The Dial-a-Ride Problem

and

Exploiting Knowledge about Future Customer Requests

Thesis Supervisor: Andreas Klose

Jesper Bjerring Hansen
2003 2423

Master's Thesis in Mathematics-Economics

Department of Mathematical Sciences
Aarhus University - 2010

Abstract

This thesis describe the dial-a-ride problem (DARP), dynamic and static, with focus on the dynaic DARP, as this version of the DARP resembles real-life instances best.

We introduce the concept of fruitful corridors, an expansion of fruitful regions, for dynamic vehicle routing problems with two locations associated with each request. Fruitful corridors is used to describe expected travel patterns for future requests.

We propose a new solution method for the dynamic DARP (FCMSA), that make use of fruitful corridors and a multiple scenario approach. The FCMSA is an expansion of an existing insertion heuristic for the dynamic DARP.

The FCMSA has been tested on simulated data with about 230 requests and an estimated degree of dynamism at 20 %. Only few testes have been made, one a single test instance. The tests suggest that the FCMSA not is able to improve the original insertion heuristic. A simulation method for providing test instances for a (dynamic) DARP is also introduced.

More tests should be carried out, preferably on real-life instances.

Resumé

I dette speciale beskrives det statiske og det dynamiske dial-a-ride problem (DARP). Fokus er på det dynamiske DARP, da det ligner autentiske tilfælde bedst.

Vi introducere et nyt begreb kaldet frugtbare korridorer for ruteplanlægningsproblemer med to beliggenheder tilknyttet en ordre. Frugtbare korridorer er en udvidelse af frugtbare regioner for ruteplanlægningsproblemer med kun en beliggenhed per ordre.

Vi fremsætter ene ny løsningsmetode (FCMSA) for det dynamiske DARP. Den ny løsningsmetode udvider en eksisterende indsættelses heuristik for det dynamiske DARP. FCMSA bruger frugtbare korridorer og en multi scenario tilgang.

FCMSA er blevet teste på simuleret data med omkring 230 kundehenvendelser og en estimeret dynamisk grad (degree of dynamism) på 20 %. Der er kun foretaget få tests i realtid. Testene antyder at FCMSA ikke formår at forbedrer den oprindelige heuristik som den udvider. En simulerings metode til at konstruere test data for et (dynamisk) DARP bliver også introduceret.

Flere tests bør foretages i realtid og helst på autentisk data.

Contents

1

Introduction

This thesis is about the dial-a-ride problem. While the thesis focuses on analytic versions of the dial-a-ride problem, it is worth keeping in mind, that there are real-world dial-a-ride problems behind the theoretical models. Real-world dial-a-ride problems is a small part of a greater public transportation problem.

There are many benefits from public transportation compared to individual transportation. A bus, train or other public transportation vehicle transporting a group of people, who otherwise would transport themselves, each in their own car, costs less than the alternative, produce less CO2 than the alternative and takes up less road space than the alternative. These benefits are also true for efficient dial-a-ride transportation, but one other benefit related to public transportation shine through in the context dial-a-ride transportation traditionally is used; namely the social aspect of public transportation.

Without a public transportation system, some groups of individuals, for instance children, elderly and physically disabled, would practically rely on family and friends when they have a transportation need. Especially for elderly and physically disabled, dial-a-ride transportation is an important part of a socially efficient public transportation system. More recent, dial-a-ride services have become important in another context in Denmark. As bus routes are closed in low populated areas, people living in these areas without the possibility to, or the wish to, transport themselves is referred to dial-a-ride services, such as Midttur (Midttrafik), Sydtur (Sydtrafik) and Flextur (Movia).

This thesis tries to make the operations of a dial-a-ride service more economically viable without having to cut down on service for the customers, by providing insights on the way to go in the area of solving analytical dial-a-ride problems, and thereby aid dispatchers in solving resembling real-world instances of the dial-a-ride problem.

Chapter 2 is a description of dynamic vehicle routing problems, the dynamic dial-a-ride problem is a dynamic vehicle routing problem. In chapter 3 the dynamic dial-a-ride problem is studied, as well as the static counterpart. In chapter 4 two existing solution methods for the dial-a-ride problem are scrutinised. Inspired by these algorithms, a new solution method for the dynamic dial-a-ride problem is introduced in chapter 7.

2

Dynamic vehicle routing

In this chapter we introduce some terminology concerning the dynamic vehicle routing problem in general. Then we state some assumptions made about the dynamic vehicle routing problem, some of them being in contrast to real life dynamic vehicle routing problem instances. After we have the assumptions and terminology in place, we look at constraints and objectives in different dynamic vehicle routing problems, making it possible to suggest solution methods. Finally we look at the differences between the static and the dynamic vehicle problem. But we start this chapter with a very rough and broad definition of the general vehicle routing problem.

The *Vehicle Routing Problem* (VRP) is the problem where a dispatcher has a fleet of vehicles and a set of geographically scattered customers requiring some service from the vehicles. The objective of the VRP is often to minimize the travel distance or travel costs of the vehicles in operation, but other objectives are also common, for instance maximising the number of serviced customers.

The vehicle routing problem is often time dependent, in the sense that service is due in certain time. This could for instance be before deadlines, in time windows or after other jobs are finished. If this is the case, the requests in question must be sequenced in a fashion that takes this time dependency into account, both in terms of feasibility and optimality.

The vehicle routing problem is also a set partitioning problem, since all customer requests must be assigned to exactly one vehicle. But the problem is more complicated than just determining a set partitioning, since the assignment must be done in a way that takes into account the sequencing of the requests. In other words, the assignment sub-problem and the sequencing sub-problem are not independent of each other. Consider for instance what would make a good solution to the independent assignment problem. This solution might make the sequencing problem infeasible.

There exist several variations of the VRP; some worth mentioning are:

1. The *Capacitated Vehicle Routing Problem* (CVRP); the vehicles can only carry a limited number of goods or passengers.

2. The *Vehicle Routing Problem with Time Windows* (VRPTW); the vehicles can only serve customers in some specific part of the day, specified for each customer.

Other variations worth mentioning in the context of this thesis are:

3. The *Pick-up and Delivery Problem* (PDP); when serving a pick-up customer/location the same vehicle later has to serve an associated delivery customer/location.

4. The *Vehicle Routing Problem with Heterogeneous Vehicles* (VRPHV); the vehicles are not necessarily alike in carrying capacity (in contrast to the widely assumed homogeneous models).

The above abstract problems can be combined, making even more variations of the VRP. On example is for instance the Capacitated Pick-up and Delivery Problem with Time Windows and Heterogeneous Vehicles, which we will not give an abbreviation, since it is very much alike the version of the dial-a-ride problem (DARP), which this thesis will focus on. The DARP will be introduced in chapter 3.

In the above definition of the VRP we make no assumption on the arrival of the customer requests. In classical VRPs, all customers are assumed to arrive in advance. In the dynamic vehicle routing problem, which we study in this chapter, some of the customer requests are assumed first to arrive, and therefore first to be known, at the time when service of other customers has begun. A new customer must be assigned to a vehicle (assuming that the given customer is not denied service). Therefore reassignment and resequencing of an initial solution becomes necessary, if not all new customers is to be denied service or a new vehicle is assigned to each customer, which most likely would not make a good solution. More on this matter in section 2.4.

2.1 Terminology

When referring to the *dispatcher* we talk about a theoretical dispatcher, not assuming that there (necessarily) is only one human dispatcher to handle the routing, or that the dispatcher necessarily is a human being. In other contexts the dispatcher would be known as the decision maker.

We talk about *requests* that have to be serviced (or in some models the dispatcher has to decide to service the request or not). A request can be made by *customers* in the model. The request can be a job that has to be done on *location* or some transporting of an object between *locations*. Handling a request is referred to as a *service*. In dynamic vehicle routing we distinguish between *static* or *advance requests* and *dynamic* or *immediate requests*. Static requests are known to the dispatcher at the starting time of execution of the services, while dynamic requests are made and become known during the performance of services.

Vehicles are different in different models. In some models all vehicles in the fleet are assumed to be identical (*homogeneous fleet*). In other models the fleet consists of vehicles with different attributes (*heterogeneous fleet*). The attributes describing a vehicle include a *crew* of one or more persons. Different crews are

able to perform different services. Another attribute is the *capacity* of the vehicle, the capacity is not necessarily one number. The vehicle may be able to perform a number of on-locations jobs before returning to a depot, and at the same time transport another number of objects. The objects that have to be carried in some models may be different, and the vehicles capacity for one kind of objects can depend on the number the vehicle carries of another object and vice versa.

A *route* is a series of locations associated with services that are to be performed. In the case where all requests can be handled on location, only one service is performed at a time. But if some of the requests concern transportation, more services can be preformed at the same time. All pairs of locations have one or two *shortest travel times* and/or *shortest distances* associated with them. The travel time from location A to location B is not necessarily the same as the travel time from B to A. The travel time might change during the day.

2.1.1 Information

An underlying feature of dynamic vehicle routing problems is that all information needed to solve the problem are not known ahead of the execution of the routes. Part of the information is dynamically revealed as time goes by. We will now describe what information is relevant for the dynamic vehicle routing problem and when it will be available.

Information relevant to the dynamic vehicle routing problem includes, but is not limited to; nodes of customer locations, distances between nodes, fleet size, capacity of vehicles, customer demands, travel times between nodes, etc.. The information can be divided into two main groups; static information, which is known to the dispatcher in advance; and dynamic information, which is first known to the dispatcher when service of the customers has begun.

Psaraftis (1995) introduces a taxonomy to characterise attributes of the information for the dynamic vehicle routing problem. We will restate this taxonomy here.

- *Evolution of information.* Dynamic information is not known when the service of customers begins, and will be revealed or updated as time goes on. Information that is static is known for the entire duration of the routing and is not updated. Even though static information is known and not updated, it can change. If static information change though, it is also known when it changes and what the changes will be.

- *Quality of information.* Some information is known with certainty, hence deterministic. Other information is known with uncertainty and forecasts are used. Again other information follow a known probability or evolve according to a known stochastic process, making the information probabilistic. The quality of information is related to a specific time, and may change. For instance, a probabilistic information can change and will become deterministic when it is realised.

- *Availability of information.* Information can be divided into local information, only known in some parts of the system, for instance to a certain vehicle (the crew) and global information known for the entire system. Even though modern information technology makes it possible (and cheap) to reveal information to the whole system, some information may be decided by the dispatcher not to be relevant for the individual vehicles. For instance when the schedules for the vehicles are recalculated, the dispatcher might choose only to reveal the next location (or the next small number of locations) to each vehicle. This choice can be made by the dispatcher, instead of providing vehicles with information that might, and probably will change when the schedule of the vehicles are recalculated.

- *Processing of information.* In some systems all information is processed by a central unit (the dispatcher), these are called centralised. In other systems, certain information is processed separately by for instance a vehicle. These models are called decentralised.

2.2 Model vs. real-world problem

Like Psaraftis (1995), we emphasise that when we use the word "problem", we refer to abstract vehicle problems with corresponding analytical models defining them. Furthermore, we always use the the term "real-world problem" when referring to a non-abstract problem. To determine if a vehicle routing problem is dynamic or not, it does not matter if the real-world vehicle problem is dynamic. A vehicle routing problem is dynamic, only if some information is not available when we start the routing (or more precise when we determine the initial routes) in the analytical model describing the real-world problem. Bearing this distinction in mind it is clear that static vehicle routing models can be used to describe real-world vehicle routing problems that are dynamic. Most real-world vehicle routing problems, if not all, are actually dynamic. Even if all customers are known in advance, other events can break the static property of the real-world problem. Such things as vehicle breakdowns or cancellations by customers can always occur.

In this thesis we assume a geographical information system (GIS) including a spatial data base (SDB) of the relevant geographical area, geocoding and a spatial adaptation (SDA) tool. Operations in the GIS is assumed handled outside the presented and proposed methods in this thesis. We limit ourself to handling orders already adapted to a mathematical network and producing mathematical output in terms of vehicle sequences and assignments for later interpretation.

For more information on the architecture of a GIS, we refer to figure 2.1 where the architecture of a decision support system is shown. The texts next to the arcs are the data going between the different components of the system. The square covers the part of the system handled by the GIS. We will focus on the part of the DSS outside the square and mainly on the last step before the output, the "Vehicle routing solver". Hence we assume when provided with customer data, we can put this data directly into our system. The figure is from

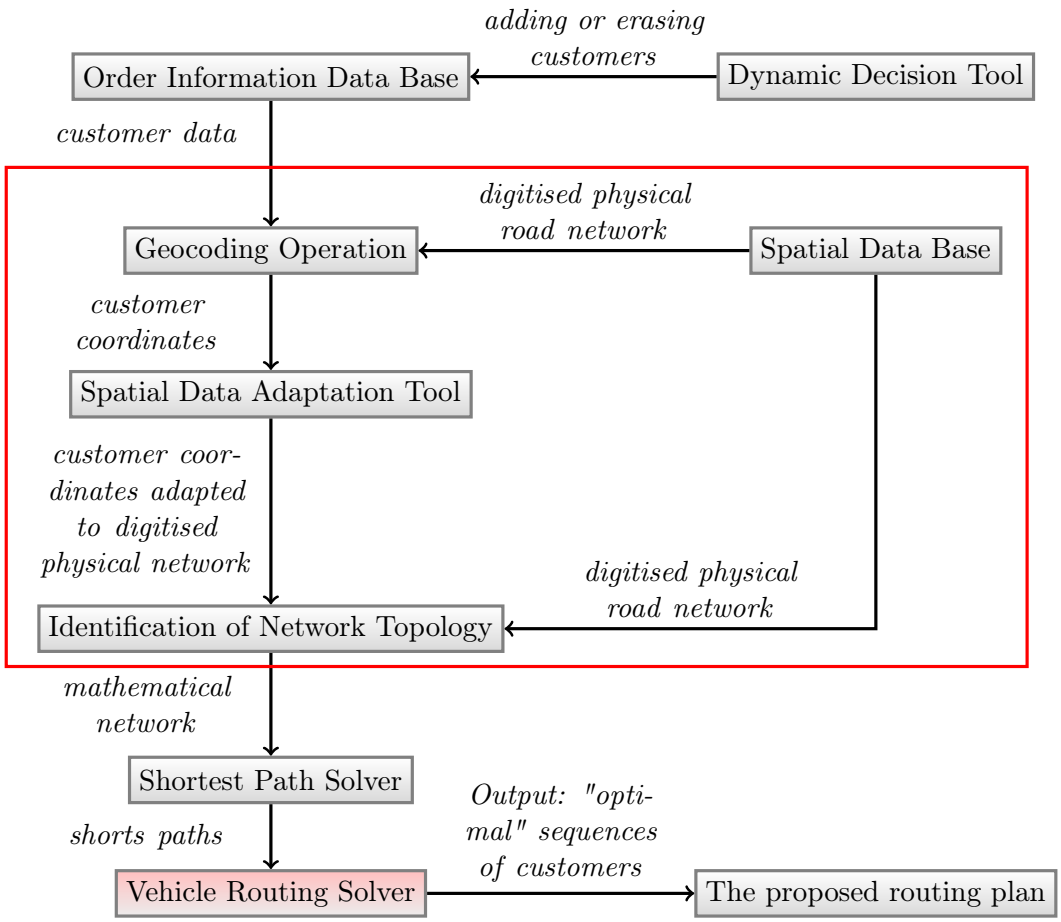

Figure 2.1: Architecture of a decision support system.

(Tarantilis, Diakoulaki, and Kiranoudis, 2003), where the components of the GIS also is described.

2.3 The dynamic vehicle routing problem

Having studied the relevant information for vehicle routing problems, we define the static vehicle routing problem in the words of Larsen (2000):

Definition 2.1 (The static vehicle routing problem.)

1. All information relevant to the planning of the routes is assumed to be known to the dispatcher before the routing process begins.

2. Information relevant to the routing does not change after the routes have been constructed.

The *Dynamic Vehicle Routing Problem* (DVRP)[1] is the problem where a dispatcher has a fleet of vehicles and a set of geographically scattered customers

[1] Also known as the *real-time* vehicle routing problem and the *on-line* vehicle routing problem.

requiring some service and some or all of the information relevant to the problem is not known by the time the vehicles start serving customers. Or in the words of Larsen (2000):

Definition 2.2 (The dynamic vehicle routing problem.)

1. Not all information relevant to the planning of the routes is known to the dispatcher when the routing process begins.

2. Information can change after the initial routes have been constructed.

Remark 2.3 In definition 2.1 and 2.2 the *routing process* refers to the route planning and the subsequent execution of the routes constructed. In the dynamic VRP a part of the routing process is *re-routing* since new information may become available after the routing process has begun. When referring only to the actual driving between and servicing of customers, according to output of the routing process, we talk about *execution of service* or just *execution*.

Some examples where DVRPs might appear, are combined pick up and delivery services, oil delivery services and dial-a-ride services, for more examples see (Psaraftis, 1995) and (Larsen, 2000). We give a short description of these examples, as they are used as examples throughout this chapter.

Pick up and delivery services

The dispatcher operates a fleet of vehicles and receives requests both in advance and while the vehicles service other customers. The dispatcher assigns a vehicle to service the incoming requests and modifies the route associated with the current vehicle if necessary.

Distribution of heating oil

The dispatcher operates a fleet of oil trucks, servicing a set of customers (normally known in advance), with a not precisely known demand for oil. Since the oil demand is not known with certainty; there is a risk of a truck running out of oil. Hence the truck needs to return to the depot, before the route is finished. Since the oil demand of the customers are not known with certainty. Situations when a request for oil is received, while servicing other customers, occur. In such a situation the new request has to be serviced as soon as possible.

Dial-a-ride services

The dispatcher operates a fleet of small buses (and taxis), and receives a combination of in advance and real-time requests to carry elderly, disabled and other people from a specified pick up location to a specified destination. Dial-a-ride services are described extensively in chapter 3.

2.3.1 Constraints

We look at some common constraints for DVRPs.

Capacity

Constraints can be on the capacity of the vehicles. Though in some problems, the capacity will never become a problem and this constraint becomes irrelevant. One example of this is the pick-up and delivery service for letters. Since letters are very small, the capacity of a vehicle can be assumed to be irrelevant.

The number of vehicles / Availability of vehicles

In some models the number of vehicles is limited. This would be the case if we operate an oil delivery service, where a marked for renting vehicles on short term basis is non-existent. In other models we might have our own main-fleet of vehicles, and some backup fleet of vehicles that could be used if not a sufficient number of own vehicles are available, which we will see is the case for some instances of the dial-a-ride service.

If the number of vehicles is limited, situations where no vehicle is available might occur. If this happens, then customers have to be rejected or offered service at a later time, where vehicles again are available. If the model does not allow to reject customers, there always have to be some vehicles available (hence the number of vehicles are assumed to be theoretical unlimited). The main fleet might still be limited, but some backup vehicles is then assumed to be available. For instance if we operate a dial-a-ride service, the backup can be a taxi suited for the customer request; or if we operate a pick up and delivery letter service, the backup vehicle might be a rival company.

Route length

Constraints can be on the length of a route or sub-route, this can be due to fuel issues of the vehicle or for instance in the problem of oil delivery the vehicle might run out of corporate oil. This constraint might also be due to driver breaks in accordance with collective and other arrangements.

Time windows

In some models there are time windows on pick up and/or delivery. For instance the pick up and delivery service, the customer should be able to decide a pick up time and/or a delivery time and then the dispatcher should provide the customer with a time window when the request is accepted.

Transport time

Constraints can be on the transport time of an object. In the dial-a-ride service there should be a maximum transport time, this transport time would often be a function of the travel time between the pick up location and the destination.

Costs of operations

The dispatcher might face a budget, hence there can be a constraint on the costs of operation. In this case throughput could be the objective to maximise.

2.3.2 Objectives

In this section we look at some different objectives used by the dispatcher associated with different goals in the models. We first look at some classic objectives for the static VRP.

Minimising the cost of operations. In a cost oriented model the dispatcher wants to minimise the cost of operations. Service goals are only enforced by the constraints with minimum limits for service.

Minimising the maximal travel time. If travel time or distance is crucial, minimisation of the maximal travel time or distance can be the objective. Limits on the costs are ensured by the constraints.

Maximising throughput. Another service goal might be to serve as many customers as possible, assuming no constraint enforces service of all customers. In this case costs are also ensured not to grow unlimited by the constraints.

The objectives presented for the static VRP are relevant for the DVRP as well, but other objectives are also common in the dynamic environment are:

Maximising service level. The objective of the model might be to maximise level of service provided for the customers. The service level can be measured in punctuality in desired service time, travel times or other.

Multiple criteria. The objective in the DVRP can also be, and often is, a combination of the above. Maximising service level include many different aspects. So maximising service level is in it itself often to be considered as multiple criteria.

In 2.3.1 we looked at a lot of constraints, in some models these constraints can be incorporated in the objective of the model. The constraints incorporated in the objective could for instance be constraints on the service level. If a constraint is incorporated in the objective, we can relax the actual constraint. The incorporation of constraints in the objective becomes much more crucial in the DVRP compared to the static VRP. This is the case because the dynamic requests make it harder to use tight constraints to ensure a minimum of service and at the same time be able to make feasible solutions. Assuming customers want a "less good" service rather than be denied service, these kind of objectives make us able to guarantee some service without setting the service levels totally aside.

We end this section about the objectives of the DVRP, remembering that all information available should be considered by the objective functions. In the

static VRP all information is known in advance with certainty. In the DVRP some information might be available in other quality, but should still be considered by the objective function nonetheless (Psaraftis, 1988).

2.4 Differences between the static and dynamic vehicle routing problems

We will now look at some differences between VRP and DVRP. Larsen (2000) lists 12 differences, which we restate here. These 12 differences are also listed in (Psaraftis, 1995).

1. *Time dimension is essential*: While time *can* be important in the static case, time is always important in the dynamic case. In the dynamic case time is essential even if there is no time constraints, since we need to know the state of information (like vehicle location) in relation of time since new events occur, and must be handled with updated information.

2. *Problem may be open ended*: Duration of the routing process in a static setting is known in advance, since the routes are known when the routing starts. In the dynamic setting the duration of the routes might not be known in some scenarios, since new customers can occur and the routes are extended .

3. *Future information may be imprecise or unknown.*

4. *Near-term events are more important*: Since information does not change in the static VRP, all events can be handled with the same weight of relevance. This is not the case for the DVRP since information concerning long-term events[2] is more likely to change than near-term events.

5. *Information update mechanisms are essential*: Information updating does not make any sense in static vehicle routing since all information is assumed static! On the other hand this is essential for dynamic vehicle routing since the available information changes, and to make good solutions we need to update the information available for the solution methods fast and very often.

6. *Resequencing and reassignment decisions may be warranted*: When new information makes a current solution suboptimal, the dispatcher is forced to reroute or reassign the vehicles to deal with the new information.

7. *Faster computation times are necessary*: See section 2.5.

8. *Indefinite deferment mechanisms are essential*: If a customer is geographical isolated when demanding service, the service of the customer may be

[2]This is information associated with the event itself, information of events near the event in time and location and new events that are near in time and location.

postponed until more demands turn up near the location of the customer. If indefinite deferment is allowed, the effects caused by it can be alleviated by using a penalising mechanism for customer waiting time in the objective function of the problem.

9. *Objective function may be different*: See section 2.3.2.

10. *Time constraints may be different*: A dynamic customer is expected to want service in a softer time window, rather than be denied service if the dispatcher is not able to provide service within a desired time window. Hence time windows and latest pick up times often are softer in a DVRP than in a static VRP.

11. *Flexiblity to vary vehicle fleet size is lower*: Since the time from obtaining a solution to the time of operations is longer in the case of the static VRP, fleet size may be easier to vary, regarding renting vehicles.

12. *Queueing considerations may become important*: A DVRP can become congested, if more immediate requests arise than the system can handle offhand. If this is the case, queueing considerations for immediate requests may be warranted, in order for the system to work meaningful.

2.5 Solution methods

In a static VRP, the dispatcher is able to solve the entire problem at once, perhaps overnight or the day before operations if all information related to the problem is available. This is not the case for the DVRP since the lack of all information in advance is what characterises the problem. The dispatcher therefore has to solve the problem in a sequential fashion (Psaraftis, 1995). The dispatcher makes an initial solution using the information available (including forecasts and probabilistic information). Then when new information arrives or changes, he tries to resolve the new problem. The dispatcher here faces the problem that he has to make the new solution very quickly, hence he does not have the luxury to "wait" for the "best solution"[3], but need to obtain a "good solution" in short time.

A solution method to the DVRP consists of an initial solution and some way to deal with information arriving later which respects the constraints in the model. The solution method should be interactive, be hierarchical, user-friendly and have a restart capability (Psaraftis, 1988).

Interactive: DVRPs are normally used in DSSs, and therefore solutions to DVRPs are often a part of a DSS. The produced solution is made in a model reflecting a real-world problem, but a human dispatcher might know features that are not considered in the model. This could be the case if the designer of the model does not know the real-world problem in

[3]Since the static vehicle routing problem is NP-hard, an optimal solution is not always an option to problems of a certain size. Notice that the dynamic vehicle routing obviously also belong to the class of NP-hard problems!

enough detail. Or more important if some events occur rarely enough to be considered "unpredictable" by the model, with the purpose of keeping it simple. If such inside observations are made by a human dispatcher, then the dispatcher should be able to override the solution method manually.

Restart capable: When updating schedules and assignments, and the execution of the routes is under way, some vehicles may not be available to carry out service and may be located differently than when the last solution was made. As well as some customers may have been serviced or are currently being serviced. Therefore the solution method should always update lists of relevant customers by removing customers already and presently served, and adding immediate customers as soon as they occur. Furthermore the solution method should be able to update vehicle positions and a list of vehicles available in some manner to be used when updating the current solution for the rest of the schedules and assignments.

Hierarchical: A solution method for a DVRP should quickly be able to determine if a request can feasibly be serviced, before determining the best way to serve it if possible. This is so, because in the case the system finds that the request can not be feasibly serviced, the dispatcher will be able to make inquiries to the customer for an alternative request. It is therefore important that feasibility checking is made early in the solution method, and feasibility checks (when more than one exists) are hierarchically carried out in a smart fashion, where checks most likely violated, taking the computational effort into account, are done first.

User-friendly: User-friendly design of a solution method is more important in a DVRP than a static VRP, because there is much more interaction between the solution method and human dispatchers feeding the immediate request. Some versions of the DVRP might have decentralised the feeding of input, so that customers can put requests in the system themselves via the Internet. This makes user-friendly design even more important.

3

The dial-a-ride problem

In chapter 2 we studied the DVRP in general. In this chapter we will describe a sub-problem to the DVRP; the dynamic dial-a-ride problem (DARP). In chapter 2 we saw that the general DVRP is closely related to the static counterpart. In this context, the dynamic DARP might be said to be even more related to the static DARP. We say so, because the static DARP is interesting for solving the dynamic DARP, since many real-life problems of the dynamic DARP is characterised with a low percentage of dynamic customers. So perhaps solution methods for the static DARP can be adopted to make good starting solutions for the dynamic DARP. We look into this in chapter 4. For now we will leave the solution methods and describe the DARP, static and dynamic.

The dial-a-ride problem (DARP) is the name of many different problems in the literature. This so because the DARP is used in several applications; (door-to-door) transportation of elderly, door-to-door transportation of physically disabled, public transport in areas with low population density, and some problem formulations incorporate more than one of these applications. The focus in this chapter and the rest of the thesis, will be on models with door-to-door transportation.

In section 3.1 we describe the information available to the dispatcher, when solving a DARP. We formalise this information for easy reference when used. In section 3.1.4 we describe the routes and schedules that will make up solutions to the DARP.

In section 2.3.1 we looked at some common constraints in relation to the general DVRP. In section 3.2.1 we will do the same, but limited to the DARP. Again we touch upon solution methods, since parallel to describing the constraints, we introduce feasibility checks for the constraints relevant, assuming an insertion heuristic is used.

In section 3.2.3, after having described common constraints and objectives for instances of the DARP, we describe a few instances of problem formulations of the DARP. Two of them found in the literature, and a new one constructed partly from a real-life DARP in the Midtjylland-region, Denmark. The last model will later be used for testing a solution method for the dynamic DARP, and is described in more detail in chapter 6.

In section 3.5 we describe some measures on requests arrivals and travel patterns to characterise an instance of DARP.

3.1 Information and assumptions

In this section we will look at the information available to the dispatcher, when solving the DARP, and assumptions made about the information available. We do not distinguish between static and dynamic information, but present all the information that may be relevant in different static and dynamic DARP models. The information is grouped in information about the fleet and vehicles, information about the customers and their requests, and finally, information about the geographical area of interest. We start of with the last.

3.1.1 Map

Let $G = (V, E)$ be a directed graph representing a map of the area of interest for some dial-a-ride service provider. The vertex set V then represent locations in the area, for instance pick up and delivery locations for customers or possible customers and depots associated with vehicles used in the dial-a-ride service. The edge set E represents *shortest distances* (δ_{ij}) and/or *travel times* (τ_{ij}) from one location in $v_i \in V$ to another location in $v_i \in V$. In some models an edge is only labelled with a shortest distance or shortest travel times, this would be the case if we assume that vehicles travel with unit speed at all time.[1] Travel times may vary over time, for instance from day to day but also during a day of operations, consequently the edge can be labelled with more than one shortest travel time. The shortest distances themselves will not change, but if some event make the graph change, for instance removal of an edge due to a congestion or road repair, the shortest travel times can be affected.

If the dispatcher at the time of scheduling knows the appearance of the graph and the travel times at all time, this information is static. If on the other hand the change in travel times and appearance of the graph are not known with certainty at the time of scheduling, the information about the graph is dynamic.

In some models, distances are assumed to be Euclidean. If so, the graph G can be made from the coordinates representing locations, using them as vertices, and letting the direct distance between them be the arcs of the graph.

3.1.2 Fleet and vehicles

The *capacity* of a vehicle was introduced in 2.1, we denote the capacity of vehicle b_k by Q_k. Q_k may, and often does, consist of multiple capacities. For instance the *capacity for wheelchairs*, Q_{1k}, (individuals in wheelchairs during the ride), the *capacity for seated individuals*, Q_{2k}, and the *capacity for seated wheelchair users* (individuals who are seated during the ride, but also need transportation of a collapsible wheelchair), Q_{3k}. Then $Q_k = (Q_{1k}, Q_{2k}, Q_{3k})$. Some vehicle might only be able to seated individuals, in that case Q_{1k} and Q_{3k} is zero, other vehicles

[1] A vehicle travelling a unit speed, traverse one unit distance in one unit of time. For instance, if we assume a vehicle travel at $60km/h$, and time is measured in minutes and distance is measured in km's, then the vehicle travels one km per minute, i.e. the vehicle travels at unit speed.

are perhaps able to carry a mix of seated individuals and seated wheelchair users, in that case Q_{2k} is zero. If the vehicle can carry a mix of individuals from the three groups, the current capacity for one group might depend on the current capacity for other groups. The multiple capacity is not only limited to three terms of capacity, as shown in the above example. In some of the models we encounter later, more groups are used, as well as groups of individuals characterised by other attributes.

The *crew* of the vehicle b_k, denoted by H_k, also may have an impact on the vehicles ability to handle different groups of individuals. For instance a vehicle has the size to handle seated wheelchair users, but might no be able to get them in and out of the vehicle if the crew just consists of one person. This means that one vehicle can have different characteristics depending on the crew.[2] H_k is assumed to be a binary variable, representing if the vehicle carries a crew of one or two members.[3]

When a vehicle drops off and picks up a customer, there is some service time associated with it. Some of the service times is directly and only related to the customer and does not depend on the vehicle and crew, for instance the time it takes the customer to get from her residence to the vehicle. We refer to this as *service times* and will get back to this in section 3.1.3. But other parts of the time used to pick up and drop off a customer, are customer depended as well as vehicle depended. For instance, if the vehicle uses a lift to load a customer type into the vehicle, this loading time might differ between vehicles and is therefore vehicle dependent. Vehicle depended service time is referred to as *loading* and *unloading times* Γ_k^+ and Γ_k^- respectively. Γ_k^+ and Γ_k^- depend on the vehicle (type) b_k and the customer request (group) r_l. In the example above with three customer groups, we would for vehicle (type) b_k get $\Gamma_k^+ = (\Gamma_{k1}^+, \Gamma_{k2}^+, \Gamma_{k3}^+)$ and likewise for Γ_k^-.

Associated with vehicles are some costs. The costs can be split into a fixed cost and some variable costs. The fixed cost for vehicle b_j driving a route, is denoted Y_j. A variable cost associated with vehicle b_j driving a km (or other unit of distance), is denoted U_j. And a variable cost associated with vehicle b_j providing a minute (or other unit of time) of service / waiting, is denoted by Z_j.

In some DARP models, we associate depots with each vehicle. The *origin depot*, O_j, is the location where vehicle b_j starts before servicing requests, and the *destination depot*, D_j, is the location where vehicle b_j drives after servicing requests. O_j and D_j might be at the same location. Other vehicles might be *non-depot-based*. This would be the case for a taxi, when the dispatcher uses such a vehicle to handle a request. No depots are associated with these vehicles, but artificial depots might be used for convenience, locating the origin depot at the pick up location for the request, and locating the destination depot at the destination of the request.

A vehicle might not be available the whole day of execution. We denote the

[2] A crew of two instead of one would also lead to a lower capacity for seated individuals and seated wheelchairs users, since the non driving crew member would occupy a seat.

[3] This means that we assume the crew's ability to help a certain customer group is only a matter of the size of the crew, and not for instance the education of the crew.

start time, of the period where vehicle b_j is available by P_j^+, and the end time, of this period, is denoted by P_j^-.

We group vehicles by types. A *vehicle type*, B_l, depends on the multi capacity, the crew, loading and unloading times, the cost and depots of the vehicle. For any given vehicle type, B_l, these values are the same for all vehicles $b \in B_l$. If the fleet is heterogeneous, the size of the fleet might not be a set of fixed numbers of different vehicles. The number of available vehicles of one type may depend on the number of available vehicles of a different kind. For instance if a vehicle with a crew of one is given a crew of two, then the first vehicle group is reduced with one vehicle and the second vehicle group is increased by one vehicle.

Vehicle availability varies because some vehicles for instance are due for repair, or the dispatcher vary the fleet size by renting more vehicles during "rush hours". Fleet size is in some models theoretically unlimited, if we assume that we at all time instantly can rent a vehicle of a desired type. In other models the fleet size is limited. Again other heterogeneous models use a mix of unlimited and limited fleet size, in the sense that the size of some vehicle types is limited, while other are theoretically unlimited. An example could be where the dispatcher in a DARP model operates a fleet of own minibuses, and hires taxis when the minibuses are not available. The total fleet in a DARP is denoted B and consists of the different vehicle types, $B = B_1 + B_2 + \cdots + B_\mu$, where μ is the number of vehicle types.

3.1.3 Customers and requests

When a customer makes a request to the dispatcher in a dial-a-ride service, the customer provides the dispatcher with some information, and the dispatcher returns back some information to the customer.

Let us start with the input information available to the dispatcher, when a customer makes a request r_i. The customer tells the dispatcher where to be picked up (pick up location v_i^+) and where to go (destination v_i^-), characteristics of the individual (or individuals) that needs transportation (*capacity request* q_i) and *desired pick up and/or delivery times* p_i^+/p_i^-. Information about *service time* to get to and from the vehicle (t_i^+/t_i^-) and the customers possible *assistance need* (h_i), for instance to get up and down stairs, is also made available to the dispatcher.[4]

When the dispatcher has the above information, he first informs the customer if the request is accepted or not (if it is allowed to turn down service in the model). If the request is accepted, the customer is also informed about *time windows* ($[e_i^+, \ell_i^+]/[e_i^-, \ell_i^-]$) and *maximum travel time* (λ_i).

If the customer is only allowed to make a request for desired pick up time *or* delivery time, then the request is called *outbound* if the customer requests a specific delivery time, and *inbound* if the customer requests a specific pick up time. We use a binary variable ω_i associated with request r_i to store if the request is out- or inbound. For $\omega_i = 0$, r_i is outbound, and for $\omega_i = 1$, r_i is inbound.

[4]The last two kinds of information might be available to the dispatcher ahead of the request in stored data with information about the customer or the customer type.

Now we have $r_i = (q_i, h_i, v_i, p_i, t_i)$ where $q_i = (q_{i1}, q_{i2}, \ldots, q_{i\nu})$ is a row-vector in $\mathbb{N}^\nu$ with the number of individuals of different groups, $v_i = (v_i^+, v_i^-)$ is a row vector in V^2 with the pick up location and destination and p_i is a row vector in T^2 or T representing *desired pick up and/or delivery times*, where T is the set of point of times during the day of operation.

Remark 3.1 The main reason to distinguish between outbound and inbound requests, is that many DARP models only let the user have a desired arrival or departure time per request. It is then evident, that outbound requests are associated with requests with desired arrival times, for instance a trip from home to a doctor's appointment. Inbound requests are associated with request with desired departure times, for instance trips homeward.

3.1.4 Routes and schedules

We define *routes* closely to the vehicles that carry them out. Let ξ_j be a route driven by vehicle b_j. Then we know the ordered set of locations $V(\xi_j)$ and the set of requests $R(\xi_j)$ associated with ξ_j. $V(\xi_j)$ is ordered according to the sequence the locations have to be visited by b_j, in accordance with ξ_j. $R(\xi_j)$ represents all the requests in vehicle b_j in accordance with ξ_j. For all locations $v_\alpha \in V(\xi_j)$, $R(\xi_j^\alpha)$ is the set of requests carried by b_j after leaving v_α. $T(\xi_j)$ denotes the *schedule* of times when vehicle b_j starts service at the corresponding locations in $V(\xi_j)$. The maximal duration of a route ξ_j is denoted Λ_j. The set of all routes for all vehicles in a solution is called the *plan* and is denoted Ξ.

Terminology and associated symbols.

Symbol	Term
$\delta_{ij} = \delta(v_i, v_j)$	Shortest distance from location v_i to location v_j.
$\tau_{ij} = \tau(v_i, v_j)$	Shortest travel time from location v_i to location v_j.
r_i	Request associated with customer i.
R	Set of request.
$q_i = r_i(q_i)$	Multi capacity request associated with customer i.
$q_{il} = r_i(q_{il})$	Capacity request for individuals of group l associated with customer i
$h_i = r_i(h_i)$	Assistance needed by customer i.
$v_i^+ = r_i(v_i^+)$	Pick up location for request i. ($v_i^+ \in V$)
$v_i^- = r_i(v_i^-)$	Destination for request i. ($v_i^- \in V$)
$p_i^+ = r_i(p_i^+)$	Desired pick up time of customer i.
$p_i^- = r_i(p_i^-)$	Desired delivery time of customer i.
ω_i	Request is outbound if $\omega_i = 0$ and inbound if $\omega_i = 1$.
$p_i(\omega_i) = r_i(p_i(\omega_i))$	Desired pick up *or* delivery time of customer i, depending on the value of ω_i.
$[e_i^+, \ell_i^+]$	Time window for picking up customer i.
$[e_i^-, \ell_i^-]$	Time window for delivery of customer i.
$t_i^+ = r_i(t_i^+)$	Service time associated with pick up of request i.

continues

Terminology and associated symbols (continued).

Symbol	Term
$t_i^- = r_i(t_i^-)$	Service time associated with delivery of request i.
$\lambda_i = r_i(\lambda_i)$	Maximal transportation time for request i .
b_j	Vehicle j.
B	Set of vehicles (the fleet).
B_l	Set of vehicles of type l.
$Q_j = b_j(Q_j)$	Capacity for vehicle j.
$Q_{jl} = b_j(Q_{jl})$	Capacity of individuals of group l for vehicle j.
$H_j = b_j(H_j)$	Crew who mans vehicle j.
$O_j = b_j(O_j)$	Origin depot location for vehicle j.
$D_j = b_j(D_j)$	Destination depot location for vehicle j.
$\Gamma_j^+ = b_j(\Gamma_j^+)$	Loading time for vehicle j.
$\Gamma_{jl}^+ = b_j(\Gamma_{jl}^+)$	Loading time for customer (group) l for vehicle j.
$\Gamma_j^- = b_j(\Gamma_j^-)$	Unloading time for vehicle j.
$\Gamma_{jl}^- = b_j(\Gamma_{jl}^-)$	Unloading time for customer (group) l for vehicle j.
$Y_j = b_j(Y_j)$	Fixed cost associated with vehicle j for driving a route.
$U_j = b_j(U_j)$	Cost associated with vehicle j for driving a kilometre (or some other fixed distance).
$Z_j = b_j(Z_j)$	Cost associated with vehicle j for each minute of service (or some other fixed amount of time).
$P_j^+ = b_j(P_j^+)$	Start time of the period where vehicle b_j is available.
$P_j^- = b_j(P_j^-)$	End time of the period where vehicle b_j is available.
ξ_j	Route driven by vehicle j.
Ξ	The plan (set of routes).
$V(\xi_j)$	The ordered set of locations associated with route driven by vehicle j.
$R(\xi_j)$	The set of all requests associated with route driven by vehicle j.
$R(\xi_j^\alpha)$	The set of requests associated with location v_α in route driven by vehicle j.
Λ^j	Maximal duration of route associated with vehicle j.

3.2 The dial-a-ride problem

3.2.1 Constraints and checks for feasibility

The constraints found relevant in the instances of the DARP studied and introduced in this thesis are the following:

(i) **Maximal route duration:** We refer to The European Union's regulation on driving and rest rules for professional drivers, which states that a driver is not allowed to drive for more than nine hours a day,[5] this is set as an upper limit, Λ, for the maximal duration for a route.

(ii) **Driver breaks:** Again we refer The European Union's regulation on driving and rest rules for professional drivers, which states that after a driving period of four and a half hours (for convenience also denoted $\Lambda/2$) the driver shall take a break of at least 45 minutes, unless it is the end of the driving for the day.[6]

When a vehicle is on a break, the vehicle has to be empty!

(iii) **Maximal travel time for a ride:** To ensure a certain service for customers, we set an upper limit on the travel time associated with a ride. *Ride* referring to the transportation of a single request from its pick up location to its destination. The rules for the maximal ride time is different in the different models for the DARP. One common way to construct a maximal ride time, which we will adopt, is to use a factor greater than 1 of the direct travel time $\tau(v_i^+, v_i^-)$:

$$\lambda_i = \sigma \cdot \tau(v_i^+, v_i^-), \tag{3.1}$$

where $\sigma \geqslant 1$ is the same for all customers.

(iv) **Time windows:** All requests have one or two time windows associated with them. For time windows on pick up, the beginning of service shall be started within the time window. For time windows on delivery, service has to be finished within the time window.

The size of the time window can theoretically be as small as the desired pick up or delivery time or as big as the rest of the day of operations.

The time windows can take many forms, but is defined in accordance with the desired pick up and delivery times. Rules to construct the time windows are different in different models.

A simple construction of a time window for the pick up of customer i could be as follows

$$[e_i^+, \ell_i^+] = [p_i^+ - \sigma_1, p_i^+ + \sigma_2] \tag{3.2}$$

[5] Ten hours of driving is allowed for two days per week per driver, but this is ignored for simplicity.

[6] This break may be replaced by a break of at least 15 minutes, followed by a break of at least 30 minutes each distributed over the period, in such a way that breaks falling before the four and half hours are done. Again, for simplicity, this exception is ignored.

where the constants $\sigma_1, \sigma_2 \geqslant 0$ are the same for all accepted customers.

(v) **Vehicle capacity.** When the vehicle capacity is a multi capacity, each current capacity of a customer group might depend on the current capacity of another group. Since the vehicle used in real-life DARPs are relatively small in capacity, we can list the different multi capacity combinations. An example of multi capacity combinations is a *type 2* vehicle from (Midttrafik, 2010), this vehicle, which we will use in an example later, can accommodate two different groups of customer requests. Group 1 are seated passengers and group 2 are wheelchair user seated outside the wheelchair. *Type 2* can carry up to either four request of group 1 and zero requests of group 2 *or* three requests of group 1 and one request of group 2 *or* two requests of group 1 and two requests of group 2.

(vi) **A request is pick up and delivered by the same vehicle.**

(vii) **A pick up of a request is always made before the delivery of the request.**

(viii) **A request is served one time and only one time.**

(ix) **All vehicles start and end their routes at the associated depots.**

Feasibility regarding constraint (vi), (vii), (viii) and (ix) is ensured by the way requests are handled in later methods for solving the DARP, all using insertion heuristics. Constraint (vi) is ensured, by the fact that the locations of a request always are inserted simultaneously. Constraint (vii) is guaranteed by the architecture of the insertion method. Constraint (viii) is ensured, by the fact that a request feasibly inserted into a route is not considered again (unless the route is cancelled). Constraint (ix) is ensured, by starting the construction of a route for a vehicle, by making the depots the first and last location to visit (if the vehicle is depot based).

We will in this section look at the rest of the constraints and introduce feasibility checks for these constraints, to help us make sure insertion of requests into routes are valid.

Assume a directed graph $G = (V, E)$ according to section 3.1.1 and assume we have a set of requests R, where

$$r_i = (q_i, v_i^+, v_i^-, p_i(\omega_i), t_i^+, t_i^-, \lambda_i, \omega_i) \quad \forall r_i \in R,$$

for which $q_i = (q_{i1}, q_{i2}, \ldots, q_{i\nu})$ and p_i is either a desired pick up or delivery time depending ω_i.

Also assume we have a set of vehicles B, where

$$b_j = (Q_j, O_j, D_j, \Gamma_j^+, \Gamma_j^-, Y_j, U_j, Z_j) \quad \forall b_j \in B$$

for which $Q_j = (Q_{j1}, Q_{j2}, \ldots, Q_{j\nu})$, $\Gamma_j^+ = (\Gamma_{j1}^+, \Gamma_{j2}^+, \ldots, \Gamma_{j\nu}^+)$ and $\Gamma_j^- = (\Gamma_{j1}^-, \Gamma_{j2}^-, \ldots, \Gamma_{j\nu}^-)$.

We introduce some variables related to the current route ξ_j for a vehicle b_j, used when describing the checks for feasibility for the constraints.

Variables and associated symbols.

$p_i^j = p(v_i, b_j)$	Variable representing the time instance when vehicle b_j starts servicing a customer at location $v_i \in V(\xi_j)$.
$p_i^{j+} = p(v_i^+, b_j)$	Variable representing the time instance when vehicle b_j starts servicing r_i's pick up location. Where r_i is a customer request in ξ_j.
$p_i^{j-} = p(v_i^-, b_j)$	Variable representing the time instance when vehicle b_j starts servcing r_i's destination. Where r_i is a customer request in ξ_j.
$w_{jl}^i = w_l(b_j, v_i)$	Current load of individuals in customer group l in vehicle b_j after leaving location $v_i \in V(\xi_j)$.

Remark 3.2 Note that the arrival time at location v_α in ξ_j not necessarily corresponds to p_α^j.

Let $\phi_j^\alpha = \phi(b_j, v_\alpha)$ be the *total service time at a location* v_α *for vehicle* b_j.

$$\phi_\beta^\alpha = \sum_{r_i \in R(\xi_\beta^\alpha)} (t_i^+(v_\alpha) + t_i^-(v_\alpha) + \sum_{l=1}^{\nu} (r_i(q_{il})\Gamma_{\beta l}^+(v_\alpha) + r_i(q_{il})\Gamma_{\beta l}^-(v_\alpha))) \quad \forall v_\alpha \in V, b_\beta \in B, \tag{3.3}$$

where $r_i(q_{il})$ is the q_{il} value from r_i,

$$t_i^+(v_\alpha) = \begin{cases} t_i^+ & \text{if } v_\alpha = v_i^+ \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

and likewise for $t_i^-(v_\alpha)$, and

$$\Gamma_{\beta l}^+(v_\alpha) = \begin{cases} \Gamma_{\beta l}^+ & \text{if } v_\alpha = v_i^+ \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

and again likewise for $\Gamma_{\beta l}^-(v_\alpha)$.

The total waiting time in route ξ_j is then:

$$\sum_{\alpha=1}^{\rho-1} ((p^{\alpha+1} - p^\alpha) - \phi_j^\alpha - \tau(v_{\alpha+1}, v_\alpha)) - \phi_j^\rho, \tag{3.6}$$

where ρ is the number of locations in ξ_j.

Then let a_k, d_k^e and d_k^ℓ denote the earliest possible arrival time, earliest departure time and latest departure time for a location v_k in a route ξ_j with associated

vehicle b_j, respectively. We will now describe some simple methods to calculate them in a recursive manner. Assume route ξ_j has l locations. Then

$$a_1 = e_j, \tag{3.7}$$
$$a_k = \max\{a_{k-1}, e_{k-1}\} + \phi_{k-1} + \tau_{k-1,k} \text{ for } i = 2, 3, \ldots, l \quad , \tag{3.8}$$

where e_j is the time where vehicle b_j is available for the dispatcher and $e_k = 0$ if no time window is active at location v_k. For the earliest possible departure time

$$d_1^e = e_j, \tag{3.9}$$
$$d_k^e = \max\{d_{k-1}^e + \tau_{k-1,k}, e_k\} + \phi_k \text{ for } i = 2, 3, \ldots, l. \tag{3.10}$$

And for the latest possible departure time

$$d_l^\ell = \ell_j, \tag{3.11}$$
$$d_k^\ell = \min\{d_{k+1}^\ell + \phi_{k+1} + \tau_{k,k+1}, \ell_k\} \text{ for } i = 1, 2, \ldots, l-1 \quad , \tag{3.12}$$

where ℓ_k equals the end of the time horizon in the day of operations, if no time window is active at location v_k. The above values are used to describe the schedule associated with a route, even though they do not state when the vehicle is to leave a location, they are useful when operating with multiple plans.

Now we let b_j be an arbitrary vehicle from B with an associated route ξ_j,[7] and try to insert a arbitrary new request r_i from R in the route. The insertion fulfil that the pick up is inserted before the destination, and if vehicle b_j is depot-based, both pick up location and destination is tried inserted between the depots. We only assume that the route is feasible, before we try to insert r_i. Hence the route might be empty before we try to insert r_i, then r_i is the first request to put in the route if feasible.

Assume r_i is inserted in ξ_j on two arbitrary places. The pick up location v_i^+ is put between v_1 and v_2 in $v(\xi_j)$ and the destination is put between v_3 and v_4 in $V(\xi_j)$. Besides obvious changes in $V(\xi_j)$ and $R(\xi_j)$, the insertion of r_i causes possible change in p_α^j and w_j^α for all $v_\alpha \in V(\xi_j)$ and thereby also for possible change in p_β^{j+} and p_β^{j-} for all $r_\beta \in R(\xi_j)$. V^{G} and R^{G} defined as the corresponding sets for ξ_j before r_i was inserted. After these possible changes are calculated, the constraints (i), (ii), (iii), (iv) and (v) has to be checked for feasibility. We now introduce these feasibility checks.

Feasibility check regarding maximal route duration. It has to be true that

$$\max(T(\xi_j)) - \min(T(\xi_j)) \leqslant \Lambda_j,$$

for the new route resulting from the insertion of r_i to be feasible. $\min(T(\xi_j))$ and $\max(T(\xi_j))$ is the first and last time element in ξ_j. If b_j is depot-based,

[7] If b_j is non-depot-based, the first location in the route, is the pick up location of the first customer, and the last location in the route, is the destination of the last customer in the route. If b_j is depot-based, the first location is O_j and the last location in the route is D_j.

they correspond to the the departure and arrival at the start and end depot respectively. The extra time needed to service and drive r_i after it is inserted in ξ^j is $t_i^+ + t_i^- + \Gamma_{ji}^+ + \Gamma_{ji}^- + \tau(v_1, v_i^+) + \tau(v_i^+, v_2) - \tau(v_1, v_2) + \tau(v_3, v_i^-) + \tau(v_i^-, v_4) - \tau(v_3, v_4)$.

Remark 3.3 If the second location inserted v_i^- is inserted directly after the first location inserted v_i^+, hence $v_3 = v_i^+$, it might not be obvious that the extra travel time due to the insertion of r_i is $\tau(v_1, v_i^+) + \tau(v_i^+, v_2) - \tau(v_1, v_2) + \tau(v_3, v_i^-) + \tau(v_i^-, v_4) - \tau(v_3, v_4)$. But this is actually the case, because if v_i^- is inserted right after v_i^+, then $v_4 = v_2$. And the result that the extra travel time is $\tau(v_1, v_i^+) + \tau(v_i^+, v_i^-) + \tau(v_i^-, v_2) - \tau(v_1, v_2)$.

The *waiting time* at a location v_α in route ξ_j is the excess time left in the schedule after vehicle have left $v_{\alpha-1}$ and arrived at v_α,

$$p^\alpha - p^{\alpha-1} - \phi_j^{\alpha-1} - \tau_{\alpha-1,\alpha} \quad ,$$

where p^α is the time b_j arrives at v_α according to route ξ_j. We assume that the vehicles leave a location as soon as possible and waiting time, if any, is located at the next location in the route not yet ready for service (this is, the next location v_α, in a route ξ_j, with a time window, where e_α is in the future when b_j arrives). This strategy is called *drive first* and is the only appropriate for the static DARP and mainly therefore this is the common strategy for the dynamic DARP as well (Mitrovic-Minic and Laporte, 2004). For more information on waiting strategies we refer to (Mitrovic-Minic and Laporte, 2004), but keep using the common *drive first* waiting strategy. Under the drive first waiting strategy the actual service time p_α^j at location v_α in a route ξ_j is $\max\{a_\alpha, e_\alpha\}$, where $e_\alpha = 0$ if no time window is present at location v_α.

Feasibility check for driver breaks.
If $\max(T(\xi_j)) - \min(T(\xi_j)) > \Lambda_j/2$, then check if their is a break in ξ_j. If there is no break in ξ_j, then the insertion of r_i in ξ_j is not feasible.

Feasibility check for maximal travel time for a ride
If one of the following equations is false, then the maximal travel time for a ride constraint is violated, and the insertion of r_i in ξ_j is not feasible.

$$p_k^{j-} - p_k^{j+} \leqslant \lambda_k \qquad \forall\ r_k \in R(\xi^j).$$

Feasibility check for time windows.
The following equations have to be true for the insertion of r_i in ξ_j to be feasible:

$$e_k^+ \leqslant p_k^{j+} \leqslant \ell_k^+ \qquad \forall\ r_k \in \{r_\alpha : r_\alpha \in R(\xi^j), r_\alpha(\omega_\alpha) = 1\}.$$
$$e_k^- \leqslant p_k^{j-} + t_k^- + \Gamma_{jk}^- \leqslant \ell_k^- \qquad \forall\ r_k \in \{r_\alpha : r_\alpha \in R(\xi^j), r_\alpha(\omega_\alpha) = 0\}.$$

This is, that vehicle b_j must start service at a pick up location within a given time window for the location, and b_j must be finished with service at a destination within a given time window for the destination. Note that not all locations in a route, necessarily have a time window associated with them, since customers are only given the option of one time window per request.

Feasibility check for vehicle capacity.
The capacity constraints hold if for all $v_\alpha \in V(\xi^j)$ there exists at least one combination of multi capacity, such that

$$w_{jl}^{\alpha} \leqslant Q_{jl}^{\mathrm{C}} \qquad \text{for } l = 1, 2, \ldots, \nu \quad ,$$

where Q_{jl}^{C} refers to Q_{jl} in some combination of multi capacity. We leave the combinations of multi capacity until we state an example constructed from (Midttrafik, 2010) later. For now we just assume one multi capacity for each vehicle. The feasibility check for vehicle capacity only has to be done for the locations that are between v_i^+ and v_i^- in $V(\xi_j)$, since locations before and after are not affected by the insertion.

If the above feasibility checks hold, then the insertion of r_i in ξ_j associated with vehicle b_j is feasible!

3.2.2 Objectives

The different objectives for the DARP are numerous, but some are more common than others. As described in section 2.3.2 for general static VRPs and DVRPs, one main difference in common objectives in the static and the dynamic DARP is that objectives in a common dynamic DARP include many aspects which in the static counterpart would be maintained by constraints.

We will now describe the objectives used in two models for a dial-a-ride service in Bologna (presented in (Toth and Vigo, 1997)) and Copenhagen (presented in (Madsen, Ravn, and Rygaard, 1995)), respectively (see section 3.2.3). The model for Bologna assume that all customer requests are static, while the model for Copenhagen allows for immediate customers.

Minimise total service costs

Toth and Vigo (1997) aim at minimising the total costs. Costs are the costs associated with the vehicles. The costs can be split up in fixed cost per vehicle used / route started (Y_j), costs for each kilometre travelled (U_j) and costs for a minute of service (or waiting) (Z_j). This objective then leads to minimising both the number of vehicle used, kilometres driven and time used for waiting and service.

Sub-objectives are included in this objective function in (Toth and Vigo, 1997). To minimise the use of a certain kind of vehicles (taxis), this vehicle

group is given an extra high cost by multiplying all costs associated with this kind of vehicle with a constant. This objective modification is used since this kind of vehicle is considered to lead to less satisfaction to users than other vehicles.

Inconvenience for the customers with regard to differences in actual and desired service time, is given an economical interpretation and included in the objective function. Toth and Vigo (1997) use a linear inconvenience function, where early arrival at destination for outbound requests is less costly, than late arrival. And vice versa for pick up locations for inbound requests:

With the expanded objective function, minimising the total costs also leads to minimising user dissatisfaction with regard to taxi use and inconvenience for users when vehicles are not punctual.

Multiple-criteria objective

Madsen et al. (1995) use a multiple-criteria objective. The objective includes sub-objectives which aim at maximising service (minimising the total waiting time for users and the deviation from actual and desired service time) and sub-objectives which aim at minimising total costs (minimising the total driving, the number of vehicles used and the routing costs). It can be argued, that minimising the total waiting time for users and minimising the total driving time includes both maximising service and minimising total costs, since lower waiting time, besides leading to lower inconvenience for customers in the vehicles, leads to lower costs for having vehicles being idle, and thereby lower total costs. And a lower driving time, besides leading to a lowering of the variable driving costs, leads to less transport time for customers in vehicles.

In the insertion heuristic in (Madsen et al., 1995) the multiple-criteria objective is addressed by the "*load*" or "cost" of inserting a request in a route. The total "load" of an insertion can be split into four different measures of fictitious cost measure. To minimise driving time a measure called "Driving time" is used. The "Driving time" measure secondarily minimise the waiting time, but another measure called "Waiting time" focus solely on this. Two other measures of fictive costs, "Deviation from desired service time" and "Capacity utilisation" focus on keeping the deviation between actual and desired service time low and minimising the number of used vehicles, respectively. It might seem that the total costs have been forgotten, but since the costs are lowered if waiting time, driving time and vehicles used are lowered, the total costs are not forgotten. See also section 4.1.5 for a more comprehensive review of the fictive cost measures of insertion.

Other objectives worth mentioning in the context of the DARP are minimising total route length, minimising operational costs for drivers and vehicles (Borndörfer, Grötschel, Klostermeier, and Küttner, 1997), minimising the size of the fleet and maximising the number of serviced requests. For an overview on objectives and constraints for different DARP models, we refer to (Cordeau and Laporte, 2007).

3.2.3 Different problem formulations

In this section we will describe formal versions of the DARP model. We will not get into detail of how many customers there are estimated to be in different segments, or certain criteria of the fleet regarding the specific number of different vehicle types, since this obvious can be different in different instances of the DARP models presented.

The objectives and constraints presented above are *not necessarily* all included in the formulation of the DARP. And surely, various other constraints and objectives than the ones above, can be found in some formulations of the DARP, but we find that those mentioned comprehensive enough to describe the underlying real-world problems of DARPs and this opinion is supported by the models for the DARP presented here.

The names of the models refer to the locations, where they have been used to describe real-world DARP instances.

Bologna

Toth and Vigo (1997) present the following version of the static DARP.[8] Customer requests are of different types and are allowed to request a desired service time for pick up or delivery. The service at each location must occur according to time windows, time windows only being imposed on locations with desired service time. There is a maximum duration of the individual rides of an request. The fleet is heterogeneous and fixed, although unlimited vehicles of the type *taxi* is assumed.

To sum up, the model can be described as the static DARP, where the group of requests R has to be served by a fleet of vehicles B, and the constraints (iii), (iv), (v), (vi), (vii), (viii) and (ix) from section 3.2.1 are imposed. The constraints (i) and (ii) are not stated in (Toth and Vigo, 1997) (maybe because they not become relevant in the instance solved in the paper).

$$\begin{aligned} r_i =& (q_{i1}, q_{i2}, q_{i3}, h_i, v_i^+, v_i^-, p_i(\omega_i), t_i^+, t_i^-) && \forall\ r_i \in R, \\ b_j =& (Q_{j1}, Q_{j2}, Q_{j3}, H_j, O_j, D_j, \Gamma_{j1}^+, \Gamma_{j2}^+, \Gamma_{j1}^-, \Gamma_{j2}^-, Y_j, U_j, Z_j) && \forall\ b_j \in B. \end{aligned}$$

The objective of the model is to minimise the total costs and user dissatisfaction. The later with regard to taxi use and inconvenience on desired pick up or delivery times. Constraint (ix) is not imposed on taxis.

Copenhagen

In (Madsen et al., 1995) the following instance of a DARP model is presented. There are several customer segments with different types of capacity request. Customers are allowed to either request a desired pick up time or delivery time, but not both. There is no guarantee for service at the desired service time, but service is guaranteed within a time window constructed from the desired

[8] Toth and Vigo (1997) do not use the term *dial-a-ride problem*, but refer to this problem as the *Handicapped persons Transportation Problem*.

service time. The fleet of vehicles is depot-based before the routing starts, and the vehicles has to return to some depot after the routing is done. Between these two events a vehicle is free to be where the dispatcher wants to put it, only obligated to fulfil constraints regarding labour regulation on driver breaks and maximal route duration. Madsen et al. (1995) also assume different vehicle speeds for the vehicles, then using their average speed.

The model can be summed up as an instance of the dynamic DARP, where the group of requests R (some dynamic) has to be served by a heterogeneous fleet of vehicles B, and all the constraints from section 3.2.1 are imposed.

$$\begin{aligned} r_i &= (q_i, v_i^+, v_i^-, p_i(\omega_i), t_i^+, t_i^-) && \forall\ r_i \in R, \\ b_j &= (Q_j, O_j, D_j, Y_j, U_j, Z_j, P_j^+, P_j^-) && \forall\ b_j \in B, \end{aligned}$$

where $q_i = (q_{i1}, q_{i2}, q_{i3}.q_{i4}, q_{i5})$ and $Q_j = (Q_{j1}, Q_{j2}, Q_{j3}, Q_{j4}, Q_{j5})$. The summary of the model omits the different vehicle speeds. The objective of the model is a mix of minimising the total driving time, the number of vehicles used, the total waiting time for customers, deviation from promised service times and the costs of routing (see *multi-criteria* in section 3.2.2). Hence the objective of the model focuses both on cost minimising and service maximising.

Jutland

Unlike the models presented for Bologna and Copenhagen, the next model is not introduced in any article, but like these models, this model is related to real-world dial-a-ride service. The Jutland model is a dynamic DARP constructed with inspiration from tender documents from the Danish public enterprise Midttraffik (Midttrafik, 2010), who is liable for public transportation in the Danish region Midtjylland.

In the Jutland model there are three customer segments with different types of capacity request. Customers are allowed to make a request for desired pick up or delivery time, but not both, as it is the case in the models for Bologna and Copenhagen. The service times associated with pick up and delivery of the request is determined by the capacity request of the request. Time windows on pick up or delivery are a function of the desired service time. The same time window function is used on all requests. Unlike the models for Bologna and Copenhagen, the vehicles have more than one combination for multi capacity. Some vehicles are depot-based, some are non-depot-based. The fleet is heterogeneous.

The model in short is the dynamic DARP instance, where the set of request R (some dynamic) has to be served by a fleet of vehicles B, and all the constraints from section 3.2.1 are imposed.

$$\begin{aligned} r_i &= (q_i, v_i^+, v_i^-, p_i(\omega_i), t_i^+, t_i^-) && \forall\ r_i \in R, \\ b_j &= (Q_j^{\mathrm{C}}, O_j, D_j, Y_j, U_j, Z_j, P_j^+, P_j^-) && \forall\ b_j \in B, \end{aligned}$$

where $q_i = (q_{i1}, q_{i2}, q_{i3})$ and $Q_j^{\mathrm{C}} = (Q_{j1}^{\mathrm{C}}, Q_{j2}^{\mathrm{C}}, Q_{j3}^{\mathrm{C}})$. C being possible combinations of multi capacity. The objective of the model is the same as the objective in the Copenhagen model. In chapter the 6 the model will be presented in more detail.

3.3 Solutions

A solution to the static DARP consists of a plan for all the vehicles (or vehicles used). The plan consists of routes and schedules. The set of routes, and associated schedules, introduces in section 3.1.4, are such a solution. For the static DARP the routes are closed, when the execution of service and driving have begun. A solution for the static DARP leaves the dispatcher with instructions on when and where to send the fleet of vehicles. It seems that this described solution for a static DARP is easy to understand.

A solution for the dynamic DARP should also leave the dispatcher with instructions on how to operate the fleet of vehicles. But dynamic events occurring, do not let a single plan as solution fulfil this goal. Since such a plan will be outdated as soon as a new immediate request arrives. So for the dynamic DARP the routes in *the* plan are open, in the sense that they can be updated. A solution to dynamic DARP consists therefore of an updateable plan of open routes and schedules *and* a policy to update this plan, when a dynamic event occur.

One could say that while methods for solving the static DARP produces solutions to the problem, methods for the dynamic DARP are themselves are part of the solution produced and suggested, since the policy part of the solution is contained in the method.

3.4 Differences between the static and dynamic dial-a-ride problem

One might think that a good solution method for a static DARP always will produce a better initial solution for the dynamic counterpart than a less good solution method for the static DARP. This is not always the case, since one central criteria for a good initial solution for a dynamic DARP should be ignored when producing a solution to a truly static DARP, namely the flexibility in the plan, that makes later decisions easier when dynamic requests occur. This insight is also made by (Borndörfer et al., 1997), even though they are looking for a solution method for the static DARP, recognising that in real life truly static DARPs rarely occur. Even in a service where dynamic requests are not accepted, vehicle breakdowns still occur or customers might cancel a request, for instance.

In chapter 7 forecasts are used to make routes and schedules more flexible in a clever way. It is worth noticing that flexibility in solutions provided by "less good" solution methods can be (and often are) beneficial when new immediate requests occur. The aim in 7 is to benefit from this observation, by structuring the way the solutions becomes flexible.

3.5 Measures to describe a problem instance of the DARP

We will now introduce some measures to evaluate instances of DARP models.

3.5.1 Degree of dynamism

The first measure, or series of measures, degrees of dynamism, only relates to the dynamic DARP, as the name suggests. Actually the measure can be used on any DVRP, and therefore we have to adjust the last measure a little bit, for it to be meaningful in the context of a DARP. It is found in (Larsen, 2000, chapter 4).

Definition 3.4 (Degree of dynamism)
The degree of dynamism, *dod*, of a model instance, is

$$dod = \frac{n_{imm}}{n_{tot}} \quad , \tag{3.13}$$

where n_{tot} is the total number of customer requests, and n_{imm} is the number of immediate requests.

Larsen (2000) extends *dod* to also consider the arrival time of the immediate customer requests.

Definition 3.5 (Effective degree of dynamism)
Assume the relevant time horizon starts at 0 and ends at Ψ. Then the effective degree of dynamism, *edod*, of a model instance, is

$$edod = \sum_{i=1}^{n_{imm}} \frac{\chi_i/\Psi}{n_{tot}} \quad , \tag{3.14}$$

where χ_i is the arrival time of the immediate request from customer i, n_{tot} is the total number of customer requests, and n_{imm} is the number of immediate customer requests.

edod is then extended to also consider the reaction time for the immediate customer requests.

Definition 3.6 (Effective degree of dynamism - time windows)
Assume the relevant time horizon starts at 0 and ends at Ψ. Then the effective degree of dynamism with respect to time windows, $edod_{tw}$, of a model instance, is

$$edod_{tw} = \frac{1}{n_{tot}} \sum_{i=1}^{n_{tot}} (1 - \frac{\ell_i - \chi_i}{\Psi}) \quad , \tag{3.15}$$

where χ_i is the arrival time of the immediate request from customer i, n_{tot} is the total number of customer requests, and n_{imm} is the number of immediate requests. ℓ_i is the end time for the pick up time windows. If only one time window is allowed and a customer request, r_i, is outbound, then $\ell_i = \ell_i^- - \lambda_i$, the latest feasible pick up time for r_i.

The degrees from definition 3.4, 3.5 and 3.6 can obviously first be calculated after the routing has been carried out for an instance of a DARP. This is so because they all involve knowledge about the number of immediate customer requests, which is not a-priori knowledge. However if we have estimates for n_{imm} and χ_i for the associated immediate requests, we can also make estimates for *dod*, *edod* and $edod_{tw}$. This can be beneficial because problem instances with certain values of *dod*, *edod* and $edod_{tw}$ are assumed to fulfil other characteristics as well.

Larsen (2000) suggests that instances of a DARP model with a low *dod* should focus on minimising routing costs. This corresponds with the focus on cost minimisation in the static (*dod*) Bologna model. Instances with high *dod* should focus on minimising response time. For instances with neither low or high *dod*, the focus should be on a mix. The last corresponds with the multiple criteria objective found in the dynamic Copenhagen model.

The dynamic DARP for transporting elderly and physically disabled people is characterised by a relatively low *dod*. On the other hand, a dynamic DARP for "regular" people (tele-bus) is characterised by a relative high *dod* (Larsen, 2000). If a DARP is a mix of these two, then we can not make a clear-cut decision on focusing on costs (minimising routing costs) or focusing on service (minimising response time).

3.5.2 Travel patterns

In (Larsen, 2000, Chapter 2) a division of DARP instance with regard to travel patterns is introduced. The DARP instances are divided into three broad categories:

1. *Many-to-one* (or One-to-many) (MTO).

   A MTO DARP is an instance of the DARP, where many requests share a destination (or pick up location).

2. *Many-to-few* (or Few-to-many) (MTF).

   A MTF DARP is an instance of the DARP, where few requests share a destination (or pick up location).

3. *Many-to-many* (MTM).

   A MTM DARP is an instance of the DARP, where none of (or almost none of) the requests share a destination (or pick up location).

Values for "many" and " few" are not stated or suggested in (Larsen, 2000). An example of MTO is driving patients to and from a hospital.

Following the terms for the DARP model introduced by (Larsen, 2000) and stated above, we introduce a measure for a part of the model or, to be more specific, a measure for a route.

Definition 3.7 (MTO route)
If the stop/customer-ratio in a route ξ_j for a vehicle b_j is

$$\frac{\#\text{locations}}{\#\text{customers}} = \frac{n+1}{n}, \tag{3.16}$$

where n is the number of customers in ξ_j, then ξ_j is said to be a *MTO route*.

Definition 3.8 (MTM route)
If the stop/customer-ratio in a route ξ_j for a vehicle b_j is

$$\frac{\#\text{locations}}{\#\text{customers}} = \frac{2n}{n} = 2, \tag{3.17}$$

where $n > 1$ is the number of customers in ξ_j, then ξ_j is said to be a *MTM route*.

Definition 3.9 (MTF route)
If the stop/customer-ratio in a route ξ_j for a vehicle b_j is

$$\frac{\#\text{locations}}{\#\text{customers}} = \frac{n+\theta}{n}, \tag{3.18}$$

where n is the number of customers in ξ_j and $\theta \in \{2, 3, \ldots, n-1\}$, then ξ_j is said to have a *MTF route* or *MTθ route*.

Definition 3.10 (Hub stop)
If a route is a MTO route, then the stop that all requests originates from or have as destination is referred to the *hub stop* of the route.

4

Algorithms for the dial-a-ride problem

In this chapter we will describe two solution methods for the DARP. One for the static DARP from (Toth and Vigo, 1997), and one for the dynamic DARP from (Madsen et al., 1995).

Even though the nexus where the heuristics used is different, they share similarities in the approach for solving the DARP instances, to which they were applied. Both heuristics use insertion of requests in routes to construct a solution. While the approach from (Madsen et al., 1995) exclusively relies on a good insertion heuristic, the method from (Toth and Vigo, 1997) follows up on the initial insertion method by using intra- and inter-route exchanges in the preliminary solution. At first sight, this might make the method from (Toth and Vigo, 1997) seem superior to the method from (Madsen et al., 1995). But such a comparison would be like comparing oranges and apples, to put it bluntly, since the method from (Madsen et al., 1995) also has to deal with dynamic occurring requests, and therefore faster computation time is much more key. Also, after applying local search to the initial plan, like in (Toth and Vigo, 1997), the routes could end up with undesirably tight schedules. This could make the later immediate requests insertions more difficult, and the final solution worse.

We start this chapter with the similarities in the two methods, hence how to make the insertions. Later we will shortly get back to the rest of the method from (Toth and Vigo, 1997), on inter- and intra-route changes. Shortly because the aim of this thesis is on solutions to the dynamic DARP, and the method in (Toth and Vigo, 1997) focuses on a static DARP.

4.1 Parameters related to vehicles, requests and insertion of requests

A good insertion method starts before even trying to insert a request. This is the case since not all insertions are alike in difficulty. Take an empty route, with a schedule only showing that the vehicle can leave the start depot, at any time and has to be back at the depot in time, such that it does not break the maximal route duration constraint. On the other hand take a route with a very tight schedule (little waiting time) and almost binding in the maximal route duration constraint. Having the job of making a request insertion in one of these two routes seems easier facing the first route than the second. The first route

resembles, what a route would look like, when we just have started inserting requests into routes, the second could be what we are facing when we are about to insert the last requests. Therefore both Toth and Vigo (1997) and Madsen et al. (1995) use a theoretical measure on how difficult it is to insert a requests in a plan. Then they sort them accordingly, and start with inserting the most difficult.

We will now study these measures or difficulty degrees related to inserting a request in a route, not knowing anything about the route the request shall/will be inserted in.

4.1.1 Difficulty degrees for insertion of requests

Difficulty degrees in Toth and Vigo (1997)

Toth and Vigo (1997, page 64) define a *difficulty degree* for the ride related to request r_i. We will try to split up the difficulty degree. The first term is a measure for the difficulty in relation to the customer type represented by the request:

$$M^i_{type} = K_{type}(a_i + y_i + z_i). \tag{4.1}$$

a_i, y_i and z_i are binary variables representing respectively (no extra personnel needed / extra personnel needed), (customer not seated in wheelchair / customer seated in wheelchair) and (walking / wheelchair). Walking customers are easier to accommodate than wheelchair customers (vehicles that can accommodate wheelchairs can also accommodate walking users). Seated wheelchair users are likewise easier to accommodate than non seated wheelchair users, and again a vehicle with extra personnel has no trouble to accommodating customers without need for help, while the opposite is not true. Therefore the most difficult type to handle with respect to customer type in Toth and Vigo (1997)'s model (a wheelchair customer with need for help, who has to be seated in her/his wheelchair) has a $M^i_{type} = K_{type} * 3$. $K_{type} \geqslant 0$ is suitable constant set by the dispatcher, used to weight this sub-difficulty degree in the overall difficulty degree for the request.

Since Toth and Vigo (1997) use binary variables to distinguish different customer groups, we can not directly translate this to the notation used elsewhere in this thesis, where customers are grouped directly by the capacity they occupy (and help they need). Later, see definition 4.3, we will present a similar term, more adopted to our notation.

The next term in Toth and Vigo (1997)'s difficulty degree relates to the difficulty to connect a request r_i "directly" to another request r_j, where directly means that there are no stops between the handling of the two requests. There are three ways to do this:

$$v_i^+ \to v_i^- \to v_j^+ \to v_j^- \ ,$$

$$v_i^+ \to v_j^+ \to v_j^- \to v_i^- \ ,$$

$$v_i^+ \to v_j^+ \to v_i^- \to v_j^- \ .$$

Giving each case equal weight, we get the average travel time between request r_i and request r_j to be

$$\begin{aligned}\Delta_{ij} = \Delta(r_i, r_j) =& \frac{1}{3}(\tau(v_i^+, v_i^-) + \tau(v_i^-, v_j^+) + \tau(v_j^+, v_j^-) \\ &+ \tau(v_i^+, v_j^+) + \tau(v_j^+, v_j^-) + \tau(v_j^-, v_i^-) \\ &+ \tau(v_i^+, v_j^+) + \tau(v_j^+, v_i^-) + \tau(v_i^-, v_j^-)).\end{aligned} \tag{4.2}$$

Normalising it with respect to the maximal travel time between request r_i and all other requests in R gives $\widetilde{\Delta}_{ij} = \Delta_{ij} / \max\{\Delta_{kl} | k, l = 1, 2, \ldots, n\}$, where n is the number of requests in R. We adopt the notation from (Toth and Vigo, 1997) and use a tilde to indicate that a value X_j has been normalised, with respect to the maximal value over all X. Using the normalised average travel time between two locations, we define *connection difficulty* for request r_i to be

$$M_{con}^i = K_{con} \frac{1}{n-1} \sum_{j \neq i} \widetilde{\Delta}_{ij}, \tag{4.3}$$

where $K_{con} \geqslant 0$ is a constant set to a suitable value by the dispatcher.

The difference between the maximal travel time and direct travel time, service time on pick up and delivery and time used to loading and unloading

$$C_i = \lambda_i - \tau(v_i^+, v_i^-) - (t_i^+ + t_i^-) - (\widehat{\Gamma}_i^+ + \widehat{\Gamma}_i^-) \tag{4.4}$$

is used to evaluate how difficult the request is with respect to maximal travel time. Like we did with the average travel time between two requests, we follow Toth and Vigo (1997) and normalise C_i with respect to the largest value over all requests and notating it $\widetilde{C}_i$. $\widehat{\Gamma}_i^+$ is the average vehicle dependent loading time for request r_i and likewise for $\widehat{\Gamma}_i^-$. We use the normalised value for C_i in the *excess travel time difficulty*, the third term in Toth and Vigo (1997)'s difficulty degree:

$$M_{exc}^i = K_{exc} \cdot \widetilde{C}_i. \tag{4.5}$$

M_{exc}^i is descending in difficulty. Later, see definition 4.2, we present a similar measure for difficulty of the excess travel time.

The fourth term in Toth and Vigo (1997) is a *service time difficulty*:

$$M_{ser}^i = K_{ser}(\widetilde{t}_i^+ + \widetilde{t}_i^- + \widetilde{\widehat{\Gamma}}_i), \tag{4.6}$$

where $\widetilde{t}_i^+, \widetilde{t}_i^-$ and $\widetilde{\widehat{\Gamma}}_i$ are t_i^+, t_i^- and the average of $\widehat{\Gamma}_i^+$ and $\widehat{\Gamma}_i^-$ normalised with respect to the largest value of each over all requests, respectively.

Toth and Vigo (1997)'s difficulty degree for inserting a request in a route becomes in our notation:

$$M_{\mathrm{TV}}^i = M_{type}^i + M_{con}^i - M_{exc}^i + M_{ser}^i. \tag{4.7}$$

Toth and Vigo (1997) set $K_{type} = 1000, K_{con} = K_{exc} = 100$ and $K_{ser} = 1$. This tells us that they prioritise to insert requests with difficult request capacity first,

i.e. wheelchair users seated in their wheelchair and with the need for help. They also prioritise to relatively early insert requests, that are characterised by having a high average travel time compared to other requests. Likewise are requests with low excess in the maximal travel time prioritised to be inserted relatively early. The service time needed for a request is also considered, but due to the low weight given by the constant ($K_{ser} = 1$), two requests almost have to be alike in the other sub-difficulty degrees before it becomes relevant, when evaluating the difficulty related to insertion of requests.

Difficulty degrees in Madsen et al. (1995)

Madsen et al. (1995) define three measures or difficulty degrees regarding insertion of customer requests in a route. We will restate the difficulty degrees in the three definitions below.

Definition 4.1 (Time window difficulty degree)
Let r_i be a request for customer i and $[e_i, \ell_i]$ a corresponding time window, assuming only time window on pick up *or* delivery. Then

$$M_{win}^{i} = \begin{cases} K_{win}^{2} \cdot (\ell_i - e_i)^{-1} + K_{win}^{1} & \text{if} \quad \ell_i > e_i, \\ K_{win}^{\max} & \text{if} \quad \ell_i = e_i, \end{cases} \tag{4.8}$$

is the *time window difficulty degree* for r_i. K_{win}^{1}, K_{win}^{2} and $K_{win}^{\max} \geqslant 0$ are constants set by the dispatcher.

Definition 4.2 (Maximal travel time difficulty degree)
Let r_i be a request for customer i and let C_i be defined in accordance to equation (4.4). Then

$$M_{trt}^{i} = \begin{cases} K_{trt}^{2} \cdot C_i^{-1} + K_{trt}^{1} & \text{if} \quad \lambda_i > \tau(v_i^+, v_i^-), \\ K_{trt}^{\max} & \text{if} \quad \lambda_i = \tau(v_i^+, v_i^-), \end{cases} \tag{4.9}$$

is the *maximal travel time difficulty degree* for r_i. K_{trt}^{1}, K_{trt}^{2} and $K_{trt}^{\max} \geqslant 0$ are constants set by the dispatcher.

Definition 4.3 (Capacity difficulty degree)
For a request for customer r_i and let $q_{i1}, q_{i2}, \ldots, q_{i\nu}$ be the capacity request for the ν different customer groups. Then

$$M_{cap}^{i} = \sum_{l=1}^{\nu} K_l \cdot q_{il} \tag{4.10}$$

is the *capacity difficulty degree* for r_i. $K_1, K_2, \ldots, K_\nu$ are constants corresponding to the different customer groups set by the dispatcher.

Remark 4.4 All three difficulty degrees above should get bigger as difficulty grows. M^i_{win} should get bigger when the time window $[e_i, \ell_i]$ gets narrower. This is true for case one ($\ell_i > e_i$) in definition 4.1 for all K^1_{win} and K^2_{win}. In case two ($\ell_i = e_i$), the narrowest time window possible, however we can never set $K^{\max}_{win}$ high enough, for this to be true, if time windows can have an infinitely small length larger than zero. In the context of the problem where this measure is used, we however assume time to be countable, so we can set $K^{\max}_{win}$ suitably large to give the measure M^i_{win} the right meaning. To ensure that M^i_{trt} fulfil its purpose, we make a similar observation to the observation about $K^{\max}_{win}$, namely that $K^{\max}_{trt}$ should be substantially large. The case where $K^{\max}_{win}$ and $K^{\max}_{trt}$ are used, states that the requests demand service at a precise time instant and the requests need to be driven directly between the pick up and delivery location. While such cases may exist in the problem instances studied in (Madsen et al., 1995), we allow ourselves to ignore requests having such characteristics. The added constants K^1_{win} and K^1_{trt} are also ignored, since they only add an equal amount of "difficulty" to all requests. For the last difficulty degree, the one on capacity, it is apparent that it grows in difficulty, and the constants should be bigger for the customer groups which few vehicles can handle than for the groups which many vehicles can handle.

The difficulty degree for insertion of a request in a route, as introduced by Madsen et al. (1995):

$$M^i_{\text{MRR}} = M^i_{win} + M^i_{trt} + M^i_{cap}, \tag{4.11}$$

The difficulty degree M^i_{MRR} and the difficulty degree M^i_{TV} have many similarities. M^i_{cap} and M^1_{type} are essentially the same measure, but different due to different data representation. M^i_{exc} and M^i_{trt} measure the same difficulty, namely the difficulty regarding maximal travel time. The first lowers the overall difficulty measure by becoming large if a lot of excess travel time exists for r_i. The second is very small in the same case and grows if the excess travel time is narrow.

M^i_{ser} ,M_{con} and M^i_{win} have no corresponding measure in the difficulty degrees M^i_{MRR} and M^i_{TV}, respectively. I.e. Toth and Vigo (1997) do not use a measure on how difficult it is to fulfil a request's time window, and Madsen et al. (1995) do not use measures on how difficult it is to service a request in terms of service time and connection to other requests.

Inspired by M^i_{MRR} and M^i_{TV} we will now define a new difficulty degree for insertion of a customer request in a route.

Definition 4.5 (Difficulty degree of a request)
Let r_i be a customer request, and let the number of different customer groups $\nu = 3$. Then the difficulty degree is set to be

$$M^i = M^i_{trt} + M^i_{cap} + M^i_{con}. \tag{4.12}$$

K^1_{trt}, K^2_{trt}, $K^{\max}_{trt}$, K_{con}, K_1, K_2 and K_3 depends on the instance of the DARP. They are set to $K^1_{trt} = 0$, $K^2_{trt} = 10$, $K_{con} = 1$, $K_1 = 100$, $K_2 = 250$ and $K_3 = 300$ in our solution method for the Jutland DARP.

M_i measures how difficult it is to fulfil constraints regarding a request's maximal travel time and capacity request, and how distant the request is from other requests. We will, like Madsen et al. (1995) , not use a measure on the difficulty on service time, since we, in the Jutland model, assume that service times for requests can be related to the capacity request of the requests. Furthermore, we will not use a difficulty measure on time windows, since we assume that they are alike for all requests, only depending on the desired pick up or delivery time.

The difficulty degree is later used to order request in a method for solving a dynamic DARP of the type presented in the Jutland model (see chapter 6).

4.1.2 Availability of vehicles

In (Toth and Vigo, 1997) the available vehicles are ordered in accordance to an *efficiency score*, M^j_{eff}, for the availability of vehicle b_j:

$$\begin{aligned} M^j_{eff} = &\sum_{l=1}^{\nu} K^l_{cap} \cdot \widetilde{Q}_{jl} + K_{help} \cdot \widetilde{H}_j - K_{fc} \cdot \widetilde{Y}_j - K_{vcd} \cdot \widetilde{U}_j \\ &- K_{vcs}\widetilde{Z}_j - \sum_{l=0}^{\nu} K^l_{load}(\widetilde{\Gamma}^+_{jl} + \widetilde{\Gamma}^-_{jl}). \end{aligned} \tag{4.13}$$

The efficiency score is translated to the notation used in this thesis, instead of the customer grouping using binaries as used in (Toth and Vigo, 1997). The order of the vehicles are used to determine which vehicles that are to be used first.

Toth and Vigo (1997), with $\nu = 3$, set $K^l_{cap} = 100$ for $l = 1, 2$, $K^3_{cap} = 1000$, $K_{help} = 100$, $K_{fc} = K_{vcd} = K_{vcs} = 10$ and $K^l_{load} = 1$ for $l = 1, 2, 3$. Where $l = 1, 2, 3$ refers to the customer groups "customers seated", "customers in wheelchairs not seated in their wheelchair" and "customers seated in their own wheelchair", respectively.

The high value for K^3_{cap} might be interpreted as vehicles with high capacity for "customers seated in their own wheelchair" (CSW), are the best candidates to accommodate a random customer request. But it can also be seen in combination with the difficulty degree M^i_{TV}, where the customer group CSW is also given a high priority, i.e. inserted early. Focusing on the last insight, it makes sense to let vehicles with (high) capacity for CSW to be used first, because it is likely these vehicles can accommodate CSW, which are the customer requests, that are most likely to be inserted first using when the constants set by Toth and Vigo (1997) for M^i_{TV}.

The order of the vehicles is only used, when constructing an initial number of routes with one requests in each route. This first request in a route is called the *pivot request* of the route. We will get back to how Toth and Vigo (1997) construct these initial routes of one request, after we show how they determine an estimate on how many vehicles that have to be used to construct the initial routes.

4.1.3 Number of vehicles/routes

Toth and Vigo (1997) determine a *minimum number of vehicles* θ_0 needed to ensure a fraction π of the requests service. After the vehicles have been ordered in accordance to the efficiency score M^j_{eff} for all vehicles $b_j \in B$. Assuming ν different customer groups and n requests, we obtain the minimum number of vehicles required to service a fraction π of the requests, from

$$\theta_0 = \min_{k=1,2,\ldots,m} \{\sum_{i=1}^{n}\sum_{l=1}^{k} Q_{lh} \geqslant \pi \sum_{i=1}^{n} r_i(q_{ih}) \quad \forall\, h = 1,2,\ldots,\nu\}, \tag{4.14}$$

where m is the total number of available vehicles in B.

(Madsen et al., 1995) do not determine a number of vehicles to be used in advance like (Toth and Vigo, 1997). The estimate in (Toth and Vigo, 1997) is on the number of initial routes to be started, if it turns out too be to small, more routes can be started by using more vehicles.

4.1.4 Priority given to pivot candidate requests

The priority given to requests as pivot candidates in (Toth and Vigo, 1997) is denoted L^i_{TV}. For the first vehicle, $\upsilon = 1$, $L^i_{\mathrm{TV}} = K_{dif} \cdot \widetilde{M}^i_{\mathrm{TV}} + K_{att} \cdot \widetilde{A}_i - K_{dep}(\widetilde{\tau}(O_\nu, p_i^+) + \widetilde{\tau}(p_i^-, D_\nu))$. For $\upsilon > 1$, L^i_{TV} is

$$\begin{aligned} L^i_{\mathrm{TV}} = K_{dif} \cdot \widetilde{M}^i_{\mathrm{TV}} + K_{prp}\frac{1}{\upsilon - 1}\sum_{j=1}^{\upsilon-1} \widetilde{\Delta}(r_i, \gamma_j) \\ + K_{att} \cdot \widetilde{A}_i + K_{dep}(\widetilde{\tau}(O_\nu, p_i^+) + \widetilde{\tau}(p_i^-, D_\nu)) \quad , \end{aligned} \tag{4.15}$$

where γ_j is the pivot request assigned to vehicle b_j for $j = 1,\ldots,\nu$. A_i is the *attractiveness* of a request r_i, and is defined as the number of requests, which have their origin or destination within 10 minutes of travel time from the origin or destination of request r_i. In equation (4.15), the different terms relate how difficult it is to insert a request, the average travel time to other pivot requests, the attractiveness of the request and the distance to the depot(s) of the vehicle b_ν, respectively. Unlike Toth and Vigo (1997), Madsen et al. (1995) start the insertion method with empty routes, consequently no initialisation of a certain number of routes is done in the method in (Madsen et al., 1995).

Remark 4.6 If vehicle b_ν is non depot based (i.e. a taxi), then $\tau(O_\nu, p_i^+) = \tau(p_i^-, D_\nu) = 0$. Considering the extra cost added to taxis in the Bologna model, this should not become relevant for the initialisation of the routes in (Toth and Vigo, 1997).

Toth and Vigo (1997) sets $K_{dif} = 1000, K_{prp} = 10$ and $K_{att} = K_{dep} = 1$. This tells us that Toth and Vigo (1997) wants to use the most difficult requests related to insertion as pivots, thus escaping the problem of inserting them in a route later. From the second term we get that pivot candidates close to other pivots in travel time is valued less, with a mark up of 10. Last we see for the last

two terms that, while it still counts in a positive direction for a candidate pivot to have many requests nearby (high $\widetilde{A}_i$) and a short distance to the depot(s) of vehicle b_ν, they are given no mark up ($K_{att} = K_{dep} = 1$).

4.1.5 The cost of inserting a request

As mentioned in section 3.2.2 - *multiple-criteria objective*, (Madsen et al., 1995) use different "*loads*" to evaluate the cost of inserting a request in a route. Thereby also maintaining the multiple-criteria objective of the Copenhagen DARP model. We will in this section start by examining these "*loads*" and thereafter look at the corresponding component in the method from (Toth and Vigo, 1997).

An insertion (of a request) into a route (plan) consists of up to six locations, one or two from the inserted object (depending on if it is a break or request) and up to four already in the route (plan) (depending on the way the new request goes into the existing schedule). We will get back to describing the structure of an insertion, when we look at insertion methods later, see section 4.2. For now we will just denote an insertion of request r_i in plan Ξ_j by $r_{ij}^{\mathrm{IN}} = r^{\mathrm{IN}}(r_i, \Xi_j)$. Notice that there can be several (feasible) insertions, r_{ij}^{IN}, for every set of r_i and Ξ_j, and mostly that is the case.

1. *Driving time*:

   Related to driving costs Madsen et al. (1995) use the measure

   $$N_{drt} = \sum_{\xi_l \in \Xi_j} \sum_{v^\alpha \in \xi_l} (K_{var} \cdot t_\alpha^{drt} + K_{const} \cdot (t_\alpha^{wait} + \phi_l^\alpha)), \tag{4.16}$$

   where t_α^{drt} is the driving time to v_α from the last location in the route, and t_α^{wait} is the waiting time at location v_α. ϕ_α^l is the total service time at location v_α for vehicle b_l (see equation (3.3) - page 23). It is worth noticing that while the change in minimum driving time, service times and unloading and loading times at best are zero, the waiting time might be reduced due to the insertion of the request.

   Remark 4.7 The sum over all locations in the plan, when calculating the effects of an insertion, is probably a bit too much, since the plan very often consists of many routes and the request is only inserted in one. Like we get back to describing the structure of an insertion in section 4.2, we will also look at which locations that are affected by the insertion, or more importantly which locations that are not affected.

2. *Waiting time*:

   The measure for the cost related to waiting time is not unambiguously positive or negative.

   $$N_{wait} = \sum_{\xi_l \in \Xi_j} \sum_{v_\alpha \in \xi_L} (K_{wait}^2 \cdot (t_\alpha^{wait})^2 + K_{wait}^1 \cdot t_\alpha^{wait}). \tag{4.17}$$

Long waiting time at a location is given a relative higher "load" by this measure.

3. *Deviation from desired service time*:

   The measure for total deviation from desired pick up or delivery times in the plan Ξ_j is

$$N_{dev} = \sum_{\xi_l \in \Xi_j} \sum_{v_\alpha \in \xi_l} \sum_{r_k \in R(\xi_l^\alpha)} K_{dev}(p_\alpha^l - r_k(p_k))^2, \tag{4.18}$$

   where p_α^l is the planed service time. If no desired service time is associated with location v_α for request $r_k \in R(\xi_l^\alpha)$ for vehicle b_l, then $r_k(p_k)$ is set to be p_α.

4. *Capacity utilisation*: To evaluate the cost on the fleet size, the unutilised capacity is considered. The following measure is used.

$$N_{cap} = \sum_{\xi_l \in \Xi_j} \sum_{v_\alpha \in \xi_l} \sum_{k=1}^{\nu} \hat{K}_k(Q_{lk} - w_{lk}^\alpha)^2 \quad , \tag{4.19}$$

   where w_{lk}^α is the actual number of requests of group k carried by vehicle b_l after leaving location v_α. The idea is to use as few vehicles as possible (this is one of the multiple-criteria of the objective function (see section 3.2.2 - *Multiple-criteria objective*). This cost will obviously go up if a new vehicle is used, and down if the request is handled in an existing route.

The four measures above are on a whole plan. In order to obtain the change, the value is calculated for the old as well as the new plan resulting from the insertion (or the relevant part of the plan, where the costs are affected by the insertion). As shown here, the measures can be used in the evaluation of solutions produced by a method for appropriate DARP models, like the Copenhagen and Jutland DARP models. All the K's $\geqslant 0$ are set to suitable values by the dispatcher, and may differ for different vehicles.

The sum over the change in all four measures is used to determine where the request is inserted. Madsen et al. (1995) only use the internal constants (constants in the sub-loads) to weight the different measures. And since all constants in a measure just can be raised by a multiple to raise the weight for a measure, the use of external constants (constants in the total load) on a measure can be considered redundant. However, it is our opinion that after determining some internal constants, that can be said to give the change in the measures equal wait, external measures then should be used to weight the change in measures. The idea is to give the dispatcher this job, and with this structure, it is less crucial that the dispatcher know the architecture of the measures, and thereby easier for him/her to use.

Toth and Vigo (1997) use the actual added costs of an insertion in a route to determine the best insertion of a request. While Madsen et al. (1995) start by determining a request for insertion and then determine the cost of all feasible

insertions of the request, Toth and Vigo (1997) combines the phase of selecting requests for insertion and calculating the costs of insertion in a route. In this combined phase the actual added costs for an request insertion are subtracted by a multiple of the degree of difficulty for the requests. In this way the difficult requests are given a fictional low insertion cost, and are therefore handled early in the insertion phase in (Toth and Vigo, 1997). Note that this subtracted value has no influence on the route the request is inserted in!

We adopt the insertion costs introduced by Madsen et al. (1995) in a method presented later to solve instances of the Jutland DARP model.

4.1.6 Setting the constant values in the scoring parameters

The different constants, Ks, are used to weight different objectives. Since measures where the constants appear are used to determine which requests (and vehicles) that are handled first, a higher measure relates to a higher priority. For instance if K_{type}, used in M^i_{type}, is relatively high, then the dispatcher likes to handle requests with difficult capacity types first, thus consequently to ensure service to all customers that can not be handled by the back up vehicles (taxis).

While the Ks in the different difficulty degrees are used to let easier requests be left to be inserted last, the last Ks presented in 4.1.5 are used by the dispatcher in (Madsen et al., 1995) to weight different objectives.

4.2 Insertion methods

We will now look at the insertion methods presented in (Toth and Vigo, 1997) and (Madsen et al., 1995), and present them using the notation of this thesis. We start with the insertion heuristic introduced by Madsen et al. (1995).

A request r_i is inserted in the existing plan Ξ in the following way.

Step 0. Take a route ξ_j from Ξ[1] and then make a new route (possibly infeasible) by inserting the locations of r_i, v_i^+ and v_i^-, in ξ_j. v_i^+ is inserted at the location after the start depot, O_j, of b_j associated with ξ_j, and then v_i^- is inserted at the locations after v_i^+. Go to *step* 1.

Step 1. The new route is checked for feasibility. If it is found feasible the extra costs (in terms of N_{drt}, N_{wait}, N_{dev} and N_{cap}) resulting from the new route is calculated. Go to *step* 2.

Step 2. Now v_i^- is moved one location up in the route, resulting in a new route (possibly infeasible) and *step* 1 is repeated. This continues until v_i^- is just before D_j, the end depot of b_j, i.e. until v_i^- can not be moved up longer in the route. Go to *step* 3.

Step 3. Now v_i^+ is moved one location up in the route and v_i^- is is placed just after v_i^+. Then *step* 1 is repeated. This continues until v_i^+ is just two locations before D_j, only with v_i^- between v_i^+ and D_j. Go to *step* 0.
Note, that at all time the existing plan consists of both routes not yet started

[1] Ξ also contains empty routes, this is routes assigned a vehicle, but not assigned any requests.

and routes the already have been finished, as well as routes that currently are being carried out by a vehicle. If no feasible insertion for r_i is found by this procedure, r_i is rejected service. The insertion procedure is carried out for all static and dynamic requests.

To save time during the insertion procedure, some possible insertions are not considered by moving v_i^+ and v_i^- "fast forward". This is the case when, in *step* 1, a feasibility check returns a constraint violation, and this constraint violation tells us that later possible routes also will be infeasible. Madsen et al. (1995) state three properties that will give this kind of information, and allow us to skip some possible insertions. We will now look at these properties.

Proposition 4.8 (Capacity property) *Let b_j be a vehicle with an associated route ξ_j and let r_i be a request with associated capacity vector q_i and pick up and delivery locations v_i^+ and v_i^- respectively. If there exists a stop $v_\alpha \in \xi_j$ for which $q_{il} > Q_{jl} - w_{jl}^\alpha$, where w_{jl}^α is the actual number of requests of group l carried by vehicle b_j after leaving location v_α.. Then r_i can only be inserted into ξ_j if both v_i^+ and v_i^- is either before or after v_α.*

Proposition 4.9 (Maximum ride time property) *Let r_i be a request. If r_i is inserted in a route ξ_j for vehicle b_j, in such a way that the maximal ride time is exceeded,*

$$p_i^{j+} - p_i^{j-} > \lambda_i \quad , \tag{4.20}$$

then this insertion is infeasible. p_i^{j+} and p_i^{j-} are the scheduled start of service by vehicle b_j at the pick up location and destination of request r_i, respectively. Placing v_i^+ earlier in the route with v_i^- in same position is infeasible as well and likewise is placing v_i^- later in the route with v_i^+ in same position also infeasible.

Proposition 4.10 (Time windows property) *Let r_i be a request, and ξ_j a route associated with a vehicle b_j. Let v_i be either the pick up location of r_i if r_i is inbound, and the destination if r_i is outbound, and $[e_i, \ell_i]$ the time window associated with v_i. If the upper limit on the time window ℓ_i is violated*

$$p_i^j + t_i > \ell_i \quad , \tag{4.21}$$

where t_i is the service time at location v_i if r_i is outbound, and zero if r_i is inbound and p_i^j is the scheduled start of service by vehicle b_j at location v_i. Then moving v_i up, one or more locations, in ξ_j would also lead to an infeasible route.

From proposition 4.8, 4.9 and 4.10 we can make a new *step* 2:
Step 2*:
If some capacity constraint is violated for some location in ξ_j, then move v_i^+ one location up in the route and place v_i^- just after v_i^+, go to *step* 1. Moving v_i^- up in the route would not change the requests carried at the location where the capacity constraint was violated.
If the maximum travel time constraint for the new request r_i has been violated, then move v_i^+ one location up in the route and place v_i^- just after v_i^+, go to *step* 1. Moving v_i^- up in the route would increase the maximal travel time violation.

If the time window for v_i^- (if any) has been violated, then move v_i^+ one location up in the current route and place v_i^- just after v_i^-, go to *step* 1. Moving v_i^- up in the route would only make the violation of the time window bigger.
If the time window for v_i^+ (if any) has been violated, then go to *step* 0 and consider a new route from Ξ. Moving v_i^- will not change that the time window is violated, and moving v_i^+ up in the route would only make the violation of the time windows bigger.
Else, move v_i^- one location up in the route and *step* 1 is repeated. This continues until v_i^- is just before D_j, the end depot of b_j, i.e. until v_i^- can not be moved up longer in the route. Go to *step* 3.

Toth and Vigo (1997) uses a parallel insertion technique, where multiple customers are inserted in a single step. They construct an *insertion cost matrix* and solve an associated min-cost Rectangular Assignment Problem, in order to determine a set of insertions of requests in current routes. The insertion cost matrix is a matrix, where each row corresponds to an active route in the current plan (for the first insertion cost matrix this is θ_0 rows). Each entry of the rows is the added cost of inserting an unrouted request in the corresponding route in the best feasible way, i.e each column corresponds to an unrouted request. Entries in the insertion cost matrix corresponding to infeasible request insertion in a route are ∞.

Every time a set of insertions has been performed, the insertion cost matrix is recalculated. Only rows corresponding to an insertion need to be updated, and columns corresponding to an insertion that were performed are erased. If a new vehicle are added, a new row is added to the insertion cost matrix, corresponding to the new route. New routes are only added if no feasible insertion exists for an request, i.e. all entries in the corresponding column is ∞.

Inserting multiple requests at the same time in the same route, without updating the costs of the insertions, might lead to the result that the added cost calculated for the individual insertion does not reflect the actual cost of the insertion. Even worse, it could lead to infeasible solutions. It is not stated in (Toth and Vigo, 1997) that the best insertions of each step need to be in different routes, but the following sentence from (Toth and Vigo, 1997, page 65) suggests that it is the case:[2]

> "At each iteration of the insertion phase the $\bar{r}$, with $1 \leqslant \bar{r} \leqslant r$, best assignments are performed. Note that at each new iteration the insertion cost matrix need not be entirely recomputed, and the corresponding Rectangular Assignment Problem need not be solved from scratch. Indeed, only the $\bar{r}$ routes where a trip has been inserted ..."

For every entry in the insertion cost matrix it is needed to calculate all possible insertions, to determine the best feasible insertion in terms of costs. The

[2] r is the number of current routes.

properties to reduce the number of possible insertions needed to be checked, introduced in (Madsen et al., 1995), could beneficially be used in (Toth and Vigo, 1997) as well.

4.3 An algorithm for the static dial-a-ride problem

In this section we will shortly summarise the algorithm from (Toth and Vigo, 1997), in the following pseudo code.

1: **procedure** TV
2: **for all** vehicles $b_j \in B$ **do**
3: Calculate the *efficiency score* (M^j_{eff}) for b_j.
4: **end for**
5: Reorder the vehicles in B in accordance to the efficiency scores, with the highest first.
6: Determine the minimum number of routes θ_0 to serve a fraction π of the requests. (see section 4.1.3).
7: **for all** requests $r_i \in R$ **do**
8: Calculate the *difficulty degree* (M^i_{TV}) for r_i.
9: **end for**
10: Set $v = 0$.
11: Set $\widehat{R} := R$.
12: **for all** vehicles b_j, $j = 1, \ldots, \theta_0$ **do**
13: **for all** requests $r_i \in \widehat{R}$ **do**
14: Calculate the priority of pivot candidates (L^i_{TV}) for r_i.
15: **end for**
16: Let the request r_k be the request with the highest L^k_{TV}, in the subset of requests in $\widehat{R}$, that feasibly can be serviced by vehicle b_j.
17: Make r_k the pivot request of route ξ_j associated with vehicle b_j.
18: Remove r_k from $\widehat{R}$.
19: **end for**
20: Determine the number of unassigned requests ($\bar{n} := |\widehat{R}|$).
21: Set $\theta := \theta_0$ (the number of currently active routes).
22: Construct the *insertion cost matrix* ($r \times \bar{n}$).
23: **while** $\bar{n} > 0$ **do**
24: Solve the corresponding Rectangular Assignment Problem.
25: Insert the $\bar{\theta}$ requests with the lowest extra cost. ($\bar{\theta} \in (1, \theta)$).
26: **if** there exists a feasible insertion in a existing route **then**
27: the *insertion cost matrix* is updated for relevant rows. I.e. rows corresponding to routes where a request is inserted.
28: **else if** a request can be handled by a new vehicle **then**
29: A new route is started.
30: The best vehicle, in terms of efficiency score and average distance from the vehicle depot to the request, is chosen to start the route.
31: The *insertion cost matrix* is extended with the new route.
32: $\theta := \theta + 1$.

```
33:         else
34:             Raise the difficulty degree of the request.
35:         end if
36:     end while
37: end procedure
```

The above code produces an initial solution to the Bologna DARP. This initial solution is then afterwards improved by a series of inter- and intrachanges of requests in and between routes. For each request r_i, Toth and Vigo (1997) define a neighbourhood of possible inter- and intrachanges of requests in and between routes. One part of the neighbourhood of r_i, is where a r_i is removed from its current route ξ_j and inserted in the best possible way in set of all routes, i.e. the plan. Another part of the neighbourhood, is where r_i is exchanged with another request from another route, the other request is then inserted in the best possible way in the former route of r_i, ξ_j, and r_i is inserted in the best possible way in the former route of the other request. The last part of r_i's neighbourhood is where r_i again is removed from ξ_j and then inserted in the best possible way in the set of all routes, and another request is removed from another route and is inserted in the best possible way in ξ_j. These neighbourhoods are alternately checked for local optimums . If the search for local optima over a certain period of time does not return any improved solution, then steps away from the current best solution, where the new solution is worse are allowed to get away from the local optimum found. These steps are allowed for a certain time period, and then the method returns to look for local optima.

4.4 An algorithm for the dynamic dial-a-ride problem

The insertion heuristic from (Madsen et al., 1995) is summarised in the below pseudo-code. It is again emphasised that the plan doing the execution contain all routes already finished, not started yet and currently being driven by some vehicle.

```
1: procedure REBUS
2:     while unassigned requests exists do
3:         ▷ Unasigned requests arrive dynamically during the day of execution.
4:         Sort the unassigned requests in accordance to difficulty of insertion
   (M_MRR).
5:         Select the request r_i with the highest M^i_MRR.
6:         for all routes ξ_j do
7:             ▷ Some of the routes might only consist of depot locations. I.e.
   they have not been assigned any requests yet.
8:             Generate all feasible insertions of the request in the current plan.
   ▷ See section 4.2.
9:             Calculate the "load" of every feasible insertion.
10:        end for
11:        if there exists a feasible insertion of the request then choose the
```

insertion with the minimum change in the objective function (minimum extra "*load*").

12: **else**
13: Return the request to a list of requests that cannot be serviced.
14: **end if**
15: Remove the request from the list of unassigned requests.
16: **end while**
17: **end procedure**

We will use the insertion method, presented in (Madsen et al., 1995) and shown in the above code, as part of an solution method for the Jutland DARP (see chapter 7). Though we will omit line 13, since we assume a unlimited number of vehicles is available for the dispatcher to rent and customers are guaranteed service.

5

Forecasting and simulation

Forecasting and simulation are two entirely different things, but nonetheless this chapter is devoted to both. We will in this chapter present two methods, one for forecasting travel patterns from historical or estimated data, and one for simulating a problem instance of the DARP. Since we have no historical data, we have implemented these two methods in a single program both making a forecast on the travel patterns of immediate requests and capable of providing us with instances of a DARP.

We start the chapter with describing the data needed to make the forecast and simulation. Afterwards we will describe the method for making forecasts on travel patterns for immediate requests, and finally we will present the method for simulating DARP instances.

5.1 Input

The dispatcher is asked to construct and estimate the following:[1]

- ***Periods***: Divide the day of execution into periods, covering the hole day of execution, and make an estimate on the expected number of immediate requests and the distribution of outbound contra inbound requests for each period. The method suggested here requires that the expected number of outbound and inbound requests are equal over all periods. This requirement seems apprehensible, since it seems fair to assume that a customer going out also will have a need for going home.

- ***Regions***: Make regions over the area of the routing. For each region it should be stated how populated it is, and relevant *attractiveness centres* are stated as well.

  Attractiveness centres could for instance be a hospital, relevant if (part of) the customers are ambulant patients. Or, for instance, a public transportation hub, like a train station, relevant for commuters.

  In this method, we require that regions are quadratic and alike in size. This requirement is made to make use of neighbour regions simple. Regions are considered to be neighbours if they share a border or corner.

[1] In this thesis all references to "files provided by the dispatcher" are examples, made up by the author of the thesis!

- ***Customer segments***: The customer segments are corresponding to the types of requests assumed in the DARP models. For each customer segment, the dispatcher is asked to make an estimate for the customer segments valuation of the attractiveness centres as travel locations. If the system offers routing for both ambulant patients and commuters, the patients are assumed to value hospitals higher as travel locations than the commuters, and opposite the commuters are assumed to value transportation hubs higher than the patients.

An example on a data file for the forecast method is shown in figure 5.1. Line 1 contains seven integers in this order: the first two stating the number of regions in the north/south axis and the west/east axis respectively, the third stating the number of customer segments, the fourth stating the number of different attractiveness centres, the fifth stating the number of time periods, the sixth stating the number of expected immediate customer requests and the seventh and last stating the side lengths of a region.

```
1  3 4 3 3 4 184 11
2  25.0 20.0 10.0 0.6
3  10.0 5.0 1.0 0.27
4  10.0 5.0 1.0 0.13
5  36669 1 0 1
6  0 0 0 0
7  1285 0 0 0
8  60227 1 0 1
9  1594 0 0 1
10 8257 0 0 1
11 4855 0 0 0
12 7945 0 0 1
13 5003 0 1 0
14 1708 0 0 0
15 9736 0 1 0
16 7180 0 0 1
17 0 240 0.95 0.05 0.39
18 240 480 0.70 0.30 0.09
19 480 720 0.10 0.90 0.29
20 720 1020 0.25 0.75 0.23
```

Figure 5.1: An example of a data file provided by the dispatcher.

Corresponding to the number of customer segments, line 2 - 4 contain information about the customer segments. Each line contains four decimal numbers in this order: three numbers reflecting the customers' valuation of the attractiveness centres and the last is an estimate on the probability that a new arriving customer request belongs to this segment.

Line 5 - 16 contain information about the regions. Each line contains the population of the region and three 0-1 values corresponding to if a attractiveness centre is present in the region. For instance if entry 2 in the line is 1, then a attractiveness centre of the first type is located in the region.

Line 17 - 20 contain data about the time periods. The first two entries in a line state when the time period starts and ends, the next two entries contain the

distribution of outbound and inbound requests in the period, and the last entry is the probability that an arriving request will fall in the time period.

5.2 Method for forecasting

We will in this section construct a method to help the dispatcher with estimating how immediate requests will occur. The method relies on the dispatcher providing some insights and estimates himself, described above, but the idea is to produce a better forecast of the general travel pattern of the immediate customers in the DARP model by combining simple estimates from historical data. Only some of the input presented in section 5.1 is used. The data needed, is data about the number of regions, customer segments and attractiveness centres, and in addition data regarding population in the regions, locations of attractiveness centres and customer segments valuation of the different attractiveness centres.

The procedure for making a forecast on travel patterns for immediate customers, is described in the procedure TRAVELPATTERN on page 54. In TRAVELPATTERN *pop* is the total population and pop_k is the population of region k. TRAVELPATTERN returns P_{trp}, an array of probabilities for travel patterns for all customer segments.

Assume an immediate request occurs at some time and that we know if the request is out- or inbound. Then the output from TRAVLEPATTERN essentially gives us probabilities on the regions, where the two locations of the request are placed and what customer segment the request belongs to. One region referred to as the "*arrival region*" of the request and the other region simply referred to as the "*second region*". Determining if the "arrival region" or the "second region" is the pick up region depends on whether the request in question is out- or inbound.

The probability for a request occurring in a "arrival region" (region j) is solely estimated to be the relative population. Assume now that a request has arrived at region j. Then the probability that the request will travel to or from region j to a certain "second region"(region k) is, beside the relative population, adjusted up by mark up values. If region k is the same region as region j (M_{I}), a neighbour region (M_{N}), contains one or more attractiveness centres (the $M_{cs,ac}$'s) or fulfils a population criteria for being an urban region (M_{U}). The different criteria for manipulating the probability are given different mark up values, indicated in the braces above.

The urban population criteria for a "second region" k is $pop_k \geqslant K_{\mathrm{U}}$, where K_{U} is the threshold line for determining what is considered an urban region. This mark up gives a large population region higher probability for becoming "second region", than just the relative population size does. The idea behind this mark up is that if a lot of people live in a small area (an urban area), then it is more likely that some things like jobs, libraries, public swimming pools, professional sporting events, etc. are in this area. That are things people will travel to and from.

The mark up values for the attractiveness centres for the different customer

```
1: procedure travelPattern
2:     for all customer segments cs do
3:         for all regions j do
4:             P_0 := 0
5:             for all regions k do
6:                 if j = k then
7:                     P_trp[j][k][cs] := M_I * pop_k / pop          ▷ Identical region mark up.
8:                 else if j and k are neighbour regions then
9:                     P_trp[j][k][cs] := M_N * pop_k / pop          ▷ Neighbour region mark up.
10:                else
11:                    P_trp[j][k][cs] := pop_k / pop
12:                end if
13:                for all customer segments ĉs do
14:                    if ĉs = cs then
15:                        for all attractiveness centres ac do
16:                            if a ac is located in region k then
17:                                P_trp[j][k][cs] = M_{cs,ac} · P_trp[j][k][cs]
18:                                    ▷ Attractiveness centre mark up for customer
19:                                      segments.
20:                            end if
21:                        end for
22:                    end if
23:                end for
24:                if pop_k ⩾ K_U then
25:                    P_trp[j][k][cs] := M_U · P_trp[j][k][cs] ▷ Urban region mark up.
26:                end if
27:                P_0 := P_0 + P_trp[j][k][cs]
28:            end for
29:            for all regions k do
30:                P_trp[j][k][cs] = pop_j / pop * P_trp[j][k][cs] / P_0
31:            end for
32:        end for
33:    end for
34:    return P_trp
35: end procedure
```

segments, $M_{cs,ac}$, reflect the customer segments' valuation of different attractiveness centres provided by the dispatcher. As already mentioned, we have made the raw data file in figure 5.1 ourself, and we have some idea about which values for the mark ups that make sense. If the method should be presented to a real-life dispatcher, a transformation of the data provided might be necessary.

The "second region"

The mark-ups are only used to determine the "second region". Assume that in a time period the probability that a request will be an outbound request is high, i.e. the arriving request's location is likely to be a pick up location. Periods with many outbound requests are typically people being picked up at home, the mark-up then increases the value on regions for the second location (the destination) to be a region attractive for them. The "second region" is the location where the request, if materialised, will have a time window.

Assume on the other hand, that a period is characterised by a high probability for inbound requests, i.e. an arriving requests location is likely to be a destination. Periods with many inbound requests are characterised by people going home, typically from areas which are attractive for them, which are made more likely in the forecast by the mark-ups on the second location (the pick up location).

Remark 5.1 Notice that P_{trp} does not tell anything about the direction of the request. Only combining it with the dispatchers estimates on the distribution of outbound and inbound requests gives an estimate on travel direction. P_{trp} does neither tell anything about what time period a request will occur, since P_{trp} is independent of time nor does it tell anything about what customer segment the request will represent, this is also given by the dispatcher.

5.3 Method to provide data for testing

In the following code we present a simulation method for creating data of a (dynamic) DARP problem. In SIMULATION, ec is the estimated number of requests, $prTp[tp]$ is the distribution of request in terms of time periods and the entries of aR is set to the arrival rates for each time period. SIMULATION places attractiveness centres by random inside a region, if they are to be found inside the given region. A request in the simulation is given two time stamps, one for the arrival time, which states when the request becomes known to the dispatcher, and one for the reaction time, which states how long time after the arrival time the request desire service at pick up or delivery.

1: **procedure** SIMULATION
2: **for all** time periods tp **do**
3: $aR[tp] = ec \cdot prTp[tp]/$(the length of tp) ▷ Determining the arrival rates for each time period.
4: **end for**
5: Generate the arrival of the first request according to a Poisson process with arrival rate corresponding to the first time period.
6: Set t to the arrival of the first request.
7: Set r to be the request (with arrival time t). ▷ r is an empty request at this point!
8: **for all** time periods tp **do**
9: **while** t smaller than the end time of tp **do**
10: Determine a reaction time for the request r.
11: The reaction time is 2 hours + a random value in minutes (following a Poison distribution with rate 10).
12: Update r with the reaction time.
13: Determine by random a capacity request type of r, using the probabilities passed from the dispatcher on customer segments.
14: Update r with the capacity request type.
15: Determine by random if r is out- or inbound, u sing the probabilities passed from the dispatcher on probabilities for out- or inbound in tp.
16: Update r with the out-/inbound value.
17: Determine by random two regions for the "arrival location" and "second location", using a travel pattern forecast.
18: Let these two regions be denoted "arrival region" and "second region" respectively.
19: Determine an "arrival location" for r.
20: Randomly place the "arrivallocation" inside "arrival region".
21: Update r with the "arrival location".
22: Determine a "second location" for r.
23: **for all** attractiveness centres ac **do**
24: $h = 0$;
25: **if** ac is in "second region" **then**
26: $h = h + M_{cs,ac}$.
27: **end if**
28: **end for**
29: Set the probability for the "second location" to be placed *randomly* inside "second region" to h.
30: **for all** attractiveness centres ac in "second region" **do**
31: Set the probability for the "second location" to be placed at ac to be $M_{cs,ac}$.
32: **end for**
33: Randomly place the "second location" inside the "second region" according to the above distribution.
34: Update r with the "second location".

35: Add r to Rs.
36: Let the nest request be a new request arriving according to a Poisson process with arrival rate aR[tp].
37: Set t to the arrival of the next request.
38: Set r to be the request (with arrival time t). ▷ r is an empty request at this point!
39: **end while**
40: **end for**
41: **return** Rs.
42: **end procedure**

Implementation of the methods

In the appendix the code for an implementation of the methods TRAVELPATTERNS and SIMULATION are described. The implementation itself is available by reference to the thesis appurtenant CD-ROM.

6

The Jutland model

In section 3.2.3 we briefly introduced a model called the Jutland model. We will in this chapter describe this model in detail, and analogously describe an instance of the model used for testing later. Due to lack of real-life instance data for the DARP or other suitable benchmark data, we simulated the instance of the Jutland model using the simulation method introduced in chapter 5.

The Jutland model is constructed as a mix of the model for *Copenhagen* from (Madsen et al., 1995) (see section 3.2.3) and another real-life instance of the DARP found in (Midttrafik, 2010). While the Copenhagen model is an analytical model, the instance of the DARP found in (Midttrafik, 2010) is a real-life problem. This forces us to translate the information found in (Midttrafik, 2010) to an analytical context. Sometimes this translation is pretty straightforward, but sometimes we need to make a good guess about the information to translate it to the new analytical *Jutland model*. For instance, it is not stated how many requests, that are expected to be immediate. However other information about the expected requests are available and from this information we estimate an excepted *dod* for the instance of the model used for testing.

We will start this chapter by describing the area of routing in the Jutland model in section 6.1. In section 6.2 we describe the characteristics of the fleet and in section 6.3 the requests, both the advance and immediate. Finally, in section 6.4, we describe the constraints and the objective of the Jutland model.

6.1 Area of routing

We assume that the routing of the dynamic DARP takes place in a setting with Euclidean distances, the distances between two locations is then easy calculated. For simplicity, we assume the travel speed to be the same for all vehicles and equivalent to the distance. I.e., a vehicle travel a unit of distance in one unit of time. The unit of time is set to minutes and the unit for distance is kilometres, the travel speed is then 60 km/h.

Unlike the models for *Bologna* and *Copenhagen* presented in (Toth and Vigo, 1997) and (Madsen et al., 1995), where the area of routing is assumed to be in an urban area,[1] the Jutland model assumes, in relation to the problem described in the tender papers (Midttrafik, 2010), that routing is a mixture of transportation of elderly, patients, disabled people and also a public transportation in low density areas. The total population in the area of the instance of the Jutland model is about 150,000 and the density is below 100 per km^2.

Under the assumption that the routing is done in a low density area, with cities of different sizes scattered around the area, it is unlikely that the locations of the requests will be equally distributed over the routing area. The simulated requests used in the test instance of the Jutland model takes this into account in the way customer requests are simulated (see chapter 5).

We will now state the relevant information to make the test data based upon a geographical region in Midtjylland, Denmark. For a general description of the relevant information, see chapter 5. Figure 6.1 shows information used, among other things, to generate data for a set of advanced customers.[2] Besides information used to simulate the customers, we can also extract information about the area of routing from this file. The first line in the file states we have 12 regions. Every region is of the size 121 km^2, which make the total region 1452 km^2. The population of each region is also stated in the data file, and the total population is 144,459.

6.2 The vehicle fleet

As already mentioned, the Jutland model is inspired by the real-life instance of the DARP from (Midttrafik, 2010). In (Midttrafik, 2010) the fleet size is divided into something called *Guarantee Vehicles* and something called *Open Vehicles*. The Guarantee Vehicles are rented for a year by Midttrafik. The hauler is paid a fixed cost for the hours where the vehicles is guaranteed disposable to the dispatcher, i.e., it does not depend on whether or not the vehicles are used by the dispatcher. The contracts for Guarantee Vehicles are negotiated in a public tender, where location, vehicle characteristics and different prices regarding the driving determines which haulers win the contracts. The Open Vehicles are in theory all vehicles that bid on the contracts for Guarantee Vehicles, but did not get a contract. The vehicles are offered to become Open Vehicles, which means that they are put in a database for vehicles that are *offered* driving at the costs put in the tender bid from the corresponding hauler. The costs are split in three parts, a fixed cost for starting a route, a cost for driving a unit of time and cost for waiting a unit of time. In (Midttrafik, 2010) a certain distance is assumed to be driven for free, depending on the fixed cost for starting a route. We ignore

[1] The population of the urban area Bologna was 376,792 (2009) with a density of 2,678 per km^2, source `http://en.wikipedia.org/wiki/Bologna`. The population of the urban area Copenhagen is 1,181,239 (2010) with a density of 6,016 per km^2, source: `http://en.wikipedia.org/wiki/Copenhagen`.

[2] The data file is made from data found for an area in Midtjylland, Denmark, containing the cities Randers, Viborg, Bjerringbro, Hammel, Hadsten and Hinnerup among others.

```
3 4 3 3 4 46 11
25.0 20.0 10.0 0.85
10.0 5.0 1.0 0.12
10.0 5.0 1.0 0.03
36669 1 0 1
0 0 0 0
1285 0 0 0
60227 1 0 1
1594 0 0 1
8257 0 0 1
4855 0 0 0
7945 0 0 1
5003 0 1 0
1708 0 0 0
9736 0 1 0
7180 0 0 1
0 240 0.95 0.05 0.4
240 480 0.70 0.30 0.1
480 720 0.10 0.90 0.3
720 960 0.25 0.75 0.2
```

Figure 6.1: Date file containing information about the area of routing, including information about population in the regions, number of regions and; the time periods, including; customer segments, including; and attractiveness centres used in the instance of Jutland modelused for testing.

this in our model. The hauler is free to accept or reject the offered driving, but we assume a hauler always accept. The idea behind this assumption is that the number of vehicles in the database is sufficiently large, and it is therefore almost always possible to find an Open Vehicle ready to accept the offered driving.

In (Midttrafik, 2010) it is stated that the Guarantee Vehicles are expected to handle 20 % of the driving assignments, and the total number of contracts for Guarantee Vehicles is 33. Since the Jutland model instance we simulate covers about 7 % of the population in the real-life problem in (Midttrafik, 2010), we set the number of Guarantee Vehicles to four in this instance, these vehicles are depot-based at a central depot. The number of Open Vehicles are set to 279, reflecting vehicles available in the area. The large fleet size guarantee that all customers receive service. The Open Vehicles are assumed to be non-depot-based, since they are assumed to handle other driving (regular taxi driving) when not used by the dispatcher in the model. The Open Vehicles are given a "home region" where they are located when not driving for the dispatcher.

The fleet of vehicles in the Jutland model is heterogeneous, and can accommodate requests of the three different types. Not all vehicles can accommodate all types of requests. The following types of vehicles are assumed available:

Type 2 vehicle (Estate car taxi)

Can handle up to four seated passengers at a time. Two of the seated passengers can be wheelchair users able to sit outside their wheelchair, which has to be a folding wheelchair.

Type 5 vehicle (Handicap bus)

Can handle six seated and two big wheelchair users at the same time. A big wheelchair is a wheelchair that needs a lift, and the customer has to be seated in his/hers own wheelchair. Other combinations of customers are assumed. We assume that a wheelchair user seated outside his/hers own wheelchair has an extra capacity requirement of one seat for the folding wheelchair, and the type 5 vehicle can carry a most two folding wheelchairs. We then get the following extra combinations of customers, that vehicle type 5 can carry at a time: four seated users, one wheelchair user seated outside his/hers wheelchair and two big wheelchair users; and two seated users, two wheelchair user seated outside his/hers wheelchair and two big wheelchair users. These extra combinations are not mentioned in (Midttrafik, 2010).

Type 2 and *type 5* are two vehicle types found in (Midttrafik, 2010). There are other types of vehicles in (Midttrafik, 2010), but we ignore them here in order to keep the model simple in this aspect.

To guarantee feasibility even if a very large instance of immediate customers are simulated, a fictitious vehicle is added to the model. This vehicle are given very high costs. The result of the very high costs, is that the dispatcher only will use this groups of vehicles, if no other are available. If it happen, that the dispatcher use vehicles from this groups, then the current fleet of vehicles is to small to handle the set of requests. Figure 6.2 shows a data file containing information about the fleet in the test instance of the Jutland model.

```
1  4 0 0 3 1 0 2 2 0 20.0 15.0 20.0 15.0 60 660 0 0 0 2
2  6 0 2 4 1 2 2 2 2 20.0 15.0 20.0 15.0 60 660 0 0 0 2
3  4 0 0 3 1 0 2 2 0 −1.0 −1.0 0 3 0 1080 60 280 280 17
4  4 0 0 3 1 0 2 2 0 −1.0 −1.0 0 3 0 1080 0 290 290 21
5  4 0 0 3 1 0 2 2 0 −1.0 −1.0 0 3 0 1080 60 260 260 18
6  4 0 0 3 1 0 2 2 0 −1.0 −1.0 2 0 0 720 0 310 300 21
7  4 0 0 3 1 0 2 2 0 −1.0 −1.0 2 0 0 1080 0 290 290 12
8  6 0 2 4 1 2 2 2 2 −1.0 −1.0 0 0 0 1080 0 410 370 44
9  6 0 2 4 1 2 2 2 2 −1.0 −1.0 0 0 0 1080 0 310 310 31
10 6 0 2 4 1 2 2 2 2 −1.0 −1.0 0 3 0 1080 60 400 340 61
11 6 0 2 4 1 2 2 2 2 −1.0 −1.0 1 1 0 720 0 310 310 25
12 6 0 2 4 1 2 2 2 2 −1.0 −1.0 1 1 0 720 0 325 325 16
13 6 0 2 4 1 2 2 2 2 −1.0 −1.0 2 2 0 1080 0 320 320 23
14 6 0 2 4 1 2 2 2 2 −1.0 −1.0 2 3 0 1080 99999 99999 99999 999
```

Figure 6.2: Date file of the fleet used in the instance of Jutland model used for testing.

A line in figure 6.2 represents a set of vehicles (or a vehicle). The first 3 entries are a combination of multiple-capacity, likewise are the next three and the next three. For the first two lines, the four next entries show the locations of the start and end depot. In this case they a the same. For all the non-depot-based sets of vehicles, the 10^{th} and 11^{th} entries in a line is -1.0 (to indicate it is non-depot). The 12^{th} and 13^{th} then states the "home region" of the vehicle.

The 14^{th} and 15^{th} entries of a line states when the vehicle is available to the

dispatcher. Entries 16, 17 and 18 states the cost for starting a route, driving a time unit and waiting a time unit, respectively. And finally it is stated in the 19^{th} and last entry how many vehicle there are in this set of vehicles.

The first two lines are the Guarantee Vehicles, line 3 to 14 is the Open Vehicles, and the last line is the fictitious vehicles of type 5.

6.3 Requests

The requests in (Midttrafik, 2010) comprise a mix of many groups of the population. The largest section is made up by ambulant seated patients, who represent about 40 % of all requests. Other sections of the population include elderly people, physically disabled passengers in wheelchairs and public passengers using the DARP service, as an alternative to public transport in low density areas. In the analytical Jutland model, we do not make this distinction on sections of the population, but since it is not stated in (Midttrafik, 2010) how many of the requests are expected to be dynamic, we use these characteristics of the customers in order to make a better estimate on the *dod* for our test instance of the Jutland model.

DARPs for transporting elderly and physical disabled people are characterised by a low *dod*, and the DARPs for transporting public users are characterised by a relative high *dod* as stated in section 3.5.1. The ambulant patients, elderly and physically disabled people make up the majority of the customers in (Midttrafik, 2010), which suggests a low *dod*. To compensate for the public passengers, that after all are in (Midttrafik, 2010), we set the expected *dod* to 0.2 in the testing instance of the Jutland model.

The expected number of requests per year is 800,000 in (Midttrafik, 2010). The instance of the Jutland model constructed for testing covers about 7 % of the population of the real-life DARP instance in (Midttrafik, 2010). So we assume about 56,000 requests per year, or about 230 for every working day, in the testing instance of the Jutland model. Using the expected *dod* of 0.2, we expect 184 of them to be known in advance, and 46 of them to occur as dynamic requests.

In the analytical Jutland model, we have the following customer segments, distinguished by capacity request:

Customer segment 1 (cs_1)

cs_1 is seated customers. The customers in cs_1 have a capacity request for one seat. In the model denoted $q_i = (q_{i1}, q_{i2}, q_{i3}) = (1, 0, 0)$ for a request r_i. This group of requests is characterised by a relative high *dod*, since most of the dynamic requests are assumed to be made by public users from this customer segment. The service time on pick up and delivery for cs_1 is set to be 2 minutes.

Customer segment 2 (cs_2)

cs_2 is wheelchair customers seated outside their own folding wheelchair. The customers in cs_2 have a capacity request of one seat and room for a wheelchair.

In the model denoted by $q_i = (q_{i1}, q_{i2}, q_{i3}) = (0, 1, 0)$ for a request r_i. This group of requests is characterised by a relative low *dod*. The service time on pick up and delivery for cs_2 is set to be 3 minutes.

Customer segment 3 (cs_3)

cs_3 is big wheelchair customers seated in their own wheelchair. The customers in cs_3 have a capacity request of room for one big wheelchair. In the model denoted by $q_i = (q_{i1}, q_{i2}, q_{i3}) = (0, 0, 1)$ for a request r_i. This group of requests is characterised by a relative low *dod*. The service time on pick up and delivery for cs_3 is set to be 5 minutes.

6.3.1 Dynamic requests

The latest time to order a request, in this model, is 2 hours before service. The open hours of the dial-a-ride service in (Midttrafik, 2010) is from 6.00 ($t = 0$) to 24.00 ($t = 1080$). Thus, the latest time to order a dynamic request is 22.00 ($t = 960$). The dynamic requests arrival are divided into four time periods:

Morning:	$6.00 - 10.00$	$(0 \leqslant t < 240)$.
Midday:	$10.00 - 14.00$	$(240 \leqslant t < 480)$.
Afternoon:	$14.00 - 18.00$	$(480 \leqslant t < 720)$.
Evening:	$18.00 - 22.00$	$(720 \leqslant t < 960)$.

When generating the data for our test instance of the Jutland DARP, we assume that a dynamic request will arrive in the morning 40 % of the time, in the midday period 10 % of the time, in the afternoon 30 % of the time and in the evening 20 % of the time. The requests are assumed to arrive in accordance to a Poisson process with an arrival rate per minute equal to the excepted number of immediate requests divided by 1080 (the minutes per day of execution). The arrival rate for each time period is then determined by the probability a request will arrive in the period and the length of the period. The arrival time is followed by a reaction time of at least 2 hours, the reaction times are assumed to follow a Poisson distribution with a mean of 10 minutes. The reaction time refers to either pick up or delivery depending on the request being out- or inbound. We refer to section 5.3 for a more profound exposition of the simulation of data.

In the morning period, arriving requests will be outbound with a probability of 0.95 and inbound with a probability of 0.05. The other periods are summarised in table 6.1.

Period	Probability	
	Outbound	**Inbound**
Morning	0.95	0.05
Midday	0.70	0.30
Afternoon	0.10	0.90
Evening	0.25	0.75

Table 6.1: Probabilities for a dynamic request will be out- or inbound in different periods.

The customer segment of the request is determined by random. The probabilities are given in table 6.2.

Customer segment	**Probability**
cs = 1	0.80
cs = 2	0.15
cs = 3	0.05

Table 6.2: Probabilities for customer segments for dynamic request.

In figure 6.3 the produced data for the immediate request in our test instance of the Jutland DARP is displayed.

```
1  1 15 129 0 0 0 3 0 3 33.29517852086738 9.37866759308674 41.63674927739082 3.3519010245294143
2  2 27 131 0 0 1 1 0 3 11.208672957569785 21.69747344198563 37.6565672119793 3.585183705673024
3  3 33 131 0 0 1 1 0 0 19.16993684528372 20.1588059703126 8.636749277390818 3.3519010245294143
4  4 43 136 1 0 2 2 2 2 24.518062436183843 27.171155686125562 28.6821698048724 26.5361909992425
5  5 53 128 0 0 2 0 0 3 0.9749247272244455 23.146212549845394 39.4928440595285 4.18141890698989
6
7
8    ... (SOME LINES ARE OMITTED) ...
9
10
11 46 902 127 0 1 0 3 0 3 39.9654786217720 0.977838812715701 41.6367492773908 3.351901024529414
12 47 918 128 0 1 1 2 0 3 28.4776161496672 14.74302885951509 41.6367492773908 3.351901024529414
13 48 936 131 0 0 2 3 0 0 36.4362448599886 22.30218689237016 8.75553908090895 9.446572802465827
```

Figure 6.3: Date file of the immediate requests used in the instance of Jutland model used for testing.

6.3.2 Static requests

The advance requests for the instance of the Jutland test model have also been generated by the simulation method introduced in section 5.3, with input file presented on page 52. The produced data (part of) is shown in figure 6.4. In the implementation of the instance of the Jutland test model, the arrival time is used as desired service time for the advance request. Therefore is the "evening" time period prolonged with two hours, when simulating the static requests. The "reaction times" for the static request are ignored, since the requests is known all time and the arrival times are used for desired service times.

```
1  1 0 128 0 0 0 3 0 0 36.460451217145355 6.195167623076001 8.669405263891967 8.900004128276365
2  2 2 123 1 0 2 2 0 3 29.605533520441032 26.010504955132323 41.9373628744579 10.23585000886395
3  3 7 127 1 0 2 2 0 3 31.39404715732172 28.048036837985062 41.66940526389197 8.900004128276365
4  4 8 131 0 0 1 1 0 3 19.994371910868523 18.62767764157259 41.66940526389197 8.900004128276365
5  5 12 131 1 0 0 0 0 3 6.308019364943658 3.9696605456359455 41.6694052638919 8.900004128276365
6
7    ... (SOME LINES ARE OMMITED) ...
8
9  186 919 127 0 1 0 3 0 3 39.73824434688275 3.04249326231804 34.5221016601562 2.01623305740877
10 187 922 129 0 0 0 3 0 3 39.77071960870581 5.99978337095169 33.0015614972930 6.05587625225643
11 188 931 127 1 0 0 3 0 3 41.84124356693064 10.9104913883629 41.66940526389197 8.9000041282763
```

Figure 6.4: Date file of the advance requests used in the test instance of Jutland model used for testing.

Note that the probabilities for customer segments are different in the simulations for the static and the dynamic requests. We set the probability lower for the two types of wheelchair users, since the arrival of dynamic requests is driven by public users (from customer segment 1).

6.4 Constraints and objective

We will restate the objectives and constraints for the Jutland model here, although they are mentioned in the introduction of the model in section 3.2.3.

Constraints

As stated in chapter 3, the Jutland model imposes constraints on time windows, multiple vehicle capacity, maximal travel time per request, maximal route duration and driver breaks. Though are the driver breaks ignored in the later implementation of the suggested solution method.

The time window for a pick up or delivery on a request is set to ten minutes before and after the desired service time of the customer of the request. Maximal travel time of a request is set to be 1.5 times the direct travel time between the pickup location and destination. Travel time of a customer is the time the customer spends in the vehicle after having been serviced at pick up until service starts at the destination of the customer.

The objective

The objective of the Jutland model is adopted from the Copenhagen model. The objective is multiple criteria, in costs, fleet utilisation, deviation from desired service time and waiting time. In the Jutland DARP the costs and fleet utilisation is considered more important than the service criteria.

7

Exploiting knowledge about future requests

The aim in this chapter is to make less good solutions! The idea is that a less good solution is more flexible, as mentioned earlier in section 3.4. However, instead of just producing "bad" solution with "bad" methods, we structure the way the flexibility is put in the routes and schedules. The cost of the flexibility is still there, but we hope that the price we get from this cost is higher.

We start this chapter by examining some methods not developed for the DARP, but for general DVRP with one location associated with each request. We will translate these methods, and insights from them, found useful to the context of a DARP, to handle requests with two locations, and then use the insights from the methods in a solution method for the Jutland DARP.

7.1 Fruitful regions and corridors

In (van Hermert and Poutré, 2004) the term *fruitful regions* is presented. van Hermert and Poutré (2004) deal with a DVRP, only with one location per request. They divided the total region where immediate requests can occur into smaller regions covering the total region. The regions are given a probability on how likely it is that an immediate request will occur in a given time period. These probabilities are the sums over the probabilities that a customer located in the region will make an request in the time period. Regions with high potential for an immediate request in the time period are referred to as *fruitful regions*.

When determining routes, the information about the regions can be used to insert *fake requests* and thereby add some flexibility to the routes. The part of the route driving to an from a fake request is called an *anticipated move*. In figure 7.1 an anticipated move is shown. *Route A* is part of a route. In this section of the route a vehicle will be driving from one location to service another, as shown in figure 7.1. The dark node represent a fake request, and the nodes inside the fruitful region are immediate requests that *might* occur at the time where the vehicle is driving *route A*. The route is changed to handle the fake request, thereby adding some flexibility to the route, making it able to handle an immediate request in the fruitful region if one occurs.

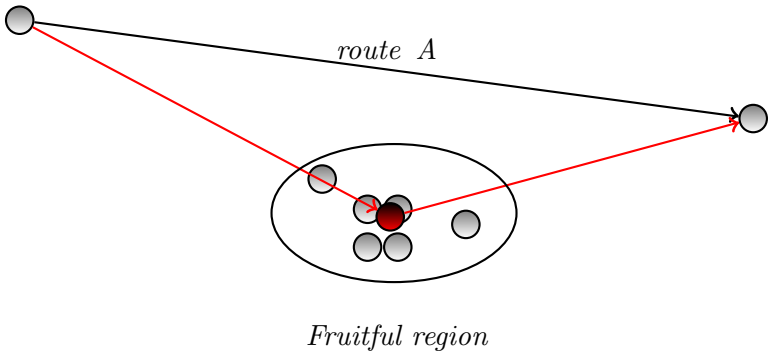

Figure 7.1: A fruitful region in a DVRP with one-location-requests.

If no immediate request occurs in the region, the vehicle will drive *route A*, but the anticipated move made, leaves *route A* with more waiting time than if the anticipated move was not put in the route. So even though the vehicle does not drive to the fake request, there are some extra costs associated with making the anticipated move. On the other hand if an immediate request occurs in the fruitful region and no anticipated move has been made, then the vehicle is less likely to be able to handle the immediate request, since the route is not constructed with the flexibility to handle immediate requests from the fruitful region. If the immediate request can not be handled by the vehicle (ignoring other vehicles in the system) then a new route must be started. The dispatcher basically faces the choice between:

1. Ignoring the fruitful region and send a new vehicle if an immediate request occurs.

2. Making the route flexible enough to handle an immediate request in the fruitful region, if one should occur.

The dispatcher can evaluate this, with information about future requests of uncertain quality, when making the choice. I.e, which choice is the optimal one depends on how likely it is that an immediate request will occur in the fruitful region and the extra cost (in economical and lower service) by making an initial flexible route, and the cost of making a new route. Assume that the cost of a new route is 100 and the extra cost of making the initial route flexible is 10 in the objective of the problem. Then already a probability of 10 % for an immediate request occurs in the fruitful region make an initial flexible route an optimal choice. Is the probability below 10 %, then is it optimal to make a tight original route, and to make a new route if an immediate request occur in the fruitful region. But other considerations than just the costs in the objective could influence this decision. For instance, choosing a flexible route offers some insurance on the maximal costs per day of execution, and how many vehicles are needed to execute the driving (in this isolated part of the system). Such "guarantees" could be useful for a dispatcher with a fleet of own vehicles and not having the option to rent vehicles. The dispatcher could lower the number of vehicles that

he needs to service all customers, which is usually most important since extra vehicles cost a lot.

Using fruitful regions in a DARP might seem like a good idea (it is not a good idea!). Assume for instance, we anticipate that a lot of pick up locations will occur close together (a fruitful region). The idea behind fruitful regions, then suggest that the dispatcher makes routes flexible in a way such that they can visit this region if a request occurs. But what if the request occurring is going in the opposite direction of the vehicle that was given a flexible route to let it visit the fruitful region? We could just make more routes flexible enough to visit the fruitful region, from different directions. But to this end we need materialised or expected customers to be included in other parts of the route. The properties of the DARP make it hard to construct routes, determining the direction of the route in advance, since directions are given in the requests of the problem already. Figure 7.2, where the dashed arrows point to the destinations of the pick up requests inside the fruitful region, illustrates the problem of using fruitful regions in a DARP.

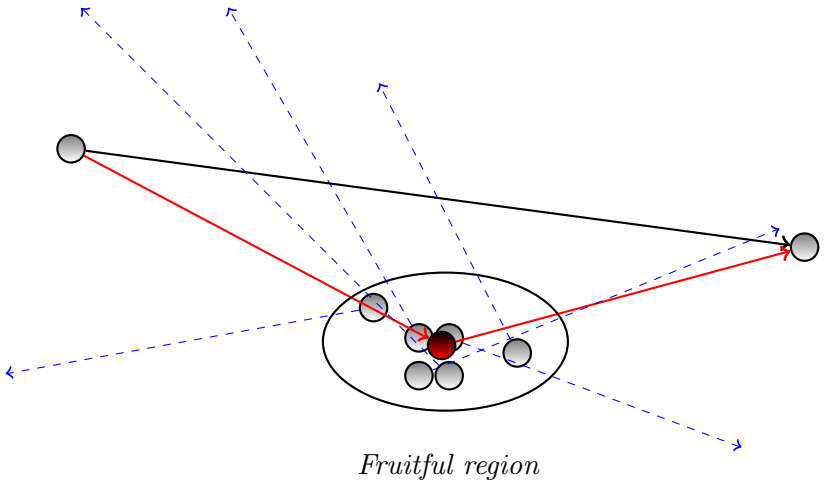

Figure 7.2: Using fruitful regions in a DARP is not a good idea.

Instead of using fruitful regions, as introduced in (van Hermert and Poutré, 2004), we introduce an extension of the concept of fruitful regions, which we call *fruitful corridors*. Like van Hermert and Poutré (2004), we divide the total region, where immediate requests can occur, into smaller regions covering the total region. A corridor is then two regions (a start region and an end region) and a direction between the two regions.

If we have θ regions, the number of corridors becomes θ^2. All regions (start regions) have a corridor pointing at all other regions including the same region (end regions). We say a corridor is fruitful if there is a high probability that a request occurring with pick up location in the start region of the corridor and destination in the end region of the corridor in a certain time period. Figure 7.3 shows a fruitful corridor.

The idea is, that if the probability of a corridor is high enough, then the

dispatcher should make a route flexible by making an anticipated move. An anticipated move, in the context of the dynamic DARP, consists of inserting a fake request with service time in the corridor and a pick up location in the start region of the corridor and destination in the end region of the corridor. In figure 7.3 an anticipated move is shown, the dark nodes represent the pick up location and destination of the fake request.

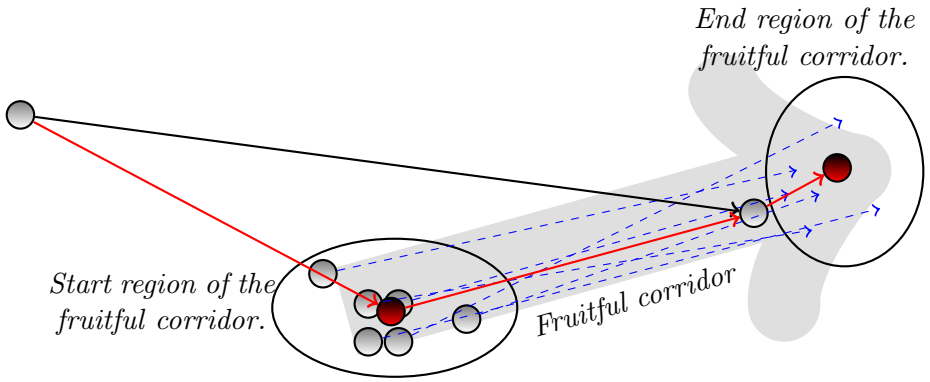

Figure 7.3: A fruitful corridor in a dynamic DARP.

7.2 Multiple scenario approach

Bent and van Hentenryck (2004) introduce a method to solve DVRPs with requests only with one location, and thereafter expand the method to deal with DVRPs with one location per request using knowledge about future requests. We will in the following describe the method for solving DVRPs with one location per request using knowledge about future requests, and parallel to the presentation of the method expand it to deal with dynamic DARPs.

In (Bent and van Hentenryck, 2004) the method is referred to as the *multiple scenario approach* (MSA). Instead of maintaining a single plan for the DVRP at hand, the method constructs multiple plans. The plans are constructed while the requests are serviced. When an immediate request occurs, the idea is that the presence of multiple plans enhances the probability that the request can be inserted in one of the plans.

A single plan must at all time t be the plan that is followed, we call this plan the *master plan*, and at time t the current master plan is denoted Ξ_t^*. To select the master plan that currently is followed, Bent and van Hentenryck (2004) introduce the *consensus function*. The consensus function could be the objective of the problem. Bent and van Hentenryck (2004) suggest another criteria for the consensus function; namely to select the plan that resembles the other plans the most. We will however use the plan with the best objective as a master plan when using the MSA as part of a suggested method for the dynamic DARP later.

The set of plans needs to be maintained over time, since some plans get outdated and others are constructed. To accomplish this maintenance issue, the MSA has to handle four types of events:

1. *Customer request*: A customer request event is when a new immediate request occur, this kind of event is not controllable by the MSA itself.

2. *Vehicle departure*: A vehicle departure event is when the master plan Ξ_t^* dictates that a vehicle leaves its current location. Using the drive first waiting strategy, this is when the vehicle completes servicing the location. If for some plan the next location in the route of the vehicle that departs is not the same as it is in the master plan this plan is no longer feasible.

3. *Plan generations*: A plan generation event is when a new plan, Ξ, is generated. This type of event is controlled by the method itself, unlike customer request events. When a new plan is generated, all requests picked up and delivered or just picked up will stay in their current route. This guarantees consistency of the new plan with decisions already made by the dispatcher concerning the execution.

4. *Time-outs*: Time-outs occur when the master plan dictates that a vehicle should wait at its current location, while one or more plans specifies that the vehicle leaves the location now. Because we use the drive first waiting strategy, we can ignore this type of event. The master plan never dictates a vehicle to wait, unless the time window of the location is not open yet and service is not done at the location. But if this is the case, then it is also the case for the other plans.

Bent and van Hentenryck (2004) formalise the events and how they affect the master plan and the set of all plans. We will also summarize this, but first we will present a set and a helping method from (Bent and van Hentenryck, 2004), which are useful in the formalised description of the events and their influence on the master plan and the set of plans.

The following set and helping method is from (Bent and van Hentenryck, 2004), but reformulated in the context of the dynamic DARP:

> DEPART(t) returns a list of requests, who have received service before or at time t. This set also includes requests where service has begun but is not finished.
>
> COMPATIBLE(Ξ, Ξ_t^*, t) is true if the plan Ξ is compatible with the master plan Ξ_t^* up to time t and false otherwise. I.e., let $r_i \in$ DEPART(t) and $r_i \in \xi_j$ where ξ_j is a route in Ξ. And let ξ_k be the route in Ξ_t^* containing r_i. Then COMPATIBLE(Ξ, Ξ_t^*, t) is true if and only if the location after $r_i(p_i^+)$ is the same in ξ_j and ξ_k for all $r_i \in$ DEPART(t).

We will now get back to describing the events of the MSA. Let us start with denoting the set of plans at time t by Ω_t.

Customer request

The arrival of an immediate request, r_i, at time t might make a plan Ξ in Ω_t infeasible if the request can not be inserted in Ξ. However r_i may feasibly be

inserted in one or more of the other plans in Ω_t. If r_i can be inserted in one or more plans, the new set of plans, Ω_{t+1}, is the plans where r_i could feasibly be inserted in a route. Since all the plans in the new set of plans are different from the old ones, we will have to go through all the plans in the new set of plans to find the best one among them. This plan is set to be the new master plan Ξ^*_{t+1}.

Otherwise, if no feasible insertion of r_i is found among the plans in Ω_t, then a new route ξ_j is started to accommodate r_i. The new set of plans, Ω_{t+1}, then becomes all the plans from Ω_t where ξ_j is added. Since the cost of the new plans all are increased by the same amount, the cost of ξ_j, the new master plan, Ξ^*_{t+1}, is the old master plan Ξ^*_t, where ξ_j is added.

Vehicle departure

When the current master plan Ξ^*_t dictates that a vehicle b_j with an associated route ξ_j in Ξ^*_t leaves the current location of b_j in accordance to ξ_j, we need to remove all plans from Ω_t not compatible with this vehicle departure. This is

$$\Omega_{t+1} := \{\Xi \in \Omega_t | \textsc{compatible}(\Xi, \Xi^*_t, t)\}, \tag{7.1}$$

$$\Xi^*_{t+1} := \Xi^*_t. \tag{7.2}$$

Bent and van Hentenryck (2004) state that Ξ^*_{t+1} is updated to the best plan in Ω_{t+1} when a vehicle departure occurs. This evaluation of the plans is unnecessary, for our choice of consensus function, the objective of the problem. Ξ^*_t is not removed from Ω_t and no new plans are added to Ω_t, when a vehicle departure occur. So if Ξ^*_t was the best plan in the old list of plans Ω_t, then it must also be the best in the new list of plans Ω_{t+1}.

Plan generation

Assume we have generated a new plan Ξ at time t, for which $\textsc{compatible}(\Xi, \Xi^*_t, t)$ is true. We set

$$\Omega_{t+1} := \Omega_t \cup \{\Xi\}, \tag{7.3}$$

$$\Xi^*_{t+1} := \arg\min_{\Xi \in \Omega_{t+1}} \{\text{"the cost of } \Xi\text{"}\} \quad , \tag{7.4}$$

where "the cost of Ξ" refers to the cost of the plan with regard to the objective of the problem, referring of the consensus function chosen.

The plan generation uses one or more fake requests, to add flexibility to the plans. After a plan has been generated, also including the fake request(s), the plan is *projected* on to the set of materialised request by removing the locations of the fake requests from the routes in the plan, consequently making the plan flexible in the "area(s)" of the fake request(s). The fake requests used in different plan generations are not necessarily the same.

We will incorporate the ideas from the MSA in our solution method for the Jutland DARP.

7.3 Exploiting knowledge about fruitful corridors

We will use the corridors introduced in section 7.1 to make the plans flexible in a clever way. While an anticipated move makes a plan more flexible in the areas near the pick up location and destination and time period around the fake request of the anticipated move, the flexibility added by the anticipated move is *limited* to just these areas and time period.

Instead of adding undesirably many fake requests in a single plan, we combine the anticipated moves made from fruitful corridors with the MSA from (Bent and van Hentenryck, 2004) modified to the dynamic DARP, in order to compensate for the limited flexibility of a single anticipated move. That is, a number of anticipated moves are made in one plan, adding flexibility to this plan in some sets of time periods and areas. Other anticipated moves are made in other plans, thereby spreading the flexibility of the plans, not in one plan, but over a sample of plans. Then when a immediate request occurs, the request can hopefully be inserted in one or more of the plans.

We put the corridors with the highest probability (the fruitful corridors) in a candidate list. From this candidate list we pick a number of corridors according to the number of expected immediate customers in the rest of the period of execution, where the same corridor from this candidate list can be picked more than once. The corridors probability of being picked corresponds to the probability that an immediate request will occur in the corridor. An anticipated move is then made by making a fake request in the corridor, with arrival time at or after the current time t.

To avoid the scenario of generating a lot of plans very alike in flexibility, we need to ensure that not only the same corridors are considered in the plan generations. If a corridor is picked in the current plan generation, the probability for the corridor being picked in the next plan generation is lowered, by ignoring the corridor the first time it gets picked, if picked at all, in the next plan generation. A corridor that is picked more than once in a plan generation obtain the same reduced probability as if it where only picked one time in the plan generation. We refer to the lowering of the probability for a corridor being picked in the next plan generation by the corridor being *labelled with a tabu*.

7.4 Suggested method for solving the Jutland DARP

We will now propose a method to solve the dynamic DARP. The solution method aims at making robust solutions with high flexibility.

We assume that a forecast on the travel patterns for immediate request in terms of (fruitful) corridors has been made beforehand. The travel patterns on immediate requests are made with the forecast method introduced in chapter 5 and expanded to time periods (using the probabilities for a request occurring in a certain time period) and directions (using the probabilities for out- / inbound for each time period). We assume an instance of DARP as described in chapter

6. Hence we have a set of vehicles B and a set of static requests R^{adv}. We also have a set of dynamic requests R^{imm} that gets revealed over time with associated reactions time. R_t is the set of materialised requests at time t, $R_t = R_{adv} + R_t^{imm}$, where R_t^{imm} is the set of immediate requests known at time t.

We call our method the FCMSA (Fruitful Corridors Multiple Scenario Approach method). FCMSA exploits knowledge about anticipated travel patterns for future request by constructing artificial requests, using the forecast on travel patterns for immediate requests, and then inserting the artificial requests in the list of unrouted requests. After constructing multiple solution plans with respect to both materialised and artificial requests, the artificial requests are removed from the plans. This makes the plans flexible and more likely to accommodate future requests.

FCMSA makes use of the insertion heuristic introduced by Madsen et al. (1995), and presented in chapter 4.3. The "load" of insertion is calculated in accordance to N_{wait} with $K^2_{wait} = 1$, $K^1_{wait} = 1$, N_{dev} with $K_{dev} = 2$ and N_{cap} with $\hat{K}_1 = 1$, $\hat{K}_2 = 1$ and $\hat{K}_3 = 1$.

N_{drt} is redefined to the actual economical cost:

$$N_{drt} = \sum_{\xi_j \in \Xi} (Y_j + \sum_{v_\alpha \in \xi_j} U_j t_\alpha^{drt} + Z_j(t_\alpha^{wait} + \phi_j^\alpha)), \tag{7.5}$$

where t_α^{drt} is the driving time to v_α from the last location in the route, and t_α^{wait} is the waiting time at location v_α.

We emphasis again that the above Ns are the total "load" in each measure, and the "load" of insertion of a request is the difference in the total "load" of the plan before a request is inserted and the total "load" of the plan after a request is inserted.

The parameters are set in this way in order to value economical costs and capacity utilisation high in the multiple-criteria objective. Though without ignoring service at desired pick up time and long waiting time at a location.

The FCMSA method is presented in the following pseudo code. The notation for the master plan (Ξ_t^*) and the set of plans (Ω_t) at time t used, unintentionally indicates that they are updated at every time instance (Bent and van Hentenryck, 2004). This is not the case, only when an event occurs (an immediate request occurs, a vehicle departs or a new plan is generated) at time t are Ω_t and Ξ_t updated to Ω_{t+1} and Ξ_{t+1}^* respectively.

1: **procedure** FCMSA (INITIALISATION)
2: **Step 0.1**: Set the "list of unrouted requests" to R^{adv}.
3: **Step 0.2**: Sort the corridors in the forecast on travel patterns for immediate requests, in accordance to the probability that a request will occur in the "start region" to the "end region", in the time period of the corridor. This list is called "*list of corridors*".
4: Make a new list of directions from "*list of corridors*", only containing corridors with a certain high probability. This list is called the "*candidate list of corridors*".
5: **Step 0.3**: Randomly take a number of corridors from the "*candidate list of corridors*". The number of corridors needed are equal to the expected

number of immediate requests. The probability that a corridor is picked is its relative probability, and a corridor can be picked more than once! Corridors picked are labelled with a tabu.

Step 0.4:

for all the corridors picked in *step 0.3* **do**

Make an artificial requests r_a in the time period associated with the corridor, with random time desired for pick up or delivery time (depending on out or inbound). The two locations of r_a is placed randomly in the regions associated with the corridor. And the customer segment of r_a is chosen randomly with respect to the expected distribution of customer segments of immediate requests in the model. Insert r_a in the "*list of unrouted requests*".

end for

The "*list of unrouted requests*" is sorted with respect to the difficulty degree M^i for all r_i in the list (see definition 4.5).

Step 0.5: Make an initial solution, by inserting the requests from the "*list of unrouted requests*" in routes (possible new ones) in a sequential manner. The solution is then projected to a plan for the materialised requests only. This is the initial "*master plan*", Ξ_0^*. Ξ_0^* is added to initial "*list of plans*", Ω_0.

end procedure

This concludes the initial pre-dynamic phase of the FCMSA model. We will now describe the rest of the method. At this time only one plan has been constructed. The next plan is calculated during the execution of the routes in the plan generation step. We emphasise that plans constructed besides future routes to be driven, also include routes already driven and routes currently driven by vehicles.

procedure FCMSA (DYNAMIC PHASE)

while $t \in$"the day of operations" **do** $\triangleright$ t is updated according to real time.

Event 1.1: (Immediate customer request)

if an immediate request r_{imm} has occurred before t **then**

for all plans Ξ in the "*list of plans*", Ω_t, **do**

if r_{imm} can be inserted in Ξ **then**

Add r_{imm} to the route $\xi_j \in \Xi$, corresponding to the best insertion.

end if

end for

if a feasible insertion of r_{imm} is found in the "*list of plans*" **then**

Remove all the plans from the "*list of plans*" where no feasible insertion was found. The resulting "*list of plans*" is Ω_{t+1}. Set the new "*master plan*", Ξ_{t+1}^*, to the best plan in Ω_{t+1}.

else

Add a new route ξ_k, with a corresponding new vehicle b_k, to all plans in Ω_t. The resulting "list of plans" is Ω_{t+1}. The new "*master plan*",

Ξ^*_{t+1}, corresponds to Ξ^*_t in Ω_t with the new route added.

end if

Set $R^{imm}_{t+1} := R^{imm}_{t+1} + \{r_{imm}\}$.

end if

Event 1.2: (Vehicle departure)

if the "*master plan*" dictates some vehicle departs from some location at time t **then**

for all plans $\Xi \in$ the "*set of plans*" Ω_t **do**

if COMPATIBLE(Ξ, Ξ^*_t, t) is **false then**

Remove Ξ from the "*set of plans*".

end if

The resulting "list of plans" is Ω_{t+1}.

Set $\Xi^*_{t+1} := \Xi^*_t$.

end for

end if

Event 1.3: (Generate a new plan)

while the method is idle with regard to event 1.1 and event 1.2 at time t **do**

Step 1.3.0: Set the "list of unrouted requests" to $(R^{adv} + R^{imm}_t) \setminus \text{DEPART}(t)$.

Step 1.3.1: Randomly take a number of corridors from the "*candidate list of corridors*". The number of corridors needed corresponds to number of expected customers in the rest of the day of operations. Corridors picked are labelled with a tabu.

Step 1.3.2:

for all the corridors picked in *step 1.3.1* **do**

Make an artificial requests r_a in the associated time period, with random time desired for pick up or delivery time (depending on out or inbound). The two locations of r_a is placed randomly in the associated regions. And the customer segment of r_a is chosen randomly with respect to the expected distribution of customer segments of immediate requests in the model. Insert r_a in the "*list of unrouted requests*".

end for

The "*list of unrouted requests*" are sorted with respect to the difficulty degree M^i for all r_i in the list.

Step 1.3.3: Make a partial plan "*plan-depart*" by projecting the "master plan" on the set of requests in DEPART(t). ▷ The partial plan, "*plan-depart*", covers all request that already have been picked up and therefore not can be moved to another route.

Step 1.3.4: Make an new plan, by inserting the requests from the "*list of unrouted requests*" in "*plan-depart*". The plan is then projected to a plan for the materialised requests only, and the projected plan is added to the current "*list of plans*", Ω_t, resulting in a new "*list of plans*" Ω_{t+1}.

Step 1.3.5: Determine the best plan with regard to the lowest "load" in Ω_{t+1} and set the new "*master plan*" Ξ^*_{t+1} to this plan. ▷ If a new event 1.1 or 1.2 occurs before *step* 1.3.5 is finished, no plans are added to the

"list of plans". And Ω_t and Ξ_t^* are still the "list of plans" and the "master plan" respectively.

end while

end while

end procedure

In chapter 8 we test the suggested method presented above on the test instance of the Jutland DARP constructed in chapter 6.

8

Experimental results

The FCMSA has been tested on the constructed instance of the Jutland DARP (see chapter 6) with 257 requests (209 advanced and 48 immediate). The tests were run on a Linux laptop with two Intel Core2 Duo CPUs in real-time. The FCMSA was able to construct a new plan in a few seconds in the start of the execution, and later over 60 plans in a minute! An upper limit on the number of plans to consider was set to 200.

The results of the testes are shown in table 8.1 (Cost = the total costs in Danish kroner, WT = total waiting time in minutes, CU = total unutilised vehicle capacity in seats[1], PG = number of plans generated).

	FM1	FM2	FM3
Routes	239	239	237
Cost	59477	61996	58761
WT	8	8	11
CU	5214	5208	5124
PG	45691	46548	49159

Table 8.1: Test results for FCMSA.

The results tells us that a large majority of the routes generated only handles a single request! A closer look at the generated solution (the last master plan) shows that no routes are constructed which handle more than three requests. The routes with three requests are all "local routes" in one of the two regions with population about 35,000 and 70,000 respectively. The short routes explain the low total waiting time, since routes with a single request generate no waiting time. The many routes also leave a lot of unutilised capacity, about 20 seats per route (this is over all locations, including the depots where the vehicle is empty). Actually no instances of more than two requests in a vehicle at a time where found. This only occurred in the "local routes" with three requests in a route. The costs indicate that service for a single request is rather expensive, at about 230 Danish kroner per request!

Due to a bug in the way the deviation of the desired service times where calculated, the results are omitted. The bug in the implementation of the deviation from the desired service time is, that instead of only calculating the deviation of service time and desired service time for the request's location with time win-

[1] Seats or space for wheelchairs.

dow, both the difference between the actual service time at the location with and without time window and the desired service time at the time window location (another location for the last!) is included in the deviation measure. Since this bug unintentionally implement a "long ride"-load, where the excess driving time is included in the load cost, this bug is expected to partly be the reason for the many short routes with only one requests and the very few detours in the rides of the customers. Unfortunately this bug was discovered to late to rerun the tests. Even though this bug has been discovered in the implementation, we do not discard the rest of results obtained, but emphasise that they shall be seen in this light.

For comparison the implementation of the FCMSA was modified to resemble other methods. The following modifications were considered:

PlainFruit (PF): To see what impact the fruitful corridors have without the multiple scenario approach, we suspend the plan generation.

PlainInsert (PI): To see how the insertions heuristic acts without the expansion, we suspend the plan generation and the fake requests (fruitful regions).

OnePlanMax (1M): To see what impact the number of plans available when a immediate requests is inserted have we limit the number of plans generation tries to one per minute.

Three tests where made for (PF) and (PI) and a single test where made for (1M). The results of the tests are shown in table 8.2.

	PF1	PF2	PF3	PI1	PI2	PI3	1M
Routes	240	240	239	238	237	239	240
Cost	55587	59766	60079	63560	60626	62882	57520
WT	4	6	4	8	8	4	5
DD	6732	6680	6631	6639	6565	6603	6536
CU	5316	5280	5280	5160	5214	5220	5328
PG	0	0	0	0	0	0	453

Table 8.2: Test results for modified FCMSAs.

The results resemble the results for the FCMSA without any modifications, but the single tests with only one plan generation per minute indicate that the massive number of plans calculated in the regular FCMSA, is undue to many. The Plain Fruit tests do better than the Plain Insert tests and on average better than the FCMSA. Combined with the result for the One Plan Max, this could indicate that the idea behind the fruitful corridors work better without being combined with the MSA with the objective as consensus function. If we accept this indication as true, then we expect that the greedy nature of the consensus function chosen destroys the flexibility added by the anticipated moves in the fruitful corridors, since this flexibility comes with a cost in the objective, and thereby flexible plans are unlikely to be appointed to be the master plan. Flexible plans become then more likely to be removed when a vehicle departure occurs.

The results of our tests does not show any promising results for the FCMSA, but in the light of the few tests on only one instance and the bug discovered in the implementation, it is very hard to conclude anything from the tests.

9

Conclusions

Suggested solution methods for the dynamic DARP are rarely found in scientific literature. Only *one* where found when exploring literature for this thesis; namely the insertion heuristic presented by Madsen, Ravn and Rygaard from 1995! All the solution methods found for the DARP, dynamic as well as static, deal with dial-a-ride services in urban areas. Non of the presented solution methods deal with exactly the same DARP.

In this thesis we have studied the DARP, both static and dynamic, closely. We break down the different parts of the problem and thereby try to describe the DARP out of the context of the characteristics of a single dial-a-ride service in some city. We have constructed a problem resembling the DARP for low populated areas, a version of the DARP, we have not found studied elsewhere.

A new solution method where presented. The idea behind the method is to insert fake customer requests in fruitful corridors, to make an solution robust enough to deal with future customer requests, by inserting them in existing routes. The method was tested without showing any promising results. Due to the limited number of test that has been carried out, it can not be ruled out that the disappointing results could be caused by other things, than bad properties of the suggested method. We suggest that furtherer tests are carried out before final conclusions are made about the suggested method. The FCMSA is not specifically constructed for a low populated area, so tests on urban areas might also be interesting, to see how the FCMSA behaves in this context. Also would it be interesting to test the method on real-life instances. Further testes without the MSA in the FCMSA might also be interesting.

Implementation

The methods have been implemented in the programming language *Java (version 1.6.0_18)*. The following classes have been constructed.

forecast, **runForecast**, **Location**, **Plan**, **Request**, **Route**, **method**, **runMethod** and **Vehicle**:

The methods for making forecasts on travel patterns and simulating requests (immediate and advance) have been implemented in the same class **forecast**. The class work independently of the other classes (except **runForecast** that is used to run it!). To run the method, make the following call:

```
" java runForecast dispIMM.data ",
```

where **dispIMM.data** is an appropriate datafile. The method constructs datafiles containing the travel patterns and simulated requests. The folder "*Forecast*" on the CD-ROM contains all necessary files to run the method. It can be run on both **dispIMM.data** and **dispADV.data**, to construct simulated immediate and advance requests, respectively. For further information about the class **forecast** we refer to the *Javadoc* in the file itself.

The output is by default named:

simEyeEasy.data - the simulated requests, in a version made easy to read.

simulation.data - the simulated requests, in a version made easy to read for the **method** class.

travelPatterns.data - the forecasts on travel patterns, in a version made easy to read for the **method** class.

travelPatternsEyeEasy.data - the forecasts on travel patterns, in a version made easy to read.

The FCMSA is implemented in **method**. Almost all work is done in the class **method** itself. The classes **Location**, **Request** and **Vehicle** solely acts as instances corresponding to their name, and do no work in the implementation. The criteria of the objective and the "*load*"-cost of an insertion of a request in relation to routes and plans calculated in **Route** and **Plan** respectively

To run the FCMSA, a call is made to **runMethod**. An example, (using the files available in the folder "*FCMSA*"):

```
" java runMethod advRequests.data fleet.data test.data
travelPatterns.data immRequests.data 46 ".
```

We refer to the *Javadoc* for more information on the classes above.

The rest of the CD-ROM

The folder "*InputData*" contains the three data files need to make the rest of the data for the test instance used. The folder "*Other*" contains different variants of **method** used:

PlainFruit - the insertion heuristic without multiple scenarios.

OnePlanMax - only one try at constructing a new plan per minute.

PlainInsert - the insertion heuristic without fruitful corridors and multiple scenarios.

All Java and data files are also available by request to `jbjerring@gmail.com`.

Bibliography

Bent, R. W. and P. van Hentenryck (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Reasearch 52*, 977 – 987.

Borndörfer, R., M. Grötschel, F. Klostermeier, and C. Küttner (1997). Telebus berlin: Vehicle scheduling in a dial-a-ride system. Technicla Report SC 97-23, Konrad-Zuse-Zentrum für Informationstechnok Berlin.

Cordeau, J.-F. and G. Laporte (2007, September). The dial-a-ride problem: models and algorithms. *Annals of Operations Research 153*(1), 29–46.

Ichoua, S., M. Gendreau, and J.-Y. Potvin (2007). Planned route optimization for real-time vehicle routing. In V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, and I. Minis (Eds.), *Dynamic Fleet Management*, pp. 1– 18. Springer.

Larsen, A. (2000). *The Dynamic Vehicle Routing Problem.* Ph. D. thesis, Technical University of Denamrk, Lyngby.

Larsen, A., O. B. Madsen, and M. M. Solomon (2007). Classification of dynamic vehicle routing systems. In V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, and I. Minis (Eds.), *Dynamic Fleet Management*, pp. 19 – 40. Springer.

Larsen, A., O. B. Madsen, and M. M. Solomon (2008). Recent developments in dynamic vehicle routing systems. In B. Golden, S. Raghavan, and E. Wasil (Eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges*, pp. 199 – 218. Springer.

Madsen, O. B. G., H. F. Ravn, and J. M. Rygaard (1995, December). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research 60*(1), 193 – 208.

Midttrafik (2010). Midttrafik og Sydtrafiks 1. udbud af koordineret kollektiv trafik i Region Midtjylland og Region Syddanmark (minus Fyn). Udbudsvilkår og kontrakt, Udbud 2010.

Mitrovic-Minic, S. and G. Laporte (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological 38*(7), 635 – 655.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. In B. Golden and A. Assad (Eds.), *Vehicle Routing: Methods and Studies*, pp. 223 – 248. North-Holland.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of Operations Research 61*, 143 – 164.

Tarantilis, C. D., D. Diakoulaki, and C. T. Kiranoudis (2003). Combination of geographical information system and efficient routing algorithms for real life distribution operations. *European Journal of Operational Research 152*, 437 – 453.

Toth, P. and D. Vigo (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science 31*(1), 60–71.

van Hermert, J. I. and J. L. Poutré (2004). Dynamic routing problems with fruitful regions: Models and evolutionary computation. In *Parallel Problem Solving from Nature - PPSN VIII*, pp. 692 – 701. Springer Berlin / Heidelberg.

The tender documents (Midttrafik, 2010) can be found at
`https://www.ethics.dk/asp4/tender/mt_0301_20091023.nsf/stddocsPub?OpenView&count=-1&ExpandView`.