algorithms. To show the urgency of including break planning in the currently used software, the influences in arrival times are important. Since there is a need for fast computation of solutions for the Vehicle Routing Problem (VRP), it is advisable if the optimal planning of breaks can be computed efficiently.

Furthermore, algorithms computing shortest paths do not only neglect breaks, but also do not consider the objective of the logistics company or truck driver. The route with the earliest arrival time is computed – independent of whether this is what the company actually wants. Preferences of avoiding toll roads, fuel-efficient driving or congestion avoidance are not taken into account when planning routes, whereas these might have their effect on the arrival time – and thus on the optimal delivery sequence resulting from the solution of the VRP.

Therefore, the objective of this thesis is to gain insight in the influence on arrival times if incorporating optimal planning of breaks and preferences, and to compute such routes efficiently. This leads to four research questions which are presented below, followed by the method how each of the questions is answered.

1. What is the influence on arrival time if breaks are planned optimally?

To answer this research question, first the problem is defined mathematically. This leads to several problem variations. This thesis addresses two of them: *RoadBlock-OneBreak* and *Parking*. Both assume that the only time-dependent information in a road network are the truck driving bans, and that only one break is allowed during the trip. The second variation extends this with the notion of parking lots: taking a break is restricted to parking lots only. Arrival times are compared to results of shortest path algorithms currently used by industry.

2. How can these routes be computed efficiently?

The context of route planning (that is, VRPs) gives importance to fast computation times. Therefore, heuristic algorithms are applied on contraction hierarchies to provide fast computation times, such that the algorithms become usable if it is needed to compute many routes.

3. What aspects identify a preferred route of a freight company?

As stated before, currently all routes are optimized such that they provide the earliest arrival time. This thesis analyses the actual objective of logistics companies and truck drivers relevant for route planning, for example avoidance of toll roads.

4. What is the influence on the arrival time if these aspects are used to compute routes?

To incorporate such preferences on route planning, first a methodology is given to use the results of a choice experiment to compute preferred routes. Next, the obtained preferences are used to plan routes and compared to the fastest routes.

1-2-1 Scope

This research is limited to route planning for trucks only. Shortest path computation is considered, which only looks into planning a route from start to destination. The focus is on routes having long distances which typically require breaks or rests during the drive. It is assumed that truck drivers follow the advised route strictly.

The routes are planned in Europe, using detailed map data of HERE [29]. The data is extended with information on road blocks, taken from governmental websites. Only road blocks are considered that are valid the entire year and on all highways, i.e. no blocks of a particular road or only valid during summer season are considered. Furthermore, it is assumed that all travel times are deterministic: no stochastic elements are taken into account. Also no real-time traffic information is used.

Since the routes are planned in Europe, the European Drivers Legislation is used, to be specific, Regulation No 561/2006 [22]. Besides that, there is a Directive 2002/15/EC that restricts the drivers' working hours which is implemented in national laws by most of the countries within the European Union. The Regulation specifies the driving itself, whether the Directive also takes other work hours into account such as administration and (un)loading. In this thesis only the Regulation is used. It should be noted that thresholds of maximum consecutive driving time and minimum break time can easily be adjusted, such that the presented algorithms can also be used for other regulations such as the Hours-of-Service regulation in the United States [24].

Concerning preferences, only stated and revealed choice data obtained from earlier experiments is used and no new experiments or surveys are conducted.

1-3 Related literature

The optimal break planning problem is a variation of a shortest path problem. A shortest path problem finds the path from start to destination for a given departure time, in such a way that the travel time is minimized. The shortest path problem can be solved using several algorithms. In this section, an overview is given of such shortest path algorithms. First the non time-dependent algorithms are discussed: algorithms that assume constant travel times along the edges. Section 1-3-2 explains the relevant time-dependent shortest path algorithms. Furthermore, algorithms are discussed that especially focus on implementing drivers legislation and incorporating preferences (see Sections 1-3-3 and 1-3-4). This leads to a conclusion which motivates the choice for the algorithms that are used in this thesis.

1-3-1 Shortest path algorithms

Computing an optimal route in a transportation network between a start and a destination node is a well-researched topic. Computing such a path is called a query. One can speed-up the querying by first applying a preprocessing operation independent of start and destination nodes, that is, shrinking the road network in such a way that optimal paths are maintained. In this section an overview is given of the most important algorithms with and without a preprocessing step, based on a review of Sommer [45].

The algorithm of Dijkstra [19] is one of the earliest querying algorithms. For general graphs with constant travel times, no faster algorithm has been developed since. However, there are some techniques that provide speed-ups on specific kind of graphs. One of such speed-up techniques is bidirectional Dijkstra search. In this algorithm, two queries are performed at the same time: one starting at the start node, one at the destination node. When they meet, the shortest path is found.

Other speed-up techniques use preprocessing and are classified in two groups: goal-directed and hierarchical approaches. An example of a goal-directed speed-up technique is A^*. This technique is based on the fact that it is possible to calculate estimates on the travel time in a road network, without knowing the exact structure of the network, by using geodesic distances. The A^* algorithm uses these estimations to direct the search toward the destination. Another goal-directed technique is SHARC [8]. This method partitions the network into regions and computes a backward Dijkstra search on the nodes at the borders of the regions. For such paths, shortcuts are added. During querying only edges are explored that lead toward the region of the destination.

Hierarchical speed-up techniques make use of the hierarchical nature of the road network: a typical route starts at a local road, travels via some B-road to the highway, probably switches to some other highway, and then gets back via the off-ramp to some local road. One of these hierarchical algorithms is Highway Hierarchies [43]. This technique uses a bidirectional Dijkstra search, which is only allowed to go to a higher level in the hierarchy of roads. Shortcuts are used to bypass several nodes, which gives an additional speed-up. Contraction hierarchies [27] also work with such a bidirectional

Dijkstra algorithm, but assigns a unique level to each node in the network. During preprocessing, each node is virtually removed and shortcut edges are added to maintain shortest paths that run through this node.

Another observation that is made in a road network is that drivers usually leave their current location via one of only a few access routes to a small set of landmarks called transit nodes. These landmarks are interconnected by a network relevant for long-distance travelling. The Transit Node Routing algorithm [5] [6] uses this observation and computes all shortest paths to landmarks nearby, and the shortest paths between all landmarks. Although the preprocessing requires lots of time, the querying only entails a few look-ups in the precomputed travel time tables.

1-3-2 Time-dependent shortest path algorithms

This thesis considers shortest paths in a time-dependent network, implying that travel times can vary for different departure times, requiring time-dependent shortest path algorithms. These algorithms are a not so well-researched topic compared to their non-time-dependent variants. This section gives an overview of the most important of them, based on a review of Delling [18].

A time-dependent shortest path query is generally hard to solve. One of the conditions that still make the problem polynomially solvable is the FIFO (First In, First Out) property of the network [16]. This means that departing later never results in an earlier arrival time. This is generally the case in road networks. Therefore, nearly all time-dependent shortest path algorithms assume the FIFO property. Using this property, also Dijkstra's algorithm can be used to solve the time-dependent shortest path problem by only replacing constant travel times with variable ones.

The computation of the time-dependent shortest paths be speed-up using preprocessing. One of such techniques is the time-dependent contraction hierarchies algorithm [7]. Like the non time-dependent variant, nodes are removed from the road network and shortest paths are remained by adding shortcuts. However, there might be different shortest paths for different departure times, which should all be maintained. This is done using special merging and linking procedures. This preprocessing method results in the currently fastest querying time among other time-dependent earliest arrival route techniques.

1-3-3 Time-dependent shortest path algorithms with drivers legislation

In the previous sections algorithms to solve the (time-dependent) shortest path problem are discussed in detail. However, none of these algorithms deal with drivers legislation. Only one study is found that deals with the optimal planning of breaks in a shortest path problem: the bachelor thesis of Bräuer [10] (written in German). He uses contraction hierarchies in such a way that parking lots on which a break can be taken are never contracted. This results in a preprocessed graph in which every query results in a path via a parking lot. Only paths adhering to the drivers legislation rule are considered to be valid. However, to select viable parking lots approximation algorithms are used, and the inclusion of road bans or congestion is not taken into account.

A related problem which is analysed for time-dependent road networks with drivers legislation is the Vehicle Routing Problem (VRP). Kok [33] assumes that all travel times between customer locations are known beforehand for every departure time. Breaks are only allowed at the customer, before or after unloading. If the travel times are too long to be able to drive the segment without breaks, additional fake customers are added to model a parking lot. It is assumed that the information on potential parking places is known beforehand. Using a dynamic programming heuristic, Kok gives a solution to the VRP on a time-dependent network with drivers legislation.

Although this is a solution method for the VRP for a time-dependent network with drivers legislation, it does not use the time-dependency in the network to optimally plan breaks. Parking lots are added as fake customers if it is not possible to reach the destination within the shift threshold, but it is not determined whether the concerned parking lots have an optimal location. It is not possible to add multiple parking lots and let the algorithm decide which one is the best, since all customers – and thus also all parking lots – need to be served.

1-3-4 Shortest path algorithms with preferences

Incorporating preferences in route planning can be seen as a multicriteria shortest path problem: an optimal path provides a balance between multiple objectives. An overview of such algorithms is given by Tarapata [48]. When only looking into algorithms that are suitable for deterministic graphs, there are two methods: using modified edge costs and using a Pareto optimal search.

Modified edge costs implies representing the preferences as a cost for travelling a road segment. For example, Eiger [21] uses such modified costs where preferences are represented with a linear-additive formula. The problem is then solved by running a Dijkstra algorithm on the graph with these modified edge weights. Another algorithm using modified edge costs is implemented in the TRIP route planner [35]. They assume that drivers prefer routes that they have taken before. Using the historical GPS data, driver specific travel time discounts are assigned to each previously traversed road segment. These modified travel costs are then used in a shortest path algorithm such as Dijkstra.

The other method of computing shortest paths while incorporating preferences uses a Pareto optimal search, a variation of Dijkstra's algorithm. This algorithm maintains multiple paths, where each of the paths does not dominate the other on all used criteria. This leads to a lot of paths, thus long running times if multiple criteria are considered for computing the paths.

1-3-5 Choice of algorithm for this thesis

In the previous sections, a review is given on shortest path algorithms, both with constant travel costs and time-dependent edge weights, as well as with drivers legislation and preferences. Dijkstra's algorithm can be used to solve nearly all mentioned variations. The algorithm is easy to adapt and is furthermore easily understandable. Therefore, it seems a good idea to include Dijkstra's algorithms as one of the algorithms in this thesis. However, a disadvantage of Dijkstra's algorithm is its long running times.

Since one of the goals of this thesis is to provide efficiently computable solutions, a second, faster algorithm needs to be considered as well. Contraction hierarchies seem to be best suitable. Not only is the algorithm suitable to use in a time-dependent case, it also provides the fastest query times currently known. Furthermore, the author of the time-dependent contraction hierarchies, Veit Batz, currently works at ORTEC, the company supporting the research for this thesis.

Therefore, this thesis uses the Dijkstra algorithm and time-dependent contraction hierarchies as a basis for the developed algorithms.

1-4 Main contributions

This thesis presents a method to incorporate truck drivers legislation considering road blocks, that is, optimal planning of breaks. An optimal Dijkstra-like algorithm is presented, as well as a heuristic method (Section 4-2). Using time-dependent contraction hierarchies a speed-up is achieved, while still remaining a very good solution quality (Section 4-3). After that, it is shown what the influence on the arrival time is, if such optimal break planning is considered. Next a method is presented to restrict taking breaks at parking lots, using an optimal Dijkstra algorithm (Section 5-2) and a heuristic method using contraction hierarchies (Section 5-3). The influence on arrival time and on route choices is given using simulations.

This thesis gives an overview on preferences of a truck company on route choice (Chapter 6). A methodology is presented to convert the results of a choice study to compute preferred routes (Section 7-1), followed by the simulations and sensitivity analysis on such routes. Finally observations are made and research suggestions are given to integrate the behavioural and algorithmic points of view of computing routes (Chapter 8).

1-5 Outline of thesis

In this chapter, the problem and thesis contributions are described. Below is an overview given of the contents of the remaining chapters of this thesis.

Chapter 2 gives the formal problem definition. The notation used throughout the thesis is introduced, as well as the specification of the several problem variations used in this thesis.

Chapter 3 provides algorithmic ingredients which are used throughout this thesis. These algorithms are Dijkstra's algorithm and contraction hierarchies. In the remainder of this thesis it is assumed that details as discussed in this chapter are known to the reader.

Chapter 4 analyses the influence on arrival time of optimal break planning by introducing optimal Dijkstra-like algorithms. Furthermore, the algorithms are optimized using time-dependent contraction hierarchies such that they become usable in practical applications.

Chapter 5 incorporates parking lots in the optimal planning of breaks. Optimal Dijkstra-like algorithms are presented and used to analyse the influence on the arrival time. To efficiently compute these routes, a heuristic method using time-dependent contraction hierarchies is used and its solution quality is obtained.

Chapter 6 gives insights in the objective of a logistic company and how this can be reflected in choices made when planning routes. This results in a list of attributes and corresponding parameters which can be used to compute the best route.

Chapter 7 gives a method to use the results of a choice study to compute preferred routes. These routes are compared to the fastest routes to obtain the influence on the arrival time.

Chapter 8 integrates the algorithmic and behavioural point of view of route planning as addressed in this thesis and gives an advice toward the research communities.

Chapter 9 are the conclusions of this thesis, summarizing the results on the research questions.

Chapter 10 reflects the consequences of the assumptions made in the scope (Section 1-2-1), and gives ideas for solving problem variations not discussed in this thesis.

Chapter 2

Problem Definition

One of the goals of this thesis is to find the fastest route in a time-dependent road network while adhering to the drivers legislation rules. Before proposing algorithms to solve the problem, it is mathematically defined in this chapter. Besides that, several variations of the problem are introduced. These are used to form several sub-problems that are simplifications of the problem. In the remainder of this thesis two of these sub-problems are addressed, giving insight in the influence on arrival time.

2-1 Problem

Given is a directed graph $G = (V, E)$ consisting of vertices (nodes) and edges. Each edge $e \in E$ is described as a pair of nodes (u, v). A *travel time function (TTF)* $f_e : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$, represents the travel time of an edge e for a given start time, see aslo Section 2-1-1.

A time-dependent earliest arrival route can be obtained by performing an *earliest arrival (EA) query*. An EA query results in the earliest arrival time $\tau_t \in \mathbb{R}$, given a start node $s \in V$, a destination node $t \in V$ and a departure time $\tau_0 \in \mathbb{R}$. The corresponding route in the road network G is represented by a path of nodes $P = \langle s, v_0, v_1, \ldots, v_k, t \rangle$.

The time-dependent earliest arrival route with drivers legislation is an extended version of the time-dependent earliest arrival route problem. The route should adhere to a set of driver legislation rules. Therefore, for each route a counter $\Delta_d \in \mathbb{R}_{\geq 0}$ is maintained, representing the consecutive driving time. This counter Δ_d may not exceed a certain threshold Δ_T, along with other restrictions that are explained in detail in Section 2-1-2. A break of duration Δ_b can be taken to reset Δ_d to 0.

A route from node s to node t is considered feasible if it can be driven without violating the drivers legislation rules. The optimal route is a feasible route with the earliest possible arrival time.

2-1-1 Travel time function

In this thesis the approach of Ichoua et al. [31] is followed for the definition of a *travel time function* (TTF). A TTF represents the travel time of a specific edge at a certain time and can be described by a *piecewise linear function* composed of straight-line sections.

A TTF f_e for an edge $e \in E$ can be composed by converting the speed limits of the edge e to resulting travel times, given the length of the edge. An example is shown in Figure 2-1. The speed limits shown in the left graph are used to compute the TTF which is shown in the right graph.

The TTFs used in this thesis fulfil the FIFO property (First In, First Out). This property ensures that it is not possible to arrive earlier if departing later, thus it holds that if $\tau \leq \tau'$, then $f(\tau) + \tau \leq f(\tau') + \tau'$. To fulfil the property, all line segments of the TTF have a slope of -1 or greater.

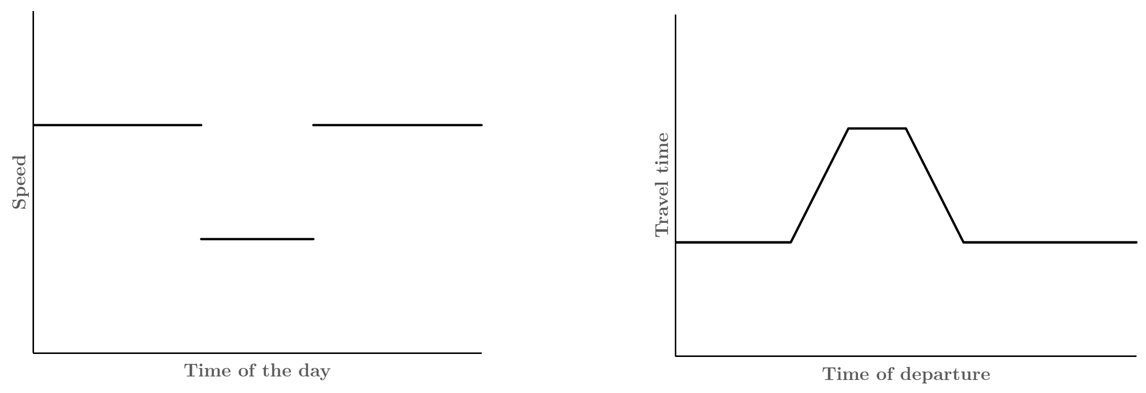


Figure 2-1: Example speed limits (left) and resulting travel time function (right) for a certain link

2-1-2 Drivers legislation

The general drivers legislation is defined as a certain consecutive driving time Δ_d may not exceed a threshold Δ_T. The driving time can be reset to 0 by taking a break of duration Δ_b. The EU regulation [22] defines several of these rules, mainly:

- **Non-stop driving time**: The non-stop driving period may not exceed the threshold of 4.5 hours. After 4.5 hours of driving, the driver must take a break period of at least 45 minutes. The break may also be split into a break of 15 minutes and a break of 30 minutes.
- **Daily driving time**: The daily driving time shall not exceed 9 hours. This may be extended to 10 hours twice a week. After each daily driving period, a daily rest of 11 hours should be held.

- **Weekly driving time**: The weekly driving time may not exceed 56 hours. This equals 2 days of driving 10 hours and 4 days of driving 9 hours. After each weekly driving period, a rest period should be taken of at least 45 hours. The driving time in two consecutive weeks may not exceed 90 hours.

2-2 Problem variations

Several variations are possible within the problem definition of the time-dependent earliest arrival route with drivers legislation. The time-dependency of the network, drivers legislation and parking lots can be simplified to create an easier solvable problem. Using these variations simplified sub-problems can be created. Solving the sub-problems gives insight in the best way to solve the complete version of the problem.

2-2-1 Time-dependency in the road network

Time-dependency of the edges is reflected by the travel time function f_e for each edge $e \in E$ as explained in Section 2-1-1. The travel time can be affected by two types of delay: road blocks and congestion. Only considering road blocks results in a special case of the general travel time function. Both road blocks and congestion are discussed in detail in the following sections.

Road block

A road block is a measure which can be taken by the government to reduce pollution or noise. At some countries in Europe, it is prohibited to drive during the night with a truck. Other countries introduced a driving ban during the weekends. Generally road blocks have a long duration: in Europe the minimum road ban duration is 7 hours. Therefore, the road blocks have a great effect on planning routes and should be taken into account.

The travel time function of a road block is a special case of a general travel time function as explained in Section 2-1-1. The road block can be described by its start and end time, where the end time is always later than the start time. The formal definition of a road block is as follows:

Definition 2.1 (Road Block). *A Road Block (*RB*) is an interval* $[\tau_{bs}, \tau_{be}]$*, where* $\tau_{bs} \in \mathbb{R}_{\geq 0}$ *is the start time of the road block and* $\tau_{be} \in \mathbb{R}_{\geq 0}$ *is the end time of the road block. Furthermore,* $\tau_{bs} < \tau_{be}$.

An example TTF of a road block on edge e is given in Figure 2-2. The edge has a *free flow travel time* of 1 hour. Between 12:00 and 14:00 the road is closed. Due to the road closure, the travel time increases instantly at 11:00 to 3 hours. For example, when starting to drive the edge at 11:30, one can drive for half an hour until 12:00, followed by 2 hours of waiting for the road block, followed by another half an hour of driving.

For arrival times between 12:00 till 14:00 the travel time decreases gradually: for every minute of arriving later, one has to wait one minute less. The slope of the segment between 12:00 and 14:00 in the graph therefore equals -1.

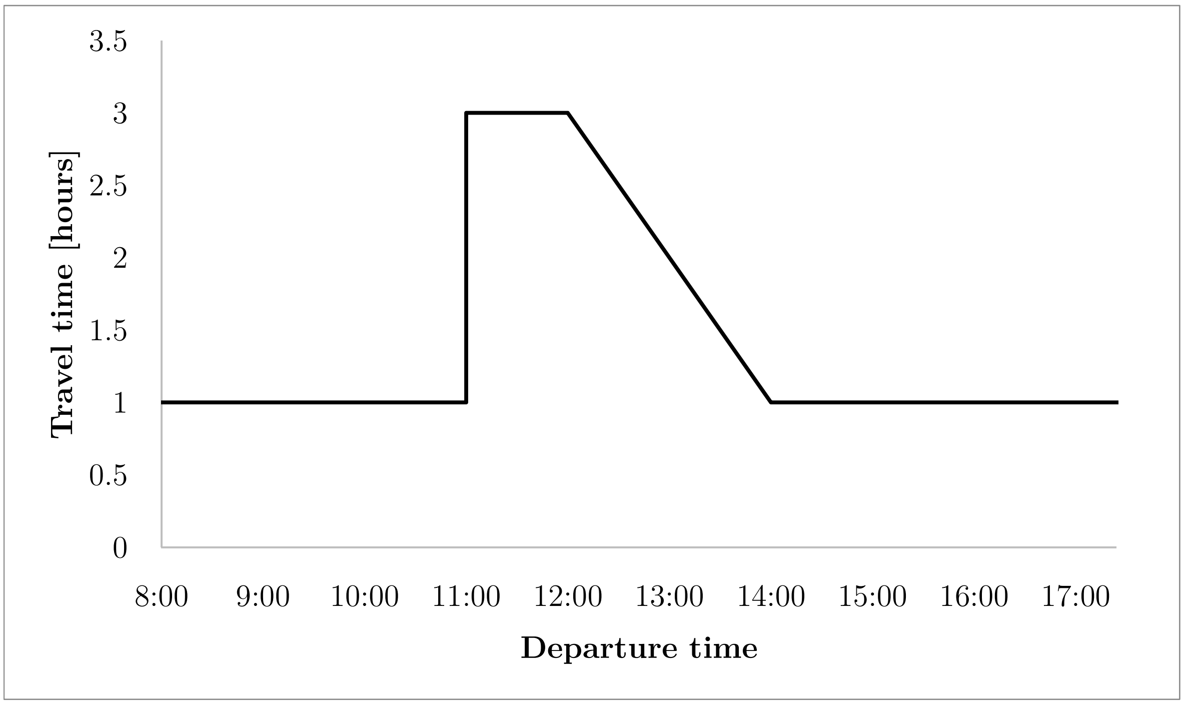


Figure 2-2: Example travel time function of a road segment having a free flow travel time of 1 hour and a road block between 12:00 and 14:00

Every edge can have multiple road blocks. For example in Switzerland: the highways are blocked for trucks every night and additionally on Sundays. The road blocks may not overlap: then they will be merged into one road block.

The travel time of an edge having one or more road blocks can be obtained by the road block information and the free flow travel time of the edge. Therefore, a Road Block Edge is introduced that contains all information to compute the travel time:

Definition 2.2 (Road Block Edge)**.** *A Road Block Edge (*RBE*) is a tuple* $\langle e, \mathrm{RBS}, \Delta_{\mathrm{FF}}\rangle$*, which describes an edge* $e \in E$ *having a sequence of road blocks* $\mathrm{RBS} = \langle \mathrm{RB}_0, \mathrm{RB}_1, \ldots, \mathrm{RB}_n\rangle$*.* Δ_{FF} *is the travel time of edge* e *when no road block is active. Furthermore, the road blocks in* RBS *are ordered and do not overlap, thus* $\tau_{be_i} < \tau_{bs_j} - \Delta_{\mathrm{FF}}$ *for any* $i < j$ *with* $\mathrm{RB}_i : [\tau_{bs_i}, \tau_{be_i}]$ *and* $\mathrm{RB}_j : [\tau_{bs_j}, \tau_{be_j}]$*.*

In order to calculate the exact travel time, it needs to be computed first which road block is *active* (i.e. which block start and end times are relevant). Remember from the example in Figure 2-2 that the time between $\tau_{bs} - \Delta_{\mathrm{FF}}$ (i.e. 11:00) and τ_{be} (i.e. 14:00) is different from the free flow travel time. The decision on the active road block can therefore be computed as follows:

$$f_e(\tau) = \begin{cases} f_{\text{RBE}}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\text{FF}}) & \text{if } \tau \in [\tau_{bs} - \Delta_{\text{FF}}, \tau_{be}] \\ & \forall \text{RB}_i \in \text{RBS with RB}_i = [\tau_{bs}, \tau_{be}] \\ \Delta_{\text{FF}} & \text{otherwise} \end{cases} \tag{2-1}$$

Using the information on the active road block, the actual travel time over an edge can be computed. The travel time function around a road block can have three different states: free flow, full waiting and partly waiting. Free flow travel time occurs if it is possible to drive the complete segment before the road block starts (i.e. $\tau < \tau_{bs} - \Delta_{\text{FF}}$), or after the block ends (i.e. $\tau \geq \tau_{be}$). If it is not possible to drive the complete road segment before the road block starts, one can stop driving halfway the road segment. Full waiting is then experienced followed by driving the remaining part. With full waiting for a block, it is meant that a traveller needs to wait from τ_{bs} till τ_{be}. Waiting partly for a road block occurs if the arrival time is later than the block start τ_{bs} (but before the block ends). The waiting is then experienced before actually driving. The resulting travel time function for a given road block is as follows:

$$f_{\text{RBE}}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\text{FF}}) = \begin{cases} \Delta_{\text{FF}} & \text{if } \tau < \tau_{bs} - \Delta_{\text{FF}} \\ & \text{or } \tau \geq \tau_{be} \\ \Delta_{\text{FF}} + \tau_{be} - \tau_{bs} & \text{if } \tau \geq \tau_{bs} - \Delta_{\text{FF}} \\ & \text{and } \tau < \tau_{bs} \\ \Delta_{\text{FF}} + \tau_{be} - \tau & \text{if } \tau \geq \tau_{bs} \\ & \text{and } \tau < \tau_{be} \end{cases} \tag{2-2}$$

Using Equation 2-1, the default travel time function f_e for an edge e can be redefined for edges with a road block. It is assumed that the set of road blocks RBS and a free flow travel time Δ_{FF} is given for every edge. This travel time function is a special case of the default travel time function, only allowing road blocks. This is not valid if looking into congestion, as explained in the next section.

Congestion

The other type of time-dependency is congestion. Traffic congestion is the result of a too high demand, reaching the capacity of the road. As a result, vehicles are driving at slower speeds and may even stand still for short periods of time. Congestion leads to a fluctuating travel time function, which does not allow modelling the travel time function as a special case such as road blocks.

Another difference between congestion and road blocks is the predictability of the travel time. Whereas road blocks are fully known beforehand, congestion can only partly be predicted (such as a daily rush hour in the afternoon), or not predicted at all (such as an accident).

2-2-2 Drivers legislation

There are three main types of drivers legislation rules, as explained in Section 2-1-2: non-stop, daily and weekly driving time. These are referred to as the *time horizon* of the problem. Varying the time horizon results in problems having different complexity to solve. Furthermore, there is some *flexibility* possible in the rules: freedom in splitting breaks or extending driving times. This is considered as another dimension of incorporating drivers legislation rules in route planning.

For time horizon, it is useless to only consider non-stop driving time. This would imply that it is not allowed to take a break and therefore does not give any insight in the problem of optimal break planning. Two rules remain: daily and weekly driving time. Daily driving time implies a time horizon of 9 hours of driving: two shifts of 4.5 hours, with a break of 45 minutes in between. Weekly driving time considers a time horizon of one week, including all night rests. Weekly driving time rules should always be considered in combination with daily driving time rules, leaving two relevant variations of the time horizon dimension.

With flexibility in the drivers legislation it is meant that there is some freedom within the drivers legislation rules. A truck driver might split a break of 45 minutes in two breaks of 15 and 30 minutes, under some special conditions. The daily driving time of 9 hours might be extended to 10 hours twice a week. This might give interesting opportunities in planning routes, but also creates a more complex problem to solve.

2-2-3 Locations for taking a break

The problem does not define the locations at which taking a break or rest is possible. In the current problem definition a break can be scheduled if needed and when favourable concerning possible congestion or road blocks, independent of the current location of the vehicle. Even parking at the shoulder is allowed.

A more realistic variation is to restrict these locations to parking lots only.

2-3 Overview of variations

The dimensions of the problem introduced in the previous section lead to several possible problem variations. Time-dependency can only include road blocks or also congestion. The time horizon can be one day or a week. The drivers legislation flexibility options can be excluded or included. Finally, the locations which are allowed to take a break can be restricted to only parking lots or not.

Considering all four dimensions and their two alternatives, 16 different problem variations can be created. These are not all useful to solve. Remember that the goal of the creation of the problem variations is to gain insight in how to solve the problem with all its facets. Therefore, it is useful to first solve the simplest problem possible, and after that solve problems that focus on each of the four dimensions.

The first identified sub-problem is variation *RoadBlock-OneBreak*, which deals only with road blocks, having a time horizon of one day, no drivers legislation flexibility options and no restriction on only taking breaks on parking lots. The following sub-problems focus on each of the identified dimensions: drivers legislation (*DriversLegislation*), parking lots (*Parking*) and time-dependency (*Congestion*). The full version of the problem (*Realistic*) includes all possibilities.

All sub-problems are shown in Table 2-1. Each of the rows indicate a problem variation, whereas the checked cells indicate what alternative within the problem dimension is active.

Table 2-1: Several possible problem variations

	Time-dependency		**Drivers legislation horizon**		**Drivers legislation flexibility**		**Restrictions for break locations**	
	Block	**Full**	**Day**	**Week**	**None**	**All**	**No**	**Yes**
RoadBlock-OneBreak	✓		✓		✓		✓	
DriversLegislation	✓			✓		✓	✓	
Parking	✓		✓		✓			✓
Congestion		✓	✓		✓		✓	
Realistic		✓		✓		✓		✓

This thesis first solves problem variation RoadBlock-OneBreak, and afterwards problem variation Parking. The discussion at the end of the thesis in Chapter 10 gives ideas and implications of dealing with the other problem variations.

2-4 Summary

The time-dependent earliest arrival problem with drivers legislation deals with finding the optimal route from start to destination at a certain start time. The roads in the network each have a certain travel time function, which give the resulting travel time given a certain departure time. Besides the general problem definition, several variations are defined in Section 2-3

The next chapter gives an overview of existing solutions to the earliest arrival problem which are used as a basis in the solutions of the problems later on. Chapter 4 gives additional insight and solution methods for the first problem variation: *RoadBlock-OneBreak*. The problem variation *Parking* is discussed in Chapter 5. The other problem variations are discussed briefly in Chapter 10.

Chapter 3

Algorithmic Ingredients

Algorithms to solve the optimal break planning problem as presented later on in this thesis are extensions of known algorithms. Therefore, this chapter discusses these earliest arrival route algorithms, that is, *Dijkstra's algorithm* and *contraction hierarchies*. For both algorithms variations with constant travel costs (Section 3-1) and time-dependent edge weights (Section 3-2) are discussed. In the remainder of the thesis it is assumed that knowledge on these algorithms is known to the reader.

3-1 Earliest arrival route algorithms with constant travel costs

For this thesis, algorithms are used that are based on time-dependent Dijkstra and time-dependent contraction hierarchies. These algorithms are extensions of their non-time-dependent versions. Therefore, the Dijkstra algorithm and contraction hierarchies for networks having constant edge weights are explained in this section. In Section 3-2 their time-dependent variants are discussed.

3-1-1 Dijkstra's algorithm

One of the classical algorithms for route planning was introduced by Dijkstra [19]. The algorithm computes all costs (travel times) for travelling from start node s to every node $u \in V$, including their corresponding shortest paths. For every node u a label with tentative travel time $d_u \in \mathbb{R}_{\geq 0}$ and a label with the tentative predecessor $p_u \in V$ is maintained. Initially, d_u is set to ∞ and p_u to $\perp$ for all nodes $u \in V$. The symbol $\perp$ means that the node does not have a predecessor yet. For the start node s, the travel time d_s is set to 0. Furthermore, every node has two possible states: *settled* or *unsettled*. At the start of the execution every node is unsettled.

During execution of the algorithm, nodes are settled one after another. Therefore, an unsettled node u is selected, which has minimal tentative costs $d[u]$ among all unsettled nodes. Next all outgoing edges $(u, v) \in E$ are relaxed, where $c_{(u,v)}$ are the travel costs of the edge between nodes u and v. Relaxing means that for each edge it is checked whether $d_u + c_{(u,v)}$ is smaller than d_v. If this is the case, tentative costs and predecessor information is updated, thus $d_v := d_u + c_{(u,v)}$ and $p_v := u$. When all outgoing edges of u are relaxed, the node is marked as settled and the next unsettled nodehaving minimal tentative costs is selected.

If node u is settled, the shortest path to u has been found by the algorithm, because u can only be settled if this node has the lowest tentative costs of all unsettled nodes. Since all edge weights are non-negative, this implies that no shorter route can be found toward u. Therefore, the execution of the algorithm can stop if destination node t has been settled. The complete path can then be retrieved using the predecessor information in p.

In an implementation of Dijkstra, the list of unsettled nodes u is usually maintained using a priority queue. In a priority queue nodes are kept in an increasing order according to their tentative costs. The unsettled node that is selected from the priority queue is the one having the minimum tentative costs d_u. The full pseudo code of Dijkstra's algorithm is shown in Algorithm 1.

Algorithm 1: Dijkstra

```
Result: Lowest travel time from $s$ to $t$
1  function dijkstra(s, t)
2      $d_u := \infty, p_u := \perp$ for all $u \in V$
3      $d_s := 0$
4      $Q := \emptyset : PriorityQueue$
5      $Q.insert(s, 0)$
6      while $Q \neq \emptyset$ do
7          $u := Q.deleteMin()$
8          if $u = t$ then
9              return $d_t$
10         foreach $(u, v) \in E$ do
11             if $d_u + c_{(u,v)} < d_v$ then
12                 $d_v := d_u + c_{(u,v)}$
13                 $p_v := u$
14                 $Q.insertOrUpdate(v, d_v)$
15     return $\infty$
```

The relaxing procedure requires $O(|E|)$ running time: each edge is relaxed at most once, since each node is settled at most once. Finding the next unsettled node with the minimum tentative costs requires $O(|V|)$ if using a simple array. This should be performed at most $|V|$ times to settle all nodes. Therefore, the running time of the algorithm is $O(|E| + |V|^2)$. Using a heap-based priority queue, this can be reduced to $O(|E| + |V| \log |V|)$. Since the average number of outgoing edges of a node in a road network is usually very small, the required running time for relaxing edges can be discarded, leaving a running time of $O(|V|\log(|V|))$.

Dijkstra's algorithm can be speed-up using various methods. First of all, it is possible to perform a *bidirectional search* [15]. In this speed-up technique, two instances of Dijkstra are run at the same time: a *forward search* starting from s and a *backward search* starting from t. When the searches meet each other, that is, a node has been settled by both forward and backward search, the search can be stopped and the complete path retrieved.

Another speed-up technique of Dijkstra's algorithm is *goal-directed search*, also known as A^*. This technique was introduced by Hart et al. [28] and tries to bias the search direction in such a way that it spreads out towards the destination node rather than toward all surrounding nodes. This can be achieved by using a potential function π_t. This function gives an optimistic estimation of the driving time toward the destination. The measure may never overestimate the actual driving time toward the destination. This can for example be achieved by taking the geodesic distance (the aerial distance) as the potential function. The travel time d_u with the added potential costs $\pi_t(u)$ is used to order the list of unsettled nodes in the priority queue. The quality of the A^* algorithm is fully dependent on the quality of the used potential function. In road networks, speed-ups might be obtained due to the 2D representation of the graph, which is not the case for fully random graphs.

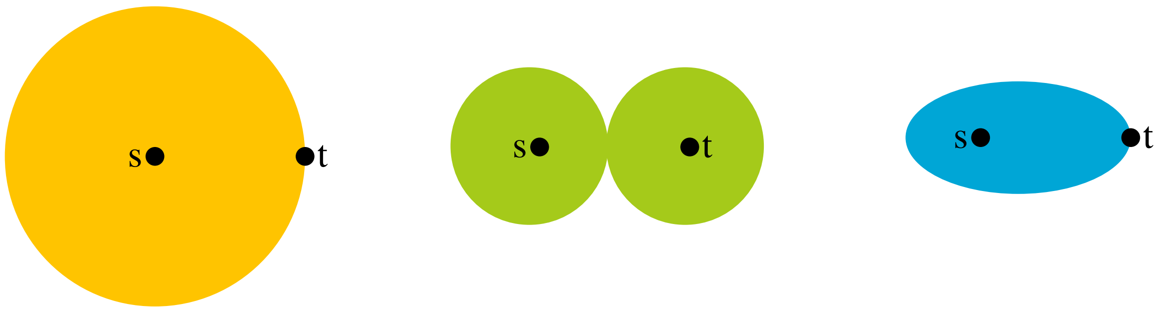


Figure 3-1: Illustration of settled nodes in the network by Dijkstra, bidirectional search and A^*

The search space of nodes for the different Dijkstra versions are shown in Figure 3-1. On the left it shows Dijkstra's algorithm which settles nodes from the start node based on the distance from the start node. The middle figure is an illustration of the bidirectional search where two searches are started: from s and t. The right figure shows the settling of nodes of the A^* algorithm.

3-1-2 Contraction Hierarchies

A completely different method of route planning makes use of the hierarchical structure of a road network. Some parts of the road network (e.g. highways) are more important than others (i.e. local roads) in the sense that optimal routes rather mainly lead through these special edges. Geisberger [27] used this idea to provide a speed-up technique for obtaining earliest arrival routes called *contraction hierarchies*.

This algorithm consists of two phases: *preprocessing* and *querying.* The preprocessing phase constructs a hierarchy of the network, independent of start and destination locations. The querying algorithm can be used to compute then earliest arrival path in a preprocessed graph for a given start and destination node.

Preprocessing

The preprocessing phase creates a hierarchical representation of the network. The hierarchy is constructed by successively removing one node after ordered from the least important to the most important node. Therefore, a node ordering is required beforehand. For example, a node ordering can be based on the number of outgoing edges. Different node orderings result in good and bad hierarchies. A good hierarchy is a *flat* and *sparse* graph, meaning that not much steps are required to reach the top of the graph from any node, and the number of outgoing edges per node is small.

The removal of a node x including its edges is called *contraction.* With the contraction of x from $G = (V, E)$, an overlay graph $G' = (V', E')$ is constructed, with $V' := V \setminus \{x\}$. The hierarchy is constructed in such a way that the minimum travel costs between the remaining nodes in V' is preserved. Therefore, it is checked whether the removal of a path $\langle u, x, v\rangle$ increases the travel time from u to v. If so, a *shortcut edge* (u, v) is added to E', in order to maintain the otherwise removed shortest path. If there is already an edge present between u and v in E', this edge is replaced.

An example contraction is shown in Figure 3-2. Node v_4 of graph G (top left) is contracted. Via node v_4 four *witness paths* are possible between the other nodes as shown in the top right graph. The path via edges (v_2, v_4) and (v_4, v_3) (red) has a cost of 2. In the remaining graph, no path exists between v_2 and v_3, thus the shortcut edge is added to the graph G' (bottom). Using the same argumentation, the blue shortcut edge is added. The green shortcut edge representing path via edges (v_3, v_4) and (v_4, v_5) has a travel cost of 4. There still exists a path in the remaining graph G' having the same length: via node V2. Therefore, the green shortcut edge is not added to graph G'. Also the purple shortcut edge does not provide a better shortest path and is therefore not added to G'. After the shortcut operation is finished, G' contains 6 nodes (instead of 7 in G) and 10 edges (instead of 13).

The preprocessing continues with subsequently contracting nodes until all nodes have been contracted. This results in $|V|$ overlay graphs in total, each having one node less. The edges between the overlay graphs are directed *upward* or *downward.* An upward edge is directed toward a more important node, whereas a downward edge is directed toward a less important node. This structure is used to form an *up-down path* in the querying phase of the algorithm.

Querying

In the querying phase an actual path from start s to destination t is computed. Therefore, a bidirectional Dijkstra search is performed on the hierarchy resulting from the

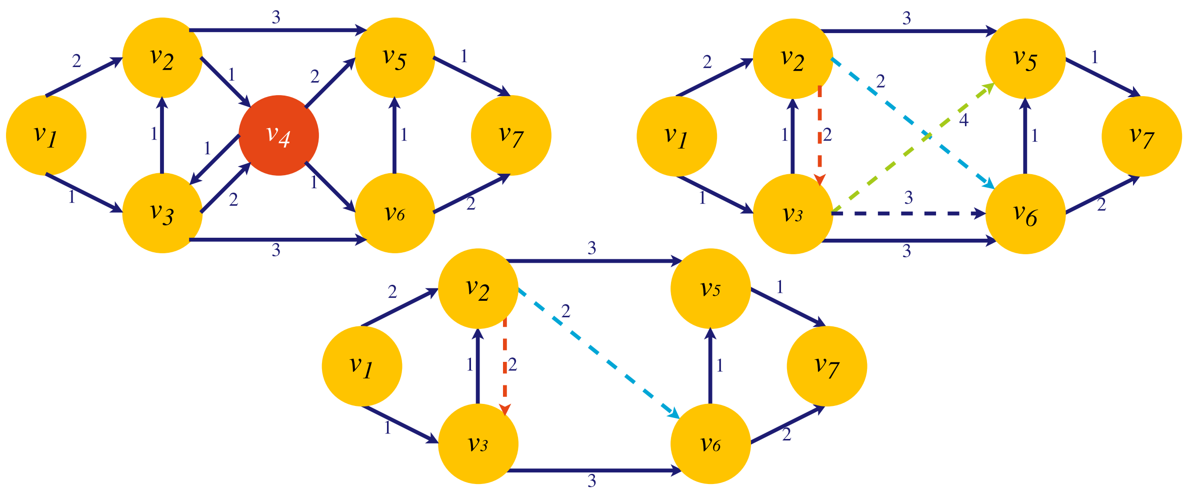


Figure 3-2: Example of contraction operation of node v_4. Left graph shows the original graph, with nodes $v_1, \ldots, v_7$ and edge weights indicated besides the edges. The right graph shows the graph including the witness paths that result from the removal of v_4.The bottom graph shows the remaining graph including shortcut edges.

preprocessing phase. The forward search starting at s only relaxes upward edges, that is, edges going toward more important nodes in the hierarchy. The backward search only relaxes downward edges, but in a reverse direction. Both searches meet each other at some node x higher up in the hierarchy. If they meet, an *up-down-path* with top node x can be retrieved. Multiple up-down-paths can be found. When the minimum value of both forward and backward priority queues exceed the travel time of the shortest up-down-path so far, the shortest path has been found, as proven by Geisberger [27].

Although the preprocessing time of contraction hierarchies is substantial (25 minutes for a network of Western Europe with about 18 million nodes on a Opteron 2.6GhZ processor), it enables fast querying of 0.2 ms per query. To compare: a query with Dijkstra's algorithm takes on average more than 5 seconds [7].

3-2 Time-dependent earliest arrival route algorithms

A time-dependent road network brings additional complexity for computing the earliest arrival route. Instead of a constant travel cost, a travel time function is defined for each edge, giving different travel times for different departure times as explained in Section 2-1-1. If the FIFO-property does not hold, it might be the case that it is beneficial to wait at some point during the route in order to get an earlier arrival travel time. Without the allowance of such arbitrary waiting, the problem becomes NP-hard [38] However, in this thesis only road networks having the FIFO-property are analysed. On these networks, the earliest arrival route can be computed in polynomial time as noticed by Dean et al. [16]. This property is assumed in the road network analysed in this thesis. If the FIFO-property does not hold, the problem becomes mostly NP-hard since the arrival time might be earlier by departing later or introducing waits [38].

In this section the (polynomial) Dijkstra and contraction hierarchies algorithm are discussed that solve the FIFO time-dependent earliest arrival route problem.

3-2-1 Time-dependent Dijktra

Dijkstra's algorithm can also be used in a time-dependent manner. Instead of storing the tentative travel costs in d_u for every $u \in V$, the tentative arrival time is stored. Therefore, the travel time function is used to compute the travel time, taking the tentative arrival time as an input. Mainly, an edge (u, v) is relaxed by comparing $d_u + f_{(u,v)}(d_u)$ with d_v.

If the travel time function $f(\tau)$ can be evaluated in constant time, the running time of the time-dependent Dijkstra algorithm is $O(|V| \log |V|)$, analogous to the non-time-dependent version.

The time-dependent version of Dijkstra can also be speed-up using A^* and bidirectional search techniques. A^* can be used directly: instead of maintaining the tentative arrival time at every node, now a tentative arrival time including the potential travel costs to the destination can be added. The potential function does not include any time-dependency and should be an optimistic estimation, such that the travel time is always underestimated.

Bidirectional Dijkstra cannot be implemented as easily as A^* in a time-dependent network. However, time-dependent bidirectional Dijkstra does provide the basis for the querying algorithm in time-dependent contraction hierarchies. Therefore, this algorithm is discussed in the next section.

3-2-2 Time-dependent bidirectional Dijkstra

The time-dependent bidirectional search is used as a basis for the time-dependent contraction hierarchy querying operations. However, it is not straight forward to implement a bidirectional search in a time-dependent network. In order to work correctly, the backward search should start at the node t with a certain arrival time. However, the arrival time at node t is not known yet: this should be computed by the algorithm! Both Nannicini et al. [37] and Batz [7] showed that it is still possible to implement the time-dependent bidirectional search. This modified version consists of four steps:

1. Bidirectional search consisting of forward search and backward interval search, stopped when both searches meet each other
2. Computation of a valid path by continuing forward search along nodes settled by the backward interval search.
3. Continuation of bidirectional search until the minimum value in the priority queues of both searches is larger than the travel time of the path computed in the previous step.

4. Computation of the optimal path by continuing the forward search along nodes settled by the backward interval search.

Each of the steps is discussed in detail in the following sections.

Step 1: Bidirectional search

The first step of the time-dependent bidirectional search consists of a normal forward search and a backward interval search. The forward search is identical to a time-dependent Dijkstra search starting from node s. The backward search is a little different due to the unknown arrival time. Nannicini et al. [37] used a backward lower bound search as backward search, whereas Batz [7] extended this to an interval search. This interval search computes a set of paths to t that may result in an earliest arrival time path, for any given departure time. Instead of a tentative cost label d_u, an interval of travel times $[q_u, r_u]$ is maintained for every node $u \in V$. This represents the minimum and maximum travel time from s to u, for any departure time. Each node u does not have one predecessor, but a set of them due to the multiple possible paths. For each relaxed edge (u, v) the minimum and maximum values of the travel time function $f_{(u,v)}$ are calculated and added to the current minimum and maximum values q_u and r_u. These new values are compared to the existing interval of v, that is, $[q_v, r_v]$. If the new minimum travel time is larger than the maximum value q_v, this edge is not of interest. If not, it might provide a good alternative route and therefore u is added to the list of predecessors. If the new maximum travel time is lower than the current minimum value r_v, all other predecessors for this edge can be discarded; these do not provide routes resulting in an earliest arrival time path for any departure time. Next the interval $[q_v, r_v]$ interval is updated with potentially found better new values. The complete backward interval search algorithm is shown in Algorithm 2.

If the forward and the backward interval searches meet each other at a node (that is, a node has been settled by both searches), both executions are stopped. The nodes settled by the backward interval search until that moment are added in a special set M and used in the next step in the time-dependent bidirectional search.

Step 2: Computation of a valid path

Step 2 starts if the forward and backward interval search have met each other at a certain node x. This implies that the algorithm of the previous step found a path from the start node s to the destination node t via some node x. The travel time of this path – not necessarily the optimal path – is retrieved by the continuation of the forward search along the nodes in set M. This set M contains the nodes that were settled by the backward interval search in step 1. If the destination node is reached, the arrival time of a path can be retrieved. The travel time of this path (i.e., arrival time at t minus the starting time at s) is called B. This travel time is used to bound the searches in the next step, such that it can be proven that the optimal path is found for sure.

Algorithm 2: Backward Interval Search

```
Data: $G = (V, E)$, destination node $t \in V$, travel time function $f$ for each edge $u \rightarrow_f v \in E$
Result:
function backwardIntervalSearch(t)
    $[q_u, r_u] := [\infty, \infty]$ for all $u \in V$
    $[q_t, r_t] := [0, 0]$
    $p_u := \emptyset$ for all $u \in V$
    $Q := \emptyset : PriorityQueue$                // Sorted on increasing lower bounds
    $Q.insert(t, 0)$
    while $Q \neq \emptyset$ do
        $u := Q.deleteMin()$
        foreach $(u, v) \in E$ do
            $[q_{new}, r_{new}] := [q_u + \min(f_{(u,v)}), r_u + \max(f_{(u,v)})]$
            if $q_{new} > r_v$ then continue
            if $r_{new} < q_v$ then $p_v := \emptyset$
            $p_v := \{u\} \cup p_v$
            if $q_{new} \geq q_v$ and $r_{new} \geq r_v$ then continue
            $[q_v, r_v] := [\min(q_v, q_{new}), \min(r_v, r_{new})]$
            $Q.insertOrUpdate(v, q_v)$
```

Step 3: Continuation of the bidirectional search

In the third step the forward and backward interval search which were stopped after step 1 are allowed to continue. This might result in additional nodes where forward and backward interval searches meet each other. If the travel time of the forward search and the minimum travel time of the backward interval search exceed the earlier computed bound B, it is known for sure that the shortest path has been found. At that moment, the searches can be stopped and the settled nodes of the backward interval search are put in the set M.

Step 4: Computation of optimal path

The actual optimal path is retrieved by the continuation of the forward search, while only allowing to settle nodes from set M. If it is finished, the earliest arrival time and corresponding path are found as proven by Nannicini et al. [37].

The time-dependent bidirectional Dijkstra algorithm as extended by Batz [7] with the backward interval search is used in the querying phase of the time-dependent contraction hierarchies.

3-2-3 Time-dependent Contraction Hierarchies

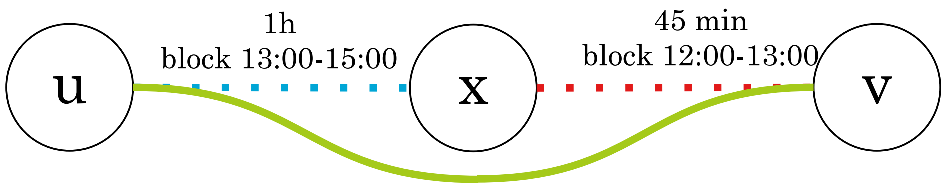
Batz [7] extended the contraction hierarchies (see Section 3-1-2) to use them with a road network including time-dependency. The problem with the original preprocessing operation is that the shortest paths might be different for different arrival times, and thus all such paths should be maintained. Therefore, two additional operations are defined that allow creation of shortcut edges: *linking* and *merging*. The used querying algorithm is a modified version of the earlier presented time-dependent bidirectional Dijkstra (see Section 3-2-2).

Preprocessing

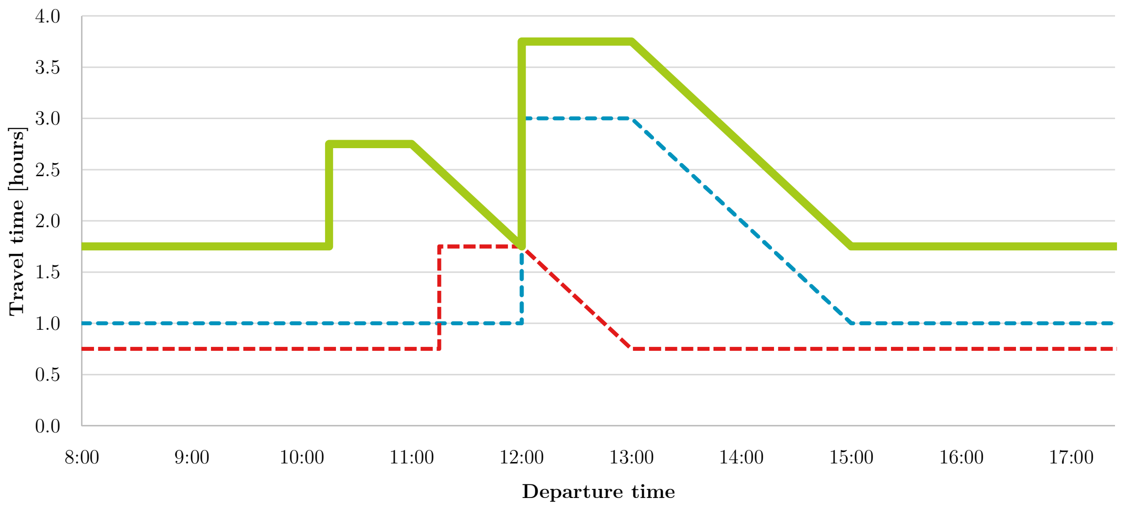
Identical to the non-time-dependent contraction hierarchies, nodes are subsequently contracted following a certain node order. If a node x is contracted, all its combinations of incident edges are transformed into potential shortcut paths. For example, assume there exists edges (u, x) and (x, v) with travel time functions $f_{(u,x)}$ and $f_{(x,v)}$. The travel time of first travelling edge (u, x) and then (x, v) is defined using the *linking* $\star$ operator as:

$$f_{(x,v)} \star f_{(u,x)}(\tau) = f_{(x,v)}(f_{(u,x)}(\tau) + \tau) \tag{3-1}$$

The linking operation gives correct results for every combination of travel time functions, as proven by Batz [7]. The exact details of the linking operation do not have to be explained to understand the idea behind it.



(a) Graph

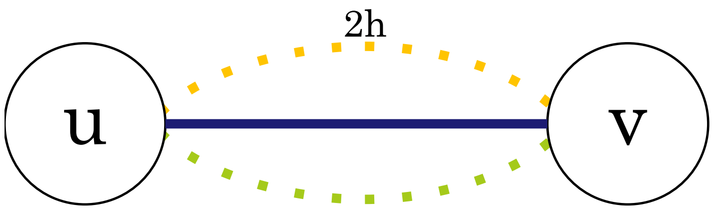


(b) Travel time functions

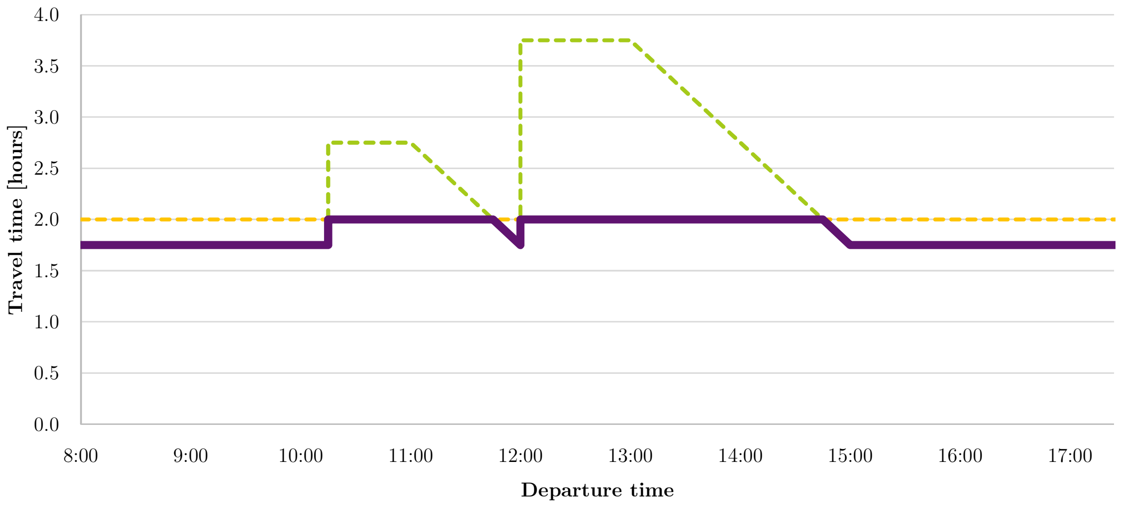
Figure 3-3: Example graph and accompanying travel time functions where node x is contracted and a shortcut path between u and v is formed by the linking operation

An example of a node x to be contracted is shown in Figure 3-3a. The edges (u, x) (blue) and (x, v) (red) both have a road block, at different moments of the day. They are linked to each other to form a shortcut path (green). The travel time functions of the original edges and the shortcut path are shown in Figure 3-3b. It can for example be seen that a departure time of 10:30 results in a travel time of 2 hours and 45 minutes. If looking into the graph itself, this is correct: a departure time of 10:30 from node u results in an arrival time of 11:30 at node x. Next, 30 minutes can be driven on the red edge before the block starts, which requires waiting till 13:00. After that the last 15 minutes should be travelled, resulting in an arrival time of 13:15 and thus indeed a travel time of 2 hours and 45 minutes.

After forming the combined paths using the linking operation, these paths are added as new shortcut edges or merged with an existing edge between nodes u and v. The merge operation is defined as the minimum of the travel time functions of the edges to be merged.



(a) Graph



(b) Travel time functions

Figure 3-4: Example graph and accompanying travel time functions where a shortcut path between u and v is merged with an existing path

Assume there existed already an edge between u and v of the previous example, having a constant travel time of 2 hours as shown in Figure 3-4a. The travel time functions are merged into the purple edge by taking the minimum of the travel time functions of the yellow and green edge. The result is shown in Figure 3-4b. It can be seen that the advice is to take the yellow route between 10:15 and 11:45 and between 12:15 and 14:45.

The complete preprocessing operation consists of contracting nodes one after each other, using a predefined node order. Travel time profiles queries between each of the neighbours of the contracted nodes are run to identify potential shortcuts. These shortcuts are created using the linking operation, followed by a merging operation if there already existed an edge between these nodes.

Querying

In the querying phase an actual path from start s to destination t for a given start time τ is computed using a modified version of the time-dependent bidirectional Dijkstra as explained in Section 3-2-2.

First of all, the forward search only runs in the upward graph, thus only considering edges that lead to higher important nodes – equal to the non time-dependent contraction hierarchies querying algorithm. Equivalently, the backward search runs in the reversed downward graph. The continuation of the forward search (step 4 in Section 3-2-2) is called *downward search* by Batz [7] and runs on the downward graph.

Besides these small modifications, some additional improvements have been applied. First of all, B (the travel time of the first found complete path, step 2 in Section 3-2-2) is not computed directly but is implicitly calculated by adding τ_u (the arrival time at node u in the forward search) to r_u (the upper bound of node u in the backward interval search). During the execution, the minimum value of B is maintained, by updating the value if a new node is settled by both searches. The forward and backward search are stopped once the minimum value of the priority queues of both forward and backward searches exceed B.

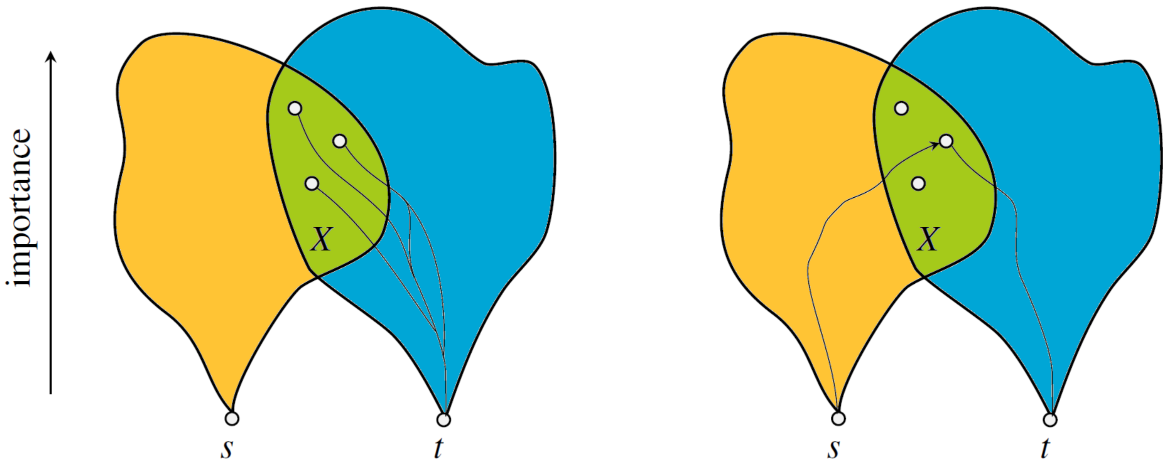


Figure 3-5: Visualisation of backward (left) and downward search (right) [7]

Furthermore, a set of candidate nodes X is maintained. A candidate node is a node which is settled by both forward and backward searches. Only nodes for which holds that $\tau_u + q_u \leq B$ are added to the set, which means that they are only added if the lower bound of the travel time from start to destination should not exceed the travel time of the shortest path found until now.

After the forward and backward interval searches are stopped, the downward search is run starting from the candidate nodes in set X. The downward search is identical to the forward search, except that it can only route through nodes that are reached by the backward search and runs in the downward graph.

A visualisation of the backward and downward searches is shown in Figure 3-5. The more important nodes are higher up in the graph. Set X (green) contains the settled nodes from the forward search (yellow) and backward interval search (blue). The downward search is run, which finds the optimal path from start s to destination t via a node in the candidate set X.

3-3 Summary

Dijkstra's algorithm and contraction hierarchies form the basis of the algorithms used later on in this thesis. Whereas Dijkstra is one of the first earliest arrival route algorithms and quite slow, its algorithm is easy to understand, prove and modify. Contraction hierarchies use a preprocessing step before querying the graph. Although preprocessing takes some time (25 minutes for a map of Western Europe), this operation only has to be performed once. After that fast querying on the preprocessed graph is possible (0.2 ms per query, or 1.5 ms per querying the time-dependent version). To compare: a run of Dijkstra's algorithm on this graph takes more than 5 seconds on average. Moreover, contraction hierarchies are proven to give optimal results and is thus not a heuristic method. In the next chapters, modified versions of the Dijkstra algorithm are used as a so called ground truth algorithm: slow but optimal. Variations on contraction hierarchies algorithms are used to obtain faster results that are not necessarily optimal.

Chapter 4

Time-dependent earliest arrival routes with drivers legislation: RoadBlock-OneBreak

The *RoadBlock-OneBreak* variation is the easiest solvable version of the optimal break planning problem. It includes time-dependent information on road blocks and computes routes having one break at maximum. The combination of drivers legislation and road blocks give an interesting opportunity: waiting for a road block does not count as driving time, but may count as break time.

The first two research questions are answered in this chapter, analysing the influence on the arrival time on optimal break planning and computing this result efficiently. First a recap of the problem variation is given in Section 4-1, as was previously defined in Chapter 2. After that a model of the problem is created in the form of a Stacked Break Graph, which is optimally solved and proven using Dijkstra-like algorithms. It is assumed that information on Dijkstra algorithms as presented in Chapter 3 is known to the reader. In Section 4-3 an algorithm is introduced to compute the solution efficiently using time-dependent contraction hierarchies. The currently used algorithm for break planning is given in Section 4-4. The results of this algorithm are compared with the optimal break planning algorithms in Section 4-5.

4-1 RoadBlock-OneBreak variation

The *RoadBlock-OneBreak* variation of the time-dependent earliest arrival route problem with drivers legislation considers road blocks as the only time-dependent information in a network. A road is closed or open: if it is closed, no one can drive the segment. If it is open the segment can be driven in *free flow travel time*. A road block is defined as an interval $RB = [\tau_{bs}, \tau_{be}]$, where τ_{bs} and τ_{be} represent the start and end times of the block. Each edge can have multiple road blocks. Such an edge is described as $RBE = \langle e, RBS, \Delta_{\text{FF}} \rangle$ with $e \in E$, RBS a set of consecutive road blocks, and Δ_{FF} the (constant) free flow travel time . The definition of a road block is explained in detail in Section 2-2-1.

The RoadBlock-OneBreak variation only deals with the drivers legislation valid on a single day. This entails that the maximum non-stop driving period Δ_d may not exceed the threshold Δ_T of 4.5 hours. After 4.5 hours of driving, a break must be taken of 45 minutes (Δ_b). For this problem variation only one break is allowed during the route, maximizing the total driving time to 9 hours.

Waiting for a road block does not count as driving time, but may count as break time. Furthermore, breaks may be planned everywhere in the RoadBlock-OneBreak problem variation. They are not restricted to a certain location (i.e. a parking lot). One may literally stand still at the middle of a highway to take a break.

The input of the problem consists of the start node s, destination node t and the departure time τ_0. Furthermore, the current driving time Δ_{d0} is required, indicating the status of truck driver, i.e. the time driven before starting with this route.

The goal of the RoadBlock-OneBreak problem is to find an *Optimal break planning path* which is defined as follows:

Definition 4.1 (Optimal break planning path)**.** *An optimal break planning path is a path from the start node s to the destination node t starting at departure time τ_0, having the earliest arrival time possible. The path adheres to the drivers legislation rules, such that the non-stop driving period Δ_d does not exceed the threshold Δ_T anywhere on the path.*

4-2 Stacked break graph

The RoadBlock-OneBreak variation is modelled using a modified graph. In order to represent the breaks, a copy of the graph is added to the original graph. Edges between these two graphs represent taking a break, as visualized in Figure 4-1.

For each edge (u_1, v_1) in the original graph, an edge (u_1, v_2) is added, with v_2 is the copy of the node v_1. The edge (u_1, v_2) represents travelling through the edge and taking a break somewhere on that edge. This graph is called a *Stacked Break Graph (SBG)* and is defined as follows:

Definition 4.2 (Stacked Break Graph)**.** *A Stacked Break Graph (SBG) is a graph $G = (V, E)$ where $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup EB$. $G_1 = (V_1, E_1)$ represents the original graph, $G_2 = (V_2, E_2)$ is an exact copy of this graph. EB consists of edges representing a break, which are possible at each edge in the graph. Thus $EB = \{(u_1, v_2) | \forall (u_1, v_1) \in E_1$ with $v_2 \in V_2\}$.*

The resulting *SBG* graph has $|V_1| \cdot 2$ nodes and $|E_1| \cdot 3$ edges: a copy of the graph G_1 with an extra break edge for each edge in the graph.

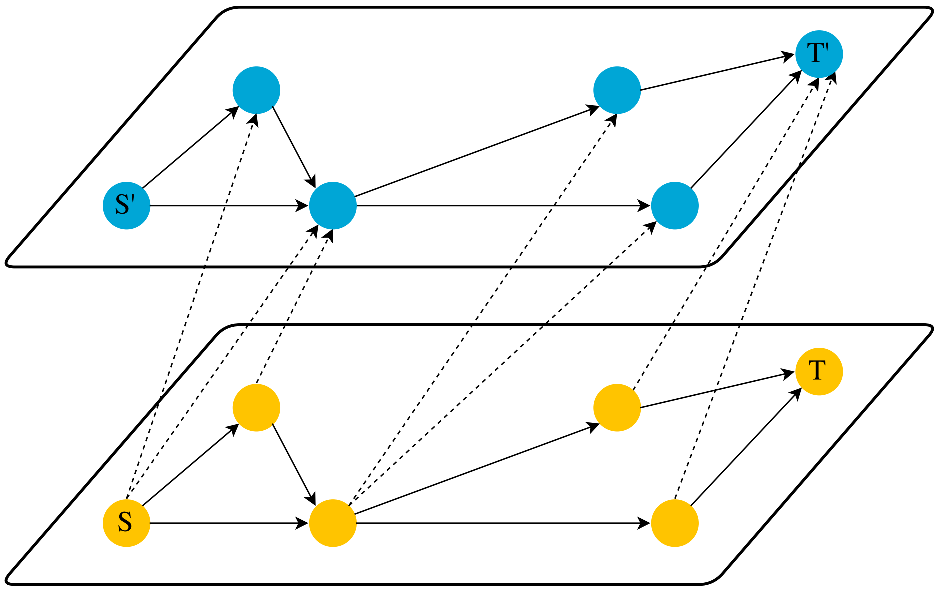


Figure 4-1: Visualisation of a stacked graph for implementing the RoadBlock-OneBreak variation. In yellow the original graph is shown. In blue the copy of the graph. Both graphs are linked with dashed edges

The travel time functions of the edges in E_1 and E_2 are equal to the functions with road blocks, as defined in Equations 2-1 and 2-2. The edges in set EB have different travel time functions: these edges should include the break time Δ_b of 45 minutes. For the planning of this break, there are three possible situations: the break is planned fully during a road block, partly during a road block, or not at all during a road block. The break is planned fully during a road block if the arrival time at the edge is later than the block start minus the free flow travel time. If the break cannot be finished before the block ends, the travel time can be calculated by determining the wait time during the block which can be used as break time $(\tau_{be} - \tau)$, and subtracting this value from the total break time. This leads to the travel time function $f_{RBE}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\text{FF}}, \Delta_b)$ for every edge in EB:

$$f_{RBE}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\text{FF}}, \Delta_b) = \begin{cases} \Delta_{\text{FF}} + \tau_{be} - \tau_{bs} & \text{if } \tau \geq \tau_{bs} - \Delta_{\text{FF}} \\ & \text{and } \tau < \tau_{be} - \Delta_b \\ \Delta_{\text{FF}} + \Delta_b - (\tau_{be} - \tau) & \text{if } \tau \geq \tau_{be} - \Delta_b \\ & \text{and } \tau < \tau_{be} \\ \Delta_{\text{FF}} + \Delta_b & \text{else} \end{cases} \tag{4-1}$$

Together with Equation 2-1 on page 15 defining the travel time function of an edge with a road block without considering breaks, this results in the following general travel time function f_e for any edge in the Stacked Break Graph:

$$f_e(\tau) = \begin{cases} f_{RBE}(\tau, RB_i = [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}) & \text{if } e \in E_1 \cup E_2 \\ & \text{and } \exists RB_i \in RBS \\ & \text{with } \tau \in [\tau_{bs} - \Delta_{\mathrm{FF}}, \tau_{be}] \\ f_{RBE}(\tau, RB_i = [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}, \Delta_b) & \text{if } e \in EB \\ & \text{and } \exists RB_i \in RBS \\ & \text{with } \tau \in [\tau_{bs} - \Delta_{\mathrm{FF}}, \tau_{be}] \\ \Delta_{\mathrm{FF}} & \text{else} \end{cases} \tag{4-2}$$

Besides the travel time, it should also be computed what the current non-stop driving time (shift duration) is after driving the edge. Therefore, a function s_{RBE} is introduced which gives the additional shift duration after driving an edge. Since waiting for a block does not count for the shift duration counter, the increase in shift duration is equal to the free flow travel time.

However, this only holds for edges in E_1 and E_2. For edges in EB the shift duration is reset after taking the break. Thus, the shift duration is equal to the driving time after the break has been taken. This leads to the function $s_{RBE}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}, \Delta_b)$ giving the current non-stop driving time after driving an edge in set EB and taking a break on this edge:

$$s_{RBE}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}, \Delta b) = \begin{cases} \Delta_{\mathrm{FF}} - \max(\tau_{bs} - \tau, 0) & \text{if } \tau \geq \tau_{bs} - \Delta_{\mathrm{FF}} \\ & \text{and } \tau < \tau_{be} - \Delta_b \\ \Delta_{\mathrm{FF}} & \text{if } \tau \geq \tau_{be} - \Delta_b \\ & \text{and } \tau < \tau_{be} \\ 0 & \text{else} \end{cases} \tag{4-3}$$

Using this formula the general shift duration function s_e for any edge e in the Stacked Break Graph can be defined, analogously to Equation 4-2:

$$s_e(\tau) = \begin{cases} s_{RBE}(\tau, RB_i = [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}, \Delta_b) & \text{if } e \in EB \\ & \text{and } \exists RB_i \in RBS \\ & \text{with } \tau \in [\tau_{bs} - \Delta_{\mathrm{FF}}, \tau_{be}] \\ \Delta_{\mathrm{FF}} & \text{otherwise} \end{cases} \tag{4-4}$$

Using the functions f_e and s_e, one can compute resulting values for, respectively, arrival time τ and shift duration Δ_d, by taking the sum of values of all edges on the path. The formulas for f_e and s_e are used in the algorithm for computing the earliest arrival route, as is explained in the next section.

4-2-1 Pareto search algorithm for the stacked break graph

Running a Dijkstra shortest path algorithm on the Stacked Break Graph does not necessarily lead to an optimal route, since the earliest arriving route might exceed the shift duration threshold whereas another route might not. Consider the example in

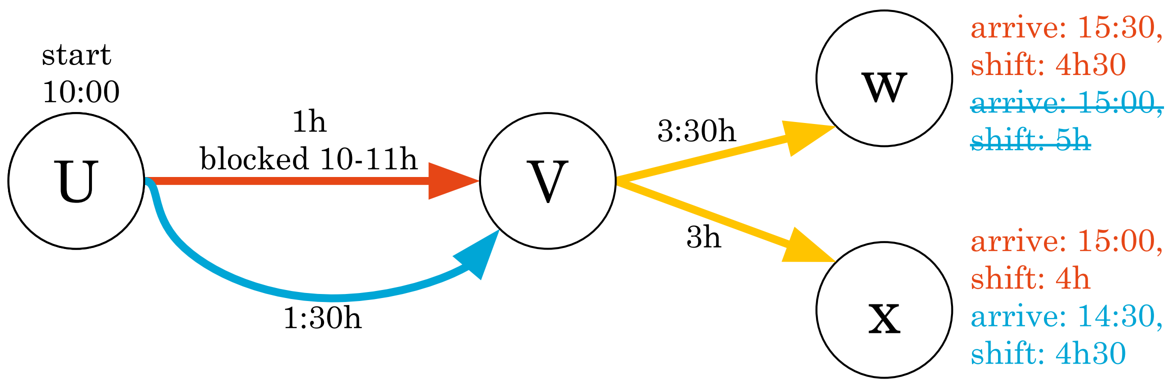


Figure 4-2: Assume node u is the start node, and the departure time is 10:00. If the destination is w, the only possible route is travelling the red route between nodes u and v given the maximum allowed driving time of 4.5 hour. If the destination is x, both routes via the red and blue routes are feasible. However, the blue route provides the earliest arrival time. Therefore, in the computation of routes from u to some destination, both red and blue routes need to be maintained at node v since either of them might provide the best solution.

Figure 4-2 where such a situation is shown. In this example, breaks are neglected such that the allowed driving time is just 4.5 hour.

Therefore, running a Dijkstra algorithm on the stacked break graph does not necessarily lead toward the optimal solution. This is due to the existence of two variables of importance: besides arrival time also shift duration determines the earliest arrival route. The optimal route can therefore be computed by keeping track of all possible *non-dominating* paths at each node, instead of just one path. Such an algorithm is referred to as a *Pareto* or *multi-label search*.

As mentioned before, instead of settling nodes, multiple possible paths toward a node are maintained in the Pareto Search algorithm. Therefore, paths are settled in the algorithm instead of nodes. These paths, containing the node, arrival time and shift duration, are defined as *labels*. Additionally, an edge of a node is only relaxed if the shift duration of the resulting label is less than the threshold Δ_T. Therefore, the resulting paths also do not violate the drivers legislation rules.

Property 4.3. *All paths resulting from the Pareto Search algorithm on the stacked break graph are valid according to the applicable drivers legislation rules, that is, the shift duration counter Δ_d never exceeds threshold Δ_T.*

Besides adhering to the drivers legislation rules, only non-dominating paths are added to nodes. Therefore, two types of domination are defined: weak and strict domination.

Definition 4.4 (Weak domination)**.** *A path i is weakly dominated by a path j, if paths i and j have the same start and destination location and the same departure time, and additionally $\tau_j \leq \tau_i$ and $\Delta_{dj} \leq \Delta_{di}$.*

Definition 4.5 (Strict domination)**.** *A path i is strictly dominated by a path j if it is weakly dominated, and additionally $\tau_j < \tau_i$ or $\Delta_{dj} < \Delta_{di}$.*

If some path i is not weakly dominated by another path j, this path is added to the list of paths at a node. Next, all paths x can be removed that are strictly dominated by i. This also implies that a path x having identical τ and Δ_d values are not removed.

The algorithm is stopped once no paths are present in the priority queue any more, that is, all possible paths exceeded the drivers legislation threshold. These resulting paths are called Pareto optimal:

Definition 4.6 (Pareto optimal path)**.** *A path i is Pareto optimal if it is not weakly dominated by any other path j that does not violate the drivers legislation rules*

From the resulting set of Pareto optimal paths at the destination node t, the path having the earliest arrival time is returned as the best path.

To prove the optimality of the algorithm, it is first shown that a Pareto optimal path is prefix optimal, followed by a proof that a Pareto optimal path is always found if it exists. Finally, it is shown that an optimal break planning path is always a Pareto optimal path. In these proofs, it is assumed that function $f(P_{st})$ represents the travel time of path P_{st} for some departure time τ_0, thus $\tau_t = \tau_0 + f(P_{st})$. Function $s(Pst)$ represents the shift duration at node t of this path for the same departure time τ_0 and current driving time Δ_{d0}, thus $\Delta_{dt} = \Delta_{d0} + s(Pst)$.

The proofs are based on work of Batz [7] and Ehrgott [20].

Lemma 4.7. *Let $G = (V, E)$ be a stacked break graph such that the travel time functions f and shift duration functions s are non-negative. Consider $s, t \in V$, where t is reachable from s in G, even if considering drivers legislation rules. Then there always exist a Pareto optimal path $P_{st} = \langle s = u_1, u_2, \ldots, u_k = t\rangle$, such that for every $i = 1, \ldots, k$, the prefix path $P_{si} = \langle s = u_1, \ldots, u_i\rangle$ is a Pareto optimal path too.*

Proof. Assume P_{si} is not a Pareto optimal path. Then there is a path $P'_{si} \neq P_{si}$ that dominates P_{si}, thus $f(P'_{si}) \leq f(P_{si})$ and $s(P'_{si}) \leq s(P_{si})$ with at least one strict equality, thus $f(P'_{si}) < f(P_{si})$ or $s(P'_{si}) < s(P_{si})$.

Therefore, a newly constructed path P'_{st} consisting of P'_{si} followed by P_{it} (the subpath of P from i to t), is an $s - t$ path which contradicts the Pareto optimality of P_{st} due to the following three reasons: First of all, $f(P'_{st}) \leq f(P_{st})$, since it is not possible to arrive later at t by departing earlier at node i due to FIFO-property of the network. Secondly, $s(P'_{st}) \leq s(P_{st})$, since $s(P'_{si}) \leq s(P_{si})$ and the additional shift duration of P_{it} is equal to the free flow travel time of this subpath, and thus independent of the departure time at node i. Third, it holds that $f(P'_{st}) < f(P_{st})$ or $s(P'_{st}) < s(P_{st})$, since the subpath toward node i also has at least one strict equality. Thus, path P_{st} is strictly dominated by path P'_{st}, violating the assumption of the Pareto optimality of P_{st}.

To conclude, there always exists a prefix path P_{si} that is a Pareto optimal path. $\square$

The prefix optimality of the Pareto optimal path is used to proof that all Pareto optimal paths are always found, as shown in the next proof:

Lemma 4.8. *A Pareto prefix optimal path P_{st} is always found by the Pareto search algorithm.*

Proof. Assume there exists some Pareto optimal path P_{si} from node s to i, that is not found by the Pareto search algorithm. Then there are two options: the label corresponding to this path is deleted during the algorithm or it was never found. If it was deleted, then P_{si} is fully dominated by some other path P'_{si} and is thus not Pareto optimal, contradicting the assumption. So, it can be concluded that the label was never found.

Let $i-1$ represent the node just before i in path P_{si}. According to Lemma 4.7, the path from s to this node is a Pareto optimal path. Since the Pareto optimal path to i was not found, the path to $i-1$ was also deleted or not found.

One can repeat this argument backwards, until the first node j in the path (unequal to s) is not found. However, it is impossible that this Pareto optimal path was not found, because the edge (s, j) is relaxed in the first step of the algorithm. If this label is removed by another dominating path, then the path from s to j is not a prefix optimal path, contradicting the assumption. Therefore, any Pareto optimal path P_{si} is found if the execution of the algorithm is stopped. □

Thus, all Pareto optimal paths toward the destination are found by the Pareto search algorithm. To complete the proof of optimality of the break planning algorithm, we now only need to prove that the algorithm indeed returns an optimal break planning path (see Definition 4.1), implying that the optimal break planning path is indeed a Pareto optimal path.

Theorem 4.9. *The Pareto search algorithm returns an optimal break planning path*

Proof. Let P_{st} be the path returned by the Pareto search algorithm, meaning that this path has the earliest arrival time of the set of paths to destination node t found by the algorithm. Now assume that this path P_{st} is not an optimal break planning path. This implies that there exists some path P'_{st} with $f(P'_{st}) < f(Pst)$. But this implies that P'_{st} is a Pareto optimal path, since it is not dominated by all other paths. However, the Pareto search algorithm finds all Pareto optimal paths (See Lemma 4.8). So, it is not possible that such a path P'_{st} exists, and the path returned by the Pareto search algorithm is an optimal break planning path. □

Although an optimal break planning path is the result of the Pareto search algorithm, its running time is bad. At the worst case, the number of paths found grows exponentially with the number of nodes in the network. Therefore, a single label heuristic is introduced in the next section which does not require to maintain multiple paths at each node and runs in polynomial time.

4-2-2 Single label algorithm for the stacked break graph

Although the proposed Pareto search in the previous section gives optimal results, lots of paths need to be maintained which yield a long running time. Therefore the idea is to introduce a Dijkstra-like heuristic, which maintains only one path (also referred to as label) at each node. First it is identified what the underlying problem is, followed by a proposed algorithm and a review on the quality of this heuristic.

Problems for running a Dijkstra algorithm

The reason why running a Dijkstra algorithm on the Stacked Break Graph (as illustrated in Figure 4-2) does not work has its cause in differences in travel time and shift duration. Waiting for a block does not count for the shift duration counter, but does count for travel time. Due to this, the route with the earliest arrival time might have a shift duration higher than allowed according to the regulations, whereas there might exist another feasible route with a lower shift duration but later arrival time.

This is mainly due to the planning of breaks: for some routes the break should be planned as late as possible to maximize the driving time instead of being planned efficiently (i.e. during a block). Therefore, the idea is to split the search into two steps, which are mentioned briefly and later explained in detail:

1. Settle nodes of V_1 until t_1 is settled or all discovered paths have a length of Δ_T, the maximum driving time without a break. During execution, also relax edges of set EB, but store the resulting paths in a different priority queue QB. If t_1 is settled, the optimal path is found and step 2 can be discarded.
2. If t_1 is not settled yet, settle nodes of V_2, starting from the nodes in QB touched in the previous step. The earliest arrival time – if there exists a feasible path – is now represented by the path to t_2.

Step 1: settling nodes in V_1

The nodes in V_1 represent the nodes that are reached without taking a break. Both edges in E_1 (all edges (u, v) with $u, v \in V_1$) and EB (all edges (u, v) with $u \in V_1$ and $v \in V_2$) are relaxed. However, we are only interested in settling nodes in set V_1. Therefore, if an edge from EB is relaxed, the node corresponding to a path to v_2 is added to a different priority queue QB. This queue is used in the next step of the algorithm.

If t_1 is settled, the algorithm stops and the path to t_1 returned. This means that there is a path from start to destination which takes less travel time than threshold Δ_T of 4.5 hours. Since waiting is never beneficial in a FIFO road network [38], the path to t_1 is optimal and can be returned.

If t_1 is not reachable, the algorithm stops once all paths have exceeded the threshold Δ_T of 4.5 hours, i.e. the minimum value in the priority queue is larger than Δ_T.

Step 2: settling nodes in V_2

In the second step the nodes in V_2 are settled: the paths with a break on the route. As a starting point of the algorithm, priority queue QB is used. This queue is filled in the previous step. It is not sorted on earliest arrival time, but on increasing shift duration. If shift durations are equal, the one having the highest arrival time is ordered first, since this is probably the route being most close to the destination.

Example

Assume the example network in Figure 4-3. The goal is to find the earliest arrival route from s to t, with a given departure time τ_0 of 8:30 and shift start of 0. In step 1 of the heuristic algorithm, nodes s_1, u_1, u_2 and v_2 are settled. For node u_2 this means that the truck drives for 3.5 hour until 12:00, followed by a break of 45 minutes taken during the block. Next another half an hour needs to be travelled, leading to a shift duration of 30 minutes and an arrival time of 13:30 (identical to the arrival time at u_1). Nodes v_1 and t_1 are not reachable in step 1 due to the exceedance of threshold Δ_T. The resulting arrival times are shown in Table 4-1. If priority queue QB is sorted on earliest arrival time, the first node to settle in step 2 would be u_2, leading to an updated arrival time at v_2 of 14:30. But the destination can never be reached via this path due to the exceedance of the threshold of 4.5 hour. The destination can only be reached if the proposed ordering on shift duration is followed: leading to an arrival time of 19:15 at t_2.

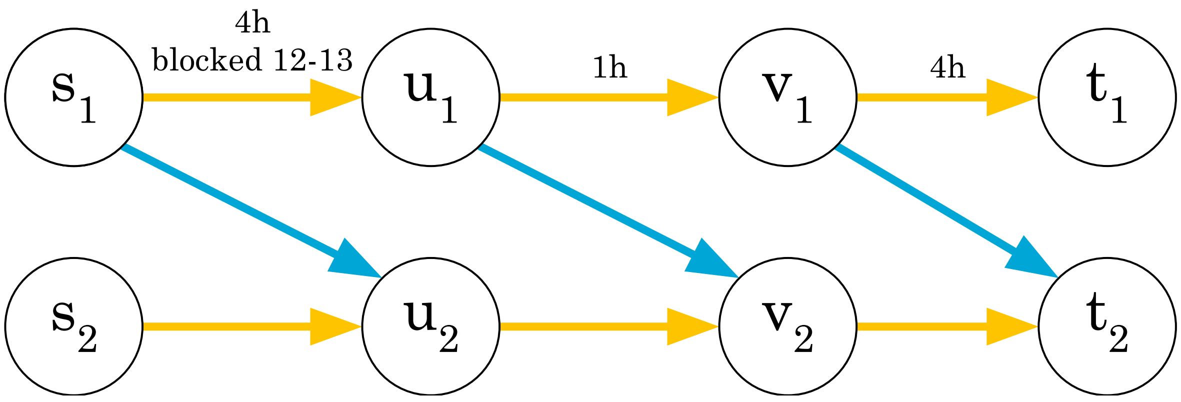


Figure 4-3: Example graph to show difference between order methods. Nodes s_1 to t_1 represents nodes where no break has been taken, the blue edges represent taking a break of 45 minutes

The example showed the justification of the ordering of the priority queue. The remainder of the settling procedure of the nodes in V_2 is identical to the time-dependent Dijkstra algorithm, with the notion that it is possible that a settled node is added to the priority queue for a second time. This occurs due to the multiple starting points of the search, whereas the queue is not sorted on the earliest arrival time.

Table 4-1: Settling of nodes of example graph in Figure 4-3. The shift duration threshold is 4.5 hours, the duration of a break is 45 minutes

After step 1 (ordered on τ)	s_1	u_1	v_1	t_1	s_2	u_2	v_2	t_2
arrival time τ	8:30	13:30	∞	∞	∞	13:30	15:15	∞
shift duration Δ_d	0:00	4:00	∞	∞	∞	0:30	0:30	∞

After step 2 (ordered on τ)	s_1	u_1	v_1	t_1	s_2	u_2	v_2	t_2
arrival time τ	8:30	13:30	∞	∞	∞	13:30	14:30	∞
shift duration Δ_d	0:00	4:00	∞	∞	∞	0:30	1:30	∞

After step 2 (ordered on Δ_d)	s_1	u_1	v_1	t_1	s_2	u_2	v_2	t_2
arrival time τ	8:30	13:30	∞	∞	∞	13:30	15:15	19:15
shift duration Δ_d	0:00	4:00	∞	∞	∞	0:30	0:30	4:30

The search can be stopped if t_2 has been settled and $\tau_{u2} > \tau_{t2}$ for all unsettled nodes $u_2 \in V_2$ The arrival time at t_2 is the best value.

Quality of heuristic

The proposed algorithm assures that the break is scheduled correctly: that is, as late as possible if needed according to drivers legislation rule, and as optimal as possible otherwise. However, there are still situations in which the algorithm does not result in an optimal result. An example of non-optimal behaviour can be seen in the earlier used example in Figure 4-2. Using the proposed heuristic still does not keep both routes from u to v whereas these are needed. However, it is noted that this situation does not occur often in the road network of Europe. This is due to the fact that a road block is active for a complete country. This example situation therefore only occurs if there are two countries close to each other where different road blocks are active in each of the countries resulting in different shift durations for identical travel times. This is for example the case at the Belgium/Luxembourg/France borders, as shown in Figure 4-4.

This situation can be detected: if a path is thrown away having a earlier shift duration but a later arrival time than the other path, this is a potential non-optimal solution. In these cases a Pareto search can be run to assure optimality.

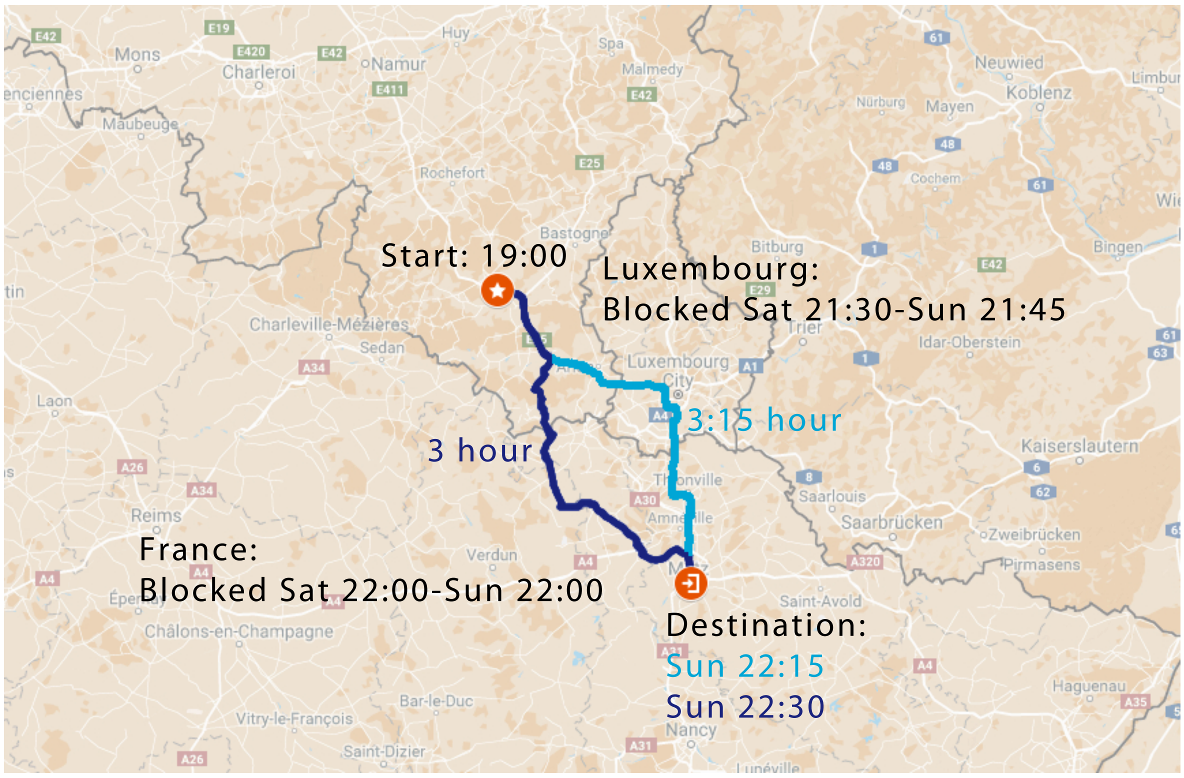


Figure 4-4: Example route where the result of the heuristic method is not optimal. The blue route through Luxembourg is a bit longer (3:15 hours) compared to the purple route through France (3:00 hours). However, the driving time through France is shorter and the block in Luxembourg ends earlier then the one in France. This causes that the blue route arrives earliest, at 22:15 compared to 22:30. However, the shift duration of the purple route is shorter: 3 hours compared to 3:15. If the path lingers further on which requires another 6:01 to 6:15h of driving time, the purple path should be used to not exceed the 9 hour threshold. But this route will not be maintained in the heuristic algorithm.

4-3 Time-dependent Contraction Hierarchies

Contraction hierarchies reduce the number of nodes and edges in a graph during a preprocessing step, while maintaining all shortest paths. The actual shortest path from start to destination is computed during the querying phase – which can generally be performed rapidly compared to traditional Dijkstra-like algorithms. In this section methods are given that enable optimal break planning while using contraction hierarchies. It is assumed that knowledge on contraction hierarchies as given in Section 3-2-3 is known.

4-3-1 Preprocessing a stacked graph

The first idea to implement the breaks in time-dependent contraction hierarchies (TCH) would be to reuse the Stacked Break Graph implementation introduced in the previous section: doubling the graph and connecting the nodes in the base layer with the copied layer by an artificial edge which represents taking a break. However, this would not result in correct answers during querying. Although the preprocessing makes sure that all shortest paths are maintained, it does not care about the shift duration. In the example network in Figure 4-5 the maintained shortest path between x_1 and z_2 at 10:00 travels through y_2: taking the break at the start results in a lower waiting time for the block between nodes x_1 and z_2. However, this results in a consecutive driving time of 6 hours between x_1 and z_2, which is not allowed concerning the threshold of 4.5 hours. The only feasible path would be via y_1.

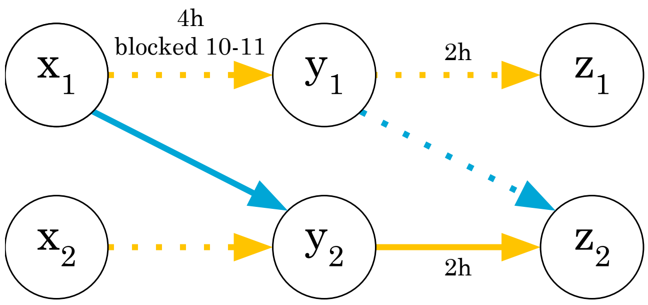


Figure 4-5: Example of a double-layered graph which gives wrong results if a query is ran on the preprocessed version of the graph. The solid arrows indicate the maintained shortest path between nodes x_1 and z_2 at time 10:00

It can be observed that the optimal location of a break depends on the start and destination locations. This information is not known at the preprocessing stage. Therefore, all possible locations for a break should be maintained as an option. This would lead to enormous amounts of parallel shortcut edges, each representing a path which takes a break at a different location on different edges. It is questionable whether the preprocessed graph then provides any advantage over working with the non-preprocessed graph. Therefore, using the stacked graph approach is not assumed as a viable option for implementing breaks in time-dependent contraction hierarchies.

4-3-2 Adjusting the querying algorithm

The previous paragraph showed that the optimal location to break can only be determined if the start and end locations are known. Therefore, it seems more promising to come up with a solution during the querying phase of the algorithm. The querying algorithm is a bi-directional Dijkstra algorithm that runs on the preprocessed graph as explained in Section 3-2-3. The idea is to apply the same Pareto search algorithm and heuristic introduced for the Stacked Break Graph, with a few modifications such that it works correctly on the preprocessed graph.

Independent of the precise implementations of the querying algorithms, we should be able to calculate shift durations of driven routes. Therefore, we need the information on the next blockstart and blockend, as well as the actual driving time (i.e. travel time minus waiting time). This information is maintained in a modified version of the preprocessing procedure.

Pareto search

The same Pareto search algorithm explained in Section 4-2-1 can be run on the preprocessed graph with a few modifications. It should be noted that the default preprocessed graph only consists of one layer of nodes, in contrast to the Stacked Break Graph. Instead of doubling the graph, we keep track of two lists of paths at each node: one for paths with a break, one for paths without a break.

The backward interval search can be kept identical - independent of breaks, shift duration or whatsoever. This is possible since the backward interval search already explores all possible paths to the top of the preprocessed tree.

The forward and downward search should deal with planning of breaks. For each to be settled label it should be checked whether a break already has been taken on this path. If no break is taken yet, two paths for each edge should be explored: one without a break (using the default travel time functions), one with a break (using the travel time functions f_{RBE} in Equation 2-1). Furthermore, also shift duration should be tracked. Paths exceeding the shift duration should be discarded. Furthermore, all upward paths in the preprocessed graph should be explored during the forward search, not just the paths providing shortest paths for the current departure time. Likewise, the downward search should explore all downward edges that are a result of the backward interval search.

It should be noted that the Pareto search algorithm does not necessarily provide optimal results in the preprocessed graph. A counter example is shown in Figure 4-6. If node v is contracted, the preprocessed graph does not contain the red coloured edges any more, since these never provide an earliest arrival path between u and t. However, in some special case, the optimal break planning path runs through these edges. Therefore, the Pareto search algorithm is not always capable of giving the optimal solution if run on a preprocessed graph.

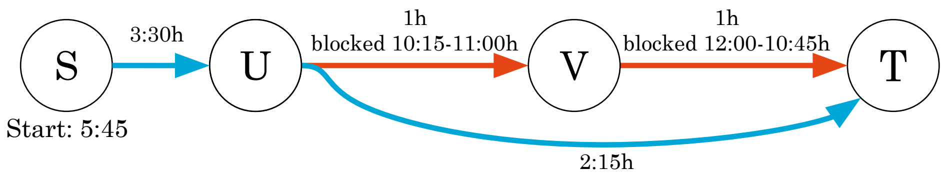


Figure 4-6: This example shows that the Pareto search algorithm does not always yield optimal results in a preprocessed graph. The red coloured edges between u and t never provide a shortest path, since always a block is experienced of at least 30 minutes waiting time. Please note that the edge between v and t can only be driven between 10:45 and 12:00 every day. Thus, if node v is contracted, only the blue edge remains in the preprocessed graph. However, if considering breaks, the red route might provide the shortest path. If starting at 5:45, one arrives at node u at 9:15. Node v is reached at 10:15, after which a break of 45 minutes is taken. At 11:00 the edge between v and t is driven, resulting in an arrival time of 12:00. The direct route between u and t results in an arrival time of 12:15, thus not the shortest path if considering breaks.

It is not likely that this situation occurs often, since in Europe there are no adjacent roads that are blocked throughout the day. More specifically, if all routes can be driven at free flow travel time, this situation does not occur, as is shown using the following proof:

Lemma 4.10. *The Pareto search on time-dependent contraction hierarchies provides optimal results, if any route can be driven at free flow travel time at some moment.*

Proof. Let $f_T(P)$ be the travel time function of path P, without considering taking breaks. Travel time function $f_B(P)$ is used to refer to the travel time function including taking a break if required, as defined in Equation 4-2. Function f_T is used to obtain the preprocessed graph.

Assume that there exists an optimal break planning path P_{st} from node s to t, that is not found by the Pareto search algorithm in the preprocessed graph. Since all Pareto optimal paths existing in the preprocessed graph are found by the algorithm (see Lemma 4.8), this means that the path does not exist in the preprocessed graph, since the Pareto search algorithm explores all edges – not just the one that provides the earliest arrival time for a specific departure time.

It is only possible if some optimal break planning path is not included in the preprocessed graph if, for any departure time τ, there exists some path P'_{st} for which it holds that $f_T(P'_{st}) < f_T(P_{st})$. This means that path P_{st} never provides a shortest path at any departure time (see example in Figure 4-6 for such a case), and is thus not included in the preprocessed graph.

Assume that P'_{st} is the earliest arrival path for the departure time τ_0 as returned by a non-modified querying algorithm, thus not assuming any breaks. This path P'_{st} is not equal to an optimal break planning path, since this path is removed during preprocessing. Then, it holds that $f_T(P'_{st}) < f_T(P_{st})$ and $f_B(P'_{st}) > f_B(P_{st})$.

So, path P'_{st} is faster than P_{st} if no break is planned, but slower if the drivers legislation rules are followed and a break is planned. This is only possible if the time required for taking a break results in a lower travel time for P_{st} than for P'_{st}, due to road blocks on the path. Thus, the free flow travel time Δ_{FF} of P'_{st} is larger than the free flow travel time of P_{st}. However, it is assumed that all routes can be driven at free flow travel time at some moment. Therefore, P_{st} should dominate P'_{st} at this specific moment, and thus provides a shortest path without considering breaks. But then, path P_{st} would not be removed during preprocessing, and is found during the execution of the Pareto search algorithm on the preprocessed graph, forming a valid up-down path. Therefore, the optimal break planning path is always found by the Pareto search algorithm on time-dependent contraction hierarchies, if any route can be driven at free flow travel time. □

Although it might seem obvious that each route can be driven at some time in free flow travel time (that is, not facing any road blocks), this is not guaranteed due to the possible creation of very long shortcut edges in the preprocessed graph. The quality of the Pareto algorithm is verified using simulations in Section 4-5-5.

As with non-preprocessed Pareto search version, lots of labels are generated. Therefore, also for the TCH a single label heuristic method is introduced.

Single label method

The single label heuristic approach introduced for the Stacked Break Graph does not work at the preprocessed graph since no double layers of nodes are available. Therefore another heuristic method is introduced. This single label heuristic approach is based on the idea that there are two options for scheduling breaks: as late as possible or as optimal as possible.

In a non-time-dependent network, it is never unfavourable to plan a break as late as possible. The planning of a break does not influence the experienced travel times and therefore the shortest path stays equal – if it is assumed that a break is allowed anywhere along the route. By planning the break as late as possible the possible driving time is maximized.

In a time-dependent network this is not the case. By taking an early break, it is possible to save time. Note that waiting during a road block does not count as driving time, but might be used as break time. Consider the example in Figure 4-7. Assume that the earliest arrival route needs to be found for a path from s to t starting at 9:00. The strategy of planning a break as late as possible (from now on called *latest-strategy*), would plan the break after 4.5 hours of driving time. This would be halfway the middle edge, at 14:30. This results in an arrival time of 18:45 at node t. A better strategy would be planning the break during the waiting time for the block (from now on called *block-strategy*). The break can be planned at 13:00, resulting in an arrival time of 18:00 at node t, and thus saving 45 minutes of time.

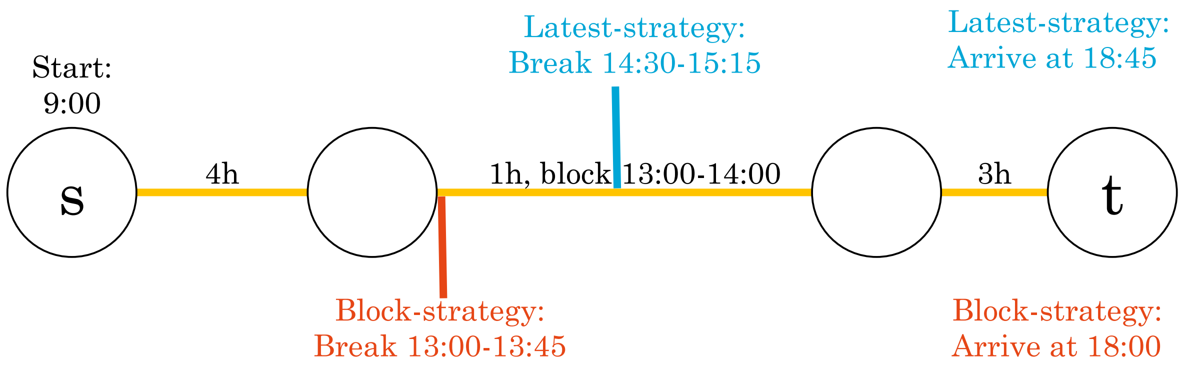


Figure 4-7: Example illustrating both block and latest strategy of planning breaks

Although the *block-strategy* results in an earlier arrival time, this does not always work out correctly. Assume the last link in the example in Figure 4-7 now takes 4 hours of travel time instead of 3 hours. By using the block-strategy and thus planning the break during the block, 5 hours of driving time still needs to be driven after the break. However, the maximum consecutive driving time is 4.5 hours. Therefore, it does not provide a feasible route. By using the *latest-strategy* (at 14:30) the destination is reached at 19:45 without violating the maximum driving time rule.

The idea of having a *latest-strategy* and a *block-strategy* was already implicitly implemented in the heuristic Stacked Break Graph Dijkstra algorithm by sorting the priority queue of the stacked graph representing breaks on shift duration. It was made sure that the destination could always be reached (latest-strategy) whether nodes were resettled if a better path was found. A better path could only be found if the break was (partly) planned during a road block (equivalent to the block-strategy).

Both strategies latest-strategy and block-strategy are discussed in detail in the following subsections.

Latest break insertion strategy

The *latest-strategy* inserts a break as late as possible to maximize the non-stop driving time d, i.e. maximize the daily driving time to 9 hours. Therefore it should be known what the resulting travel time and shift duration is when a break is inserted on an edge. This leads to small modifications of Equations 2-1 and 2-2, taking current shift duration Δ_d and threshold Δ_T into account. This leads to the equations $f_{LS}(\tau, \Delta_d)$ and s_{LS} for travel time and shift duration:

$$f_{LS}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}, \Delta_d) = \begin{cases} \tau_{be} + \Delta_{\mathrm{FF}} - (\tau + \Delta_T - \Delta_d) & \text{if } \tau + \Delta_T - \Delta_d + \Delta_b \geq \tau_{bs} \\ & \text{and } \tau + \Delta_T - \Delta_d \leq \tau_{bs} \\ f_{RBE}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}) + \Delta_b & \text{else} \end{cases} \tag{4-5}$$

$$s_{LS}(\tau, [\tau_{bs}, \tau_{be}], \Delta_{\mathrm{FF}}, \Delta_d) = \Delta_{\mathrm{FF}} - \Delta_T + \Delta_d \tag{4-6}$$

For example, take a situation where you arrive at 11:20 (τ) at a road segment which takes 1 hour to drive and which is closed from 12:00 (τ_{bs}) till 14:00. The current shift duration is 4 hours (Δ_d), resulting in a remaining driving time of 30 minutes considering the threshold Δ_T of 4.5 hours driving time. This results in a driving scheme of driving for 30 minutes until 11:50, followed by a 45 minute break till 12:35. Next you have to wait until 14:00 before driving the remaining 30 minutes, resulting in an arrival time of 14:30. The travel time is then 3 hours and 10 minutes. In the formula this is calculated as follows:

$$\tau_{be} + \Delta_{\mathrm{FF}} - (\tau + \Delta_T - \Delta_d) = 14:00 + 1:00 - (11:20 + 4:30 - 4:00) = 3:10 \tag{4-7}$$

4-3-3 Block break insertion strategy

The *block-strategy* inserts a break at a moment which is as optimal as possible in terms of arrival time. This implies that the break is planned (partly) during a road block. Remember: waiting for a road block does count as travel time but does not count for driving time.

It is therefore most optimal to drive until the block actually starts. However, this might not be possible due to the remaining driving time. Therefore, the remaining shift duration is depended on the start time of the road block and not on the current shift duration. This leads to the following set of equations for calculating the travel time and shift duration:

$$f_{BS}(\tau,[\tau_{bs},\tau_{be}],\Delta_{\mathrm{FF}},\Delta_d)=\begin{cases}\tau_{be}-\tau_{bs}+f_{RBE}(\tau) & \text{if } \tau+f_{RBE}(\tau)>\tau_{bs}\\ & \text{and } \tau<\tau_{bs}\\ & \text{and } \tau_{bs}-\tau<\Delta_T-\Delta_d\\ f_{RBE}(\tau)+\tau_{be}-(\tau+(\Delta_T-\Delta_d)) & \text{if } \tau+f_{RBE}(\tau)>\tau_{bs}\\ & \text{and } \tau<\tau_{bs}\\ & \text{and } \tau_{bs}-\tau\geq\Delta_T-\Delta_d\\ f_{RBE}(\tau)+\tau_{bs}-\tau & \text{if } \tau\geq\tau_{bs}\\ & \text{and } \tau\leq\tau_{be}\\ & \text{and } \tau+\Delta_b\leq\tau_{be}\\ f_{RBE}(\tau)+\tau+\tau_{bs}-\tau & \text{if } \tau\geq\tau_{bs}\\ & \text{and } \tau\leq\tau_{be}\\ & \text{and } \tau+\Delta_b>\tau_{be}\end{cases} \tag{4-8}$$

$$s_{BS}(\tau,[\tau_{bs},\tau_{be}],\Delta_{\mathrm{FF}},\Delta_d)=\begin{cases}\Delta_{\mathrm{FF}} & \text{if } f(\tau)>\Delta_{\mathrm{FF}}\\ \Delta_{\mathrm{FF}}-(\tau_{bs}-\tau) & \text{if } \tau_{bs}-\tau<\Delta_T-\Delta_d\\ & \text{and } \tau_{bs}-\tau<\Delta_{\mathrm{FF}}\end{cases} \tag{4-9}$$

4-4 Current situation

Before testing the presented algorithms with a test-set, first the algorithm is introduced that is used by the current routing software of ORTEC. This algorithm is used to compute the influence in arrival time when optimal break planning is applied.

In the software of ORTEC breaks are not taken into account when planning routes, but inserted afterwards. This can be imitated by using a modification of Dijkstra's algorithm. First the shortest path is computed using the time-dependent Dijkstra algorithm (see Section 3-2-1) from start s to destination t at departure time τ_0. No breaks or thresholds on driving time are assumed, but road blocks are taken into account. Next, a break is inserted on this path at the moment that the driving time exceeds threshold Δ_T. After that the arrival time of the remainder of the path is updated according to the inserted break. Finally it is checked whether the threshold of driving time is not exceeded after the break has been taken.

This simple algorithm replicates the behaviour of the current routing software provided by ORTEC. However, the current software does not take road blocks into account. Only in experimental test versions this is included. However, testing without road blocks is useless. Therefore they are included in this algorithm.

4-5 Simulations

Three algorithms are introduced in this chapter: *Current* (representing the current situation of break planning), Dijkstra and TCH. The last two algorithms come with a Pareto search and a single-label variant. The goal of this simulations section is to analyse the influence on arrival time compared to the current algorithm and the proposed algorithms and to give insight in resulting routes. Furthermore it is analysed what the resulting running time is of each of the algorithms.

Therefore, first a test-set is created representing the trips made by trucks through Europe. Next all routes of the test-set are computed using the different algorithms. The results are used to analyse the average improvement in travel time between the currently used algorithm and optimal break planning. The properties of improving routes are identified to make a first step to prediction of possible improvement. The performance of the heuristic algorithms are computed, as well as the running time for each of the algorithms.

4-5-1 Test set-up

The algorithms are tested on the network of Europe provided by HERE [29]. The network includes all highways and secondary roads and has 7 million nodes and 13 million edges. Road blocks active throughout the year are considered; no specific holiday blocks are taken into account. The road blocks in Europe are visualized in Figure 4-8. Yellow indicates that this country has some block active on Sunday, for example Saturday 21:30-Sunday 21:45 (Luxembourg) or Sunday 13:00-22:00 (Czech Republic). The blue countries Austria and Switzerland have road blocks for trucks during every night, as well as on Sunday. The minimum block duration is 7 hours (night blocks in Austria and Switzerland), the maximum block duration is 38 hours (Saturday 15:00-Monday 5:00 in Austria). Appendix A lists all road blocks used in the simulations.

The test instances (i.e. pairs of start and destination locations) are a representation of the freight flows through Europe. Data on freight flows is taken from the ETIS-project (European Transport policy Information Systems) [23]. This project combines data from several trade databases to create an Origin-Destination matrix between countries of Europe. Actual test instances are created by randomly choosing a node in the country of origin and destination according to the Origin-Destination distribution given by ETIS [23]. In total, 10000 queries are generated.

The start time for each of these test instances is picked randomly from an uniform distribution between Monday 0:00 and Sunday 23:59. It is assumed that all routes start with a shift duration of 0:00.

All tests are run on a laptop, having 16GB memory and an Intel Core i7-3740QM 2.70GHz processor with 4 cores.

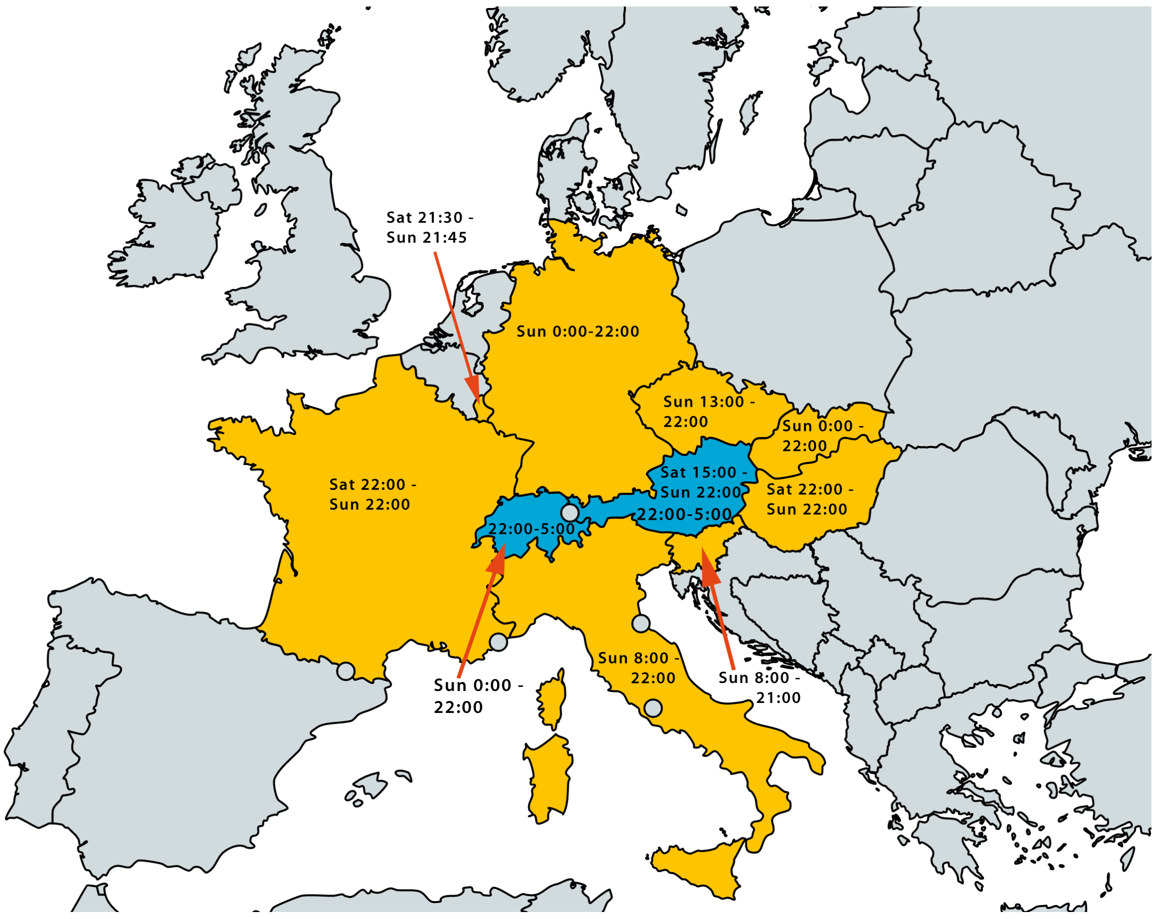


Figure 4-8: Road blocks in Europe, where yellow coloured countries indicate a driving ban during Sunday and blue coloured countries indicate a driving ban every night and on Sundays

4-5-2 Verification of results

Writing computer programs always has a chance of producing bugs, despite creating tests or testing with small instances. Therefore, first samples of the results are checked for correctness. All outliers in terms of long travel time or large differences between latest and optimal planning of breaks were checked manually. This did not lead to any suspicious routes. Furthermore, for 10 random routes it was checked whether the travel time and route choice corresponds with the results of a route planner like Google Maps. Although all simulated travel times were longer, that is, about 1.5 times as long, this is assumed correct due to slower travel speeds of trucks.

Due to these verification tests, it is assumed that the produced routes are correct.

4-5-3 Comparison of current and optimal break insertion

Not all 10000 start and destination pairs are possible to drive within the 9 hours of driving time (i.e. 4.5 hours of driving, 45 minute break, 4.5 hours of driving): only 40%.

This is mainly due to the long distances that trucks drive through Europe: even a ride from the Netherlands to Germany can already exceed this limit. Therefore, all instances are run a second time for a longer driving time, representing two days of driving instead of one. This entails a driving threshold of 9 hours, followed by an 11 hour (night) rest and again a maximum driving time of 9 hours. Of these instances, 79% was able to reach the destination before the threshold was exceeded. These variations are called OneDay and TwoDays in the remainder of this section – not necessarily meaning that the driving time is indeed one or two days.

Of the possible OneDay instances 65% needed a break, of the TwoDays instances this was 49%. In total 163 (6%) and 664 (17%) of the routes improved using scheduling breaks as optimal as possible instead of the latest strategy. The improving routes had an average 2:45 or 5:14 hour improvement in travel time, respectively, 17% (OneDay) or 19% (TwoDays) of the total travel time. On average, the improved route had a travel time of 84.9% or 89.9% of the original route. The results are shown in Table 4-2.

Table 4-2: Results of OneDay and TwoDay

	One day	**Two days**
Total number of routes	10000	10000
Feasibile routes	4042	7894
Routes with breaks	2645 (65%)	3908 (49%)
Routes improving	163 (6%)	664 (17%)
Average travel time of improving routes	15h 49min	27h 31min
Average absolute improvement	2h 45 min	5h 14 min
Maximum absolute improvement	37h 54 min	28h 49min
Average relative improved travel time	84.9%	89.9 %
Maximum relative improved travel time	36.2%	11.9%

Example routes

The most improving route for OneDay is visualized in Figure 4-9. The route starts near Venice, Italy, and ends in Regensburg, Germany. Whereas the fastest route – without assuming breaks – is travelling a large part through Austria, this is not beneficial if a break needs to be inserted. Due to the break between 14:13 and 14:58, the truck ends up in the long Sunday block of Austria lasting between Saturday 15:00 and Monday 5:00. This leads to an arrival time of 37 hours and 54 minutes later than the optimal route, which leaves Austria as fast as possible, thus skipping the Sunday block in Austria.

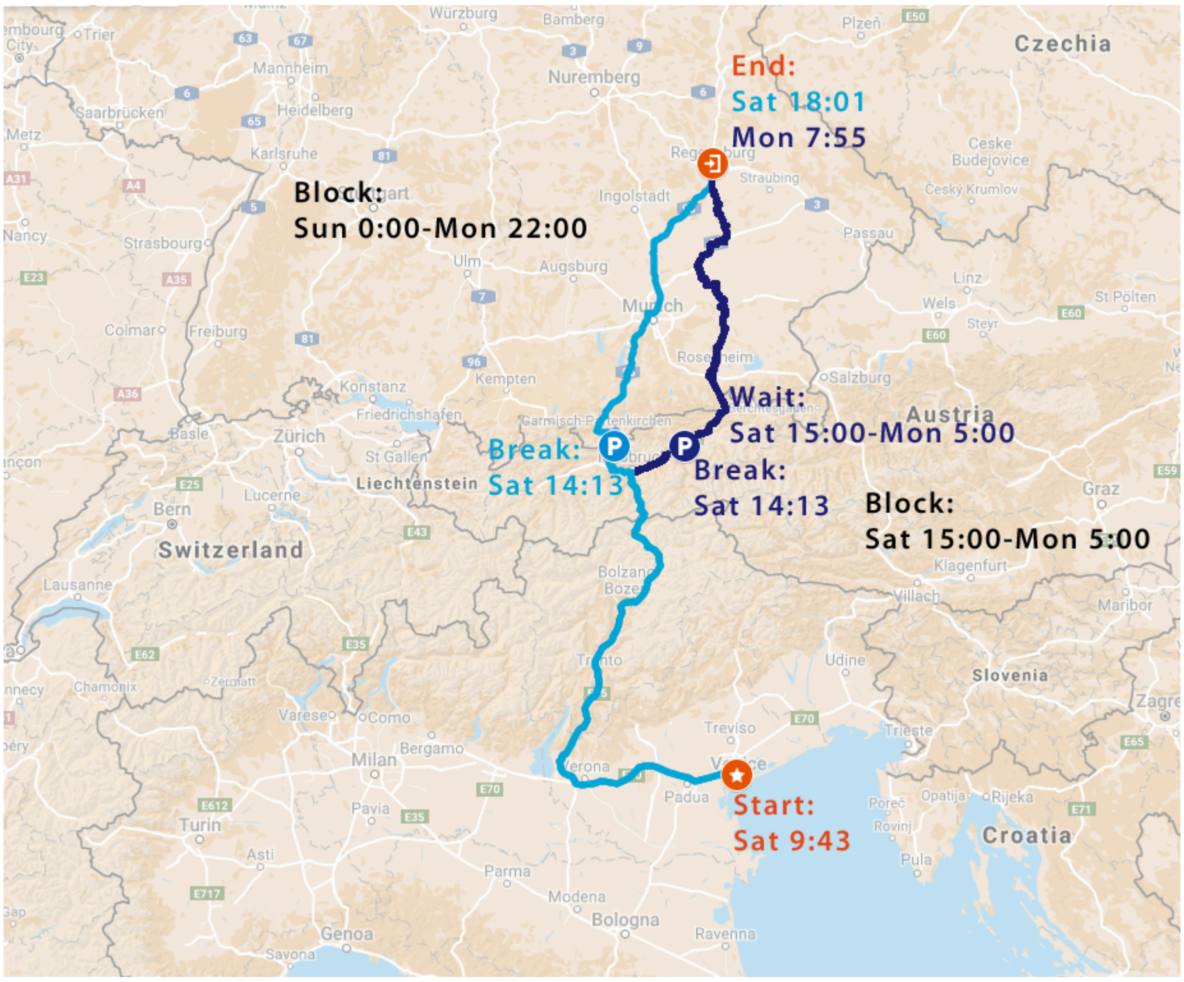


Figure 4-9: Example route of OneDay: start in the North of Italy, destination is located in Germany. A different route choice leads to a shorter travel time if considering breaks

The most improving route for the TwoDay variation is shown in Figure 4-10. Different from the OneDay example is that both routes are identical. However, the time of taking a break on this route is different. This route passes through two potential blocks: the night block in Austria (22:00 till 5:00) and the Sunday block in Germany (0:00 till 22:00). If the night block in Austria is used for a night rest (11 hour break), this results in reaching the destination before the block in Germany starts. However, if just waiting in Austria (the latest break scheduling example), and taking a break after 9 hours of driving in Germany – just 33 minutes of driving away from the destination – causes to just end up in the road block in Germany causing a delay of about 29 hours!

Duration of improving routes

Besides the most improving routes, it is interesting to see the distribution of travel time improvements. Therefore, a histogram of the OneDay and TwoDay test set is shown in Figure 4-11. A clear peak can be identified at 45 minutes of improvement for OneDay,

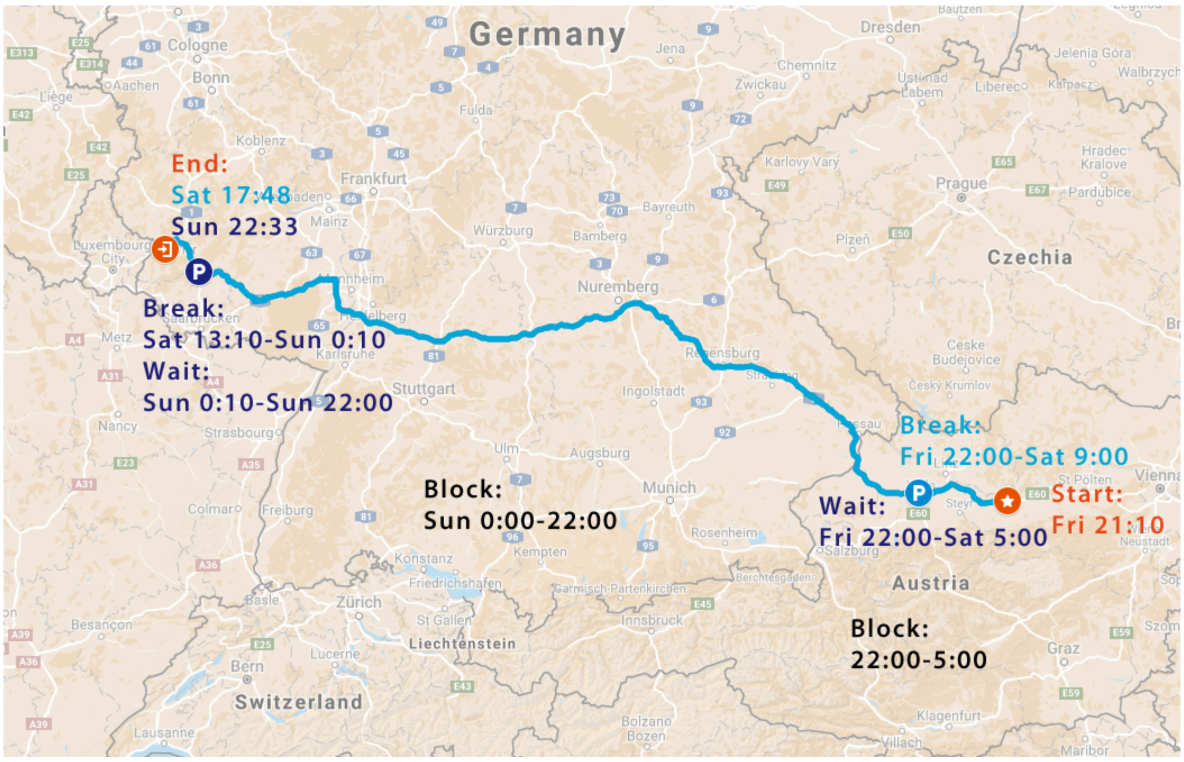


Figure 4-10: Example route of TwoDay: start in Austria, destination in Germany. Optimally planning the break (during the night block of Austria) leads to a shorter travel time

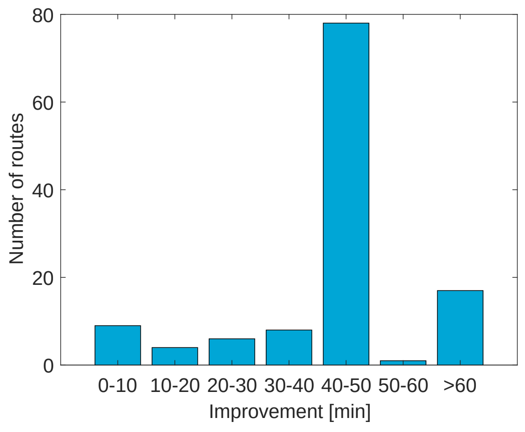
also the mode of the data. This period is exactly the break time, thus meaning that the break can fully take place during a road block instead of afterwards. TwoDays also has a peak around the break time of 11 hours.

Properties of improving routes

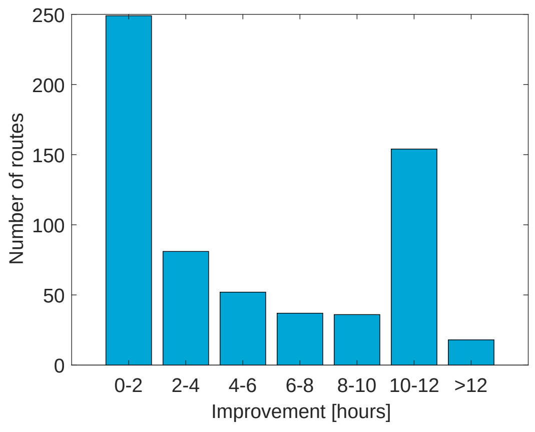
In addition to the distribution of the actually improving routes, it is interesting to see which routes are improving, to be able to predict whether a route might improve if optimal break scheduling is applied. The distribution of the starting times of the improving routes are shown in Figure 4-12. It can be seen that most improving routes start on a Saturday or Sunday, which is logical due to the large amount of road blocks active on Sunday (see Figure 4-8). During the weekdays, only night blocks are relevant. Therefore the routes between 8:00 and 16:00 on weekdays are not improving much.

4-5-4 Optimality gap of Pareto and single-label Dijkstra

As shown in the counter example in Figure 4-2, there are possible situations in which the Dijkstra single-label heuristic algorithm does not result in the correct answer. However, it was also noted that due to the structure of road blocks in Europe (i.e. a road block is active for the complete country), only errors could occur at the borders of countries.



(a) OneDay



(b) TwoDays

Figure 4-11: Distribution of number of improving routes for OneDay and TwoDays

This indeed does not happen much: the results of the Pareto search and the heuristic are compared, and these match fully. For this specific test set, no differences in results are found.

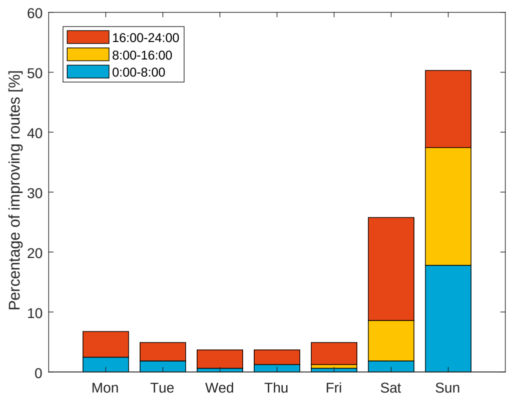
However – albeit without consequences – there do exists situations in which a potential route is thrown away in the heuristic algorithm, that is, a route having a later arrival time but an earlier shift duration than the maintained path. These potential mistakes can be easily detected and recorded, leading to 816 of 10,000 routes where a non-dominated route exists at a certain route but is not considered in the rest of the algorithm. When setting the drivers legislation to a two-day threshold (i.e. threshold of 9 hours and break time of 11 hours) there are 2537 of 10,000 routes that have a potential mistake.

A solution would be to rerun these testcases with the Pareto Search algorithm to ensure correctness.

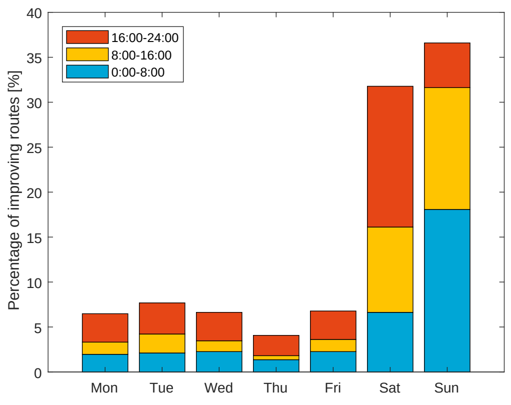
4-5-5 Quality of the Time-Dependent Contraction Hierarchies algorithms

As was shown with a counter example in Figure 4-6, the Pareto search algorithm does not necessarily give an optimal solution. However, no differences in results are found between the Pareto Dijkstra and Pareto TCH algorithms. This means that for this specific road network and test set, there do not exist edges that provide shortest paths for optimal break planning but are removed during preprocessing.

Also the results of the single-label TCH heuristic were checked with the optimal results. These also fully match, indicating that all heuristics given in this thesis are of good quality.



(a) OneDay



(b) TwoDays

Figure 4-12: Time and day distribution of improving OneDay and TwoDays routes

4-5-6 Running time analysis

The Dijkstra and TCH algorithms provide identical results. However, the difference in running time between them is large as can be seen in Table 4-3. The difference between running the Pareto and improved versions is another factor 3. This yields to average query times of 0.0067 (OneDay) or 0.012 (TwoDays) seconds.

Table 4-3: Average query running times of the different algorithms

		One Day		**Two Days**	
		Pareto	**Single-label**	**Pareto**	**Single-label**
Dijkstra	**Average**	12.77 sec	10.18 sec	23.91 sec	18.25 sec
	Maximum	49.90 sec	31.66 sec	66.77 sec	47.41 sec
TCH	**Average**	0.064 sec	0.0067 sec	0.154 sec	0.012 sec
	Maximum	0.670 sec	0.030 sec	1.014 sec	0.034 sec

4-6 Discussion

It should be noted that both shown example routes in Figures 4-9 and 4-10 as produced by the latest break scheduling algorithm are probably not driven in practice. A truck driver is aware of road blocks himself and would probably ask the planner to depart earlier or later and will drive in such a way that he will not unluckily bump into road blocks by just being a few minutes too late. Furthermore, planners will probably take the road blocks into account as common sense when planning routes: i.e. it is not favourable to start a route 21:10 in Austria, while you know that the road is blocked from 22:00 onwards.

Furthermore no differences between results in arrival time of the Pareto and heuristic algorithms are found in the conducted experiments. This is mainly due to the nature of road blocks in Europe (see Figure 4-2): a road block is active for a complete country and if road blocks are active in two neighbouring countries, these always overlap. Pareto search approaches therefore seem useless in this application. However, when using the same algorithms while using additional time-dependent information one would probably find much more differences due to the far more complex structure of congestion compared to road blocks.

The implemented algorithms are not optimized fully. Lots of gain in running times could be reached by memory-efficient programming. However, it was not the goal of this thesis to optimize on running times as much as possible. This thesis did show that it is possible to incorporate planning of breaks in time-dependent contraction hierarchies and obtain reasonable speed-ups. It is up to the programmer implementing the algorithms to optimize the memory consumption and running times as much as possible.

4-7 Summary

In this chapter the influence of incorporating drivers legislation in optimal route planning considering a network only having road blocks is analysed. Therefore, a Pareto search algorithm and a heuristic single-label Dijkstra based algorithm are formed and applied on a Stacked Break Graph and on time-dependent contraction hierarchies (TCH). The Dijkstra Pareto algorithm is proven to give optimal results, whether this might not be the case for the TCH variant. However, all four algorithms yield identical results, so the quality of the algorithms is considered to be good.

A test set representing the freight flows of Europe is used to identify the differences between optimal break scheduling and the currently used break scheduling method. This shows that 6 % of the routes improves if only considering a one-day time limit. For a two day time limit this grows up to 17 % of the routes. The average absolute improvement of improving routes is large: 2:45 hours (for a maximum driving time of 9 hours) or 5:14 hours (for a maximum driving time of 18 hours). It is therefore advisable to take optimal break planning into account when planning routes facing road blocks due to the large impact they can have on arrival times.

Chapter 5

Time-dependent earliest arrival routes with drivers legislation: Parking

In the previous chapter the optimal time and location for taking a break is computed to analyse the influence on the arrival time of optimal break planning, one of the goals of this thesis. However, in reality it is not allowed to take a break at any location, for example, it is not allowed to park your truck at the shoulder of a highway. Therefore, in this chapter, taking breaks is restricted to parking lots. Analogously to the previous chapter, the goal is not only to analyse the influence on the arrival time of this restriction, but also to come up with a method that computes the result efficiently.

In this chapter, first a recap is given of the problem variation, as was previously defined in Chapter 2. After that the earlier introduced Stacked Break Graph is extended in Section 5-2 in such a way that it models the parking lot problem. The Pareto search algorithm defined in the previous chapter can be used to compute optimal routes in this modified graph. To speed-up the computation, a heuristic is introduced that uses time-dependent contraction hierarchies. After that, simulations are run giving insight in the difference in arrival time as well as differences in actual routes.

5-1 Parking variation

The *Parking* variation of the time-dependent earliest arrival routes with drivers legislation is alike the *RoadBlock-OneBreak* variation discussed in Chapter 4. For each road segment road blocks may be defined. If a road block is active, no one can drive the segment. The definition of such a road block is given in Section 2-2-1. Furthermore the route should adhere to the drivers legislation which is valid on a single day. Therefore, the non-stop driving period Δ_d may not exceed the threshold Δ_T of 4.5 hours. After

4.5 hours of driving a break should be taken of 45 minutes (Δ_b), after which again 4.5 hours can be driven. This break can only be taken on a specific link which is located at a parking lot suited for trucks. If the parking lot does not contain links but is directly located next to a road, this road segment of at maximum 3 km of length is considered as the parking lot. It might therefore be possible that a shortest path runs across a parking lot while no break is taken at this parking lot.

5-2 Stacked break graph with parking lots

The previous chapter introduced the stacked break graph that has two layers: one representing the nodes where no break has been taken, one where a break has been taken. In this graph all edges are doubled in such a way that on every location in the network a break can be taken. This definition can easily be adjusted for parking lots by only adding links between the different layers if the link is representing a parking lot. This idea is shown in Figure 5-1.

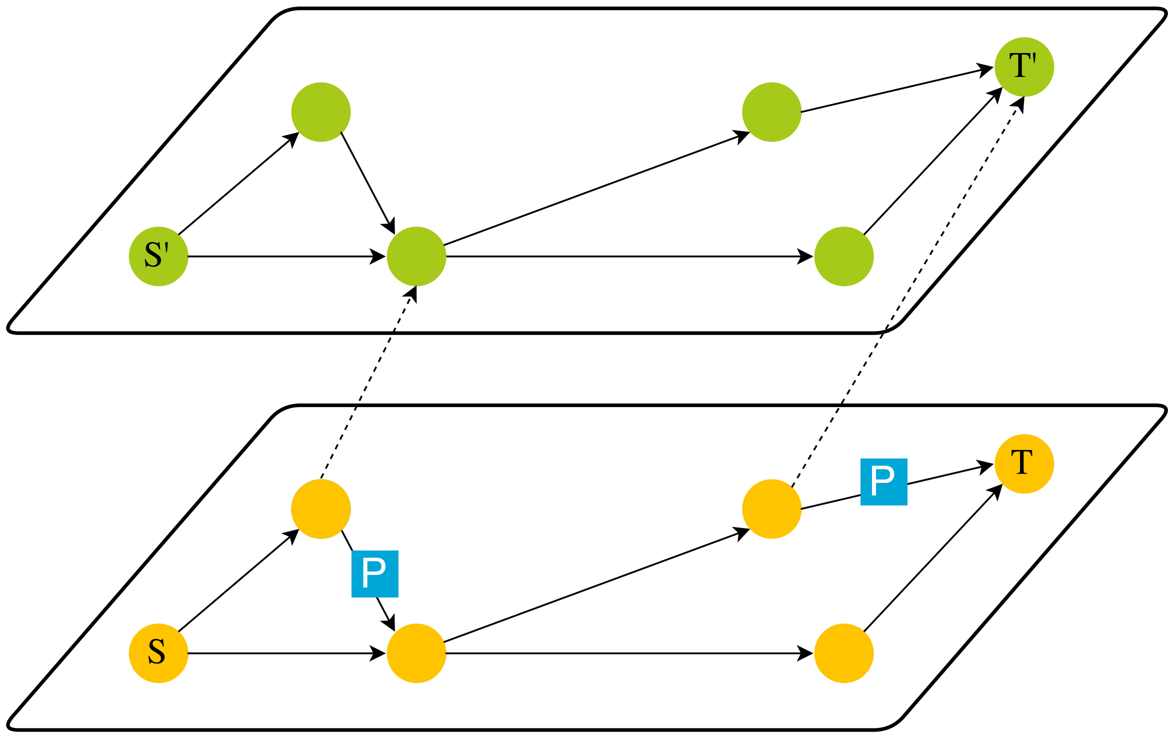


Figure 5-1: Example of the stacked break graph with only allowing breaks at parking lots

After constructing this graph, the Pareto search algorithm introduced in Section 4-2-1 can be applied without any modifications. This again yields optimal results as was already proven in the previous chapter.

5-3 Heuristic using time-dependent Contraction Hierarchies

The Dijkstra-like algorithm introduced in the previous chapter can easily be adjusted such that breaks are only allowed at parking lots. For time-dependent contraction hierarchies, this is not the case.

During the preprocessing phase of the time-dependent contraction hierarchies only links of shortest paths are considered in the final hierarchy. Parking lots are generally not located on a shortest path, but are a (small) detour of this path. Brauer [10] overcomes this problem by giving the parking lots the highest node order and makes sure that these are not contracted during preprocessing. Using non-modified forward and backward queries, optimal paths via parking lots are then retrieved. Disadvantage of this method is the long query time due to the form of the preprocessed graph: it is not flat and sparse any more which is required to obtain fast running times. Computing an optimal route using this method takes 14 seconds on average on a network of Germany with 7 million nodes and 5449 parking lots.

To speed-up the computation compared to Brauers approach, the idea is to introduce a heuristic method. This idea is based on the fact that it is possible in a preprocessed graph to obtain a shortest path from start to a parking lot, and from a parking lot to the destination. The optimal route can then be computed by planning routes from start to every parking lot in the network, and from every parking lot toward the destination. The route having the earliest arrival time while not violating the threshold Δ_T of maximum consecutive driving, is then the optimal route. With an average query time of 0.2 ms [7], and about 10 000 parking lots in the network, this still leads to a total query time of 30 seconds per start-destination pair (i.e. $2 \cdot 10000 \cdot 1.5ms$). Instead, the idea is to find a subset of parking lots that form viable options.

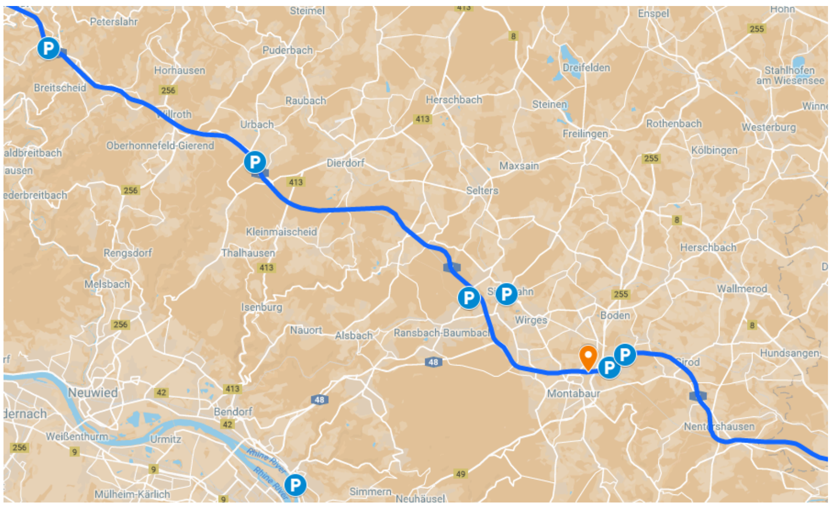


Figure 5-2: The optimal route runs from North-West to South-East in this map. The optimal break location is indicated with the orange marker. Using the backward Dijkstra with a time limit of 30 minutes, parking lots in the surroundings are searched. During the next phase, routes are computed to each of these parking lots, and from these parking lots toward the destination.

From the computations in Chapter 4 it is known where the optimal break location is – without concerning parking lots. Most likely, the optimal parking lot is near this location. Since this break is scheduled as late as possible (without giving up on earliest arrival time), a backward Dijkstra search is enough to obtain the possible options for parking lots. Depending on the desired solution quality a maximum time limit of reachable parking lots can be set. An example of the results of a backward Dijkstra searching for parking lots is shown in Figure 5-2.

To conclude, the overall procedure for the heuristic approach using time-dependent contraction hierarchies is as follows:

1. Compute the optimal break location using algorithms of Chapter 4
2. Compute a set of nearby parking lots using a backward Dijkstra search
3. For each of these parking lots:
    (a) Compute the route from start to the parking lot using TCH (no breaks considered)
    (b) Compute the route from the parking lot to the destination using TCH, starting at Δ_b minutes later than the arrival time at the parking lot
    (c) Check whether both routes are feasible (i.e. do not exceed the shift duration threshold Δ_T)
4. Return the route with the earliest arrival time at the destination

5-4 Simulations

Two algorithms are introduced: the optimal *SBG-Parking* and the heuristic approach *TCH-Parking*. Both algorithms are analysed in this section, to identify the influence if restricting to parking lots, as well as the solution quality of the heuristic approach. Illustrations of maps with produced routes are used to visualise the behaviour of the algorithm and to give insight in the problem.

5-4-1 Test set-up

Both algorithms are tested on a detailed network of Europe provided by HERE [29]. This network has 31 million nodes and 66 million edges, which is about 5 times as large as the less-detailed network used in the simulation tests of the previous chapter. This level of detail is needed since links at parking lots are otherwise not included in the network.

Data on parking lots is taken from the database of Truck POIs (Points of Interests) provided by HERE. Used parking lots used are marked as restaurants, truck parkings or truck rest areas. This leads to a list of 9306 parking lots, spread along the 48 countries included in the dataset.

Road blocks are identical compared to the test set-up in Section 4-5-1. The 10000 test instances are converted such that they represent nodes in the more detailed network used in this chapter. Furthermore, the test instances are rerun on the detailed map with the optimal Pareto search algorithm without concerning parking lots to be able to make good comparisons.

All tests are run on a laptop, having 16GB memory and an Intel Core i7-3740QM 2.70GHz processor with 4 cores.

5-4-2 Verification of results

The result are verified by comparing optimal routes without restrictions on parking lots with optimal routes that do consider parking lots. It is never the case that a route becomes faster if the break was restricted to parking lots – which is indeed expected. There are many routes having identical arrival times. Some of these were checked by plotting the route on the map. All of these had a parking lot along the optimal path, close to the optimal break location. These parking lots were located directly along the road, thus not causing any additional travel time.

Furthermore some example routes with and without parking lots were plotted on a map to see the differences. These routes seem feasible and logical.

5-4-3 Influence on arrival time of incorporating parking lots

First the results of the Pareto Search with parking lots are compared to the results without parking lots. Of the original test set, only routes are considered having a break in the OneDay simulations, leading to a total of 2609 routes.

The restriction of parking lots results in 143 (5.5%) routes not possible to drive any more due to the exceedance of the drivers legislation rules. Their average travel time is 11 hours and 36 minutes, indicating that these are generally long routes including waiting time somewhere in the route (recall that the maximum driving time is 2 times 4.5 hours). On average a route takes 3 minutes longer if a break should be taken on a

Table 5-1: Comparison of with and without parking lot restriction

Average difference in travel time	3 minutes
Maximum difference in travel time	382 minutes
Minimum difference in travel time	0 minutes
Standard deviation of difference in travel time	14 minutes
Number of routes with identical travel time	765 (29 %)

parking lot. This indicates that there are enough parking lots available along the routes and that generally only small detours are required causing little delay. There are also routes not affected by the restriction of parking lots: 29% did not change in travel time, due to parking lots directly located along the road. The maximum difference in travel time was 382 minutes, caused by a combination of an unfortunate road block and an optimal break location which was not along a highway, where most truck parking lots are situated.

Breaks planned along such secondary roads is one reason for deviating travel times. Another reason is the lack of parking lots in certain countries. For example in the Baltic countries, Albania and Bosnia and Herzegovina little number of parking lots for trucks are available. It is questionable whether there are indeed less facilities for trucks in these countries, or whether there is just a lack of data. An example of such a route is shown in Figure 5-3.

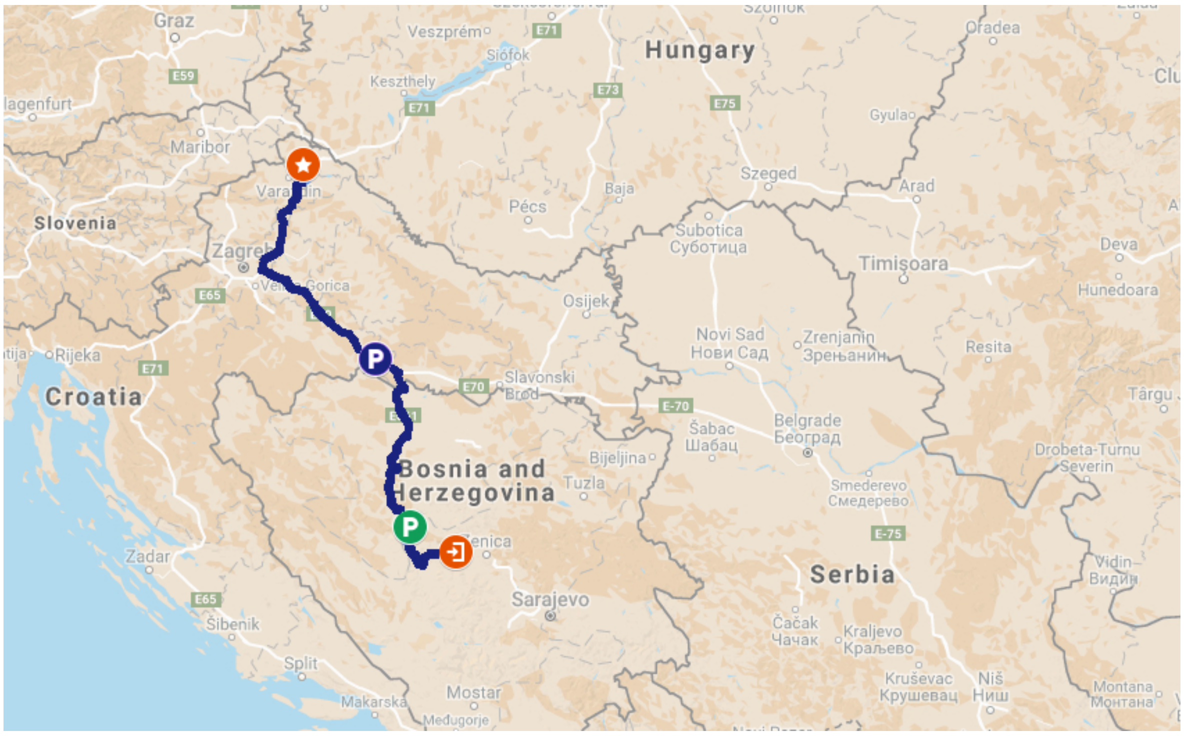


Figure 5-3: Example route with large difference in locations of parking lots: the route travels from North Croatia toward Bosnia and Herzegovina. The optimal break location (green) is located near the destination. However, no parking lot is available around there. This causes the break to be planned at the border of Croatia (purple), leading to a higher shift duration at the destination.

5-4-4 Influence on route choice if restricted to parking lots

Besides the influence on arrival time it is also interesting to see the influence on route choice of the routes which are restricted to parking lots. Therefore, 50 routes with and without parking lots were plotted in detail to see the differences between both. Results of overlap between these routes is shown in Table 5-2. A large amount of these routes (80%) have an overlap of 99 up to 100 %. These routes can be seen identical – except for possibly taking the off-ramp and next on-ramp in order to travel toward the parking lot. An example of such a small detour is shown in Figure 5-4.

Table 5-2: Overlap between routes that do or do not restrict taking breaks at parking lots

Overlap of 99-100 %	80 % of the routes
Overlap of 90-99 %	2 % of the routes
Overlap of 50-90 %	2 % of the routes
Overlap of 0-50 %	12 % of the routes
No feasible solution found	4% of the routes

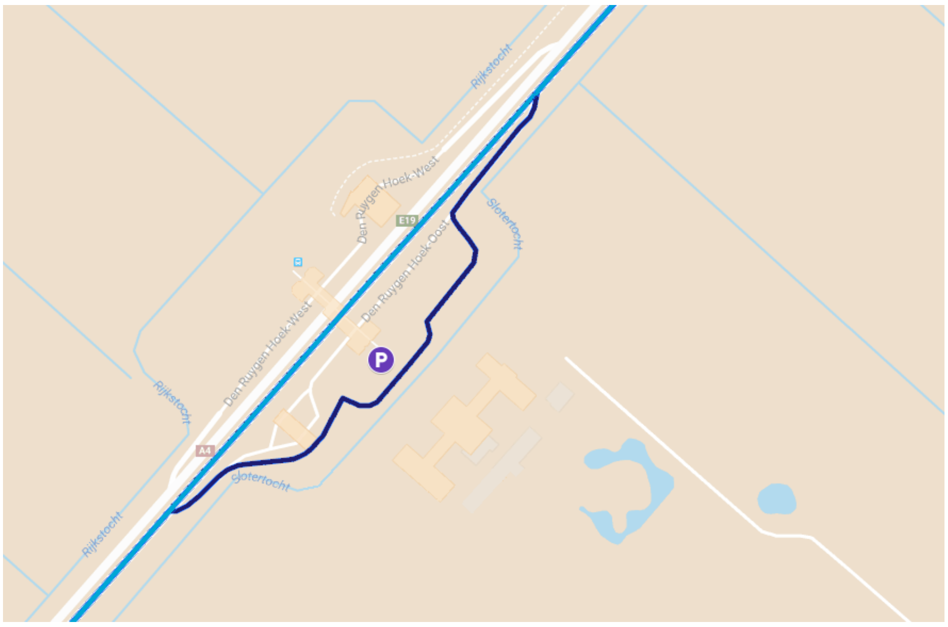


Figure 5-4: Example of a route with limited effect on route choice: the truck driver only needs to take an off-ramp and next on-ramp to get to the parking lot, rest of the route is identical to the optimal route

An example of a large difference in routes is shown in Figure 5-5. Due to the absence of parking lots around the optimal break location which is situated along a secondary road, a completely different route through Germany turns out to be the best route if restricted to parking lots.

It can be concluded that the difference in routes is generally small, but for 16 % of the routes parking lots cause large changes in routing decisions. Some of them are caused by the absence of parking lot data in specific countries, but most are caused by absence of parking lots on the specific routes chosen. Another 4 % of the routes are not feasible any more if breaks are restricted to parking lots.

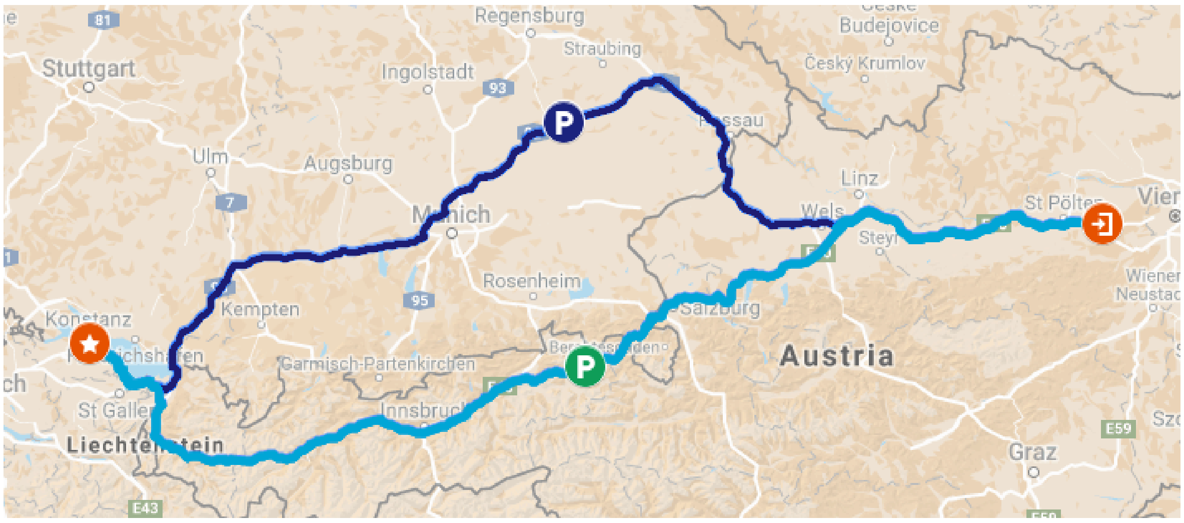


Figure 5-5: Example of a large difference in routes due to restriction of parking lots. The Southern route is the optimal route as computed in the previous chapter. The optimal break location is indicated with the green sign. However, this location is situated along a secondary road in Austria ("Autostraße" instead of "Autobahn"). No parking lots can be found in the surroundings. The best alternative restricted to parking lots is driving through Germany using the highways and visit a parking lot near the highway

5-4-5 Evaluation of heuristic approach

The heuristic approach first searches for parking lots nearby, followed by computing routes from start to each of the parking lots, and from each of the parking lots toward the destination. For this simulation experiment only feasible routes of the previous section are considered, that is, routes having a break and that are feasible when restricted to parking lots. This results in 2467 routes.

First parking lots nearby are computed for different ranges of search horizons around the optimal break location. All parking lots found within 10 to 120 minutes (with steps of 10 minutes) are obtained. This leads to an average number of parking lots per route as shown in Figure 5-6. For 10 minutes of travel time nearly no parking lots are found, whereas 120 minutes lead to 34 possible parking lots per route on average.

Table 5-3: Comparison of deviation between no parking lots and solution which are restricted to parking lots. Only the best solution of the heuristic is taken into account when calculating averages.

	Optimal	30 minutes	60 minutes	90 minutes	120 minutes
Average	3 minutes	27 minutes	34 minutes	33 minutes	35 minutes
Maximum	382 minutes	2231 minutes	2231 minutes	2280 minutes	2280 minutes
Minimum	0 minutes	0 minutes	0 minutes	0 minutes	0 minutes
Median	1 minute	3 minutes	3 minutes	2 minutes	2 minutes
# of solutions	2467	1325	1936	2129	2221
% of solutions	100%	53.7%	78.5%	86.3%	90.0%

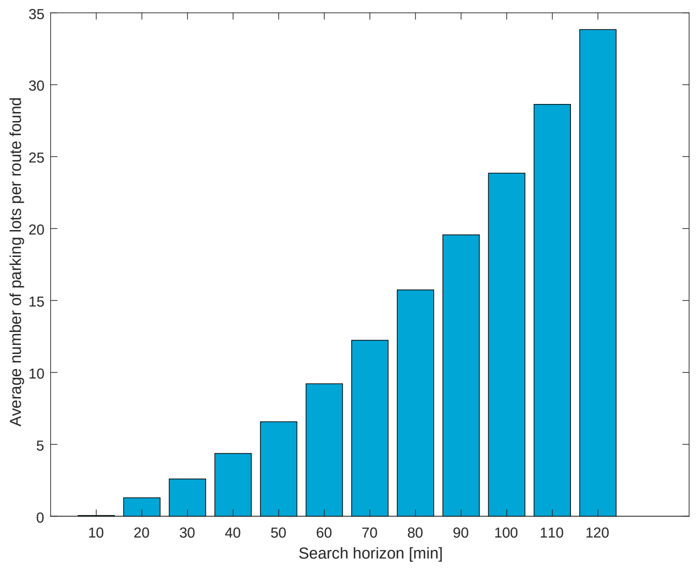


Figure 5-6: Average number of parking lots per route (for each of the 2467 feasible routes)

Next the routes from start to each of the parking lots and from each of the parking lots toward the destination are computed. The best route (with the earliest arrival time) is checked against the optimal solution. A comparison between the route without parking lot restriction and the results of the optimal and heuristic algorithm is shown in Table 5-3. It can be seen that the average deviation for the heuristic is about 30 minutes and increases slightly with an increased search horizon. However, one would expect a better solution quality (and thus a lower average) as the number of potential parking lots increases. However, the number of feasible solutions increases as well. It seems that some of these far away parking lots give a bad solution, which cause a worse average deviation. This can also be seen from the median which is generally low: 2 minutes for 120 minutes of search time. This indicates that many solutions provided by the heuristic are of good quality, but few are not.

The composition of best routes is visualised in Figure 5-7. For a search horizon of 10 minutes, most routes are not possible to compute due to a lack of parking lots found: in nearly 95% of the cases no parking lot is found for the route. Of the remaining routes that did find one or more parking lots, most test instances do not have a feasible route. From 20 minutes onward optimal routes are found. For 120 minutes of search time about 80% of the routes are optimal or near optimal (arrival time less than 10% difference). Ecen with a seearch horizon of 120 minutes, some routes remain unfeasible. This is caused by routes travelling through countries with limited parking lot availability. For example, in Figure 5-3, the driving time between the optimal break location and the optimal parking lot is 3 hours: resulting in not being found by the heuristic method within 2 hours of search time.

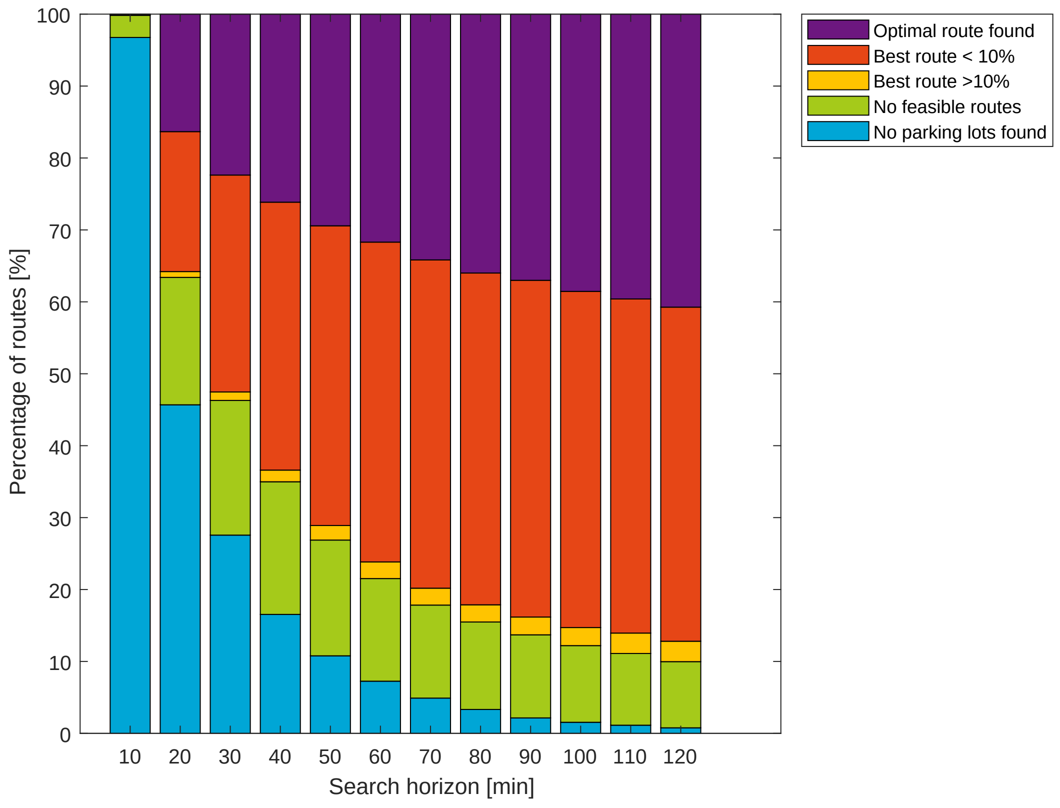


Figure 5-7: Composition of best results of heuristic, for each of the search ranges of 10 to 120 minutes. 100% corresponds to the 2467 feasible routes

5-4-6 Running time analysis

The Dijkstra Pareto search algorithm with and without parking restrictions, respectively, requires some running time: 41.5 seconds (no parking) and 85.9 seconds (parking) on average. It should be noted that the network is detailed (having 31 million nodes) and all considered routes are long (that is, all routes require a break and thus last longer than 4.5 hours of driving time). Result are shown in Table 5-4.

Table 5-4: Running time

	Dijkstra Pareto search		**Time-dependent contraction hierarchies**	
	No parking	**With parking**	**60 minutes**	**120 minutes**
Average	41.5 sec	85.9 sec	0.46 sec	1.61 sec
Maximum	120.4 sec	167.1 sec	3.43 sec	7.19 sec

The running time of the heuristic method can be split into two parts: finding parking lots using the backward Dijkstra, and computing all possible routes. The running time of both components per search horizon is shown in Figure 5-8. For the computation of routes most time ($> 95\%$) is spent in computing the exact route (i.e. all nodes in the path) and checking the feasibility by running a (slow) non time-dependent Dijkstra over this computed path. This could be significantly improved by implementing the tracking of shift durations within the preprocessed contraction hierarchies.

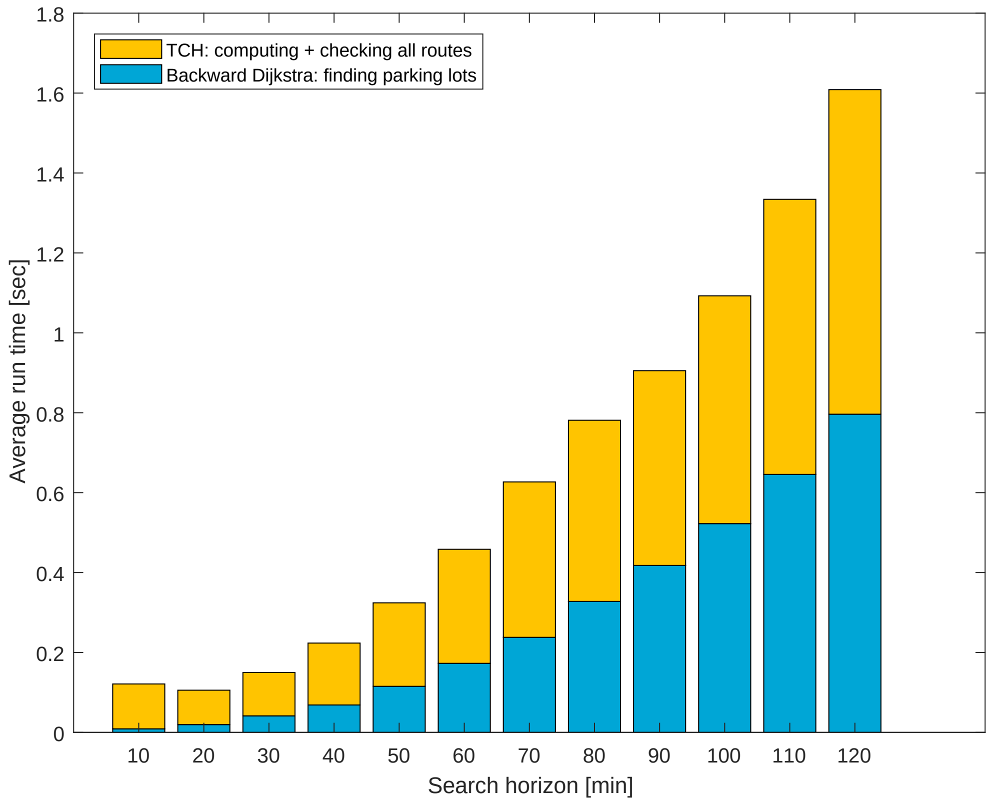


Figure 5-8: Composition of running time for the heuristic

Overall, a significant improvement in running time can be reached if using the heuristic method. For a search horizon of 60 minutes the running time can be improved with a factor 160, while 75 % of the solutions have a good quality.

If compared with Brauer [10] who did all computations using contraction hierarchies, the optimal running time is comparable if scaled for the less-detailed network size he uses: he got 14 seconds of average running time for a network of 7 million nodes. Scaled for the 31 million nodes, this leads to a imaginary running time of 62 seconds using Brauer's algorithm, only a bit less than the Dijkstra Pareto search algorithm. However, the heuristic provides significant improvement in running time, only requiring 0.46 seconds using a search horizon of 60 minutes: a speed-up of more than 100 times for 80% chance on a route having less than 10% difference with the optimal route.

5-5 Discussion

The lack of data on parking lots in some countries has its influence on the computed routes. It is questionable whether there are indeed less parking lots suitable for trucks or whether it is allowed to park your truck anywhere at the shoulder in these countries.

The computation of parking lots nearby is based on a backward Dijkstra search. Another method is computing the parking lots nearby by using geodesic distances. This is a generally much faster method than Dijkstra queries. It is interesting to see whether the geodesic distances would lead to the same quality of final routes computed.

The average influence in arrival time of incorporating parking lots is only 3 minutes. It is questionable whether this is enough to include the exact planning of parking lots in the used algorithms. Probably a better heuristic method would be to check where the optimal break location is. If this location is not at a highway or if it is in a country known to have little parking availability, then the route can be recomputed with incorporating optimal break planning restricted to parking lots.

5-6 Summary

In this chapter it is investigated what the influence on the arrival time is if breaks are restricted to parking lots. On average, this influence is only 3 minutes and does not change the actual routes driven. However, for 20% of the analysed routes the difference is much larger, leading up to 6 hours of additional travel time (on routes having a maximum driving time of 9 hours). Large differences are caused by lack of (data on) parking lots in specific countries and an optimal break location around other places than highways.

Besides an optimal algorithm also a heuristic method is presented. This heuristic searches for parking lots near the optimal break location and plans routes via these parking lots. For a search horizon of 1 hour, 80% of the solutions have a deviation of less than 10%. This suggests that the optimal parking lot is generally nearby the earlier computed optimal break location.

Chapter 6

Preferences in route choice

The second part of this thesis focusses on incorporating preferences in route planning. It is assumed that the fastest route is not necessarily the best route according to the goals of the logistics company. Therefore, this chapter identifies the aspects determining a preferred route. This is done by first identifying the goal and sub-goals of the freight company in Section 6-1. Each of these sub-goals have their influences on routing decisions. These are identified, leading to a list of attributes which should be taken into account to compute the best route. Since the goal of this thesis is to identify the influence on arrival time if incorporating preferences, also a good estimation on the influence of these attributes is needed. Therefore, several choice studies are identified, after which one is selected as the basis for the remainder of this thesis to compute best routes.

6-1 Goals of a freight company

The freight company wants to maximize profit by transporting goods using their trucks. Srinivas et al. [46] identified three factors which compose the overall profit of a truck company:

1. **Minimizing costs of the entire route plan.** The costs of a truck company consists of fuel consumption, vehicles, driver wages and more.
2. **Driver satisfaction.** A dissatisfied driver results in losses for the firm on the longer term. A truck driver having the same goals as the company is most beneficial.
3. **Customer satisfaction.** Delivering the right products on time satisfies the customers. Satisfied customers are likely to put new orders at their current transport company.

These three factors together form the overall logistics performance, which is considered as the profit of the company. Each of the three factors can be influenced by route choices made. For example, costs can be minimized by avoiding toll roads. All three factors arediscussed in the following sections, identifying the relevant route attributes.

6-1-1 Minimizing costs

Several costs are involved in operating a truck. The break-down of costs in the USA is shown in Figure 6-1 [50]. It can be seen that the driver wages are the largest factor (35%), followed by the fuel costs (34%).

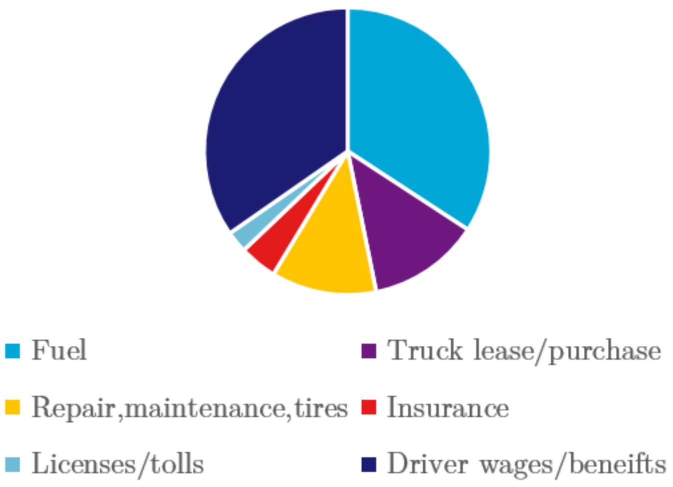


Figure 6-1: Break-down of Average Marginal Costs per Hour in 2014 in the USA

Driver wages are directly influenced by the number of working hours of truck drivers. For the context of this thesis it is assumed that driver wages are only influenced by the duration of the travel time, i.e. minimizing the travel time leads to lower wages.

Costs spend on fuel is another factor that can be influenced by different route choices. First of all, the fuel consumption can be reduced by minimizing the driving time. Secondly, one can compute more energy efficient paths. Best paths are then computed using the fuel consumption per link. Scora et al. [44] noticed that shortest paths are generally also the most fuel saving paths, if neglecting road grades and heavily congested roadways. Third, truck drivers can be motivated to change their driving behaviour. Van der Voort et al. [55] presents a prototype fuel-efficiency support tool, which gives in-vehicle advice. Eco-friendly techniques such as quickly shifting up gears and anticipating on traffic conditions to avoid unnecessary braking are implemented. With a simulator experiment it was shown that a fuel reduction of 16% could be reached with the use of their prototype compared to normal driving. However, it is not within context of this thesis to directly correct or anticipate on the behaviour of the driver. This option is therefore not considered as an option to reduce fuel consumption by making different route choices. The last way to reduce fuel consumption is by letting trucks cooperate by forming platoons [32]. By letting trucks drive closely after each other using smart technology, fuel consumption is decreased. However, since the scope of this thesis is just routing one vehicle, this option is not taken into account.

The costs of hiring or purchasing trucks, maintenance and paying the insurance is not assumed to be influenced by choosing different routes.

The last category of costs are toll and license costs. By skipping a toll road, costs are reduced. In a route choice experiment [47] it was shown that truck drivers are insensitive to the inclusion of toll roads if they do not have to pay for toll costs themselves. However, if they own their own truck, they are very sensitive to avoiding such toll roads. However, if the time pressure is high – such as with transporting refrigerated goods – this effect is not seen.

To conclude, three route attributes are important in minimizing costs in the perspective of route choices:

1. Travel time
2. Fuel-consumption
3. Toll roads

6-1-2 Driver satisfaction

The second goal of a freight company is to ensure the satisfaction of the truck driver. A dissatisfied driver might not do its ultimate best to reach the goals of the company and deliver orders as fast as possible to make profit. A higher satisfaction of a truck driver can be reached by incorporating its preferences during route planning.

For truck drivers it is important to be able to drive on comfortable roads. One aspect is the type of road: truck drivers like to stick to highways [3]. Furthermore some analysis showed the influence of road curvatures, but only for heavy goods vehicles [42].

Facilities along the route is a factor analysed by multiple studies. Experiments showed that long-haul truck drivers prefer routes having enough parking availability and refuelling possibilities [42] [47]. Arentze et al. [3] also comes to this conclusion, but considers restaurants instead of just parking lots.

Besides facilities, truck drivers also have preferences on the planning of visiting such facilities, that is, the planning of breaks. Some truck drivers prefer sleeping during the night, whereas others have strong preferences of driving during the nights, when roads are mostly empty [51]. However, the time of the day at which the route is driven is mostly the result of the starting time of the trip, which cannot be shifted in the context of this thesis. Furthermore, some truck drivers prefer to split their breaks to prevent sleepiness or to be able to rest longer at a more pleasant facility. Others just like to drive as far as they get without splitting breaks or rests [52].

Some other factors analysed in experiments with truck drivers are not found to be significant. Fuel consumption was not a reason to change routes [47], probably because fuel costs are paid by the company and not by the driver himself. For the same reason also avoiding tolls was not found to be significant in the perspective of drivers [42] [47].

To conclude, three factors are important in obtaining a good driver satisfaction in the perspective of route choices:

1. Road type
2. Parking lots
3. Planning of breaks

6-1-3 Customer satisfaction

The third goal of a company is to keep customers satisfied, by offering good services. Coulter [12] identified five service dimensions for freight transportation:

1. Reliability (i.e. precision of predicted transit time)
2. Risk Avoidance (i.e. accident-free transportation, insurance)
3. Customer Service (i.e. quality of personnel)
4. Personalizing (i.e. personal service, knowing needs of customer)
5. Handling (i.e. efficient and effective handling of shipment)

Since this thesis focuses on routing of trucks and not on the process of handling customers and loads, only the dimensions Reliability (1.) and Risk avoidance (2.) are of interest for route choices and are discussed in the remainder of this section.

Many different definitions for travel time reliability exist, including different methods for determining the travel time reliability of a certain route. What most definitions have in common is that they all relate to the (shape of the) travel time distribution [54]. The wider the time distribution shape is, the more unreliable the travel time is considered. Four main methods of calculating the travel time reliability are [36]:

1. **Statistical range methods**. These use standard deviation statistics to present the range of travel times the traveller might be presented to.
2. **Buffer time measures**. These indicate the extra percentage travel time which a traveler should take into account to arrive on time with a high chance.
3. **Tardy trip measures**. These indicate the amount of trips that result in late arrivals or the delay that occurs during the worst trips.
4. **Probabilistic measures**. These calculate the probability that travel times are larger than some threshold depending on the travel time itself (for example the median travel time + 20%).

All methods calculate the travel time reliability for the complete trip, using the statistical data available for the travelled road segment. In a road network, variances in travel times are mainly due to congestion. Roads having higher chances on congestion result in a lower reliability of travel time. For route choices it is therefore assumed that increasing arrival time reliability can be obtained by avoiding congestion, independent of the actual method for computing the reliability.

In terms of customer satisfaction, risk entails safe and accident-free transportation. Although roads get safer and fatalities with passenger cars decrease, this is not the case for truck drivers. Reasons for the constant number of injuries are given by Craft [13]. They concluded that fatigue of drivers is responsible for 13% of the truck crashes overall. A lot of attention has been given to reduce this sleepiness of drivers.

Chen et al. [11] showed that having two or three breaks instead of one break in a 10-hour driving period, decreases the crash risk significantly. Although this study was performed in the USA, having different drivers legislation rules than Europe, it is still possible to take some of these findings into account in obtaining risk avoided routes, since splitting of breaks is also allowed in Europe. Pylkkönen et al. [41] focussed their research on sleepiness itself instead of actual accidents. They found out that the first night of driving had a higher chance of sleepiness than the consecutive nights. The morning shift was safest, directly followed by the day and evening shifts. This would imply that the best decision concerning risk avoidance would be to avoid driving at night.

The risk of a truck driver can also be reduced by avoiding certain countries or regions. Recently the Calais "jungle" of migrants caused lots of problems of migrants trying to clamp on trucks secretly to travel toward Great Britain [49]. This can be a reason for truck companies to avoid routes along Calais if not necessary. Using another ferry might provide an alternative. Also other countries or road sections might be less safe and better be avoided to obtain a higher safety level.

To conclude, two factors are important in keeping customers satisfied, in the perspective of route choices:

1. Congestion
2. Risk

6-1-4 Conclusion

The three objectives of minimizing costs, ensuring driver satisfaction and customer satisfaction lead to a list of route attributes that are important to consider when computing routes that are in line with the goals of the company. These are shown in Table 6-1 and used in the next section.

Table 6-1: Route attributes important for maximizing profit

Costs	**Driver satisfaction**	**Customer satisfaction**
Travel time	Road type	Congestion
Fuel consumption	Parking lots	Risk
Toll roads	Planning of breaks	

6-2 Choice experiments on route choice

The goals of the truck company and corresponding attributes in route choices are given in Table 6-1. However, the goal of this thesis is to analyse the influence on arrival time of a generally preferred route. Therefore, it is relevant to know the relative importance of each of these attributes. This could be obtained by running an experiment where participants can rate the attributes. However, it is not into the scope of this thesis to perform such experiments. Therefore, alike experiments are identified after which one is selected that covers most attributes and is best usable in a route planning application.

6-2-1 Experiments and their attributes

Five surveys are found that analyse routing decisions: Arentze et al. [3], Rowell et al. [42], Sun et al. [47], Hess et al. [30] and Prato et al. [40]. Their methods, participants and attributes are described in this section.

Arentze et al. [3] performed a *stated choice experiment* for the Dutch Ministry of Transportation among 100 drivers active in the Eindhoven region. In a stated choice experiment the researcher itself constructs a set of hypothetical choice alternatives, according to a list of attributes and their possible values. The participants choose the alternative which they prefer most from a choice set. Using this technique a set of reliable parameters for each of the attributes in the experiment is obtained. These parameters are used in a *utility function* that gives the score of a route giving its properties. The route having the highest score within a set of routes is assumed to be the most preferred one.

The research included the following route attributes: travel time, congestion, road category, road pricing, passing through urban areas and the presence of a restaurant facility. To make the choice alternatives more realistic, context of the situation is provided. *Context variables* that were included are travel time, time of day and size of truck. Including these context variables increases the validity of the model, but the effect should be measured to be able to generalize choice decisions made. The travel time context variable takes values of at maximum 30 minutes, thus only short routes are considered in this experiment. An example choice alternative including the context is shown in Figure 6-2.

Rowell et al. [42] did not conduct a stated choice experiment, but asked their respondents instead to rate items that are potentially influencing their routing decisions. They choose this approach to be able to easily obtain a large set of responses. The goal was not to determine how the respondent's priorities vary, but which priorities were common and different among the respondents. The test group consists of 850 truckers in

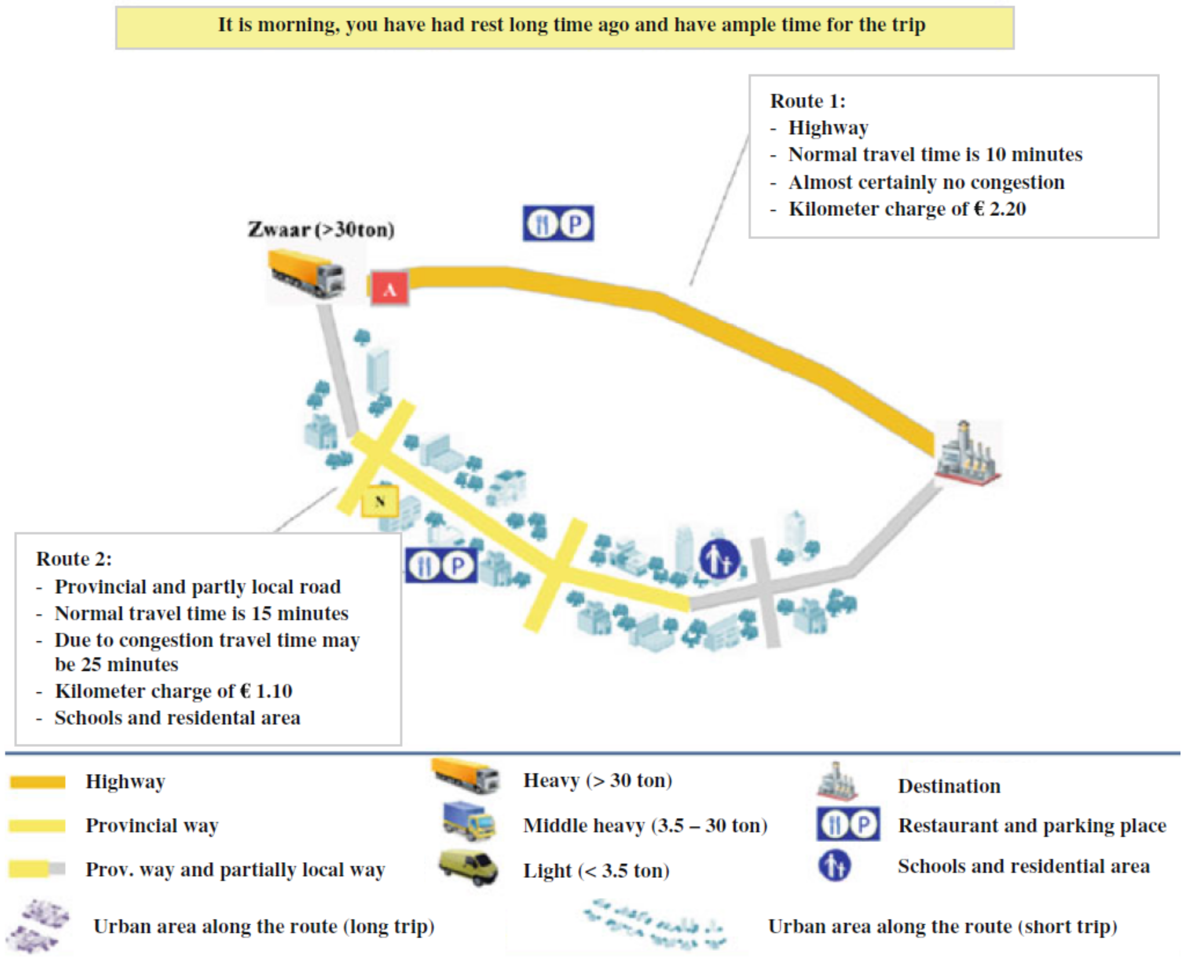

It is morning, you have had rest long time ago and have ample time for the trip
Route 1:
- Highway
- Normal travel time is 10 minutes
- Almost certainly no congestion
- Kilometer charge of € 2.20
Zwaar (>30ton)
A
N
Route 2:
- Provincial and partly local road
- Normal travel time is 15 minutes
- Due to congestion travel time may be 25 minutes
- Kilometer charge of € 1.10
- Schools and residental area
Highway
Provincial way
Prov. way and partially local way
Heavy (> 30 ton)
Middle heavy (3.5 – 30 ton)
Light (< 3.5 ton)
Destination
Restaurant and parking place
Schools and residential area
Urban area along the route (long trip)
Urban area along the route (short trip)

Figure 6-2: Example choice alternative of Arentze et al. [3]

the Washington area. Attributes taken into account concerning routing decisions were travel distance, time, costs, avoiding congestion, avoiding tolls, road grade and curvatures, refuelling locations and availability of support along the road. The data was analysed using an *Item Response Theory (IRT)* and a *Latent Class Analysis (LCA)*. IRT results in attributes that are most differentiating in the respondent's replies, whereas LCA groups respondents into classes.

Sun et al. [47] uses again another survey method. They asked how often a certain factor influences the decisions of a participant in its route decisions. Participants rated four factors: travel time predictability, availability of parking locations, the presence of fuel stations and the effect on fuel consumption. The test group consisted of 252 truck drivers that were asked to fill in the survey at rest areas in the USA.

Hess et al. [30] uses GPS data to analyse route choices made by 709 heavy goods vehicles in England. Unlike the previous three experiments, this is a *revealed preference* study, meaning that no surveys are used that might bias the results. The interest of the experiment lies mainly in the usage of different road types in England. Therefore, only travel time and road type are included as attributes in the experiment.

Prato et al. [40] also runs a revealed preference study. The used 13000 observations around the Copenhagen area in Denmark and analysed travel time, congestion, costs, left turns and right turns. Disadvantage is that the research is focussed on car drivers instead of truck drivers. It is questionable what the effect of vehicle type is on the resulting parameter values.

The five studies and their attributes are compared with each other in terms of suitability for this thesis in the following section.

6-2-2 Overview of experiments

The five analysed experiments and their included attributes are shown in Table 6-2. The goal of this section is to select one of the studies that best represents the preferences identified in this chapter and is most usable in the next chapter where it is used for actually planning routes.

Table 6-2: Five experiments on route choices and their included attributes. Stated preference studies are marked as SP, revealed preference studies as RP.

		Arentze	**Rowell**	**Sun**	**Hess**	**Prato**
		SP	**SP**	**SP**	**RP**	**RP**
Costs	**Travel time**	✓	✓	✓	✓	✓
	Toll roads	✓	✓			✓
	Fuel-consumption			✓		
Driver	**Road type**	✓	✓		✓	
	Parking lots	✓	✓	✓		
	Planning of breaks					
Customer	**Congestion**	✓	✓	✓		✓
	Risk					

A Revealed Preference (RP) study would be the best to use since no survey bias is present. Furthermore, preferences are obtained from real routes in the road network and therefore these represent existing options. However, the identified revealed preference studies contain only little number of attributes. A stated preference study seems therefore the option to go for.

Arentze et al. [3] and Rowell et al. [42] include the most attributes. Only fuel-efficient driving, break planning and safe driving are not included. It seems plausible to select one of the two.

Arentze et al. [3] conducted a stated choice experiment, whereas Rowell et al. [42] used Item Response Theory to analyse their data. A stated choice experiment is more practical in usage when it comes to computing routes due to the resulting exact parameters instead of a list of most discriminating attributes. Furthermore Rowell et al. did analyse toll costs, but did not include these in their results due to a high correlation with travel time. This reflects that in this experiment truck drivers do not have to pay toll themselves, making them insensitive to additional pricing. Another advantage of the study of Arentze et al. is that the experiment was run in the Netherlands instead of the USA, which is in line with the European scope of this research.

Therefore, the resulting parameters of Arentze et al. are used in the next chapter to compute actual preferred routes. From now on, fuel-efficient driving, break planning and safe driving is therefore omitted. Since Arentze et al. included urban areas in their experiment, this attribute is included in further analysis as well. All used parameters are shown in Table 6-3.

Table 6-3: Values of parameters taken from Arentze et al. [3]. Effect coding is used to express the parameters, resulting in 2 parameter values for each 3-level attribute. Section 7-2-1 contains more information on effect coding.

$\beta_{TravelTime}$	-4.579 (log)	
β_{road}	1.132 (Highway)	-0.799 (Local roads)
β_{urban}	0.450 (No urban area)	-0.669 (Urban area with school)
$\beta_{congestion}$	1.512 (No congestion)	-0.989 (Long delay)
$\beta_{pricing}$	1.036 (No pricing)	-1.053 (With pricing)
$\beta_{restaurant}$	-0.127	

6-3 Summary

In this chapter three goals of a truck company are identified: minimizing costs, maintaining a good driver satisfaction and keeping customers satisfied. Each of these three aspects has its own influence and can be affected by different routing objectives. Since the objective of this thesis is to know the influence on the arrival time if these attributes are incorporated, exact parameter values giving the relative influence of each of these aspects should be known. Because of performing choice experiments is not one of the goals of this thesis, five earlier performed studies are analysed as potential candidates for providing parameter values. The stated choice experiment of Arentze et al. [3] includes most earlier identified attributes and is best suited, and is therefore selected to use in the next chapter. It should be noted that with the selection of this study, the aspects of fuel-consumption, break planning and safe driving are omitted from obtaining route choices.

Chapter 7

Incorporating preferences in time-dependent earliest arrival routes

The goal of this thesis is to analyse the difference in arrival time between the best and fastest route. Therefore, the previous chapter identified the route attributes that should be taken into account when obtaining such a best route, from the viewpoint of a logistics company. The stated choice experiment of Arentze et al. [3] best represents these attributes and is therefore used as a basis in this chapter.

This chapter starts with the introduction of a method to compute a route based on the utility function. Next, the effect coded parameters of Arentze et al. are converted to dummy coding, weighted and calibrated such that the utility function is usable for the computation of routes. After that, in Section 7-4 fastest routes are compared to best routes according to the utility function. In Section 7-5 a sensitivity analysis is conducted to test the influence on arrival time of each of the considered route attributes.

7-1 Using utility functions to compute the best route

Arentze et al. [3] conducted an experiment analysing the influence of travel time, road type, passing through urban areas, congestion, pricing, bonus and restaurants along the routes. This results in a *utility function* which expresses the utility for a given route, as follows (with τ the number of minutes travelled):

$$U = \beta_{TravelTime} \cdot \log(\tau) + \beta_{road} + \beta_{urban} + \beta_{congestion} + \beta_{pricing} + \beta_{bonus} + \beta_{restaurant} \quad (7\text{-}1)$$

There are mainly two ways how this utility function can be used to compute the best route according to the utility function:

1. Score all routes of a precomputed set of routes, select the one having the highest utility
2. Change edge weights in such a way that they represent (dis)utility, run a Dijkstra algorithm on the modified graph maximizing the utility

The first method lies most in nature with the conducted stated choice experiment, where the participant can choose between several options of routes from start to destination. Therefore, the set of routes needs to be computed on beforehand. Previous studies used for example the K-shortest path algorithm to compute the set [26]. Problem with K-shortest path is that this generally does not provide enough good routes, i.e. the second shortest path is a path where one leaves the highway via an off-ramp and enters the highway again at the next on-ramp. It is not likely that such a path provides a good solution, which means that lots of paths need to be computed before the best path is found. There are solutions to overcome this problem by using algorithms such as Google Maps is using to compute multiple routes as shown in Figure 7-1. Their algorithm behaves like the Alternative Routes algorithm introduced by Abraham et al. [2]. The computation of the set of routes is completely based on travel time and overlap of the routes. This method therefore creates a biased selection of routes. A route more or less identical to the fastest route but with a small detour to visit a truck parking is not likely to be contained in this resulting set of routes. Also other methods for computing a set of routes seem to give biased results. It can be concluded that there are no suitable methods available that can create a good set of alternative route usable for reflecting preferences of different logistics companies.

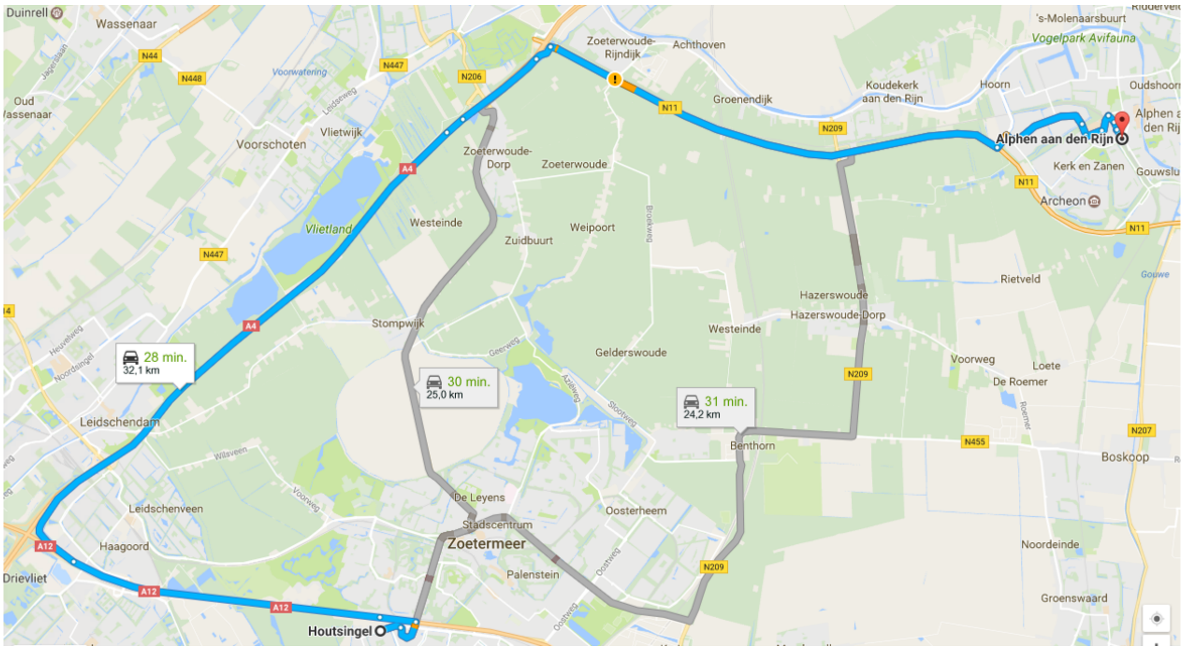
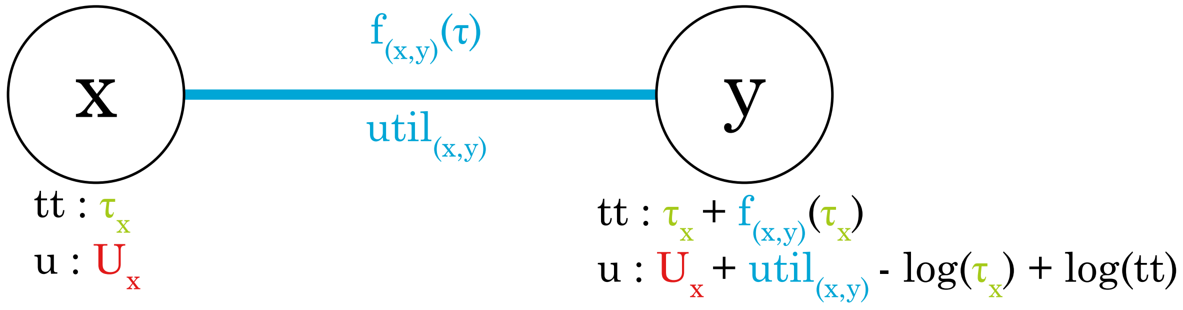


Figure 7-1: Example of alternative routes as presented by Google Maps for a trip from Zoetermeer to Alphen aan den Rijn

The second method of computing the best route according to a utility value per link, therefore seems more promising. However, lots of modifications need to be made such that the utility function can be used as an edge weight function. Currently, the utility function expresses a value for a complete route. This function should therefore be converted in such a way that it is possible to obtain a utility contribution per link.

It should be noted that using this method a value for each of the attributes is assumed for each of the links. For example, a link is considered to be a highway, a main road or a local road and an attribute value is added to the utility accordingly. This is different than Arentze et al. assume in their experiments. They assume that a complete route is associated with one property: that is, it either travels along the highway, main roads, or local roads, not something in between. Interestingly, the presented choice alternatives do consist of multiple road types. For example, route 2 in the choice alternative shown in Figure 6-2 travels along *Provincial and partly local road.* The route is coded as *Provincial road.* A route marked as *Provincial road* therefore travels mainly along provincial roads instead of only along these roads. Although this might result in differences between the method used in this thesis and the actual experiment, it is assumed to better reflect the actual routes taken.

A requirement for an edge weight function is linear additivity, implying tat the value of a path (for example, the arrival time) can be computed by summing up the values of edges on this path (for example, the travel time). However, the utility function as used by Arentze et al. is not completely linear additive: for travel time a logarithmic function is used. Therefore it is not possible to assign a certain (static) utility contribution per link, since it matters what the trip duration is before driving across this specific link. This is easily fixed by not only maintaining the total disutility for each node, but also the travel time until that node. If the utility function of an edge (x, y) without the travel time component is referred to as $util_{(x,y)}$, the total utility u can be calculated using a combination of the $util_{(x,y)}$ function and the travel time function $f_{(x,y)}$, as shown in Figure 7-2. This procedure can be implemented using Dijkstra's algorithm, where nodes to be settled are sorted on maximum utility value.



$f_{(x,y)}(\tau)$
$util_{(x,y)}$
x
y
tt : τ_x
u : U_x
tt : $\tau_x + f_{(x,y)}(\tau_x)$
u : $U_x + util_{(x,y)} - \log(\tau_x) + \log(tt)$

Figure 7-2: Computing the utility of node y originating from node x. First, the travel time is updated according to the travel time function $f_{(x,y)}$,. After that, the utility u is updated by adding the linear additive part of the utility edge weight function, followed by updating the value by the new logarithmic value of the travel time.

What is left, is the conversion of the utility function valid for a complete route, to a utility contribution per link. This is explained in detail in the next section.

7-2 Computing the utility contribution per link

The utility function should be converted in such a way that it represents a utility contribution per link travelled. Therefore the used coding scheme of the utility function is explained in detail in this section. The *effect coding* used by Arentze et al. [3] is converted to a *dummy coding* to express absolute (dis)utility instead of relative (dis)utility. Next the attributes are weighted and scaled such that it is usable in a routing application.

7-2-1 Dummy and effect coding

The earlier presented utility function included parameter values for each of the attributes, for example β_{road} for the road type. However this does not represent just one value: there are different betas, depending on the type of road. To express these multiple options, coding schemes are used. In most stated choice experiments, this is done using *dummy variables* and *dummy coding.* A dummy variable has the value of 0 or 1. For road type, this can be represented using the following equation:

$$\beta_{road} = D_1 * \beta_{highway} + D_2 * \beta_{main} + D_3 * \beta_{local} \tag{7-2}$$

In this example, D_1 is 1 if the specific route is a highway, 0 otherwise. D_2 likewise represents whether the route consists of main road or not, D_3 whether it represents a local road or not. The sum of dummy variables D_1, D_2 and D_3 is always one, such that each route can only have one property.

Arentze et al. [3] uses a different coding scheme, called effect coding. In effect coding dummy variables are used that can take 3 instead of 2 values: besides 0 and 1 also -1. As a result, only 2 dummy variables are needed to code the same number of attribute levels, as is shown in Table 7-1.

Table 7-1: Effect and Dummy coding

	Effect coding			**Dummy coding**			
	E_1	E_2	**Parameter**	D_1	D_2	D_3	**Parameter**
Highway	1	0	$\gamma_{highway}$	1	0	0	$\beta_{highway}$
Main road	0	1	γ_{main}	0	1	0	β_{main}
Local road	-1	-1	$-\gamma_{highway} - \gamma_{main}$	0	0	1	β_{local}

The parameter values of the three attribute levels sum up to one. However, the effect coding now only expresses a relative effect on an average utility value. This average utility value is the constant C_{road} in the effect coding formula, that is:

$$\beta_{road} = C_{road} + E_1 * \gamma_{highway} + E_2 * \gamma_{main} \tag{7-3}$$

In case of road type, this constant is a negative value, in such a way that the resulting β_{road} is always negative, that is, always results in a disutility, even if driving over the highway. In the context of trucks it is assumed that driving is never done for pure pleasure and thus always results in some disutility, albeit small.

The use of this constant and the fact that the parameters do not express absolute (dis)utility but relative (dis)utility, make the coding useless to directly use in a routing application. By maximizing the utility, this might result in non-negative cost cycles, as shown in Figure 7-3.

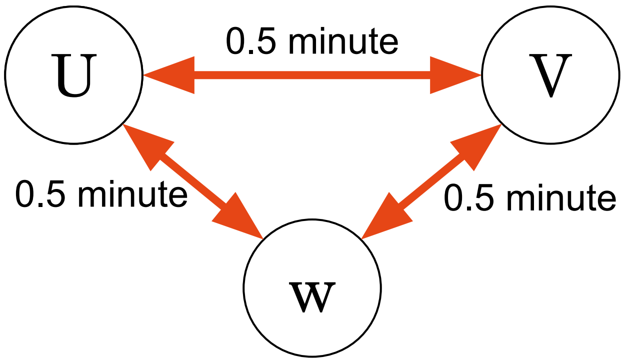


Figure 7-3: Example graph containing a cycle. If we assume the roads are local roads (parameter value -0.799) in an urban area (0.219), without congestion (1.512) or pricing (1.548) and no restaurant (-0.127), then driving this cycle gives an increase in utility of $-4.579 \cdot \log 1.5 - 0.799 + 0.210 + 1.512 + 1.548 - 0.127 = 1.54$. Driving this cycle thus increases utility, which results in endlessly driving cycles without reaching the destination when optimizing for a maximum utility.

If the constant values for each of the attributes are known, one could easily overcome this problem by subtracting the constant value from the corresponding parameter values. However, these values are not given by Arentze et al. [3]. Therefore, first the effect coded parameters are converted to dummy coded parameters, and after that the values are weighted in such a way that the importance of each of them is reflected correctly in the resulting utility function.

7-2-2 Effects to dummy coding

A method to convert an effect coding to a dummy coding is introduced by Daly et al. [14]. First an additional dummy variable is added to the effect coding, such that the number of dummy variables for effect coding corresponds with the number of dummy variables needed. This is done without loss of generality if the newly added parameter is set to zero, as shown in Table 7-2.

Table 7-2: Introduce additional variable for effect coding

	E_1	E_2	E_3	**Parameter**
Level 1	1	0	0	γ_1
Level 2	0	1	0	γ_2
Level 3	-1	-1	1	$-\gamma_1 - \gamma_2 - \gamma_3$

Next, the effect coded parameters are converted to dummy coded parameters by setting the reference level β_3 to 0, and determining β_1 and β_2 using the following formula:

$$\beta_i = \gamma_i + \sum_{l<K} \gamma_l \tag{7-4}$$

With β the dummy coded parameter, γ the effect coded parameter and K the number of attribute levels.

For road type, this gives the following parameters:

Table 7-3: Conversion from effect to dummy coded for road type

	Effect coded	**Dummy coded**
Highway	$\gamma_1 = 1.132$	$\beta_1 = \gamma_1 + (\gamma_1 + \gamma_2) = 2.23$
Main road	$\gamma_2 = -0.333$	$\beta_2 = \gamma_2 + (\gamma_1 + \gamma_2) = 0.466$
Local road	$-\gamma_1 - \gamma_2 - \gamma_3 = -0.7999$	$\beta_3 = 0$

The resulting dummy coded parameters correctly reflect the differences between each of the attribute levels. However, the parameters still do not reflect the actual disutility of an attribute and their relative influence. Therefore the parameters need to be weighted. After that a scaling procedure should take place to determine the utility contribution relative to the actual travelled distance or time.

7-2-3 Weighting of attributes

The previous section explained the process from effect coded parameters to dummy coded parameters. The parameters now correctly represent the relative differences in utility between each of the attribute levels. However, the start level is still unknown, that is, the actual disutility received such that it is correctly reflected that certain attributes are more important than other attributes. Therefore, each of the attributes needs to be weighted by subtracting a certain value.

Actually, what needs to be computed is the relative value of each of the attributes compared to another attribute. This process is comparable to obtaining priors for a choice study. Bliemer et al. [9] gives several options for obtaining such prior values, of which using *revealed preference data* is the most common. Important to realise is that parameter coefficients of other studies cannot be used directly due to scale differences. Instead, ratios of such coefficients to a common attribute should be used. This common attribute should appear in each of the studies analysed. For routing, travel time is an attribute that is included in every study. Therefore, this attribute is selected as the common attribute. The revealed preference studies used to identify relative differences compared to the travel time are the studies earlier explained in Section 6-2-1.

Roadtype

Preferences for roadtypes are analysed in the revealed preference study of Hess et al. [30]. They developed a route choice model for heavy goods vehicles using GPS data in the United Kingdom. They identified three main road types: motorway, A-road and B-road. These are assumed comparable with highway, main road and local road as used in the experiment of Arentze et al. [3]. For motorways they found a parameter value of -0.052, with a logarithmic time parameter value of -6.195. Scaled according to the time parameter value of Arentze et al. of -4.579, this leads to a utiliy value of -0.034 for highways. Given the highway dummy coded value of 2.23 (see Table 7-3), each of the attribute levels need to be adjusted with a constant of -2.264, leading to attribute level values of -0.1254 for main road and -0.1494 for local roads.

Urban

For travelling through urban areas no revealed preference data is found. However, there exists a strong correlation between road type and urban areas in the used road networks of HERE [29]. A highway is never considered to drive through urbanized areas, whereas most local roads do travel through urban areas. Therefore, the prior value for road type as given by Hess et al. [30] is used for urban areas as well.

Congestion

Prato et al. [40] analysed the route choice behaviour of car drivers given a GPS dataset. Although this study considered car drivers instead of truck drivers it is assumed that it still reflects the value of congestion properly – in absence of revealed preference data of trucks. Prato et al. computed the value of congested time compared to free flow travel time. He concluded that the value of congestion is 1.46. That is, 1.46 minutes of congestion is rated equally as 1 minute of free flow travel time. This value can be used to weight the long delay attribute level. This attribute level of Arentze et al. means that there is a chance that the route takes up to twice as long. This leads to a disutility of a long delay of -0.3066 per minute of travel time. Corrected for the average travel time of the routes in the choice experiments, results in a parameter value of -4.8820

Pricing

Besides congestion Prato et al. also analysed the *value of time*. They found a value of time of €17.31 per hour. The toll roads in Europe are priced €0.22 per kilometer on average. This means that avoiding 1 kilometer of toll is worth 0.76 minute of additional driving, equal to an utility contribution of -0.1596. Corrected for the average travel time of the routes in the choice experiments, results in a parameter value of -2.5413.

Restaurant

No revealed preference studies are found on the presence of a restaurant along the route. Arentze et al. uses one value for a restaurant: there is one along the route, or there is none. The associated utility is very little and not found significant. It is questionable whether the presence of a restaurant is indeed not important for route choice, or whether it is not important in the choice alternatives Arentze et al. analysed. The used choice alternatives are short: 10 to 30 minutes. It seems reasonable that in such a short trip no stops are made by a truck driver halfway the route, thus the presence of a restaurant does not matter. For longer routes it is assumed that a restaurant does provide some advantage, if it is located at a spot where a break must be taken according to the drivers legislation. Therefore a restaurant is awarded with a high utility contribution if it is located on the route after 3 to 4.5 hours of driving time. Before or after that time limit, generally no break is taken. In this thesis a utility contribution of 0.01 is used for a restaurant located at a not so favourable time, 1.05 for a favourable time. A utility of 1.05 corresponds to 10 minutes of additional driving time. During the sensitivity analysis it is discussed whether the actual chosen values matter much.

7-2-4 Overview of parameter values after converting and weighting

The weighing of parameters according to prior values obtained from revealed preference data leads to parameter values of each of the attribute levels as shown in Table 7-4.

Table 7-4: Parameter values of utility function, after conversion to dummy coding and weighting accordingly

$\beta_{TravelTime}$	-4.579 (log)		
β_{road}	-0.0344 (Highway)	-0.1254 (Main roads)	-0.1494 (Local roads)
β_{urban}	-0.0344 (No urban area)	-0.0464 (Urban area)	-0.0924 (Urban area with school)
$\beta_{congestion}$	-2.828 (No congestion)	-4.499 (Medium delay)	-4.8820 (Long delay)
$\beta_{pricing}$	-0.057 (No pricing)	-2.5413 (With pricing)	
$\beta_{restaurant}$	1.05 (Visited between 3 and 4.5 hours of driving)		0.01 (Other)

7-3 Calibrating the utility function

The weighted utility function parameters shown in Table 7-4 cannot be used directly as edge weights. First of all, the parameter values should represent a value per link. Therefore, the values need to be scaled using the travel time or distance of the choice experiment. Secondly, the travel time is modelled as a logarithmic function in the results of Arentze et al. [3], whereas other choice studies use a linear travel time component. In Section 7-3-2 the difference between these two options is analysed.

7-3-1 Scaling for travel time or length

The utility contributions of road type, presence of urban area, congestion, pricing and restaurant are given for the complete route by Arentze et al. [3]. If routes are computed using the method as explained in Section 7-1, a utility contribution need to be given for each link. Therefore, the utility contributions need to be scaled to represent a value per link. With scaling the utility function it is meant that the parameter values shown in Table 7-4 are divided by the average travel time or distance of the choice experiment such that a utility contribution per kilometre or minute is obtained.

Especially for longer routes a distance scaling gives strange results: the best routes are routes having minimum distances instead of representing a combination of multiple route aspects. An example is shown in Figure 7-4. The routes all travel along non-priced roads and have identical results for urbanized areas and road types. Differences are found in congestion, travel time and length of the route. Scaling for distance (orange route) leads to a short route in terms of distance, but at the costs of about 40 minutes of additional travel time while the chance on congestion is increased. The purple route has 10 minutes of additional travel time compared to the fastest route, for a decreased chance on congestion. Logically, the blue route seems to be the best one: a little bit of additional travel time results in less congestion. The orange route does not seem to be the best route at all: it takes additional travel time and results in more congestion than the fastest route.

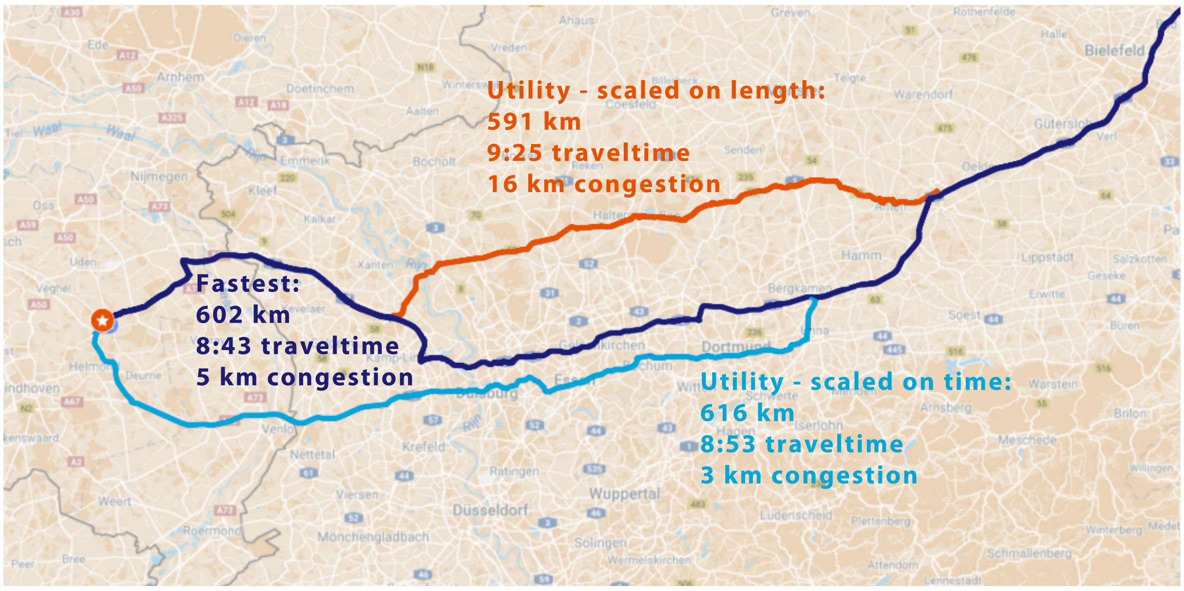


Figure 7-4: The fastest route, the best utility route using travel time scaling and the best utility route using distance scaling. The last part toward the destination is not shown.

Due to the varying lengths of routes used in the simulations of this thesis, it matters a lot whether utility contributions are scaled according to distance or travel time. Scaling for distance results in short routes that do not necessarily provide good routes. Scaling for travel time seem to result in better options. Therefore, in the remainder of this thesis the utility contributions are scaled according to travel time to calculate the respective value of travelling a link. Results in differences in arrival time when scaling for distances instead, are given in Appendix B.

7-3-2 Logarithmic or linear travel time utility contribution

Arentze et al. [3] uses a logarithmic function to represent the travel time. The idea of using a logarithmic function is that people care less on travel time differences if they drive for longer periods. For example, a detour of 5 minutes is expected to have a larger effect on the total utility of a trip if the total trip duration is 30 minutes compared to a total trip duration of 5 hours. A logarithmic function expresses this neatly. This method especially works well with two distinct ranges of travel times, such as in the experiment of Arentze et al. where a short route (15 km, 10 minutes) and a long route (30 km, 30 minutes) are compared with each other. The fit of the logarithmic function than yields a large disutility contribution per minute for a short trip compared to the disutility contribution of a long trip. However, this might result in incorrect modelling if the routes become much longer than 30 minutes of driving time, since other attributes are modelled linearly. For example, driving 1 kilometer along a toll road compared to a non-paid road results in a disutility of -0.156. After 6 hours of driving, this same disutility is reached with 12 minutes of additional driving time (i.e. $-0.156 = -4.579 * \log(x) - -4.579 * \log(360)$, yields $x = 372$). With a toll price of 22 eurocent per km, this corresponds to a *value of time* of only €1.10 per hour! Far too low to be a reasonable value compared to an average value of time of about €17 per hour [40].

A solution is to model the travel time as a linear function. It is assumed that the values for utility contribution for 10 and 30 minutes (that is, the time limits used in the experiment of Arentze) are correctly measured. Next a line is drawn between these points to obtain a linear travel time utility contribution function.

Due to the much longer routes used in this thesis compared to the experiment of Arentze et al. the linearized travel time utility contribution function is used. For convenience, results for influences on arrival time with the logarithmic travel time function are added in Appendix B.

7-3-3 Final parameter values

After the conversion from effect to dummy coding, weighting, scaling and using a linear travel time component instead of a logarithmic one, the final parameters representing the disutility per minute of travel time are shown in Table 7-5.

Table 7-5: Values of utility contribution per minute of travel time, as used in the simulations

$\beta_{TravelTime}$	-4.579 (log)		
β_{road}	-0.0138 (Highway)	-0.1048 (Main roads)	-0.1288 (Local roads)
β_{urban}	-0.0138 (No urban area)	-0.0258 (Urban area)	-0.0718 (Urban area with school)
$\beta_{congestion}$	-0.1776 (No congestion)	-0.2826 (Medium delay)	-0.3066 (Long delay)
$\beta_{pricing}$	-0.0036 (No pricing)	-0.1596 (With pricing)	
$\beta_{restaurant}$	1.05 (Visited between 3 and 4.5 hours of driving)		0.01 (Other)

7-4 Simulations

The utility function of Arentze et al. [3] representing a route preferred by a truck company is converted in the previous sections in such a way that a Dijkstra algorithm can be run on a road network with adjusted edge weights in order to compute the route having maximum utility.

7-4-1 Test set-up

In order to compute the most preferred route, the road properties need to be known to compute resulting utility contributions. The map of Europe of HERE [29] consisting of 31 million nodes and 66 million edges is used. This map includes meta data which is used to obtain the relevant road properties.

Each link in the map of HERE has a specific road type. *Motorways* and *A-roads* are mapped to *Highways*, *B-roads* are mapped to *Main roads* and *Regional and Local roads* are mapped to *Local roads*. For each road type it is indicated whether it is in an urban area or not. Besides the identification of being an urban area or not, Arentze et al. [3] also makes a distinction between urban areas with and without schools. Due to lack of data on schools, it is assumed that a local road through a city is mapped to *Urban area with school*. All other road types through urban areas are mapped to *Urban area*, the rest to *No Urban area*. Values of average travel times provided by HERE are used to compute whether there is a chance on a medium or long delay. A *Medium delay* is considered to have a travel time which is up to 50 % longer. If the delay lasts even longer, it is considered to be a *Long delay*. Whether a road is a toll road or not is included in the dataset of HERE and used directly. Data on the availability of restaurants is taken from the Truck POI dataset used in Chapter 5.

Road blocks are identical compared to the test set-up in Section 4-5-1. Also drivers legislation rules are identical (i.e. at maximum 4.5 hours of consecutive driving followed by a break of 45 minutes). A subset of 175 randomly selected feasible routes is taken from the original subset of 10000 routes. The average travel time of these routes is 6 hours and 47 minutes. All tests are run on a laptop, having 16GB memory and an Intel Core i7-3740QM 2.70GhZ processor with 4 cores.

7-4-2 Verification of results

First of all, it is checked whether utility best routes are never faster than the fastest route – which should not be possible. This is indeed the case. Next some routes are plotted on a map and compared with the fastest route. Routes are checked according to the linear travel time function and other attributes scaled with travel time instead of distance. This leads to a few examples which give insight in the resulting choices due to the usage of the utility function.

A first example is shown in Figure 7-5. This shows a situation in which the best route is 7 minutes longer than the fastest route, while avoiding toll road. The toll paid in

Poland for this section for a normal truck is about €6.76, a bit less than the assumed average of 22 eurocent per kilometer in Europe. Given the value of time of €17.31 [40], a truck company would allow an additional driving time up to 23 minutes to avoid such a toll road segment. The 9 minutes additional travel time as shown in this example is therefore perfectly acceptable and correct.

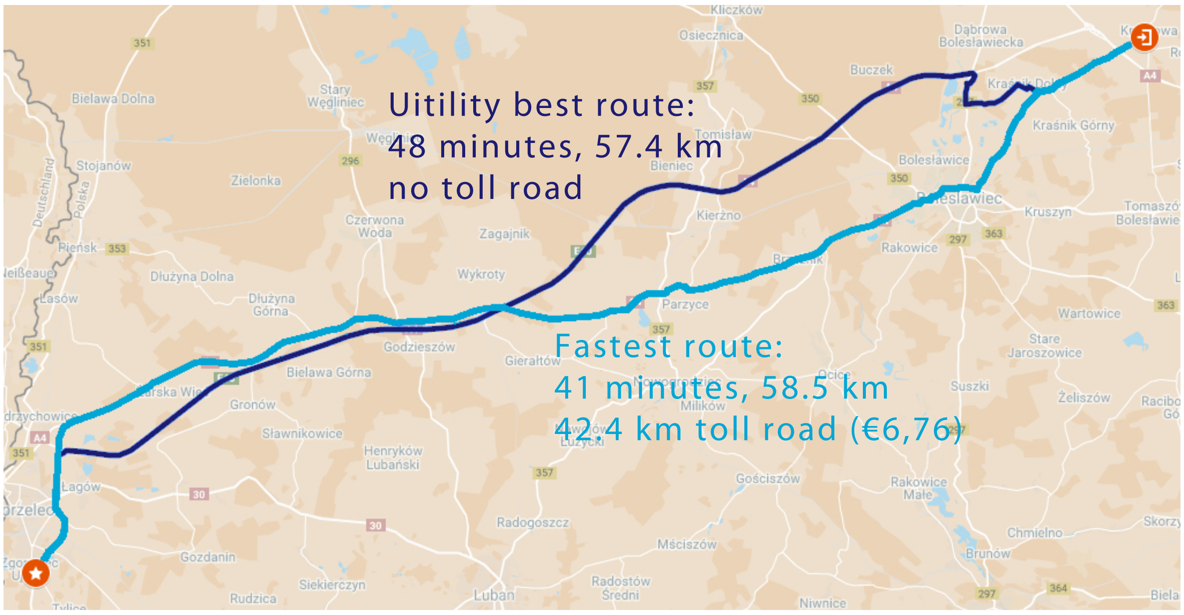
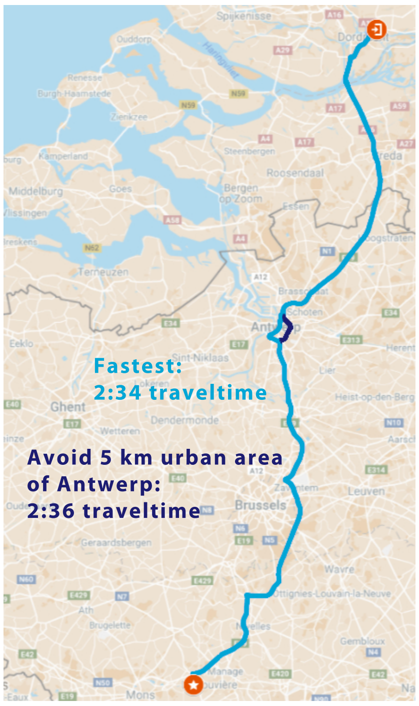


Figure 7-5: Example difference between the route which is fastest (purple) and best according to the utility function, i.e. 7 minutes of additional driving time is acceptable for saving e6.76.

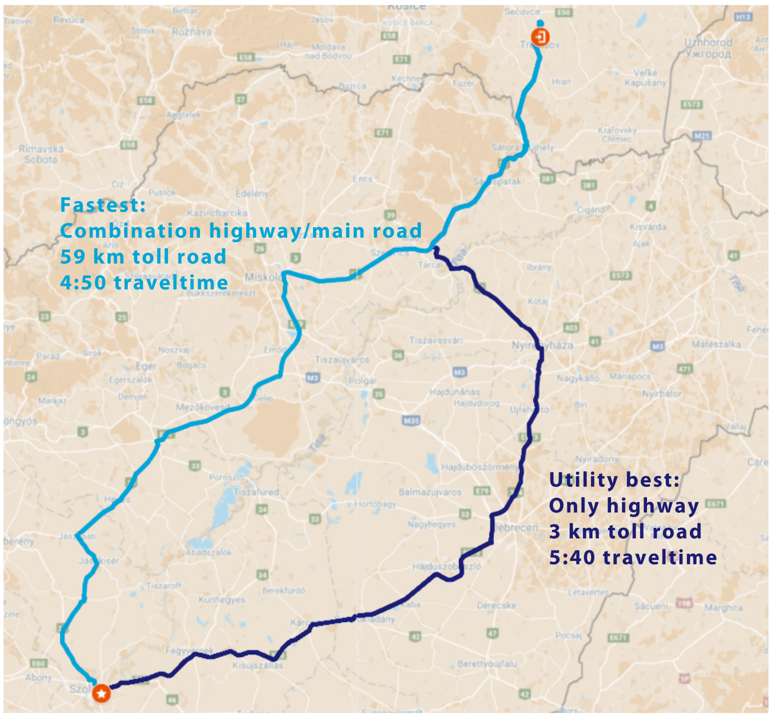
A second example is shown in Figure 7-6a. This example focuses on urbanized areas. The route having the highest utility is slightly longer (2 minutes). Avoiding an urban area results in a increased utility of 0.012 per minute, far less than the utility contribution of -0.21 per minute travel time. However, not only urban area is avoided: these roads through Antwerp are heavily congested, giving an additional 0.129 utility contribution per minute of travel time. This results in a total utility increase of 0.705, thus worth 3.5 minute of additional travel time.

The third and last example (see Figure 7-6b) deals with road types along the route. The fastest route travels partly across main roads (30 km). Furthermore it drives along 59 km of toll road, whereas the utility best route only drives 3km on such roads. Avoiding 56 km of toll road is worth 42 minutes of additional travel time, which would not be enough to choose for this longer route of 50 minutes. However, also 30 km of main road is avoided, worth another 13 minutes of additional driving time.

The checks and examples justify the correct working of the algorithm and conversion of the utility function. In the next sections, the average influence on the arrival time is shown.



(a) Avoid congested urban area through Antwerp



(b) Avoid main road and toll road

Figure 7-6: Two examples of differences between fastest and utility best route

7-4-3 Influence of preferences on the arrival time

For each of the 175 routes the best route is computed and compared with the travel time of the fastest route. The results on the differences in travel time are shown in Table 7-6. The relative influence in travel time is obtained by dividing the difference in travel time by the total duration for each of the routes. This leads to an average relative increase in travel time of 1.8%. In absolute numbers, this corresponds to a 9 minute difference on average. The route having largest deviation has a relative difference in travel time of 18 %. This route also had the largest absolute difference in travel time: 109 minutes. Such a large difference in travel time is mainly caused by avoiding long segments of congested and toll roads. The example in Figure 7-6 already showed that avoiding 56 km of toll road is worth 42 minutes of additional travel time. Likewise, 109 minutes of additional travel time equals skipping 145 km of toll road.

Table 7-6: Differences in travel time between fastest routes and routes with maximum utility

	Relative	**Absolute**
Average difference	1.8%	9 min
Minimum difference	0 %	0 min
Maximum difference	18 %	109 min
Standard deviation of difference	3%	18 min

In 17 % of the 175 tested routes, no difference is found between fastest and most preferred route as shown in Table 7-7. Another 29 % of the routes have a relative difference of less than 0.1 %. In 11 % of the planned routes the difference between fastest and preferred route is more than 5 %.

Table 7-7: The distribution of differences in travel time between the fastest and best routes

	Percentage of total number of routes
No difference	17 %
0 to 0.1%	29%
0.1 to 1 %	22 %
1 to 5 %	21 %
More than 5 %	11 %

7-4-4 Influence of preferences on route composition

Besides the influence on the arrival time, it is interesting to see the difference in road properties. This is shown in Figure 7-7. The fastest route consists on average of 97% highways and 3% of other road types. The best route consists for 98% of highways, reflecting a small difference. Larger differences in composition of the route can be found for toll roads. The fastest routes travels on average a 9 % longer distance along toll roads than the preferred routes.

Interestingly, the utility best route travels more through urban areas than the fastest route: 3%. It is assumed that this is caused by avoiding toll roads or congested roads. A toll road can for example sometimes be avoided by taking a route through the city centre. This is also reflected by the used parameter values (see Table 7-5): travelling a toll road results in far more disutility than travelling through an urban area.

More restaurants are found along congested and toll roads. The influence of the restaurant parameter is considered too little to affect the choice of route. For example, one rather skips 8 km of toll road than driving along a restaurant at this toll road. Therefore, best routes have a 40 % lower chance of facing a restaurant along the route.

7-5 Sensitivity analysis

The process of converting and scaling the parameters might not result in realistic parameters. Especially the selection of prior values for weighting is a critical element having much influence on the resulting parameters. Furthermore, the used choice study of Arentze et al. [3] might not be applicable for long journeys as used in the simulations of this thesis. Whereas Arentze et al. uses routes of 10 or 30 minutes in length, the considered test set contains routes with an average travel time of more than 6 hours. Truck companies might have stronger or less stronger influences of such longer trips compared to the shorter trips.

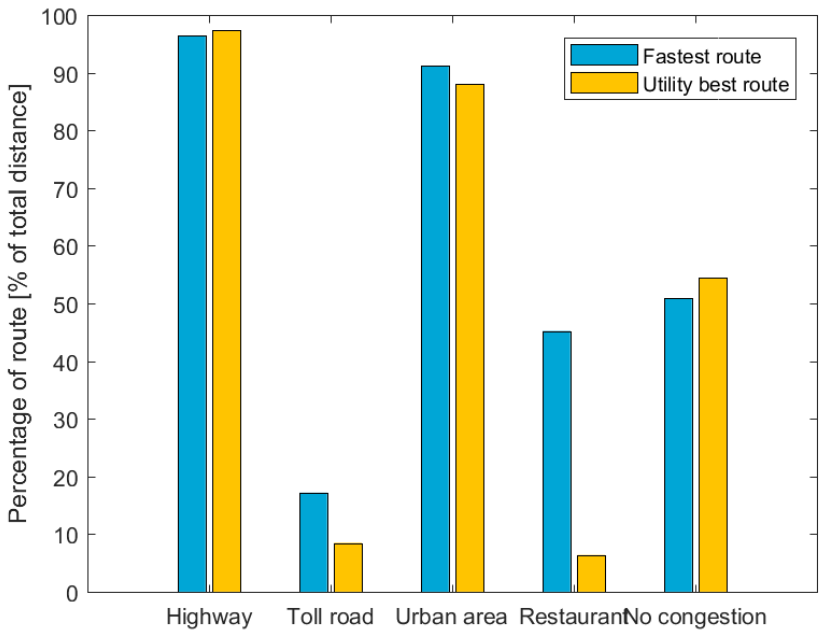


Figure 7-7: Difference in composition of routes of utility and fastest routes. Shown is the average percentage of a route, i.e., the fastest route consists of 18% of toll roads whereas the utility best route consists of 9 % of toll roads.

Therefore, this section tests several tastes of the used preference parameter values. For each attribute, the parameter value is set to -100% (no influence), -50%, -25%, -10%, +10%, +25%, +50% and +100% (twice as much influence). The other attribute values are kept equal. This allows to see the difference in travel time if a truck company has a (less) stronger preference for each of the attributes. If an attribute has multiple levels (i.e. for road type: highway, main road, local road) with according parameters, then all parameters are increased or decreased with the same percentage. The relative importance and direction of attribute levels is kept identical: for example, it is not tested what a route would look like if some one dislikes highways and prefers to drive along local roads only.

For the sensitivity test, 25 randomly selected routes are computed for each of the 48 parameter sets. Next, the travel time is compared to the travel time of the original preferred route. The average percentual difference between these two travel times is shown in Table 7-8. Table 7-9 show the average percentual overlap between the newly computed and original best route.

First of all, the direction can be concluded: being more sensitive (+100%) for travel time results in a shorter route. Being more sensitive for pricing results in longer routes (i.e. one is willing to take additional detours to avoid toll roads). Also being more sensitive to avoiding urban areas results in longer routes. Avoiding congestion costs travel time. It should be noted that with congestion, the chance on congestion is meant. The resulting longer travel times due to congestion are not taken into account

Table 7-8: Percentual differences in travel time compared to utility function. For example, if the pricing utility contribution is decreased by 100 % (i.e. avoiding tolls is less important), the travel time

	-100%	**-50%**	**-25%**	**-10%**	**+10%**	**+25%**	**+50%**	**+100%**
Travel time	2.62% [11 min]	1.34%	0.44 %	0.29 %	-0.13 %	-0.15 %	-0.23 %	-0.40 % [2 min]
Pricing	-1.67 % [7 min]	-1.37 %	-0.48 %	-0.44%	0.00 %	0.51%	0.51%	0.77% [3 min]
Urban	-0.09% [0 min]	-0.09%	-0.08%	0.00 %	0.00 %	0.36 %	0.36%	0.42% [1 min]
Road type	0.28 % [1 min]	0.27 %	0.27 %	0.00 %	-0.07%	-0.07%	-0.07%	-0.09% [0 min]
Congestion	-1.60 % [7 min]	-0.77%	-0.50 %	0.00 %	0.37%	0.38%	0.41%	0.86% [4 min]
Restaurant	0.00 % [0 min]	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 % [0 min]

Table 7-9: Percentual overlap of routes compared to the routes produced by the utility function

	-100%	**-50%**	**-25%**	**-10%**	**+10%**	**+25%**	**+50%**	**+100%**
Travel time	90%	96%	96%	97%	99%	98%	97%	93%
Pricing	83%	88%	98%	98%	99%	98%	98%	97%
Urban	97%	98%	99%	100%	100%	98%	98%	95%
Road type	94%	97%	97%	100%	99%	99%	99%	99%
Congestion	78%	92%	98%	100%	98%	97%	96%	91%
Restaurant	100%	100%	100%	100%	100%	100%	100%	100%

when computing routes. In reality, this could mean that a preferences for avoiding congestion leads toward a faster travel time instead of a slower one. For restaurants, the direction is unknown.

The table also gives insight in which parameter values most matter. The travel time does not change much for the adjusted urban area, road type and restaurant parameter values. This is also reflected in the average overlap of routes, which does not differ much for these attributes. Most likely, these attributes do not influence the travel time that much. One can imagine that in some cases long detours are required to avoid an urban area, while its additional gained (relative) utility does not compensate the additional travel time. Recall the example around Antwerp in Figure 7-6a: avoiding 5 km of urban area is not even worth 2 minutes of additional travel time.

Travel time is most sensitive, followed by pricing and congestion. These three aspects also have large effects on actual routes taken, as can be concluded from the average percentual overlap of routes.

An example route showing the sensitiveness of toll roads is shown in Figure 7-8. It can be seen that being not sensitive to toll roads compared to avoiding toll roads completely, gives a difference in travel time of more than 1.5 hour. If we look at the sensitivity on average, the differences are much smaller: 1.67% less or 0.77% additional travel time. This is mainly caused by routes through countries that do not have toll systems: difference in toll sensitivity does not matter for route choice because the fastest route does not run along toll roads anyway.

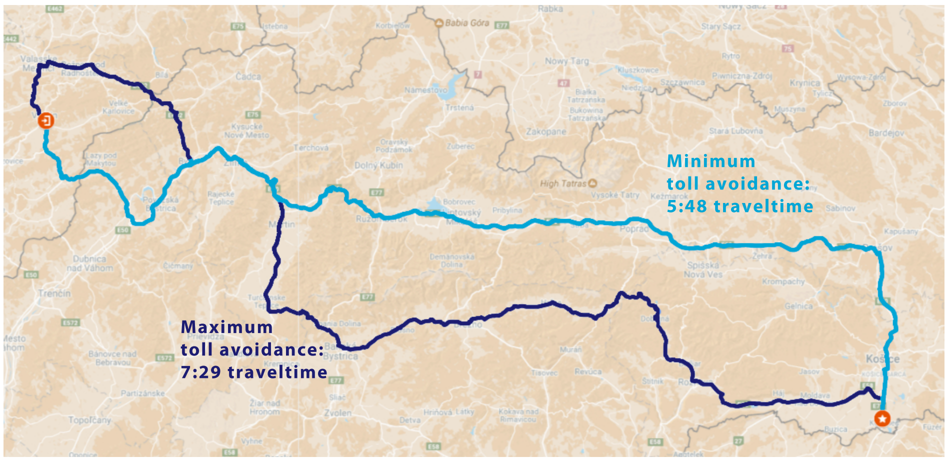


Figure 7-8: Maximum toll avoidance (i.e. pricing parameter +100%) results in a travel time of 7:29. No influence of the toll attribute (i.e. pricing parameter -100%) results in a travel time of 5:48.

Congestion is another attribute that is highly sensitive in terms of travel time. An example route is shown in Figure 7-9. If congestion avoidance is not taken into account, this results in the fastest route from start to destination with a travel time of 7:15 hours. If the congestion avoidance parameter is set to having twice as much influence, the travel time increases with more than 1.5 hour, avoiding busy areas around Vienna. Again, the average influence on the travel time for these sensitivity levels is much lower: 1.6% shorter travel time or 0.86% longer travel time. However, multiple routes do not face any congestion due to a lack of detailed data on congestion, especially on routes that are not motorways.

From the sensitivity analysis it can be concluded that the parameters of travel time, pricing and congestion have most influence on the arrival time and route choice.

7-6 Discussion

This chapter gives a methodology for using a stated choice experiment to compute preferred routes. The proposed method uses *stated preference* (SP) data of Arentze et al. [3] as a base, and uses *revealed preference* (RP) data for prior selection to weight the parameters. A better method would be to use the RP data as a basis, and add missing parameters from SP data later on, to prevent a survey bias. However, the analysed RP studies are not suitable for the context of this thesis. Hess et al. [30] focusses on trucks, but only includes road types in its study, whereas Prato et al. [40] has car drivers as participant that possibly have other preferences then truck drivers. To obtain better

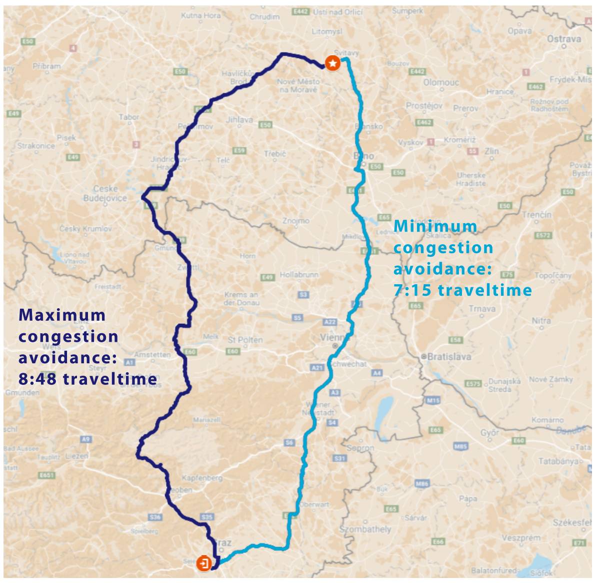

Minimum congestion avoidance: 7:15 traveltime
Maximum congestion avoidance: 8:48 traveltime

Figure 7-9: Maximum congestion avoidance (i.e. congestion parameter +100%) results in a travel time of 8:48. Minimum congestion avoidance results in the fastest route between start and destination of 7:15 hour

parameters than used in this thesis, a revealed preference experiment should be held with truck companies, having all attributes of interest included, that is, travel time, congestion and pricing.

The value of time of €17.31 per hour used in this thesis to determine the maximum additional driving time for skipping a toll road, is taken from the RP study of Prato et al. that has car drivers as its participants. It is questionable whether this is indeed a good value for trucks: driver wages, fuel and truck availability might costs more than €17.31 per hour for a logistics company. A higher value of time leads to less toll avoidance, which results in a 1.67% average shorter travel time as shown in Section 7-5.

The congestion parameter is only considered as a chance on congestion, whereas the actual resulting longer travel times are not taken into account when obtaining arrival times. Furthermore, the simulations showed that avoiding congestion lead to a higher travel time, but in reality this might not be true. Furthermore, the chance on congestion was computed independent of departure time. During peak hours one would expect a higher chance on congestion than during off-peak hours. This might result in different preferred routes at different times of the day.

Another observation made during the simulations was the low influence of restaurants. This is also an insignificant attribute in the experiment of Arentze et al. [3]. To correct for the small trip durations of Arentze et al. it is assumed that the presence of a restaurant results in a higher utility contribution if its location is near the moment in time that a break should be taken. However, this still does not influence the routing decisions, maybe due to an absence of data on restaurants along other roads than toll roads.

7-7 Summary

This chapter analyses the influence on the arrival time by taking preferences of the logistics company into account. The stated choice experiment on route choice of truck drivers of Arentze et al. [3] is taken as a basis. To compute the best preferred route, link weights are changed in such a way that these represent preferences instead of travel times. To overcome positive weight loops in the network, the utility function is changed from effect to dummy coding and weighted with priors obtained from revealed preference studies.

The simulations show that the difference in travel time between the fastest and preferred route is 9 minutes on average, or 1.8% relatively. However, for 11% of the routes, differences in travel time of more than 5% are found. These preferred routes generally contain less toll roads and have a lower chance on congestion than the fastest route. The sensitivity analysis showed that the parameters for travel time, toll roads and congestion cause the largest differences in arrival time. It is advisable that these parameters are set according to the objective of the company to obtain the best route as good as possible.

Chapter 8

Integrating Algorithmic and Behavioural points of view in route planning

This thesis analyses a route planning problem from two perspectives: algorithmic and behavioural. By combining the computation of earliest arrival routes with actual preferences other than travel time, several remarks can be made on currently used developments in algorithms and on methods for gaining behavioural insights. In this chapter three such observations are made, concerning the consequences for optimizing for running times, obtaining relevant preference attributes and computing routes using preferences.

8-1 Optimizing for running time

Within the algorithmic research community, most interest lies in optimizing running times when it comes to shortest path computation. This incentive comes mostly from industry where orders need to be scheduled as quickly as possible to minimize the time that trucks are standing still.

There is one large disadvantage: to realise small running times, lots of assumptions are made to speed-up the process. For example, drivers legislation rules are neglected or later on inserted. Analysis in this thesis showed that the influence of incorporating drivers legislation can be large: up to 37 hours in unfortunate situations where road blocks are faced due to the insertion of breaks.

The behavioural research community asks the relevant question: why optimize running times that much, such that the quality is reduced? It seems that the usage of many-to-many queries is unknown to large groups of researchers. It seems plausible to compute a route within a few seconds from the perspective of a driver, whereas this is not the case from the perspective of the planning process of a company.

Possibly it is a better idea to split up the work of route computation in two steps: first decide upon the exact ordering of routes using currently fast many-to-many algorithms, after that compute each of the routes in detail including drivers legislation, parking lots, preferences and possibly other aspects. If the precise computation of routes leads to a very large deviation, this can be used as an input to reconsider the delivery order sequence.

8-2 Relevant preference attributes

The behavioural community researches the effect of lots of attributes on routing. However, not all analysed attributes provide useful insights. For example, road type is one of the most important attributes to consider in routes as follows from the experiment of Arentze et al. [3]. However, when planning routes on real road networks, this attribute is strongly correlated with travel time: if a route mostly drives on highways, a lower travel time is obtained compared to driving local roads. Simulations showed that an average fastest route consists of more than 95% highways (see Figure7-7). It is therefore questionable whether a preference for highways indeed reflects a preference for specific road types, or whether this implicitly means that the participant of the experiment wants to reach its destination as fast as possible, and knows from experience that this objective is reached by following highways.

Furthermore, the attribute of facilities along the route is included in many choice experiments. What is left out, is the specific location of such facilities along the route. However, a restaurant near the start or destination would possibly be rated very low compared to a restaurant at the parking lot where a break should be taken according to the drivers legislation. This is not included in any of the analysed choice experiments.

Some included attributes deal mostly with associated costs, such as avoiding toll or fuel efficient driving. However, most truck drivers do not pay for toll or fuel themselves. It seems plausible that therefore these factors are not found significant in many experiments [42] [47]. However, studies counting the number of trucks present on toll roads, came to the conclusion that forecasts based on preference studies highly overestimate the actual usage of toll roads by trucks [4]. This can be explained by the influence of the logistics company on the route choice of truckers: whereas truckers might prefer driving on toll roads, they are not always allowed to. Therefore, the aspect of who pays for what costs and what routes are allowed to be taken according to the company, should clearly be defined in any choice experiment to prevent such cost-based biases.

To conclude, best results of choice experiments are obtained if attributes are not highly correlated with each other in real routes on a map. Additionally, restaurants and parking lots should only be included in combination with drivers legislation, and the context of who pays which costs should be made clear for any route choice experiment.

8-3 Dealing with preferences in obtaining a route

In this thesis two methods are identified in Section 7-1 for actually obtaining a route given a stated choice study: first computing a set of routes and then selecting the best, or by changing edge weights such that these represent disutilities. For both methods remarks could be made and improvements in research upon these topics identified.

8-3-1 Computing a set of routes

Resulting utility functions can be used to calculate the utility of a certain route. If a set of routes is given, the best route can be selected easily by computing their utility. However, computing such a set of routes is hard. Within the behavioural community, K-shortest path algorithms are used to compute such sets. Experiences of the algorithmic community shows that these methods do not work on very detailed maps: the second best route is likely to be identical to the fastest route except for a small detour of taking an off-ramp and the next on-ramp on a highway. In a detailed road network it is likely that only after computing the first 1 million shortest paths, there is a chance that the best route is contained in it.

Computing a good set of routes between two locations also gains attention of the algorithmic community. However, there focus is purely based on travel time and being distinguishable from each other. The Alternative Routing algorithm [2] selects routes that are not much longer than the shortest route (advise of maximum 25% longer) and a maximum overlap of routes (advise of maximum 80% overlap). If we consider some analysed examples in Chapter 7 the alternative (preferred) route is nearly never included in the resulting route set using the Alternative Routing algorithm: Figure 7-6a (avoiding urban area of Antwerp) shows an overlap of more than 95%, whereas Figure 7-8 (maximum toll avoidance strategy) results in a 30% longer route. It is questionable whether such criteria purely based on travel time and overlap indeed provide good alternative routes. At least, these routes do not (always) give routes that are the best according to the utility function.

It would be a good idea if the algorithmic community focusses on computing sets of alternative routes based on road properties such as congestion and toll instead of only travel time based criteria. These route sets could be used to easily identify the best route for a specific truck driver by rating them using a utility function. Although the set is still biased and it is unsure whether the ultimate best route is contained in the set, chances are much higher compared to the currently available alternative routing algorithm.

8-3-2 Computing the best route

To overcome the biases resulting from using precomputed sets of routes, this thesis changes weights of edges such that these represent preferences and uses these to compute best routes. However, this required lots of modifications to the reported parameters of the used choice experiment. There is room for improvement in stated and revealed preference studies such that they become directly usable in route planning applications in future, as is shown in the following paragraphs.

First of all, utility functions are valid for a complete route. Transforming them to link-based functions can be done by scaling, that is, dividing the parameters by the average travel time of the considered routes in the experiment's test set. It is questionable whether this method is fully correct, especially because in route choice studies routes are modelled quite binary: a route consists of highways or main or local roads, not something in between. To make choice alternatives more realistic, entrance roads toward the highway are added in the choice alternatives, for example a road marked as "highway" actually consists of 5 % local road, 10 % main road and 85 % highway. To better calibrate parameters it is advisable to investigate the possibilities of performing link-based route choice studies. If real link-based values can be obtained, it's easy to obtain the correct best route by changing the edge weights and running a default shortest-path algorithm.

Besides shortest-path algorithms and some Pareto search approaches, little effort is made by the algorithmic community to compute a route that is not simply the fastest but also adheres to other criteria. For example, the Google Maps route planner provides options to avoid tolls: however, fully avoiding tolls is not necessarily what a logistics company wants. Recall the example in Figure 7-6b: 3 km of toll road is still included in the preferred route. Skipping this 3 km results in a much longer route, which is not worth the additional travel time.

A very neat example of incorporating preferences using an edge weight function is given by the Fietersbond Routeplanner [25], a routing application especially focussed on cyclists in the Netherlands. At this website you can create your own preference profile (see Figure 8-1) which is used to compute routes. Preferences ranging from street lightning to travelling through forests and penalties for traffic lights can be set using sliders after which an optimal route for your personal profile is computed. It is unknown what method they are using to compute routes, but it is expected that some linear combination of each of the attributes is used, identical to the method used in this thesis.

Currently efforts are made in the algorithmic community for so called *Customizable Route Planning* [17]. The idea is to apply a metric-independent preprocessing step, after which a second preprocessing step lasting only a few seconds is applied with the specific metrics. By creating such a graph, fast querying is possible. The research of customizable route planning mostly focusses on being able to incorporate real-time congestion updates. However, a good opportunity lies in incorporating preferences. In combination with a user-friendly method of selecting a preference profile such as the example of the Fietsersbond, it is then possible to compute routes for any preference profile very fast.

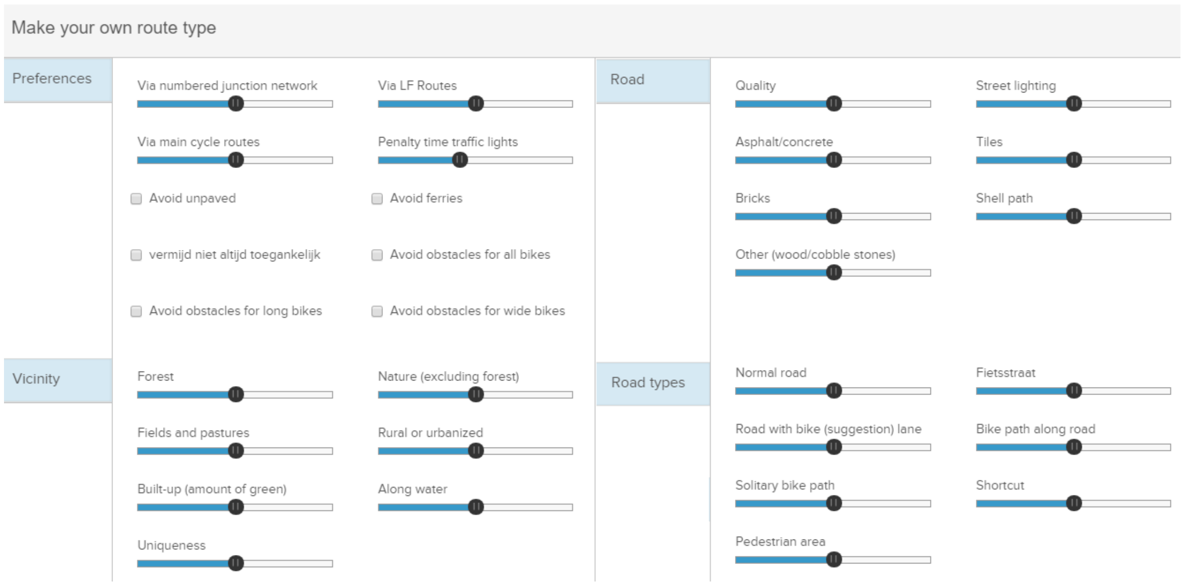


Figure 8-1: Overview of preference selection of the Fietersbond Routeplanner [25]

8-4 Conclusion

This reflection on the algorithmic and behavioural perspective on route planning results in a few observations. First of all, the algorithmic research community should change it focus from purely trying to compute the fastest route as fast as possible, to finding methods of computing the best route. Research can be performed on splitting the process of a typical Vehicle Routing problem into two steps: first obtaining an ordering of routes using fast many-to-many algorithms, after that computing precise travel times using slow but good algorithms. Secondly, progress can be made by finding better methods to compute multiple alternative routes, that are not only based on travel time or overlap of routes but also consider other properties such as toll roads or congestion.

The research on driver behaviour should keep in mind that currently, not all attributes included in stated choice experiments are realistic in terms of route planning. Some are highly correlated with travel time (such as road type) or are useless if too little information is specified (it matters whether a restaurant is at the beginning or halfway the route). Furthermore efforts could be made in link-based stated choice experiments, such that the resulting values can be easily used to compute routes in a network by changing edge weights.

Instead of conducting stated choice experiments, it is also possible to let the planner tweak parameters using illustrative sliders such that results of his parameter choices are directly visible in planned routes. Such a method could be used in combination with the promising research on Customizable Route Planning [17], a new algorithm that is focussed on real-time traffic updates but could also be useful for incorporating different preference profiles.

Chapter 9

Conclusions

The objective of this thesis is to gain insight in the influence on arrival times if incorporating optimal planning of breaks and preferences, and to compute such routes as efficiently as possible. In the previous chapters all research questions to reach this goal are answered. A recap on the results is given in these final conclusions of this thesis.

The first research question, “What is the influence on arrival time if breaks are planned optimally?”, is answered by comparing the results of an optimal break planning algorithm to the currently used algorithm of ORTEC [39], that inserts breaks after planning the route. Of the test set – representing average freight flows through Europe – 6% of the trips lasting up to one day improves, with an average improvement of nearly 3 hours. Differences are caused by planning breaks during road blocks – the driving bans which are imposed to trucks in some countries on Sundays or during the night. Also longer trips having a duration of maximum two days are analysed. For these routes 17% improves with optimal break scheduling, with an average improvement of 5 hours. Also the influence of restricting these breaks to parking lots is analysed: this requires an additional 3 minute travel time on average. However, 5% of the routes become infeasible due to absence of parking lots nearby. Another 15% of the routes change completely – although not always affecting the travel time.

The second research question, “How can these routes be computed efficiently?”, is answered by presenting an algorithm using time-dependent contraction hierarchies. It is shown that this algorithm may not provide optimal answers in theory, but on the considered test set no differences are found with the optimal solutions. This algorithm gives running times of several milliseconds, making it possible to use it to compute large numbers of routes in short times. For the problem of incorporating parking lots in break planning, another heuristic method using time-dependent contraction hierarchies is presented, leading to a solution quality of 90% with running times a factor 80 less.

The third question “What is the objective of route planning of relevant stakeholders?”, is answered by providing a literature study on several route choice experiments.

This leads to a set of important preferences, including fuel-efficient driving, highway preference, congestion avoidance and toll avoidance.

The fourth question “What is the influence on the arrival time if this objective is used to compute routes?” is answered by using one stated choice experiment as a basis and using their parameters to plan routes. To be able to plan routes, first the effect coded parameters are changed to dummy coded parameters, followed by weighting them with data from revealed preference studies and after that scaling for travel time of the original choice study. This leads to a linear-additive formula which is used as an edge weight function in Dijkstra’s algorithm. The results show that the average influence of incorporating preferences is little: only 9 minutes, with trips with an average travel time of 7 hours. However, for some test cases large differences are found. The largest influences in travel time are caused by avoiding toll and congestion.

After answering the four research questions, an opinion is given on how the algorithmic and behavioural research community can help each other, such that in future by default best routes are computed instead of fastest routes. A suggestion is made on research on Alternative Routes, as well as to adapt Customizable Route planning toward usage with preferences. Furthermore, it is encouraged that choice studies are conducted or reported in such a way that results are directly usable in route planning applications.

Chapter 10

Discussion

This thesis addressed the problem of planning routes from start to destination for trucks, in the context of long-distance trips. Prior to the research, some assumptions were made (see Section 1-2-1). The expected influence of these assumptions is discussed in this chapter. First, the assumption on truck drivers strictly following routes is addressed, followed by the consequences of using the dataset of HERE [29]. In Section 10-3 the omitting of working hour rules and American drivers legislation is discussed. Finally, Section 10-4 discusses the influence of using earlier performed choice experiments instead of conducting new experiments.

Besides the assumptions made in Section 1-2-1, it is decided in Section 2-3 that only the problem variations *RoadBlock-OneBreak* and *Parking* are discussed in detail. However, by working on these variations, also ideas arose for solving the other variations. These ideas for *DriversLegislation*, *Congestion* and *Realistic* are given in Section 10-5.

10-1 Truck drivers are not dumb

This thesis assumes that trips are planned in the context of the *Vehicle Routing Problem* (VRP). Planning routes from and to each of the customers is used as an input to determine the optimal delivery sequence. The routes computed by the algorithms are then assumed to be strictly followed by truck drivers. Furthermore, it is assumed that no disruptions occur, such that computed arrival times correspond with actual arrival times.

However, most truck drivers will not strictly follow silly routes as computed by the software. Consider the earlier used example in Chapter 4, also shown in Figure 10-1. The purple route is the route computed by the current planning algorithms. This leads to a driving time of about two days. It is questionable whether a truck driver would strictly drive this silly route: he knows of the active road blocks himself and will definitely make sure he leaves Austria before the block starts. This results in an earlier arrival time at the destination. If it is not possible to avoid the road block, the truck driver would generally communicate on beforehand with the planner, requesting an earlier or later start time such that unnecessary waiting is avoided.

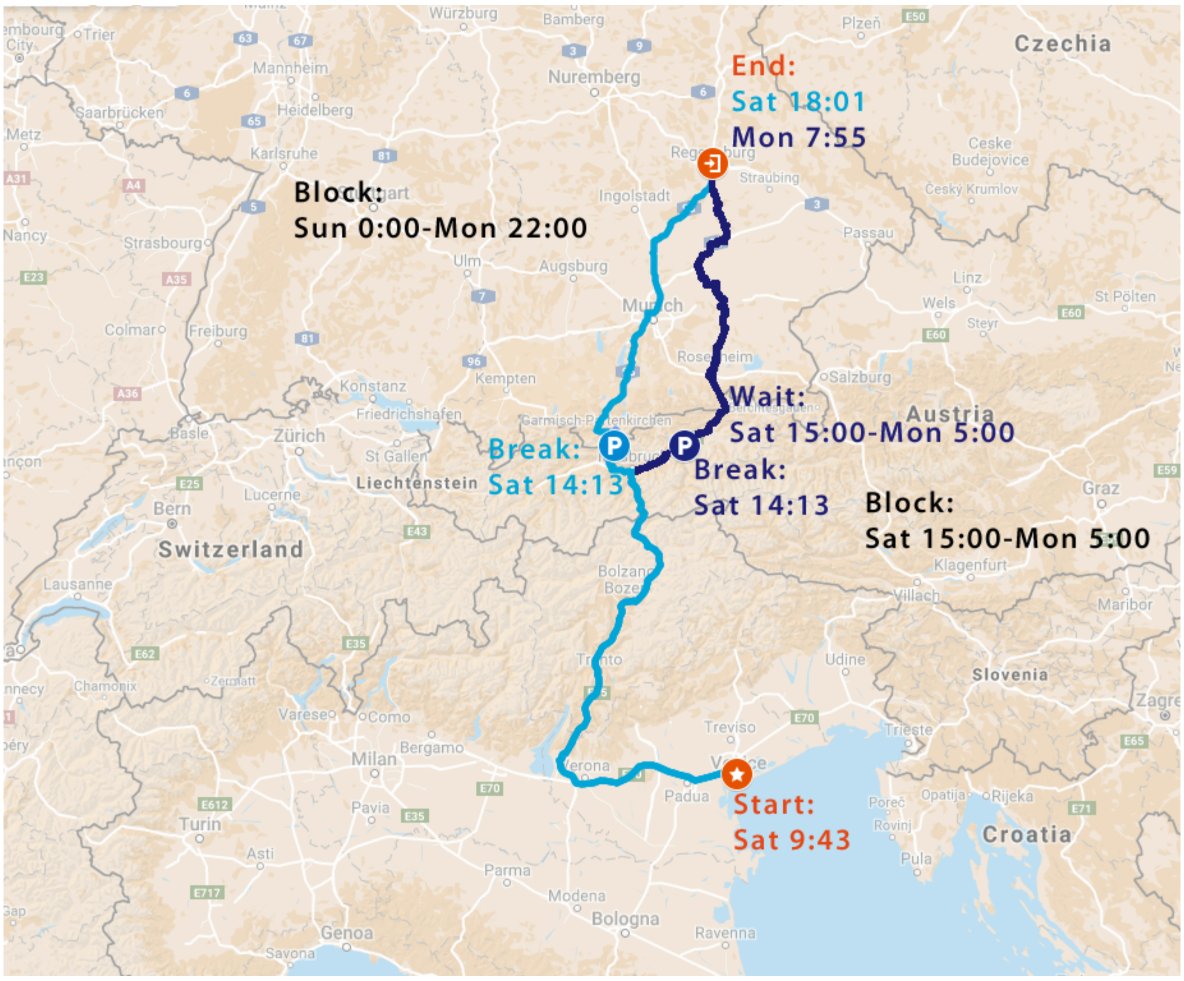


Figure 10-1: Example route of OneDay: start in the North of Italy, destination is located in Germany. A different route choice leads to a shorter travel time if considering breaks

It can be concluded that such worse routes are probably never driven in reality. The computed influences on arrival time are therefore probably overestimated. However, including optimal break planning does provide lots of opportunities for optimizing delivery sequences and for automating the route planning process without needing the knowledge of planners or truck drivers.

10-2 Inaccurate data on parking lots, congestion and road blocks

The data on the road network is provided by HERE [29]. Although excellent information is given on the links and nodes in the network, additional information is incomplete.

Precise data on parking lots seem to be missing in certain countries, resulting in some large distances between the optimal location to take a break and the nearest parking lot. It is reasonable that part of the 5.5% of routes not feasible if considering parking lots (Section 5-4-3), actually would be feasible if the used data on parking lots was more detailed.

Data on congestion used for computing best routes that avoid congestion, is taken from HERE [29] as well. The dataset provides travel time profiles that give the average travel time for every 15 minutes of a day. These average travel times do not reflect the chances on congestion: a 1% chance on 100 minutes delay or a 50% chance on 2 minutes delay both lead to an average delay of 1 minute, although reflecting possibly different disutilities. However, no detailed data was available on the reliability of travel times. It is advisable to use more detailed or even real-time traffic data if a certain company has a strong preference for avoiding congestion. However, for the context of this thesis, the influence on routing decisions as computed in Chapter 7 is not considered to be too high.

Furthermore, no data on road blocks is included in the dataset. This is added manually by searching through all governmental web sites for rules on truck allowance. The information on road blocks is complex: every country has its specific rules on time restrictions, but also on truck types for which these are valid. At special days (i.e. during Christmas) or periods (i.e. summer holiday) sometimes additional road blocks are imposed. For this thesis only road blocks are concerned that are valid throughout the year. Secondly, it is assumed that all road blocks are active for all trucks in the simulations. This is not the case in reality: some road blocks are active for trucks weighing more than 16.5 tons (i.e. in Great Britain), other for trucks weighing more than 7.5 tons (i.e. in Italy) or even for trucks weighing just more than 3.5 tons (i.e. in Switzerland). Furthermore, there are lots of exceptions for the bans. For example, cooled transport is not affected by road blocks in most countries. Although the used data on road blocks is not so precise, it still gives good insight in problems in route planning occurring due to these bans. For a real application it is advisable to implement the full set of rules dependent on the actual truck type of the logistics company.

10-3 Only European drivers legislation

In this thesis, only the European Drivers Legislation [22] are used in the simulations. However, in Europe there is also a Directive on the working hour rules. These working hour do not just consider driving, but also time used for administration and loading. This might have its effect on computed routes, but can be easily implemented in the algorithms. By adding a counter for consecutive working time and checking whether the working time threshold is not exceeded, one can easily obligate taking early breaks if needed. The influence on the arrival time is fully dependent on the amount of work to be performed besides driving, and therefore no estimations can be made for a general case.

Only rules valid in Europe are considered. Other countries have different regulations, such as the Hours-of-Service regulation in the United States [24]. However, all regulations are combinations of limits on driving time and certain break durations. Therefore, these can be easily implemented using the algorithms introduced in this thesis, by changing the threshold and break values. The influence on routing decisions using these different regulations is unknown, but will mostly be affected by the active road blocks in these countries.

10-4 Using choice studies to compute the preferred route

This thesis only uses stated and revealed preference data for computing preferred routes. No new choice experiments or surveys are conducted. The effect of using the specific choice studies and specific conversion method of the utility function is already discussed in Section 7-6. It is advised to conduct a new revealed preference study with only relevant parameters (probably travel time, congestion and toll) included before using in real applications.

However, it is also possible to not use any choice studies but let the freight company or planner interactively decide upon its own preferences. Such a method is used by the Fietsersbond (see Figure 8-1), which uses sliders to determine a personal profile. By letting planners or truck drivers select their own preferences and viewing the resulting route, they can decide for themselves which set of parameters results in their personal best route. Probably, this method leads to better parameters than conventional choice studies due to the direct link with actual routes in a road network.

10-5 Non-discussed problem variations

In Section 2-3 five different problem variations are defined, of which only two are discussed in this thesis. However, by working on the *RoadBlock-OneBreak* and *Parking* variations, ideas are formed for solving the other variations. In this section, ideas for each of the three problem variations are discussed.

10-5-1 DriversLegislation

Problem variation *DriversLegislation* is one of the variations which did not gain any attention. This variation deals with a full time horizon of driver legislation, that is, the possibility for multiple breaks during a trip. Secondly, the flexibility in the regulations, such as the allowed splitting of breaks or extending of driving times, is included in this variation.

In this thesis, one break is implemented using a *Stacked Break Graph*, where the upper layer represents the nodes at which a break has been taken. One could extend this idea by not adding one copy of the graph, but multiple copies, each representing a certain number of breaks taken during the trip. However, this still leads toward a fixed number of breaks during a route, directly reflecting the number of layers added to the extended stacked break graph.

Therefore, a better idea is to keep track of the *status* of a driver using several counters. Labels containing the status and arrival time are then settled in the Dijkstra Pareto algorithm instead of nodes. The used edge weight function then takes both status and arrival time as an input, and gives a list of new labels as an output, each reflecting a possibility for taking different kind of breaks as allowed according to the regulations.

Although all drivers legislation rules (including flexibility rules) can be implemented in such an edge weight function, the disadvantage is that enormous amounts of labels are generated reflecting all options of taking breaks, resulting in long running times and high memory usage of the Pareto algorithm. Optimization can be reached by throwing away labels that are not likely to form a good solution, such as labels representing paths that have low shift duration counters but arrive much later than others due to taking many breaks. However, results are not guaranteed to be optimal any more with such a heuristic method.

The influence of incorporating the full set of drivers legislation on the simulation results, will mainly be reflected by not having any infeasible routes in the test-set. It is expected that adding flexibility of splitting breaks only has marginal influence since optimally, breaks are scheduled during road blocks, and road blocks last for longer time periods than breaks. Splitting is therefore not beneficial.

10-5-2 Congestion

The *Congestion* variation adds congestion as additional time-dependent component. Instead of computing travel times using the free flow travel time, a function representing the variable travel times due to congestion should be used. The travel time functions for break edges should be adjusted such that the travel time after taking a break is adjusted to the actual traffic situation at that moment. In the stacked break graph, these travel time functions can be implemented in the Pareto algorithm without any further required modifications.

For the time-dependent contraction hierarchies this is different, since it cannot be directly retrieved what the travel time over a shortcut edge is, if a break is taken at this shortcut edge. For example, the rush hour might start while taking a break along the road, resulting in a longer travel time than at the departure time at the start of the shortcut edge. It is unknown how this problem can be solved efficiently in time-dependent contraction hierarchies.

The effect on the arrival time mainly depends on the predictability and duration of the congestion. It is expected that the planning of short breaks (with a duration of 45 minutes) is mainly affected by congestion. For example, it might be beneficial to take an early break during rush hour.

10-5-3 Realistic

The solutions to the problem variations *RoadBlock-OneBreak*, *Parking*, *DriversLegislation* and *Congestion* can be combined into a full version of the problem, referred to as *Realistic*. Since incorporating congestion and parking lots do not require much effort, it seems the best idea to take the *DriversLegislation* problem solution as a basis.

To compute the possible travel times over an edge (that is, with or without taking a break), one can use a black box approach, that takes the arrival time and driver status as an input, and gives one or more labels as an output. Such an example flow chart approach is shown in Figure 10-2.

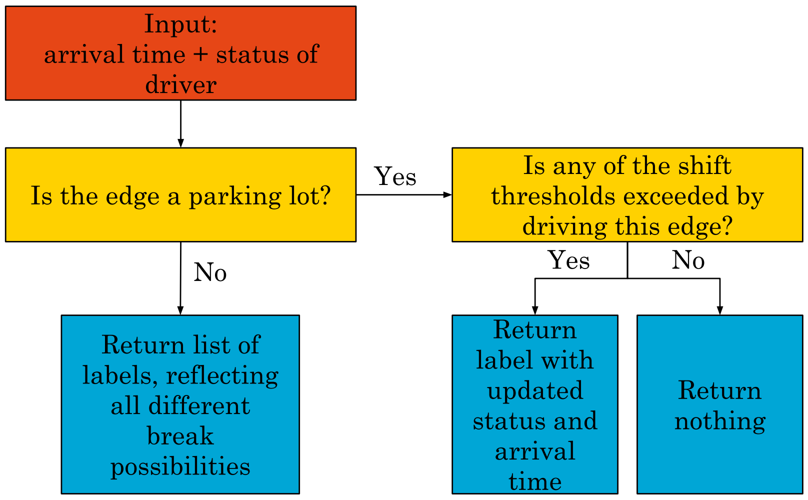

Input:
arrival time + status of driver
Is the edge a parking lot?
Yes
Is any of the shift thresholds exceeded by driving this edge?
No
Yes
No
Return list of labels, reflecting all different break possibilities
Return label with updated status and arrival time
Return nothing

Figure 10-2: Example process reflecting travelling options with break planning. The function used to obtain the travel time incorporates congestion.

However, such an approach results in enormous amounts of possible Pareto optimal paths, especially due to the numerous break options possible for every edge as already mentioned for the *DriversLegislation* variation. The number of possible paths grows even more due to considering congestion, resulting in new possibilities for smart planning of breaks. This requires lots of running time and memory usage. Furthermore, it is unknown how contraction hierarchies can be used to incorporate optimal break planning with congestion, which make it hard to obtain speed-ups in running time.

However, it is assumed that congestion and flexibility in drivers legislation have less influence on the arrival time than considering road blocks. Also the influence on the arrival time if restricting to parking lots is small, as is shown in Chapter 5. Furthermore, real-time data on congestion (i.e., accidents) probably have larger influence on the travel time than the daily rush hour periods. Therefore, it is questionable whether it is worth the hassle to compute the results of the *Realistic* problem variation, since most gain in travel time is expected to be reached by planning breaks during road blocks, thus by using the *RoadBlock-OneBreak* algorithms.

Appendix A

Road blocks in Europe

This thesis uses road blocks in the simulations. However, the data on road blocks is not provided within the road network dataset of HERE [29]. Therefore, the information on road blocks is found by searching governmental websites. Each country has its specific restrictions on road blocks and has its own definition of a truck for which the restriction is valid. In this thesis no specific weight restrictions are taken into account: all road blocks for trucks are considered. Furthermore, some countries have road blocks that are only valid on some specific roads, or only at special periods in the year. These are not included as well.

Of the following European countries, no information on active road blocks is found: Albania, Andorra, Armenia, Belarus, Belgium, Bosnia and Herzegovina, Estonia, Finland, Gibraltar, Iceland, Ireland, Kosovo, Latvia, Lithuania, Macedonia, Malta, Moldova, Monaco, Montenegro, the Netherlands, Norway, San Marino, Serbia, Sweden, Ukraine and Vatican.

Of the remaining countries, the table on the next page shows the information found for each of the countries, and the used road block information in this thesis.

	Road block in reality	**Assumed road block in simulations**
Austria	Blocked every night between 22:00 and 5:00, additionally on Sunday from 15:00 till Monday 5:00.	Monday 22:00-Tuesday 05:00
Bulgaria	Blocks on some specific road sections.	No
Croatia	Some roads are blocked on Sundays from 15th of June till 15th of September.	No
Cyprus	No	No
Czech Republic	Sunday 13:00-22:00	Sunday 13:00-22:00
Denmark	Night driving ban around Copenhagen between 19:00 and 7:00	No
France	Saturday 22:00 till Sunday 22:00.	Saturday 22:00-Sunday 22:00
Germany	Sunday 0:00-22:00	No
Great Britain	Some night and weekend bans around London	No
Greece	Blocks valid on some highways for 1st of June till 30th of September.	No
Hungary	Blocked between 1st of September till 30th of June for Saturday 22:00 till Sunday 22:00. In July and August the block is valid between Saturday 15:00 and Sunday 22:00.	No
Italy	Blocked on Sundays between 8:00 and 22:00. From June to September the starting time changes to 7:00.	Sunday 8:00-22:00
Luxembourg	Saturday 21:30 till Sunday 21:45, for transport between Belgium or Germany and France.	Saturday 21:30-Sunday 21:45
Poland	Some blocks during public holidays	No
Portugal	Tunnels prohibited for trucks.	No
Romania	Different Sunday blocks for different road segments.	No
Slovakia	Sunday 00:00 - 22:00	Sunday 00:00 - 22:00
Slovenia	Sunday 8:00-21:00	Sunday 8:00-21:00
Spain	Sunday ban for some roads toward Madrid and Barcelona.	No
Switzerland	Blocked every night between 22:00 and 5:00, additionally on Sunday from 0:00 till Monday 5:00.	Every night 22:00-5:00, Sunday 0:00 - Monday 5:00

Appendix B

Results for variations of the utility function

In Section 7-3 it is decided that the used utility function should contain a linear instead of logarithmic travel time component, and should be scaled according to the travel time of the original experiment. The decision is made by reasoning and comparing routes obtained by running simulations for each of the scaling and travel time combinations.

For the considered test instances, for each of the four combinations of travel time and scaling method, the differences in travel time compared to the fastest route are shown in the table below. The relative differences are computed by first dividing the travel time of the utility best route by the fastest route for each of the routes, and then taking the average/minimum/maximum/standard deviation of these numbers.

		Distance scaling		**Time scaling**	
		Logarithmic	**Linear**	**Logarithmic**	**Linear**
Average difference	**Absolute**	69 min	26 min	15 min	9 min
	Relative	21.3%	7.2%	3.3%	1.8%
Minimum difference	**Absolute**	1 min	0 min	0 min	0 min
	Relative	0.36%	0%	0%	0%
Maximum difference	**Absolute**	1431 min	1344 min	120 min	109 min
	Relative	531.12%	499.01%	35.32%	18.89%
Standard deviation	**Absolute**	178 min	111 min	25 min	18 min
	Relative	54.8%	40.5%	5.4%	3.3%

It can be seen that scaling for distance leads to large deviations: on average 69 or 26 minutes, at maximum even more than 20 hours! Recall that the average trip duration is 7 hours. These large differences are mainly caused by trying to obtain the shortest route as possible – which lasts much longer compared to the fastest route – and then bumping into some road block.

Differences between logarithmic and linear travel time functions are much less. Especially if scaling for time, differences are small. However, as explained in Section 7-3-2 the logarithmic travel time function results in a low Value of Time for long distance trips, since the other attributes are linear. If all attributes are in a logarithmic form, this would work perfectly well.

Bibliography

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "A hub-based labeling algorithm for shortest paths in road networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6630 LNCS, pp. 230–241, 2011.

[2] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "Alternative routes in road networks," *Journal of Experimental Algorithmics*, vol. 18, no. June, pp. 1.1–1.17, 2013.

[3] T. Arentze, T. Feng, H. Timmermans, and J. Robroeks, "Context-dependent influence of road attributes and pricing policies on route choice behavior of truck drivers: Results of a conjoint choice experiment," *Transportation*, vol. 39, no. 6, pp. 1173–1188, 2012.

[4] R. Bain and L. Polakovic, "Traffic Forecasting Risk Study Update 2005 : Through Ramp-Up And Beyond," *Standard & Poor's Global Project Finance Yearbook*, vol. 104, no. October, pp. 65–68, 2005.

[5] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, "In Transit to Constant Time Shortest-Path Queries in Road Networks," *Proc. of the Workshop on Algorithm Engineering and Experiments*, pp. 45–59, 2007.

[6] H. Bast, S. Funke, P. Sanders, and D. Schultes, "Fast Routing in Road Networks with Transit Nodes," *Science*, vol. 316, no. 5824, pp. 566–566, 2007.

[7] V. Batz, *Time-Dependent Route Planning with Contraction Hierarchies.* Karlsruher Institut fur Technologie (KIT), Diss., 2014.

[8] R. Bauer and D. Delling, "SHARC: Fast and robust unidirectional routing," *Journal of Experimental Algorithmics (JEA)*, vol. 2, pp. 13–26, 2009.

[9] M. C. J. Bliemer and A. T. Collins, “On determining priors for the generation of efficient stated choice experimental designs,” *Journal of Choice Modelling*, vol. 21, pp. 10–14, 2016.

[10] C. Bräuer and M. Baum, *Optimale zeitabhängige Pausenplanung für LKW-Fahrer mit integrierter Parkplatzwahl.* Bachelorarbeit, Karlsruhe Institute of Technology, 2016.

[11] C. Chen and Y. Xie, “The impacts of multiple rest-break periods on commercial truck driver’s crash risk,” *Journal of Safety Research*, vol. 48, pp. 87–93, 2014.

[12] R. L. Coulter, W. R. Darden, M. K. Coulter, and G. Brown, “Freight transportation carrier selection criteria. Identification of service dimensions for competitive positioning,” *Journal of Business Research*, vol. 19, no. 1, pp. 51–66, 1989.

[13] R. Craft and D. Blower, “The large truck crash causation study,” *Paper presented and distributed at the November*, vol. 17, no. November, p. 2004, 2004.

[14] A. Daly, T. Dekker, and S. Hess, “Dummy coding vs effects coding for categorical variables: Clarifications and extensions,” *Journal of Choice Modelling*, vol. 21, no. September, pp. 36–41, 2016.

[15] G. B. Dantzig, *Linear programming and extensions.* Princeton university press, 1962.

[16] B. C. Dean, “Shortest paths in FIFO time-dependent networks: Theory and algorithms,” *Rapport technique, Massachusetts Institute of ...*, pp. 1–13, 2004.

[17] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, “Customizable Route Planning,” *International Symposium on Experimental Algorithms*, pp. 376–387, 2011.

[18] D. Delling and D. Wagner, “Time-dependent route planning,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5868 LNCS, pp. 207–230, 2009.

[19] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[20] M. Ehrgott, *Multicriteria Optimization.* Springer Science & Business Media, 2006.

[21] A. Eiger, P. B. Mirchandani, and H. Soroush, “Path Preferences and Optimal Paths in Probabilistic Networks,” *Transportation Science*, vol. 19, no. 1, pp. 75–84, 1985.

[22] European Parliament Council of the European Union, “Regulation (EC) No 561/2006 of the European Parliament and of the Council of 15 March 2006 on the harmonisation of certain social legislation relating to road transport and amending Council Regulations (EEC) No 3821/85 and (EC) No 2135/98 and repealing Co,” 2006.

[23] European Transport policy Information Systems. Etis database. [Online]. Available: http://www.etisplus.eu/ (Accessed 25-02-2017).

[24] Federal Motor Carrier Safety Administration. Hours of service. [Online]. Available: https://www.fmcsa.dot.gov/regulations/hours-of-service (Accessed 23-02-2017).

[25] Fietsersbond. Fietsersbond routeplanner. [Online]. Available: https://routeplanner.fietsersbond.nl/ (Accessed 05-05-2017).

[26] M. Fiorenzo-Calatano Stella, *Choice Set Generation in Multi-Modal Transportation Networks*. Netherlands TRAIL Research school, 2007.

[27] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5038 LNCS, no. July, pp. 319–333, 2008.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[29] HERE. Here map data. [Online]. Available: https://here.com/en/products-services/data/here-map-data/ (Accessed 17-02-2017).

[30] S. Hess, M. Quddus, N. Rieser-schüssler, and A. Daly, "Developing advanced route choice models for heavy goods vehicles using GPS data," *Transportation Research Part E*, vol. 77, pp. 29–44, 2015.

[31] S. Ichoua, M. Gendreau, and J. Y. Potvin, "Vehicle dispatching with time-dependent travel times," *European Journal of Operational Research*, vol. 144, no. 2, pp. 379–396, 2003.

[32] G. Janssen, H. Zwijnenberg, I. Blankers, and J. de Kruijff, "Future of Transportation Truck Platooning," *TNO Automotive*, no. TNO 2014 R11893, pp. 1–36, 2015.

[33] A. L. Kok, *Congestion Avoidance and Break Scheduling Congestion Avoidance and Break Scheduling within Vehicle Routing*. Beta Research Scool for Operations Management and Logistics, 2010.

[34] R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems: From a taxonomy to a definition," *European Journal of Operational Research*, vol. 241, no. 1, pp. 1–14, 2015.

[35] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (trip): Incorporating personalization into route planning," *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, no. 2, p. 1795, 2006.

[36] T. Lomax, D. Schrank, S. Turner, and R. Margiotta, "Selecting Travel Reliability Measures," *Texas Transportation Institute monograph (May 2003)*, no. May 2003, 2003.

[37] G. Nannicini, D. Delling, L. Liberti, and D. Schultes, “Bidirectional A* search for time-dependent fast paths,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5038 LNCS, no. 2, pp. 334–346, 2008.

[38] A. Orda and R. Rom, “Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length,” *Journal of the Association for Computing Machinery*, vol. 37, no. 3, pp. 607–625, 1990.

[39] ORTEC. [Online]. Available: http://ortec.com/nl-nl/ (Accessed 29-09-2016).

[40] C. G. Prato, T. K. Rasmussen, and O. A. Nielsen, “Estimating Value of Congestion and of Reliability from Observation of Route Choice Behavior of Car Drivers,” *Transportation Research Record: Journal of the Tranpsortation Research Board*, vol. 2412, pp. 20–27, 2014.

[41] M. Pylkkönen, M. Sihvola, H. K. Hyvärinen, S. Puttonen, C. Hublin, and M. Sallinen, “Sleepiness, sleep, and use of sleepiness countermeasures in shift-working long-haul truck drivers,” *Accident Analysis and Prevention*, vol. 80, pp. 201–210, 2015.

[42] M. Rowell, A. Gagliano, and A. Goodchild, “Identifying truck route choice priorities: the implications for travel models,” *Transportation Letters*, vol. 6, no. 2, pp. 98–106, 2014.

[43] P. Sanders and D. Schulters, “Engineering Highway Hierarchies,” *ACM Journal of Experimental Algorithmics*, vol. 42, no. 42, 2006.

[44] G. Scora, K. Boriboonsomsin, and M. Barth, “Value of eco-friendly route choice for heavy-duty trucks,” *Research in Transportation Economics*, vol. 52, pp. 3–14, 2015.

[45] C. Sommer, “Shortest-path queries in static networks,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–31, 2014.

[46] S. S. Srinivas and M. S. Gajanand, “Vehicle routing problem and driver behaviour : a review and framework for analysis,” *Transport Reviews*, vol. 0, no. 0, pp. 1–22, 2016.

[47] Y. Sun, T. Toledo, K. Rosa, M. Ben-Akiva, K. Flanagan, R. Sanchez, and E. Spissu, “Route Choice Characteristics for Truckers,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2354, pp. 115–121, 2013.

[48] Z. Tarapata, “Selected Multicriteria Shortest Path Problems: An Analysis of Complexity, Models and Adaptation of Standard Algorithms,” *International Journal of Applied Mathematics and Computer Science*, vol. 17, no. 2, pp. 269–287, 2007.

[49] The Telegraph. Calais protests: Lorry drivers begin blocking roads amid anger over violent tactics of migrants trying to reach uk. [Online]. Available: http://www.telegraph.co.uk/news/2016/09/04/

calais-migrants-urged-to-stay-in-jungle-during-port-blockade/ (Accessed 17-11-2016).

[50] T. Trego and D. Murray, "An Analysis of the Operational Costs of Trucking," *American Transportation Research Institute*, no. September, 2010.

[51] Truckers report forum. Night driving vs day time. [Online]. Available: http://www.thetruckersreport.com/truckingindustryforum/threads/night-driving-vs-day-time.287664/ (Accessed 17-11-2016).

[52] Truckers report forum. Split sleeper break. [Online]. Available: http://www.thetruckersreport.com/truckingindustryforum/threads/split-sleeper-break.189180/ (Accessed 17-11-2016).

[53] A. Van den Engel, "Driving restrictions for heavy goods vehicles in the european union," *Report, Zoetermeer*, 2010.

[54] J. W. C. van Lint, H. J. van Zuylen, and H. Tu, "Travel time unreliability on freeways: Why measures based on variance tell only half the story," *Transportation Research Part A: Policy and Practice*, vol. 42, no. 1, pp. 258–277, 2008.

[55] M. V. D. Voort, M. S. Dougherty, and M. V. Maarseveen, "A prototype fuel-efficiency support tool," *Transportation Research Part C*, vol. 9, pp. 279–296, 2001.