(DRAFT)

Demand-Responsive Transport: Models and Algorithms

Lauri Häme

January 16, 2012

Abstract

Demand-responsive transport (DRT) is an advanced, user-oriented form of public transport between bus and taxi, involving flexible routing of small or medium sized vehicles. This dissertation presents mathematical models for demand-responsive transport and algorithms that can be used to solve combinatorial problems related to vehicle routing and journey planning.

Tiivistelmä

Demand-responsive transport (DRT) is an advanced, user-oriented form of public transport between bus and taxi, involving flexible routing of small or medium sized vehicles. This dissertation presents mathematical models for demand-responsive transport and algorithms that can be used to solve combinatorial problems related to vehicle routing and journey planning.

Sammanfattning

Demand-responsive transport (DRT) is an advanced, user-oriented form of public transport between bus and taxi, involving flexible routing of small or medium sized vehicles. This dissertation presents mathematical models for demand-responsive transport and algorithms that can be used to solve combinatorial problems related to vehicle routing and journey planning.

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Lauri Häme. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows. *European Journal of Operational Research*, 209, p. 11–22, February 2011.

II Esa Hyytiä, Lauri Häme, Aleksi Penttinen, Reijo Sulonen. Simulation of a Large Scale Dynamic Pickup and Delivery Problem. In *SIMUTools*, Malaga, Spain, March 2010.

III Lauri Häme, Jani-Pekka Jokinen, Reijo Sulonen. Modeling a competitive demand-responsive transport market. In *Kuhmo Nectar Conference on Transport Economics*, Stockholm, Sweden, June-July 2011.

IV Jani-Pekka Jokinen, Lauri Häme, Esa Hyytiä, Reijo Sulonen. Simulation Model for a Demand Responsive Transportation Monopoly. In *Kuhmo Nectar Conference on Transport Economics*, Stockholm, Sweden, June-July 2011.

V Teemu Sihvola, Lauri Häme, Reijo Sulonen. Passenger-Pooling and Trip-Combining Potential of High-Density Demand Responsive Transport. In *Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2010.

VI Lauri Häme, Harri Hakula. Dynamic journeying under uncertainty. Submitted to *Under review for publication in European Journal of Operational Research*, 20.12.2011 .

VII Lauri Häme, Harri Hakula, Saara Hyvönen. Dynamic journeying in scheduled networks. Submitted to *Under review for publication in IEEE Transactions on Intelligent Transportation Systems*, 16.1.2012 .

VIII Lauri Häme, Harri Hakula. A Maximum Cluster Algorithm for Checking the Feasibility of Dial-A-Ride Instances. Submitted to *Under review for publication in Transportation Science*, 16.1.2012 .

IX Lauri Häme, Harri Hakula. Routing by Ranking: A Link Analysis Method for the Constrained Dial-A-Ride Problem. Submitted to *Under review for publication in Operations Research Letters*, 16.1.2012 .

Author's Contribution

Publication I: "An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows"

This article was written by the author.

Publication II: "Simulation of a Large Scale Dynamic Pickup and Delivery Problem"

Parts of this paper were written the author, including most of Section 3 and parts of Section 1. The simulations reported in Section 3 were designed and conducted by the author.

Publication III: "Modeling a competitive demand-responsive transport market"

The author was the main author of this article, which was nominated for the best student paper award in Kuhmo Nectar 2011. The article is under review for publication in Economics of Transportation.

Publication IV: "Simulation Model for a Demand Responsive Transportation Monopoly"

Parts of this paper were written the author, including major parts of Sections 2 and 3. The market mechanisms were programmed into the simulation model and the simulations were executed by the author. The author produced the figures in this paper and the idea of using the simulation model reported in "Simulation of a Large Scale Dynamic Pickup and Delivery Problem" to study market mechanims was originally

suggested by the author. This paper is also under review for publication in Economics of Transportation.

Publication V: "Passenger-Pooling and Trip-Combining Potential of High-Density Demand Responsive Transport"

The author programmed and executed the simulations reported in this paper.

Publication VI: "Dynamic journeying under uncertainty"

The author was the main author of this article.

Publication VII: "Dynamic journeying in scheduled networks"

The author was the main author of this article.

Publication VIII: "A Maximum Cluster Algorithm for Checking the Feasibility of Dial-A-Ride Instances"

The author was the main author of this article.

Publication IX: "Routing by Ranking: A Link Analysis Method for the Constrained Dial-A-Ride Problem"

The author was the main author of this article.

1. Introduction

1.1 Demand-responsive transport today...

Demand-Responsive Transport (DRT) is often referred to as a form of public transport between bus and taxi involving flexible routing and scheduling of small or medium sized vehicles. This means that the vehicle routes are updated daily or in real time by incorporating information on the demand for transportation. Usually, the customers of a DRT service are required to request and book their trips in advance by placing trip requests including information on the origin and destination of the trip as well as the desired pick-up or drop-off time. The vehicle operator uses this information to provide service in a way that the passenger needs are satisfied.

DRT systems are typically used to provide transportation in areas with low transportation demand, where a regular bus service might not be as efficient. Another common application of DRT arises in door-to-door transportation of elderly or handicapped people (paratransit). DRT services are often fully or partially funded by local authorities, as providers of socially necessary transport. Most services that are provided by private companies for commercial reasons are related to transporting passengers between airports and urban areas.

The implementation of demand-responsive transport is strongly dependent on the target group or the business concept of the service. In some services, the vehicle routes are built freely according to customer requests, whereas other services make use of so-called skeleton routes and schedules, that are varied as required. Some DRT systems make use of terminals, at one or both ends of a route, such as an urban center or airport. In these applications (one-to-many or many-to-one), customers may specify either the origin or destination of the desired trip. Some systems provide door-to-door service within a certain service area and others provide service between a set of specified stops. For example, a DRT service operating in Nurmijärvi (Finland) aims to improve the level and accessibility of services in a sparsely populated area

and to reduce the costs of public transport. The service operates on a "many-to-many" basis, that is, there are no predefined routes. The stop points are located at a maximum of 900 m from origins and destinations. In the case of special users, the stop points are non-predefined (door-to-door service).

The popularity of demand-responsive transport has recently grown mainly due to the shortcomings of conventional bus and taxi services and new technical developments. In addition, flexible public transport services provided by local authorities and bus operators in partnerships with employers, stores and leisure centres are thought to help to break down social exclusion [18]. However, current DRT services have often been criticised because of their relatively high cost of provision, their lack of flexibility in route planning and their inability to manage high demand [17].

At the present moment, a large number of demand-responsive transport services are in operation. Most of such services operate within relatively small neighbourhoods and during low-use daytime hours, when there is not enough demand for traditional public transport. Thus, while the current services meet their current needs, demand-responsive transport remains a relatively small business compared to traditional transportation services, not to speak of private cars.

What if a DRT system was implemented in large scale, in a way that service could be provided for an entire metropolitan area?

1.2 . . .and tomorrow

Several new ideas and concepts related to demand-responsive transport services operating in urban areas have been recently presented, see for example [9]. These ideas are often motivated by problems arising from the congestion of urban areas caused by the increasing number of private cars. Thus, one of the main present goals of planning demand-responsive transport is seen to be the developing of *functional public transport services able to compete with private car and taxicab traffic*.

The popularity of the private car as a means of transport is partly based on a direct connection between the origin and destination of a trip and a short total travel time. In order to compete with private cars, public transport should thus offer connections as direct and fast as possible, in which the walking and waiting time are minimized. Another major advantage of the private car is seen to be the availability of the car at any time, even without planning beforehand. The study of large scale demand-responsive transport has therefore been directed towards highly dynamic services, which allow customers to request service not long before they are willing to depart. In addition, a demand-responsive public transport system should be able to offer an alternative for transportation without the inconvenience related to conventional public

transport.

The total travel time in public transport consists of *walking time* from origin to pick-up point, *waiting time* at the pick-up point, *ride time*, that is, the time spent in the vehicle, possible *transfer time* and walking time from drop-off point to destination. In order to attract people with private cars, it is necessary that the waiting and riding times are within acceptable bounds. In addition, it can be suggested that the service should be a near door-to-door service and the amount of transfers between vehicles should be minimal.

Intuitively, the idea of a large scale DRT system seems promising. With state-of-the-art engineering, there should be no insuperable technical hindrances in implementing such a service.

1.2.1 Opportunities

The fact that demand-responsive transport is "there for you when you want and where you want" is thought to be a major advantage compared to conventional public transport. While it may not be feasible to think that DRT could provide a level of service substantially better than that offered by taxi cabs, a system that could combine customers' trips efficiently could be more cost-efficient than a conventional taxi organization. This would make it possible to provide more inexpensive service without compromising too much on the level of service experienced by customers.

Compared to private cars, demand-responsive transport is thought to have several advantages in urban areas. For a model of a hypothetical large-scale demand-responsive public transport system for the Helsinki metropolitan area, simulation results published in 2005 demonstrated that "in an urban area with one million inhabitants, trip aggregation could reduce the health, environmental, and other detrimental impacts of car traffic typically by 50 - 70%, and if implemented could attract about half of the car passengers, and within a broad operational range would require no public subsidies" [27]. In addition to providing affordable transportation without the additional expenses related to maintenance, taxes and parking fees, demand-responsive transport could eliminate many other, possibly concealed, concerns related to private cars, including the difficulty of finding parking space and the stress related to driving in hazardous conditions or traffic jams.

At this point, one might ask: If the large scale demand-responsive transport system is superior compared to the alternatives, why has it not been implemented in practice?

1.2.2 Possible issues

While it is clear is that implementing a large-scale demand-responsive transport system would require significant investments, it is not clear whether there would be enough demand for such a service were it implemented.

For example, it might not be realistic nor beneficial from the social point of view to think that a conventional heavy rail system was replaced by demand-responsive minibuses, due to the high efficiency of heavy rail. Moreover, traditional public transport in general has many significant advantages compared to demand-responsive transport: Taking into account the current experience from DRT services, a major issue can be seen to be the reliability of the service. So far, estimating ride times accurately in a service with no fixed routes has proven to be somewhat insuperable, not least because of the human drivers, who are required to follow routes that are constantly changing, and the differences in their driving styles. Another disadvantage of DRT arises when customers are required to book their trips in advance and thus commit themselves to the service or payment at the time the trip is booked. In traditional bus services, this problem does not exist since customers may adjust their personal schedules dynamically according to known timetables, without pre-commitments. A commitment to a trip can be even more binding than in a taxi service: A normal taxi can wait for some time for the customer, for example, if the customer is at home when the taxi arrives, but it might not be reasonable that a demand-responsive minibus with customers on board would wait many minutes for one customer with the expense of other customers.

While demand-responsive transport has many rewards compared to the private car as argued above, the car has many characteristics that are hard to compensate with public services. Firstly, a person who has already invested in a car and thus settles yearly taxes and maintenance costs, is often not willing to use other transportation services since it would cost more than the marginal cost of using the car. Secondly, the car is unbeatable in many cases when speaking of flexibility: It is available at any time of the day, even without planning beforehand. Even if a demand-responsive transport service accepted immediate requests without a minimum pre-order time, the customer would still be committed to wait for the designated vehicle to arrive. Thirdly, the private car is thought to be the most convenient way of carrying large amounts of luggage and goods. The car is also often used for storing equipment, which is not likely to be possible in a public service.

Despite the above threats related to large scale demand-responsive transport, the concept should be studied carefully. Even if the private car has its advantages in the current state of the world, it may become practically useless in congested urban areas.

1.3 Problem statement

This work is focused on the discrete and combinatorial problems arising in the planning of public transport in general and demand-responsive transport in particular. The main goals are (i) to develop models for a priori studying different forms of transportation services without having to implement them in practice and (ii) to develop algorithms for solving combinatorial problems related to public transport.

The following chapters present three approaches to the demand and supply of public transport:

In Chapter 2: *Vehicle routing*, the demand for transportation is assumed to be known at the individual level. Using a fleet of vehicles, what is the best way to satisfy the demand?

In Chapter 3: *Journey planning*, the schedules and routes of transport services are assumed to be known during a specific time horizon. Using the scheduled transport services, what is the best way for a commuter to travel from a given origin to a given destination?

In Chapter 4: *Economic models*, we define the demand model by assuming that customers seek trips with small travel times and the supply model by assuming that transport service providers aim to maximize profit. Given these models, where does the demand for transportation meet the supply? How do regulation policies affect the economic equilibrium?

2. Vehicle Routing

A vast majority of theoretical studies related to demand-responsive transport are formalized as combinatorial optimization problems involving the construction of vehicle routes with respect to a set of customers, whose pick-up and drop-off points are known a priori [26]. This problem is often referred to as the *dial-a-ride problem*. A large scale system operating in real time induces new challenges: In order to be able to compete with private cars, service should be available within a short period of time from the trip request. This calls for adaptive routing algorithms, since the modifications in vehicle routes have to be executed in real-time. In order to ensure a sufficient level of service, the customers' waiting and ride times should be relatively limited. The vehicle dispatching algorithms should be designed in a way that the constrained nature of the problem is taken into account.

In Section 2.1, we study the *single-vehicle dial-a-ride problem* which involves the construction of an optimal single vehicle route serving a set of customers. This problem is motivated by the fact that solutions to the single-vehicle problem can be used as subroutines in environments with multiple vehicles [20, 21]. The extension from the single-vehicle problem to the multi-vehicle case is dicussed in Section 2.3.

2.1 Single-vehicle dial-a-ride problem

Different types of dial-a-ride services give rise to different types of mechanisms for controlling vehicle operations. For example, if a dial-a-ride service requires customers to request service during the previous day, the nature of vehicle dispatching will certainly differ from a service in which customers may request immediate service. In this section, a specific version of the dial-a-ride problem (DARP) is examined, namely, the *single-vehicle DARP with time windows*, in which the goal is to determine the optimal route for a single vehicle serving a certain set of customers.

The consideration of narrow time windows means that the vehicle route is restricted by relatively strict time limits for pick-up and delivery of each customer. Narrow

time windows emerge in real-time dial-a-ride services, in which each customer is given an estimate or guarantee regarding the pick-up and delivery times in the form of time windows. These time windows are examined as hard limits to be met by the vehicle. Time windows have been incorporated in many early and recent studies of the DARP, see for example references [21, 15, 16, 25, 4, 11, 28, 3]. In these studies it is noted that in dynamic settings, time windows eliminate the possibility of indefinite deferment of customers and strict time limits help provide reliable service. While the main focus is on fixed time windows, maximum ride time constraints that limit the time spent by passengers on the vehicle between their pick-up and their delivery [13], are considered as well.

In [20], the objective function is defined as a generalization of the objective function of the Traveling Salesman Problem (TSP), in which a weighted combination of the time needed to serve all customers and of the total degree of dissatisfaction they experience until their arrival to the destination of the trip is minimized. The dissatisfaction of customers is assumed to be a linear function of the time each customer waits to be picked up and the time each customer spends riding in the vehicle until his/her delivery. In [21], the approach is extended to handle time windows on departure and arrival times, but only the route duration is minimized.

In Publication I, both aspects of the problem (general objective function and time windows) are considered. The main contribution is a solution method designed in a way that i) it is capable of handling practically any objective function suitable for dynamic routing and ii) the computational effort of the algorithm can be controlled smoothly: If the problem size is reasonable, the algorithm produces optimal solutions efficiently and as the problem size is increased, the search space may be narrowed in order to achieve locally optimal solutions.

Publications VIII and IX discuss a single-vehicle algorithm based on hyperlink-induced topic search [**?**] for maximizing the number of customers in a single vehicle route. The method is seen to be useful in determining the feasibility of multi-vehicle instances.

Literature review

In the following paragraphs, a short review on the studies related to the single vehicle dial-a-ride problem is presented, based on [5, 7, 6] and [1], to which the reader is referred for more exhaustive summaries.

The first exact approach to the solution of the dynamic dial-a-ride problem was presented by [20]. In this work the single-vehicle, "immediate-request" case is studied, in which a set of customers should be served as soon as possible. In this first model no time windows are specified by the customers, but the vehicle operator incorpo-

rates *maximum position shift* constraints limiting the difference between the position of a request in the chronological calling list and its position in the vehicle route. The objective is to minimize a weighted combination of the time needed to serve all customers and customer dissatisfaction. The complexity of this algorithm is $O(n^2 3^n)$ and only small instances can thus be solved. However, the computational effort is decreased as more strict constraints are imposed.

The algorithm is first constructed for the static case of the problem and then extended to solve the dynamic case, in which new requests occur during the execution of the route but no information on future requests is available. In this version of the problem, the use of maximum position shift constraints is essential, in order to prevent a request from being indefinitely deferred. In a later study [21], the approach is extended to handle time windows on departure and arrival times. In this extension, only the route duration is considered in the objective function.

In [23, 24], a heuristic approach to the single vehicle DARP with one-sided time windows was introduced. In this problem formulation, the objective function is defined as a function of (i) the difference between the actual travel time and the direct travel time of a user and (ii) the difference between desired drop-off time and actual drop-off time. The algorithm was tested on real-life problems involving 7 to 20 customers.

In [10], an exact dynamic programming algorithm for the single vehicle DARP, formulated as an integer program, was presented. The formulation includes time windows as well as vehicle capacity and precedence constraints. Optimal solutions with respect to total route length were obtained for $n = 40$.

In [2], exact and heuristic procedures for the traveling salesman problem with precedence constraints are introduced, based on a bounding procedure. Computational results of the algorithm are given for a number of randomly generated test problems, including the dial-a-ride problem with the classical TSP objective function.

Psaraftis noted that *although single-vehicle Dial-A-Ride systems do not exist in practice, single-vehicle Dial-A-Ride algorithms can be used as subroutines in large scale multivehicle Dial-A-Ride environments. It is mainly for this reason that one's ability to solve the single-vehicle DARP is considered important.* A similar approach is used in this work, where the idea is to use the solution of the single-vehicle DARP as a subroutine in a dynamic multiple-vehicle scenario. An effort is made to solve the single vehicle problem up to optimality whenever the problem size is reasonable.

2.1.1 Problem formulation [20, 21]

Let there be N customers, each of which have been assigned a number i between 1 and N. For each customer $i \in \{1, \ldots, N\}$, let u_i be the pick-up point and u_{N+i} be the delivery point, $[e_i, l_i]$ be the pick-up time window and $[e_{N+i}, l_{N+i}]$ be the delivery time window[1], q_i be the load (number of passengers) associated to customer i. Let A be the starting point of the vehicle (location of the vehicle at $t = 0$). It is assumed that the time to go from any one of the pick-up and delivery points u_i, where $i \in \{1, \ldots, 2N\}$, directly to another point u_j is a known and fixed quantity $t(i, j)$.

The goal is to find a vehicle route starting from A and ending at one of the delivery points so that the following conditions hold.

1. The quantity
$$w_1 T + w_2 \sum_{i=1}^{N} (\alpha T_i^W + (1 - \alpha) T_i^R) \tag{2.1}$$
is minimized, where $w_1, w_2 \in \mathbb{R}$ are given weight parameters, the parameter α is the customers' time preference constant ($0 \leq \alpha \leq 1$), T is the duration of the route, T_i^W is the waiting time of customer i, from the earliest pick-up time until the time of pick-up, T_i^R is the riding time of customer i, from the time of pick-up until the time of delivery.

2. The vehicle route should be legitimate, namely each customer should be picked up before he/she is delivered.

3. The vehicle has a certain capacity of C passengers that cannot be exceeded.

4. All time constraints must be satisfied.

In equation (2.1), the quantity $\sum_{i=1}^{N} (\alpha T_i^W + (1 - \alpha) T_i^R)$ is the assumed form of the total degree of dissatisfaction experienced by the customers until their delivery, where α is the customers' time preference constant describing the impact of ride time and waiting time on the degree of dissatisfaction.

In reference [19], it is noted that in static and deterministic transportation models, finding an appropriate objective function is fairly easy and that the objective function is usually a good measure for evaluating the solution. In dynamic models, the objective function used to find the solution over a rolling horizon has often little to do

[1] In addition to the fixed time windows $[e_i, l_i]$ and $[e_{N+i}, l_{N+i}]$, to each customer may be associated a specific maximum ride time constraint $T_i^{\max}$. The handling of these constraints is discussed in section **??**.

with the measures developed to evaluate the overall quality of a solution. In stochastic models it might be useful to minimize the probability of violating time windows. However, the advanced insertion method to be described is designed in a way that several objective functions, that are thought to be suitable for dynamic and stochastic problems, may be incorporated with minimal work. Generally, the choice of the objective function depends on the particular version of the problem and is discussed only briefly in this document. For a study related to choosing an appropriate objetive function for the dynamic DARP, the reader is referred to [14].

The feasibility of time windows is governed by the following two assumptions.

1. If the vehicle arrives at any node (either pick-up or delivery) later than the upper bound of that node's time constraint (l_j), then the entire vehicle route and schedule is infeasible. In other words, those upper bounds constitute hard constraints that should be met by the vehicle.

2. If the vehicle arrives at any node earlier than the lower bound on that node's time constraint (e_j), then the vehicle will stay idle at that point and depart immediately at e_j.

2.1.2 Adaptive insertion algorithm [Publication I]

In general, exact procedures for solving routing problems are computationally very taxing, since the complexity is always more or less equal to the classical traveling salesman problem. In addition, exact optimization can be seen to be needless at runtime if routes are modified often. Despite these facts, exact algorithms are useful in a way that the performance of different heuristics may be compared to the optimal solution.

The main idea in the algorithm presented in Publication I is that customers are added to the vehicle route one by one by using an *exhaustive insertion* method, which leads to a globally optimal solution, that is a vehicle route which is feasible with respect to all customers and minimizes a given cost function.

Many studies related to the dial-a-ride problem, see for example [15, 16, 11, 28], make use of what is called the insertion procedure, in the classical version of which the pick-up and delivery node of a new customer are inserted into the *current optimal sequence* of pick-up and delivery nodes of existing customers. In these references, several improvements to the insertion algorithm have been suggested. However, the main idea of the classical insertion algorithm a priori excludes the possibility that the appearance of a new customer may render the optimal sequencing of already existing

customers no longer optimal. While the basic insertion algorithm can be seen to produce relatively good results for unconstrained problems, the performance compared to an exact algorithm is decreased as the problem becomes more constrained.

The main idea is to construct the optimal route iteratively by implementing an insertion algorithm for each customer, one by one *for all feasible sequences of pick-up and delivery nodes of existing customers*. Namely, the procedure involves two steps for each customer:

1. Perform insertion of the new customer to all feasible service sequences with respect to existing customers.

2. Determine the set of feasible service sequences with respect to the new customer and existing customers.

It can be readily shown that the insertion of a new customer to all feasible service sequences with respect to existing customers produces all feasible service sequences with respect to the union of existing customers and the new customer and leads to a globally optimal solution but is computationally expensive if the number of feasible service sequences grows large. However, if the route is constructed under relatively narrow time window constraints, the number of feasible routes with respect to all customers will be small compared to the number of all legitimate routes. Furthermore, the algorithm may be easily extended to an adjustable heuristic algorithm able to handle any types of time windows, see Section **??**.

The idea of the advanced insertion method is clarified by the following example, where no capacity or time constraints are taken into account. Let $i^{\uparrow} = i$ denote the *pick-up node* of customer i and let $i^{\downarrow} = N + i$ denote the *delivery node* of customer i. A *service sequence* is defined as an ordered list consisting of pick-up and delivery nodes. For instance, the service sequence $(i^{\uparrow}, j^{\uparrow}, j^{\downarrow}, i^{\downarrow})$ indicates the order in which customers i and j are picked up and dropped off.

Let us start the advanced insertion process with customer 1. Since the pick-up $1^{\uparrow}$ of customer 1 has to be before the delivery, $1^{\downarrow}$, the only possible service sequence at this point is $(1^{\uparrow}, 1^{\downarrow})$. Thus, the set of potential service sequences with respect to customer 1 consists of this single service sequence. By insertion of customer 2 into the service sequence $(1^{\uparrow}, 1^{\downarrow})$ we get the six service sequences presented in table 2.1.

By inserting the pick-up and delivery node of customer 3 into all of these service sequences we get a total of $6(5 + 4 + 3 + 2 + 1) = 90$ new potential service sequences. However, if the time and capacity constraints are taken into account, not all service sequences described above are necessarily feasible.

Table 2.1. Potential service sequences with respect to customers 1 and 2. No capacity or time constraints are taken into account. $i^{\uparrow}$ denotes the pick-up node and $i^{\downarrow}$ denotes the delivery node of customer i.

A: $1^{\uparrow}$ $1^{\downarrow}$ $2^{\uparrow}$ $2^{\downarrow}$	B: $1^{\uparrow}$ $2^{\uparrow}$ $1^{\downarrow}$ $2^{\downarrow}$	C: $1^{\uparrow}$ $2^{\uparrow}$ $2^{\downarrow}$ $1^{\downarrow}$
D: $2^{\uparrow}$ $1^{\uparrow}$ $1^{\downarrow}$ $2^{\downarrow}$	E: $2^{\uparrow}$ $1^{\uparrow}$ $2^{\downarrow}$ $1^{\downarrow}$	F: $2^{\uparrow}$ $2^{\downarrow}$ $1^{\uparrow}$ $1^{\downarrow}$

For example, if after the insertion of customer 2 it can be seen that only the service sequences A and B in table 2.1, namely $(1^{\uparrow}, 1^{\downarrow}, 2^{\uparrow}, 2^{\downarrow})$ and $(1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow})$, are feasible with respect to time constraints, then it can be shown that all feasible service sequences including customer 3 are included in the sequences given by insertion to these two service sequences, which gives a maximum of $2(5 + 4 + 3 + 2 + 1) = 30$ new potential service sequences.

Briefly, the structure of the advanced insertion algorithm may be sketched as follows. At first, the set of feasible service sequences S_i is determined recursively for each customer $i \in \{1, \dots, N\}$ by inserting the pick-up and delivery nodes of customer i into each feasible service sequence with respect to customers $1, \dots, i-1$. In this way the algorithm produces the set S_N of all feasible routes with respect to customers $1, \dots, N$. Then the solution to the static problem is obtained by choosing the sequence $s \in S_N$ with minimal cost $C(s)$. In the general form (2.1) of the cost function, this involves the calculation of waiting times and ride times for all customers for all feasible service sequences. If only route duration is minimized, we can simply choose the sequence for which the arrival time at the last node is smallest.

A priori clustering

In problems involving a large number of customers, assuming that the time windows are relatively narrow, a significant portion of service sequences can be eliminated before the actual insertion process by simply studying the mutual relationships between nodes, similarly as described in [12]. More precisely, assume that the vehicle departs from a node i at the lower bound e_i of the time window and moves directly to node j. If the vehicle does make it in time to j, that is, if

$$e_i + t(i, j) > l_j, \tag{2.2}$$

it is said that the transition $i \to j$ is *a priori infeasible*. A priori infeasibility could be defined similarly for capacity constraints. However, assuming that the load associated to each customer is at most $C/2$, where C is the capacity of the vehicle, each transition is a priori feasible with respect to capacity. In other words, two customers i and j with loads satisfying $q_i \leq C/2$ and $q_j \leq C/2$ may be picked up and dropped of in any order without the capacity constraint being violated.

Note that if for any two nodes i and j, both transitions $i \to j$ and $j \to i$ are infeasible a priori, the entire problem is infeasible. Otherwise, either $i \to j$ or $j \to i$ or both are a priori feasible and the pick-up and delivery nodes of customers can be divided among m preceding clusters by using the following rule.

Definition. *Cluster C_k precedes cluster C_l if and only if the transition $x \to y$ is a priori infeasible for all $x \in C_l$ and $y \in C_k$. In this case, we shall use the notation $C_k \prec C_l$.*

Clearly, assuming that there exists a feasible solution, the above precedence relation of clusters is a strict order: (i) $C \not\prec C$ for all clusters, (ii) $C_a \prec C_b$ implies $C_b \not\prec C_a$ and (iii) $C_b \prec C_c$ and $C_a \prec C_b$ implies $C_a \prec C_c$. In addition, note that if a single node i has an infinite time window $[-\infty, +\infty]$, there is only one cluster since all transitions $j \to i$ and $i \to j$ are feasible a priori.

In practice, the clusters can be determined by means of an adjacency matrix F for which $F_{ij} = 1$ if $i \to j$ is feasible and $F_{ij} = 0$ if $i \to j$ is infeasible a priori. By arranging the rows and columns suitably we get a block upper triangular matrix where each block corresponds to a cluster. Figure 2.1 shows the adjacency matrix of a sample problem involving four customers. The rows and columns are arranged in a descending order with respect to row sums. From the matrix five clusters can be identified (blocks on the diagonal), namely $\{1^\uparrow\} \prec \{1^\downarrow, 3^\uparrow, 2^\uparrow\} \prec \{2^\downarrow\} \prec \{3^\downarrow, 4^\uparrow\} \prec \{4^\downarrow\}$. Thus, since the positions of nodes $1^\uparrow, 2^\downarrow$ and $4^\downarrow$ are fixed, the problem falls down to determining the optimal ordering of nodes $1^\downarrow, 3^\uparrow, 2^\uparrow$ and $3^\downarrow, 4^\uparrow$. In general, by

	$1^\uparrow$	$1^\downarrow$	$3^\uparrow$	$2^\uparrow$	$2^\downarrow$	$3^\downarrow$	$4^\uparrow$	$4^\downarrow$
$1^\uparrow$	1	1	1	1	1	1	1	1
$1^\downarrow$	0	1	1	1	1	1	1	1
$3^\uparrow$	0	1	1	1	1	1	1	1
$2^\uparrow$	0	1	1	1	1	1	1	1
$2^\downarrow$	0	0	0	0	1	1	1	1
$3^\downarrow$	0	0	0	0	0	1	1	1
$4^\uparrow$	0	0	0	0	0	1	1	1
$4^\downarrow$	0	0	0	0	0	0	0	1

Figure 2.1. A priori adjacency matrix of a sample problem involving 4 customers. From the matrix five clusters can be identified, namely $\{1^\uparrow\} \prec \{1^\downarrow, 3^\uparrow, 2^\uparrow\} \prec \{2^\downarrow\} \prec \{3^\downarrow, 4^\uparrow\} \prec \{4^\downarrow\}$.

clustering the nodes a priori, a significant amount of insertions need not be checked for feasibility.

Structure and complexity

The number of insertions that are tested for feasibility for customer i is bounded above by the formula

$$n(i) \leq m_{i-1} \sum_{x=1}^{2i-1} x = \frac{m_{i-1}(2i)(2i-1)}{2}, \tag{2.3}$$

where m_{i-1} is the number of feasible service sequences with respect to customers $1, \ldots, i-1$. The inequality can be justified by the fact that not all possible insertions are checked for feasibility. In the worst case, where capacity and time constraints are *not restrictive*, we get

$$m_1 = 1, \qquad m_2 = (4 \cdot 3)/2 = 6, \qquad m_3 = 6(6 \cdot 5)/2 = 90,$$
$$m_i = \frac{(2i)(2i-1)}{2} \frac{(2(i-1))(2(i-2)-1)}{2} \cdots \frac{4 \cdot 3}{2} = \frac{(2i)!}{2^i}.$$

Thus the maximum number of insertions required in the solution of the static case is $\sum_{i=1}^{N} \frac{(2i)!}{2^i} \approx \sum_{i=1}^{N} 2^{1-i} i^{2i+\frac{1}{2}} \sqrt{\pi}$. Thus, in the worst case, the number of feasible solutions is of order $O(\sqrt{N}(N^2/2)^N)$.

Suppose that, due to strict time (and capacity) constraints, the number of potential service sequences m_i for customers $1, \ldots, i$ is bounded by some function $m : \mathbb{N} \to \mathbb{N}$ such that $m(i) \leq \frac{(2i)!}{2^i}$. Then the number of insertions for customer i is bounded by $n(i) \leq \frac{m(i-1)(2i)(2i-1)}{2}$. The total number of insertions is bounded by $\sum_{i=1}^{N} \frac{m(i-1)(2i)(2i-1)}{2}$. For example, if $m(i) = kN$ for some constant k, the computational complexity of the screening phase is reduced to $O(N^4)$. If $m(i) = k$, the computational complexity is reduced to $O(N^3)$.

On the grounds of previous calculations, it can be stated that the adaptive insertion algorithm will generally not be able to produce exact solutions efficiently in cases where the capacity and time constraints are not restrictive. However, if the number of feasible service sequences is bounded due to strict constraints, the algorithm will lead to an exact solution computationally inexpensevely. In addition, the advanced insertion algorithm has a special property of being extendable to an adjustable heuristic, as described in the following subsection.

A heuristic extension

Even if the capacity and time constraints were not highly restrictive, the algorithm can be modified easily by bounding the size of the set S_i of service sequences, in which new customers are inserted, by including only a maximum of L service sequences for each customer i. More precisely, if after inserting customer i, the number of feasible service sequences with respect to customers $1, \ldots, i$ is larger than L, the set of feasible service sequences S_i with respect to customers $1, \ldots, i$ is narrowed by including only L service sequences, that seem to allow the insertion of remaining customers (see

part 2.1.2). After the last customer has been inserted, the feasible service sequences are evaluated by means of the objective function (2.1).

This modification leads to a heuristic algorithm, in which the computational effort can be controlled by the parameter L, referred to as the *degree* of the heuristic. The resulting algorithm is somewhat sophisticated in a way that it produces globally optimal solutions for small sets of customers and when the number of customers is increased, the algorithm still produces locally optimal solutions with reasonable computational effort. In the special case where $L = 1$, the algorithm reduces to the classical insertion algorithm. If $L \geq \frac{(2N)!}{2^N}$, the heuristic coincides with the exact version of the algorithm as no routes are discarded.

Objective functions

In order to be able to efficiently make use of the above heuristic extension idea, the set of service sequences is narrowed by means of a certain heuristic objective function after the insertion of each customer. Since the main purpose of heuristics at the operational level is to always produce some implementable solutions very quickly, even if they were only locally optimal, such an objective function should be defined in a way that the algorithm is capable of producing *feasible* solutions even if the complexity of the problem was high.

Looking only at the cost defined in formula (2.1) may eliminate from consideration sequences that are marginally costlier but would easily allow the insertion of remaining customers in the route. Thus, more sophisticated criteria should be considered to help ensure that the heuristic will find a feasible solution when one exists.

In other words, the function should favor service sequences with enough *time slack* for those customers, that have not been inserted into the sequences. Publication I suggests the following heuristic objectives. Given the service sequence $s = (p_1, \ldots, p_m)$, we wish to optimize one of the functions

$$f_{rl}(s) = t_m, \qquad \text{(Route duration)} \tag{2.4}$$

$$f_{ts}(s) = \sum_{j=1}^{m} l_j - t_j, \qquad \text{(Total time slack)} \tag{2.5}$$

$$f_{min}(s) = \min_{j \in \{1,\ldots,m\}} l_j - t_j, \qquad \text{(Max-min time slack)} \tag{2.6}$$

where $[e_j, l_j]$ is the time window and t_j is the calculated time of arrival at node p_j.

In general, each of the above objective functions aim to maximize the temporal flexibility of service sequences in different ways.

Route duration (2.4) favors service sequences in which the time to serve all customers is as small as possible. This objective can be justified by the fact that it is likely that new customers may be inserted at the rear of a route that is

executed quickly.

Total time slack (2.5) stores sequences in which the sum of excess times (or the average excess time) at the nodes is maximized, that is, sequences which are likely to allow the insertion of a new customer before the last node.

Max-min (2.6) seeks sequences in which the minimum excess time at the nodes of the route is maximized. In other words, the sequences in which there is at least some time slack at each node are considered potential.

A simple example motivating the use of the above objective functions is presented in figure 2.2.

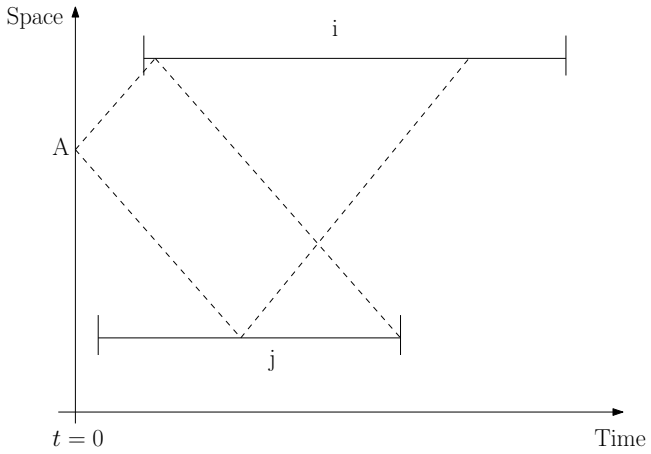


Figure 2.2. Route flexibility. A vehicle is located at A at $t = 0$, and two customers are due to be picked up within the presented time windows at i and j. The dashed lines represent two possible routes for the vehicle. If the route duration were minimized, i should be visited before j. However, since there would be no "slack time" at j, this decision would a priori exclude the possibility that new customers could be inserted between i and j. On the other hand, if j were visited before i, there would be more possibilities for inserting new customers on the route before i. However, the route $A \rightarrow i \rightarrow j$ is shorter and thus it is more likely that customers can be inserted at the end of the sequence.

Tuning

In order to ensure that the heuristic always produces a feasible solution when one exists, the parameter L can be tuned during run-time by using the following idea.

1. At first, the problem is solved by using an initial degree L_0.

2. Each time the algorithm is unable to find a feasible solution, the degree is increased and the problem is solved again.

Let us study the expected CPU time of the tuning method. Let $P(S|L = l)$ denote the probability that a solution is found by using the degree $L = l$ and let $T(l)$ denote

the average running time of the algorithm with $L = l$. The average CPU times of feasible runs are assumed equal to those occurring during infeasible runs, although infeasible runs are actually slightly less expensive. However, it can be seen that this approximation will not affect the results of the following examination. The expected CPU time for a given tuning strategy $(L_0, L_1, \ldots)$, where $L_0 < L_1 < L_2 < \ldots$, is given by the formula

$$T(L_0) + \sum_{i=1}^{\infty} T(L_i) \prod_{j=0}^{i-1} (1 - P(S|L = L_i)).$$

It should be noted that the above probabilities are actually conditional: If a solution is not found with some value of L, there is no point in solving the problem again with the same value. In addition, a relatively small increase in the value of L will not significantly affect the possibilities of finding a solution. Thus, the tuning approach used in the following examination is based on multiplying the value of L by a number K each time a feasible solution is not found. This method will be referred to as K-multiplying.

Assuming that the complexity grows linearly with l, that is $T(l) = lT(1)$, and that the probability of finding a solution is sufficiently high, it can be shown that the optimal initial degree for the K-multiplying method has to be relatively small. At first we will show that if $P(S|L \geq 1) > 1/2$, then any initial degree $L_0 = mK$, where $m \in \mathbb{N}$, is suboptimal.

Theorem 1. *Let $E(\tau(L_0))$ denote the expected CPU time of the K-multiplying method for the initial value L_0. If $T(l) = lT(1)$ for $l \in \mathbb{N}$ and $P(S|L \geq 1) > 1/2$, then*

$$E(\tau(L_0)) \leq E(\tau(KL_0))$$

for all $L_0 \in \mathbb{N}$ and $K \geq 2$.

Proof. Since it is clear that $E(\tau(l)) \geq T(l)$ for $l \in \mathbb{N}$, for any $K \geq 2$, we get

$$\begin{aligned} E(\tau(L_0)) &= T(L_0) + (1 - P(S|L = L_0))E(\tau(KL_0)) < T(L_0) + \frac{1}{2}E(\tau(KL_0)) \\ &= \frac{1}{k}T(KL_0) + \frac{1}{2}E(\tau(KL_0)) \leq (\frac{1}{K} + \frac{1}{2})E(\tau(KL_0)) \leq E(\tau(KL_0)). \end{aligned}$$

□

Then, the following theorem states that if the probability of finding a solution is at least $1 - 1/K$, the optimal initial value is bounded above by the logarithm of the smallest natural number l for which $P(S|L = l) = 1$.

Theorem 2. *Let l' denote the smallest natural number l for which $P(S|L = l) = 1$. If $P(S|L \geq 1) > 1 - 1/K$, then*

$$E(\tau(L_0)) < T(L_0(1 + \log_K l'/L_0))$$

for all $L_0 \in \mathbb{N}$ and for all $K > 1$.

Proof. The expected CPU time is given by the formula

$$E(\tau(L_0)) = \sum_{i=0}^{\infty}(1 - P(S|L = K^i L_0))^i T(K^i L_0).$$

Since $(1 - P(S|L = K^i L_0)) = 0$ for $K^i L_0 > l'$, that is, $i > \log_K l'/L_0$, we get

$$E(\tau(L_0)) = \sum_{i=0}^{\lfloor \log_K l'/L_0 \rfloor} (1 - P(S|L = K^i L_0))^i T(K^i L_0) \leq \sum_{i=0}^{\lfloor \log_K l'/L_0 \rfloor} (1 - (1 - 1/K))^i K^i T(L_0)$$
$$\leq T(L_0)(1 + \log_K l'/L_0).$$

□

In general, the two theorems imply that the expected CPU time is minimized when the initial degree L_0 is small. However, this is true only if the probability of finding a solution with small values of L is relatively high. For example, if $l' = 16$, $K = 2$ and $P(S|L \geq 1) > 0.5$, theorem 2 states that that $E(\tau(1)) < T(5)$. By theorem 1 we know that the values 2 and 4 are suboptimal for $K = 2$. Thus, the optimal initial value is either 1 or 3 depending on the actual probabilities of finding solutions with different values of L.

Finally, it should be noted that the optimal choice of K and L_0 will in general depend on the type of the problem to be solved and that by increasing the values, the optimality of the solution with respect to the objective of the problem is increased as well since more sequences are evaluated, as will be seen in section 2.2.

2.1.3 Maximum cluster algorithm [Publication VIII]

2.1.4 The dynamic case

Let us discuss the extension of the predescribed single-vehicle algorithms to the dynamic case, in which new customers are appended to the route in real time. Generally, it can be seen that there is no need to compute a new route except when a new customer request occurs, a customer does not show up at the agreed pick-up location, the vehicle is ahead or behind of schedule or the vehicle has lost its way.

A simple example, similar to the one presented in [20], on the real-time updating of a vehicle route is shown in figure 2.3. Initially, customers 1 and 2 have been assigned pick-up and delivery points and corresponding time windows. The vehicle located at A follows the tentative optimal route $(1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow})$, where "$\uparrow$"-symbols denote pick-up nodes and "$\downarrow$"-symbols denote delivery nodes. (see figure 2.3a). At the time the vehicle is at B, the vehicle route is updated with respect to customers $1, 2$ and 3. At this instant a new tentative optimal route shown in figure 2.3b is produced.

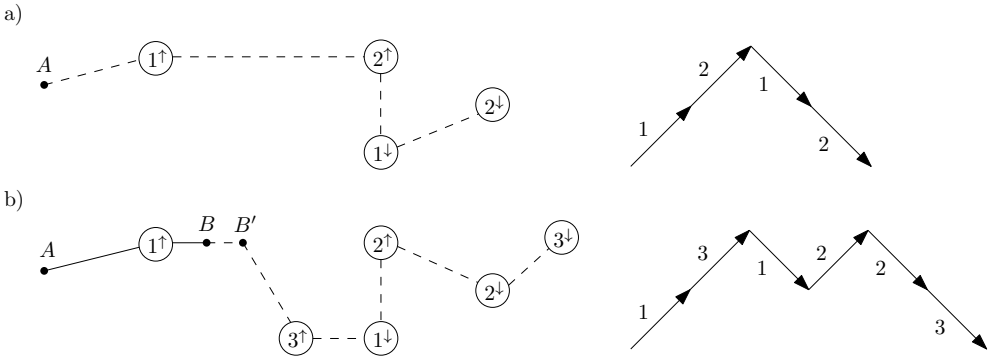


Figure 2.3. Modifications in the vehicle route. The route is updated when the vehicle is at B and a new tentative optimal route beginning at B' is produced. The figures on the right show the routes as so-called labeled Dyck paths [8], in which each pick-up $i^{\uparrow}$ precedes the correspoding drop-off $i^{\downarrow}$. At the time a new customer is added to the route, a new path is formed. Clearly, the "height" of the path shows the number of customers aboard in different parts of the route.

Note that the new route does not have B as origin, but a point B', slightly ahead of B. This is due to two facts: i) It will take some time for the algorithm to process the new input and reoptimize, ii) It will take some time for the driver of the vehicle to process the information regarding the new route. The distance between B and B' depends in general on the particular dial-a-ride service examined. For example, in a highly dynamic setting, in which the modification of the route is not allowed to take more than a few seconds, it may be assumed that the latter of the facts is more restrictive. Any dynamic dial-a-ride service should make use of a mechanism to update the point B' at sufficient time intervals. In this work, however, it is assumed that the point B' is known at each instant, as well as the estimated time of arrival T' at B'. The *vehicle checkpoint* (B', T') acts as the starting point for all service sequences.

For customers that have already been picked up, the pick-up point is not a part of the input of the problem. Thus, for such customers only the delivery point is considered.

In dynamic models, the objective function used to find the solution over a rolling horizon has often little to do with the measures developed to evaluate the overall quality of a solution [19]. However, the adaptive insertion method described in Publication I is designed in a way that several objective functions, that are thought to be suitable for the dynamic problem, may be incorporated with minimal work. For example, if the vehicle route is subject to several modifications in short periods of time, one of the flexibility measures (2.6), (2.5) or forward time slack defined in [22] may be used as an objective function for the problem in order to be able to insert as many future customers in the route as possible. Generally, the choice of the objective function depends on the particular version of the dynamic routing problem and the performance of different objective functions may be sensitive to, for example, constraints, demand intensity or the number of vehicles. For a study related to choosing

an objetive function for the immediate-request DARP, the reader is referred to [14].

2.2 Numerical experiments

The following paragraphs present a summary of the computational results reported in Publication I. The exact and heuristic versions of the algorithm were tested on a set of problems involving different numbers of customers and different time window widths determined by a travel time ratio R describing the maximum allowed ratio of travel time to direct ride time. The pick-up and drop-off points of customers were chosen randomly from a square-shaped service area and the ride times between the points were modeled by euclidean distances.

At first, the complexity of the problem was studied with respect to three parameters, namely i) the number of customers N, ii) travel time ratio R and iii) the average time interval μ between customer requests. The complexity of the exact algorithm was measured in terms of the average number of feasible sequences in *feasible problem instances*, that is, randomized problems for which at least one feasible solution was found. The number of feasible sequences gives us an insight on the complexity of the problem itself, since any enumeration algorithm would have to evaluate the same number of feasible sequences.

Then, the performance of the heuristic with different objective functions was evaluated. The complexity was measured in terms of the number of sequences evaluated by the heuristic.

2.2.1 Experiments

The following experiments were performed on a standard laptop computer with a 2.2 GHz processor. The CPU times and the number of evaluated sequences appeared to have a roughly linear relationship. A typical problem instance involving 20 customers could be solved up to optimality within less than a second.

Experiment 1: Number of customers

At first we study the complexity of the exact algorithm with respect to the number of customers. In practice, the number of customers assigned to a single vehicle is governed by the pre-order time of the dial-a-ride service: If customers may request service in advance, the planned vehicle routes are expected to be longer than in immediate-request services. Figure 2.4 shows the average number of feasible sequences in feasible problem instances on a logarithmic scale, computed over 10000 randomized instances for $N = 1 \ldots 20$, $R = 2, 2.5, 3$ and $\mu = 1800$s.

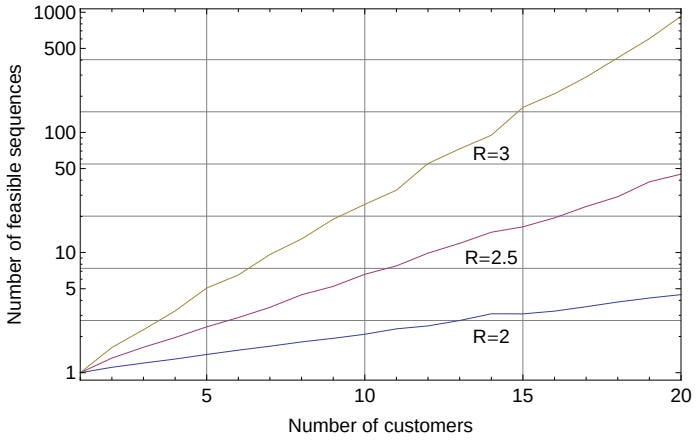


Figure 2.4. Experiment 1. The complexity of the exact algorithm with respect to the number of customers on a logarithmic scale. The three curves represent, as a function of the number of customers, the average number of feasible sequences for $R = 2, 2.5, 3$ and $\mu = 1800$s. Clearly, the complexity increases exponentially with respect to the number of customers.

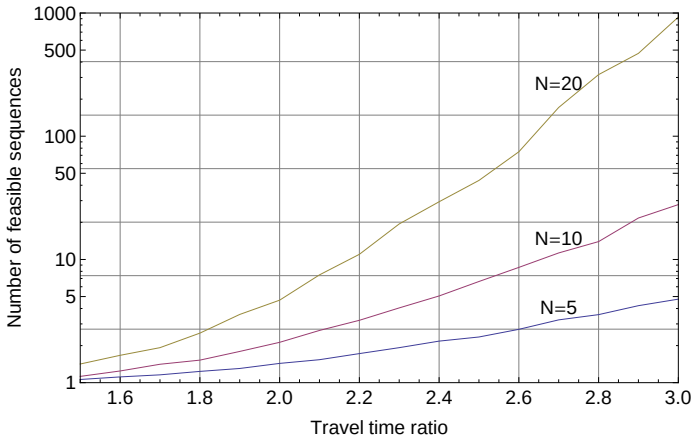


Figure 2.5. Experiment 2. The complexity of the exact algorithm with respect to travel time ratio on a logarithmic scale. The three curves represent, as a function of the travel time ratio, the average number of feasible sequences for $N = 5, 10, 20$ and $\mu = 1800$s.

Referring to the figure, it can be seen that the complexity of the problem increases exponentially with respect to the number of customers with all studied values of the travel time ratio. In addition, the complexity is increased with the travel time ratio.

Experiment 2: Travel time ratio

Let us study the complexity of the exact algorithm as a function of travel time ratio. Figure 2.5 shows the average number of feasible sequences in feasible problem instances on a logarithmic scale, computed over 10000 randomized instances for $R = 1.5 \ldots 3$, $N = 5, 10, 20$ and $\mu = 1800$s.

The figure shows that the effect of the travel time ratio on the complexity of the problem is significant. The fact that the slopes of the curves increase with R on the logarithmic scale indicates that the relation between complexity and R is superexpo-

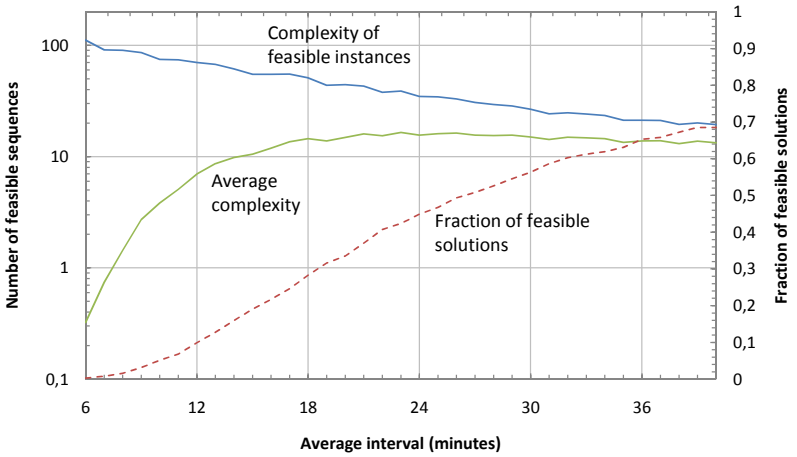


Figure 2.6. Experiment 3. The complexity of the exact algorithm with respect to the average time interval between customers. The solid lines represent the average number of feasible sequences in feasible problem instances and all problem instances, on a logarithmic scale for $N = 10$ and $R = 3$. The dashed line corresponds to the fraction of problem instances, for which at least one feasible solution was found.

nential.

Experiment 3: Time interval

Let us conclude the study of the exact algorithm by examining complexity with respect to the average time interval between customer requests. The solid lines in figure 2.6 represent the average number of feasible sequences in i) feasible problem instances and ii) all problem instances, on a logarithmic scale, computed over 10000 randomized instances for $N = 10$, $R = 3$ and $\mu = 6 \ldots 40$ minutes. The dashed line corresponds to the fraction of problem instances, for which at least one feasible solution was found.

The figure indicates that the complexity of feasible problem instances decreases exponentially with respect to the average time interval μ. On the other hand, the probability of finding at least one feasible solution is increased with μ. By looking at the curve corresponding to the average complexity of all problem instances (including infeasible cases), it can be seen that the complexity is maximized at a certain time interval ($\mu = 24$ minutes in this case), in which both the probability of finding a feasible solution and the number of feasible sequences in feasible cases are relatively large.

At this point, it should be emphasized that the above results apply to randomized instances for the single vehicle problem. In a dynamic multiple vehicle setting, an effort is made to divide the customers among available vehicles optimally. Thus, it is suggested that a larger number of customers can be served by a single vehicle in less time than in the case in which the customers' pick-up and drop-off locations

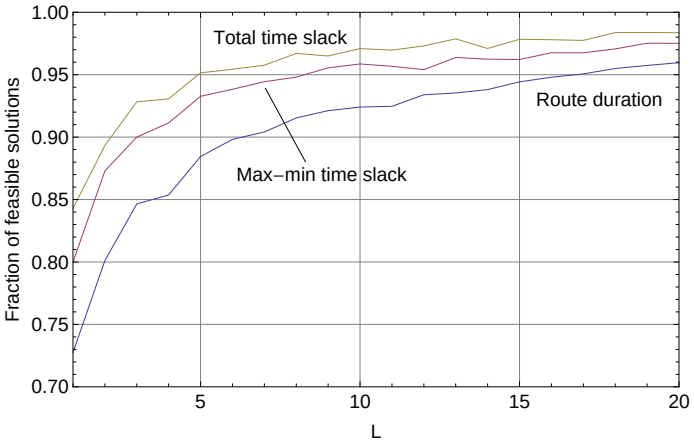


Figure 2.7. Experiment 4. The performance of three different heuristic objective functions as functions of degree L. The curves represent the fractions of instances for which a feasible solution was found by the objective functions (compared to the exact algorithm), for $N = 20$, $R = 3$ and $\mu = 1800$s. The total slack time objective function outperforms the other two in all studied cases.

are completely random. However, the above results give us an insight on how the complexity of the single vehicle subroutine behaves with respect to the average time interval between the earliest pick-up times of customers assigned to a single vehicle.

Experiment 4: Objective functions

Let us study the performance of the three different heuristic cost functions (2.4), (2.5) and (2.6) as a function of the degree L of the heuristic. Figure 2.7 shows the fraction of problem instances for which a feasible solution was found by the heuristic (compared to the exact algorithm), computed over 10000 randomized instances. The number of customers was set to $N = 20$, the travel time ratio was 3 and the average time interval $\mu = 1800$s was used.

Referring to the figure, it can be seen that the total time slack cost function (2.5) is capable of finding a feasible solution to randomized problems most often, while the performance of the route duration cost function (2.4) is worst of the three algorithms. Note that as the degree L is increased, the fraction of feasible solutions converges to 1 for any heuristic cost function, since whenever $L \geq \frac{(2N)!}{2^N}$, the heuristic coincides with the exact algorithm regardeless of the studied problem.

2.2.2 Conclusions

In this work, an exact optimization procedure is developed to solve the static and dynamic versions of the single vehicle dial-a-ride problem with time windows. Using complete enumeration to solve the problem with respect to a generalized objective function is motivated by the dynamic nature of online demand responsive transport

services, in which looking only at the tentative route duration, as in existing algorithms for the problem, may decrease the possibilities of serving future customers.

In addition, an adjustable heuristic extension to the algorithm is introduced, in order to be able to control the CPU times: If the problem size is reasonable, the proposed solution method produces globally optimal solutions. If the problem size is increased, the algorithm adjusts itself to produce locally optimal solutions, closing the gap between the classical insertion heuristic and the exact solution and thus making the algorithm applicable to any static or dynamic dial-a-ride problem.

2.3 Multi-vehicle dial-a-ride problem

3. Journey planning

4. Economic models

5. Conclusions

This work is focused on combinatorial problems arising from the planning of demand responsive transport.

As a final conclusion, it can be stated that there are many theoretical results that support the technical viability of large scale demand responsive transport. However, as suggested in [9], one of the key issues in a large scale demand responsive service ever becoming a reality, is the institutional inertia against change in transit paradigms. No models exist that are directly applicable in finding to what extent a completely new transportation system is possible in real life. How to model demand for a hypothetical transportation service remains a relatively open question. In order to be able to access such practical problems, future work calls for real-life pilot services, which would probably give valuable information on the demand for demand-responsive transport.

Bibliography

[1] G. Berbeglia, J-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202:8–15, 2010.

[2] L. Bianco, A. Mingozzi, S. Ricciardelli, and M. Spadoni. Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. *INFOR*, 32:19–31, 1994.

[3] J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586, 2006.

[4] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research B*, 37:579–594, 2003a.

[5] J.-F. Cordeau and G. Laporte. The dial-a-ride problem (darp): variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, 1:89–101, 2003b.

[6] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153:29–46, 2007.

[7] J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M.W.P. Savelsbergh. Transportation on demand. In *Transportation*, pages 429–466. Amsterdam: North-Holland, 2007.

[8] R. Cori. Indecomposable permutations, hypermaps and labeled dyck paths. *Journal of Combinatorial Theory, Series A*, 116(8):1326–1343, 2009.

[9] C. E. Cortés, M. Matamala, and C. Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200:711–724, 2010.

[10] J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6:301–325, 1986.

[11] M. Diana and M.M. Dessouky. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B*, 38:539–557, 2004.

[12] Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

[13] B. Hunsaker and M. W. P. Savelsbergh. Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters*, 30:169–173, 2002.

[14] E. Hyytiä, L. Häme, A. Penttinen, and R. Sulonen. Simulation of a large scale dynamic pickup and delivery problem. In *3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010)*. 2010.

[15] J.J. Jaw, A.R. Odoni, H.N. Psaraftis, and N.H.M. Wilson. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. *Transportation Research Part B*, 20:243–257, 1986.

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

[17] J. Mageean and J. D. Nelson. The evaluation of demand responsive transport services in europe. *Journal of Transport Geography*, 11(4):255–270, 2003.

[18] Department of the Environment Transport and the Regions. *Social exclusion and the provision and availability of public transport*. DETR, London, 2000.

[19] W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In *Network Routing*, volume 8, pages 141–295. Amsterdam: North-Holland, 1995.

[20] H.N. Psaraftis. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–154, 1980.

[21] H.N. Psaraftis. An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17:351–357, 1983.

[22] M.W.P. Savelsbergh. The vehicle routing problem with time windows: minimizing route duration. *ORSA Journal on Computing*, 4:146–154, 1992.

[23] T. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science*, 19:378–410, 1985a.

[24] T. Sexton and L. D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: Ii. routing. *Transportation Science*, 19:411–435, 1985b.

[25] P. Toth and D. Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31:60–71, 1997.

[26] P. Toth and D. Vigo. *The vehicle routing problem*. Philadelphia, PA, 2002.

[27] Jouni T Tuomisto and Marko Tainio. An economic way of reducing health, environmental, and other pressures of urban traffic: a decision analysis on trip aggregation. *BioMed Central, November 25*, 2005.

[28] K.I. Wong and M.G.H. Bell. Solution of the dial-a-ride problem with multi-dimensional capacity constraints. *International Transactions in Operational Research*, 13:195–208, 2006.

Publication I

Lauri Häme. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows. *European Journal of Operational Research*, 209, p. 11–22, February 2011.

European Journal of Operational Research 209 (2011) 11–22

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows

Lauri Häme *

Aalto University School of Science and Technology, PL 11000, 00076 Aalto, Finland

ARTICLE INFO

Article history:
Received 29 January 2010
Accepted 24 August 2010
Available online 27 August 2010

Keywords:
Transportation
Dial-a-ride problem
Exact algorithm
Heuristics

ABSTRACT

The dial-a-ride problem (DARP) is a widely studied theoretical challenge related to dispatching vehicles in demand-responsive transport services, in which customers contact a vehicle operator requesting to be carried from specified origins to specified destinations. An important subproblem arising in dynamic dial-a-ride services can be identified as the single-vehicle DARP, in which the goal is to determine the optimal route for a single vehicle with respect to a generalized objective function. The main result of this work is an adaptive insertion algorithm capable of producing optimal solutions for a time constrained version of this problem, which was first studied by Psaraftis in the early 1980s. The complexity of the algorithm is analyzed and evaluated by means of computational experiments, implying that a significant advantage of the proposed method can be identified as the possibility of controlling computational work smoothly, making the algorithm applicable to any problem size.

1. Introduction

Dial-a-ride involves the dispatching of a fleet of vehicles to satisfy demands from customers who contact a vehicle operating agency requesting to be carried from specified origins to other similarly specified destinations, as defined in Psaraftis (1980).

Different types of dial-a-ride services give rise to different types of mechanisms for controlling vehicle operations. For example, if a dial-a-ride service requires customers to request service during the previous day, the nature of vehicle dispatching will certainly differ from a service in which customers may request immediate service. The problem of dispatching vehicles in a dial-a-ride service is generally called the *dial-a-ride problem* (DARP). In this work, a specific version of this problem is examined, namely, the *single-vehicle DARP with time windows*, in which the goal is to determine the optimal route for a single vehicle serving a certain set of customers.

The consideration of narrow time windows means that the vehicle route is restricted by relatively strict time limits for pickup and delivery of each customer. Narrow time windows emerge in real-time dial-a-ride services, in which each customer is given an estimate or guarantee regarding the pickup and delivery times in the form of time windows. These time windows are examined as hard limits to be met by the vehicle. Time windows have been incorporated in many early and recent studies of the DARP (see for example Psaraftis, 1983; Jaw et al., 1986; Madsen et al., 1995; Toth and Vigo, 1997; Cordeau and Laporte, 2003a; Diana and Dessouky, 2004; Wong and Bell, 2006; Cordeau, 2006). In these studies it is noted that in dynamic settings, time windows eliminate the possibility of indefinite deferment of customers and strict time limits help provide reliable service. While the main focus of this work is on fixed time windows, maximum ride time constraints that limit the time spent by passengers on the vehicle between their pickup and their delivery (Hunsaker and Savelsbergh, 2002), are considered as well.

The problem formulation studied in this work follows closely the one presented in Psaraftis (1980, 1983). In the first reference, the objective function is defined as a generalization of the objective function of the Traveling Salesman Problem (TSP), in which a weighted combination of the time needed to serve all customers and of the total degree of dissatisfaction they experience until their arrival to the destination of the trip is minimized. The dissatisfaction of customers is assumed to be a linear function of the time each customer waits to be picked up and the time each customer spends riding in the vehicle until his/her delivery. In the second reference, the approach is extended to handle time windows on departure and arrival times, but only the route duration is minimized.

In this work, both aspects of the problem (general objective function and time windows) are considered. The main contribution is a solution method designed in a way that (i) it is capable of handling practically any objective function suitable for dynamic routing and (ii) the computational effort of the algorithm can be controlled smoothly: if the problem size is reasonable, the algorithm produces optimal solutions efficiently and as the problem size is increased, the search space may be narrowed in order to achieve locally optimal solutions.

* Tel.: +358 40 576 3585; fax: +358 9 451 2474.
E-mail address: lauri.hame@tkk.fi

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.08.021

This document is organized as follows. Section 2 introduces an exact algorithm for the static version of the problem, in which all requests are known a priori. As the complexity of the problem is increased, the algorithm is extended to a heuristic procedure with slight adjustments in order to be able to solve the problem more efficiently (Section 2.3). This is followed by a discussion of the dynamic case in Section 3 and by computational results presented in Section 4, where the complexity and performance of the proposed solution method is evaluated.

1.1. Literature review

As mentioned above, different versions of the dial-a-ride problem motivate several different solution methods. Most recent studies are related to the static multiple vehicle DARP, in which a set of vehicle routes is designed for a set of customers, whose pickup and drop-off points are known a priori (see for example Cordeau and Laporte, 2003a; Cordeau, 2006; Bent and Van Hentenryck, 2006; Ropke and Pisinger, 2006; Xiang et al., 2006; Melachrinoudis et al., 2007; Parragh et al., 2010; Garaix et al., 2010). In addition, a few heuristics for the dynamic version have recently been reported (for example Madsen et al., 1995; Horn, 2002; Attanasio et al., 2004; Coslovich et al., 2006; Xiang et al., 2008).

In the following paragraphs, a short review on the studies related to the single-vehicle dial-a-ride problem is presented, based on Cordeau and Laporte (2003b), Cordeau et al. (2007), Cordeau and Laporte (2007) and Berbeglia et al. (2010), to which the reader is referred for more exhaustive summaries.

The first exact approach to the solution of the dynamic dial-a-ride problem was presented by Psaraftis (1980). In this work the single-vehicle, "immediate-request" case is studied, in which a set of customers should be served as soon as possible. In this first model no time windows are specified by the customers, but the vehicle operator incorporates *maximum position shift* constraints limiting the difference between the position of a request in the chronological calling list and its position in the vehicle route. The objective is to minimize a weighted combination of the time needed to serve all customers and customer dissatisfaction. The complexity of this algorithm is $\mathcal{O}(n^2 3^n)$ and only small instances can thus be solved. However, the computational effort is decreased as more strict constraints are imposed.

The algorithm is first constructed for the static case of the problem and then extended to solve the dynamic case, in which new requests occur during the execution of the route but no information on future requests is available. In this version of the problem, the use of maximum position shift constraints is essential, in order to prevent a request from being indefinitely deferred. In a later study (Psaraftis, 1983), the approach is extended to handle time windows on departure and arrival times. In this extension, only the route duration is considered in the objective function.

In Sexton and Bodin (1985a,b), a heuristic approach to the single-vehicle DARP with one-sided time windows was introduced. In this problem formulation, the objective function is defined as a function of (i) the difference between the actual travel time and the direct travel time of a user and (ii) the difference between desired drop-off time and actual drop-off time. The algorithm was tested on real-life problems involving 7 to 20 customers.

In Desrosiers et al. (1986), an exact dynamic programming algorithm for the single-vehicle DARP, formulated as an integer program, was presented. The formulation includes time windows as well as vehicle capacity and precedence constraints. Optimal solutions with respect to total route length were obtained for $n = 40$.

In Bianco et al. (1994), exact and heuristic procedures for the traveling salesman problem with precedence constraints are introduced, based on a bounding procedure. Computational results of the algorithm are given for a number of randomly generated test problems, including the dial-a-ride problem with the classical TSP objective function.

Psaraftis noted that although single-vehicle Dial-A-Ride systems do not exist in practice, single-vehicle Dial-A-Ride algorithms can be used as subroutines in large scale multi vehicle Dial-A-Ride environments. It is mainly for this reason that one's ability to solve the single-vehicle DARP is considered important. A similar approach is used in this work, where the idea is to use the solution of the single-vehicle DARP as a subroutine in a dynamic multiple-vehicle scenario. An effort is made to solve the single-vehicle problem up to optimality whenever the problem size is reasonable.

2. The static dial-a-ride problem with time windows

In this case of the dial-a-ride problem, it is assumed that there are N customers to be served, to each of which is associated a pickup point, a drop-off point and relatively narrow time windows for pickup and delivery. The problem is to determine for a single vehicle the optimal route with respect to the N customers at some instant ($t = 0$).

Since the main application of the algorithm is related to dynamic dial-a-ride services, vehicle routes will be examined as *open paths*, starting with the location of the vehicle at $t = 0$ and ending with the delivery of a customer, instead of round trips, as in most static vehicle routing and dial-a-ride problems. The use of open paths causes no loss of generality, as the open and closed route problems are reducible to one another in linear time. This has been shown for the Traveling Salesman Problem by Papadimitriou (1977).

In addition, the computational effort of the algorithm should be minimal in order to be able to produce optimal solutions in a short time, even though the number of customers is large. In Psaraftis (1980, 1983), an exact algorithm based on dynamic programming is introduced for the single-vehicle DARP similar to the one described above. It is shown that if the problem is solved with strict constraints, the algorithm proves to solve the problem to optimality in short CPU times. However, if the constraints are not restrictive, the computational effort grows as an exponential function of the number of customers N. In the approach presented in this work, an effort is made to solve the problem up to optimality whenever the computational effort is reasonable. If the computational effort needed to produce globally optimal solutions grows too large, the algorithm should still produce locally optimal solutions with relatively little computational effort. In other words, the algorithm should be scalable in a way that the computing time may be somehow controlled regardless of the number of customers.

2.1. Problem formulation (Psaraftis, 1980, 1983)

Let there be N customers, each of which have been assigned a number i between 1 and N. For each customer $i \in \{1,\ldots,N\}$, let u_i be the pickup point and u_{N+i} be the delivery point, $[e_i, l_i]$ be the pickup time window and $[e_{N+i}, l_{N+i}]$ be the delivery time window,[1] q_i be the load (number of passengers) associated to customer i. Let A be the starting point of the vehicle (location of the vehicle at $t = 0$). It is assumed that the time to go from any one of the pickup and delivery points u_i, where $i \in \{1,\ldots,2N\}$, directly to another point u_j is a known and fixed quantity $t(i,j)$.

The goal is to find a vehicle route starting from A and ending at one of the delivery points so that the following conditions hold.

[1] In addition to the fixed time windows $[e_i, l_i]$ and $[e_{N+i}, l_{N+i}]$, to each customer may be associated a specific maximum ride time constraint $T_i^{\max}$. The handling of these constraints is discussed in Section 2.2.1.

1. The quantity

$$w_1 T + w_2 \sum_{i=1}^{N} \left(\alpha T_i^W + (1-\alpha) T_i^R \right) \quad (1)$$

is minimized, where
   - $w_1, w_2 \in \mathbb{R}$ are given weight parameters,
   - the parameter α is the customers' time preference constant ($0 \leqslant \alpha \leqslant 1$),
   - T is the duration of the route,
   - T_i^W is the waiting time of customer i, from the earliest pickup time until the time of pickup,
   - T_i^R is the riding time of customer i, from the time of pickup until the time of delivery.
2. The vehicle route should be legitimate, namely each customer should be picked up before he/she is delivered.
3. The vehicle has a certain capacity of C passengers that cannot be exceeded.
4. All time constraints must be satisfied.

In Eq. (1), the quantity $\sum_{i=1}^{N} \left(\alpha T_i^W + (1-\alpha) T_i^R \right)$ is the assumed form of the total degree of dissatisfaction experienced by the customers until their delivery, where α is the customers' time preference constant describing the impact of ride time and waiting time on the degree of dissatisfaction.

In Powell et al. (1995), it is noted that in static and deterministic transportation models, finding an appropriate objective function is fairly easy and that the objective function is usually a good measure for evaluating the solution. In dynamic models, the objective function used to find the solution over a rolling horizon has often little to do with the measures developed to evaluate the overall quality of a solution. In stochastic models it might be useful to minimize the probability of violating time windows. However, the advanced insertion method to be described is designed in a way that several objective functions, that are thought to be suitable for dynamic and stochastic problems, may be incorporated with minimal work. Generally, the choice of the objective function depends on the particular version of the problem and is discussed only briefly in this document. For a study related to choosing an appropriate objective function for the dynamic DARP, the reader is referred to Hyytiä et al. (2010).

The feasibility of time windows is governed by the following two assumptions.

1. If the vehicle arrives at any node (either pickup or delivery) later than the upper bound of that node's time constraint (l_j), then the entire vehicle route and schedule is infeasible. In other words, those upper bounds constitute hard constraints that should be met by the vehicle.
2. If the vehicle arrives at any node earlier than the lower bound on that node's time constraint (e_j), then the vehicle will stay idle at that point and depart immediately at e_j.

The algorithm to be presented in the following section will check that the time constraints are satisfied in a fashion particularly suitable for a highly restricted problem. In addition, the vehicle route is constructed in such a way that the pickup of each customer precedes the delivery and thus there is no need to check the validity of the legitimacy condition. While capacity constraints are also taken into account, the algorithm is designed and works best on problems in which the time limits are assumed to be more restrictive.

No exact solution to the problem described above has been reported. Similar approaches include the following.

- Psaraftis (1980), Bianco et al. (1994), Hernández-Pérez and Salazar-González (2009). Time limits are substituted by maximum position shift constraints.
- Psaraftis (1983), Desrosiers et al. (1986). Time windows are considered but the objective is to minimize route duration.
- Sexton (1979), Sexton and Bodin (1985a,b). A heuristic approach to a similar problem with one-sided time windows is presented.

2.2. Static case solution

In general, exact procedures for solving routing problems are computationally very taxing, since the complexity is always more or less equal to the classical Traveling Salesman Problem. In addition, exact optimization can be seen to be needless at run-time if routes are modified often. Despite these facts, exact algorithms are useful in a way that the performance of different heuristics may be compared to the optimal solution.

The main idea in the following approach to the solution of the static case is that customers are added to the vehicle route one by one by using an *advanced insertion* method, which leads to a globally optimal solution, that is a vehicle route which is feasible with respect to all customers and minimizes a given cost function.

Many studies related to the dial-a-ride problem (see for example Jaw et al., 1986; Madsen et al., 1995; Diana and Dessouky, 2004; Wong and Bell, 2006), make use of what is called the insertion procedure, in the classical version of which the pickup and delivery node of a new customer are inserted into the *current optimal sequence* of pickup and delivery nodes of existing customers. In these references, several improvements to the insertion algorithm have been suggested. However, the main idea of the classical insertion algorithm a priori excludes the possibility that the appearance of a new customer may render the optimal sequencing of already existing customers no longer optimal. While the basic insertion algorithm can be seen to produce relatively good results for the unconstrained TSP, the performance compared to an exact algorithm is decreased as the problem becomes more constrained (see Section 4).

In this work, the main idea is to construct the optimal route iteratively by implementing an insertion algorithm for each customer, one by one *for all feasible sequences of pickup and delivery nodes of existing customers*. Namely, the procedure involves two steps for each customer:

1. Perform insertion of the new customer to all feasible service sequences with respect to existing customers.
2. Determine the set of feasible service sequences with respect to the new customer and existing customers.

It can be readily shown that the insertion of a new customer to all feasible service sequences with respect to existing customers produces all feasible service sequences with respect to the union

Table 1
Potential service sequences with respect to customers 1 and 2. No capacity or time constraints are taken into account. $i^{\uparrow}$ denotes the pickup node and $i^{\downarrow}$ denotes the delivery node of customer i.

A:	$1^{\uparrow}$	$1^{\downarrow}$	$2^{\uparrow}$	$2^{\downarrow}$	B:	$1^{\uparrow}$	$2^{\uparrow}$	$1^{\downarrow}$	$2^{\downarrow}$	C:	$1^{\uparrow}$	$2^{\uparrow}$	$2^{\downarrow}$	$1^{\downarrow}$
D:	$2^{\uparrow}$	$1^{\uparrow}$	$1^{\downarrow}$	$2^{\downarrow}$	E:	$2^{\uparrow}$	$1^{\uparrow}$	$2^{\downarrow}$	$1^{\downarrow}$	F:	$2^{\uparrow}$	$2^{\downarrow}$	$1^{\uparrow}$	$1^{\downarrow}$

of existing customers and the new customer and leads to a globally optimal solution but is computationally expensive if the number of feasible service sequences grows large. However, in the following algorithm the route is constructed under relatively narrow time window constraints, which means that the number of feasible routes with respect to all customers will be small compared to the number of all legitimate routes. Furthermore, the algorithm may be easily extended to an adjustable heuristic algorithm able to handle any types of time windows, as will be seen in Section 2.3.

The idea of the advanced insertion method is clarified by the following example, where no capacity or time constraints are taken into account. Let $i^{\uparrow} = i$ denote the *pickup node* of customer i and let $i^{\downarrow} = N + i$ denote the *delivery node* of customer i. A *service sequence* is defined as an ordered list consisting of pickup and delivery nodes. For instance, the service sequence $(i^{\uparrow}, j^{\uparrow}, j^{\downarrow}, i^{\downarrow})$ indicates the order in which customers i and j are picked up and dropped off.

Let us start the advanced insertion process with customer 1. Since the pickup $1^{\uparrow}$ of customer 1 has to be before the delivery, $1^{\downarrow}$, the only possible service sequence at this point is $(1^{\uparrow}, 1^{\downarrow})$. Thus, the set of potential service sequences with respect to customer 1 consists of this single service sequence. By insertion of customer 2 into the service sequence $(1^{\uparrow}, 1^{\downarrow})$ we get the six service sequences presented in Table 1.

By inserting the pickup and delivery node of customer 3 into all of these service sequences we get a total of $6(5 + 4 + 3 + 2 + 1) = 90$ new potential service sequences. However, if the time and capacity constraints are taken into account, not all service sequences described above are necessarily feasible.

For example, if after the insertion of customer 2 it can be seen that only the service sequences A and B in Table 1, namely $(1^{\uparrow}, 1^{\downarrow}, 2^{\uparrow}, 2^{\downarrow})$ and $(1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow})$, are feasible with respect to time constraints, then it can be shown that all feasible service sequences including customer 3 are included in the sequences given by insertion to these two service sequences, which gives a maximum of $2(5 + 4 + 3 + 2 + 1) = 30$ new potential service sequences.

To formalize the above procedure, an insertion function is defined as follows. Let s be a service sequence $s = (p_1, p_2, \ldots, p_m)$ consisting of m nodes, where $p_j \in \{1, \ldots, 2N\}$ for all $j \in \{1, \ldots, m\}$ and let h and k be natural numbers such that $h \in \{1, \ldots, m + 1\}$ and $k \in \{h + 1, \ldots, m + 2\}$. The insertion $I(s, i, h, k)$ of a new customer i into the sequence s is defined as the service sequence

$$\begin{aligned} I(s, i, h, k) &= (r_1, \ldots, r_{m+2}) \\ &= (p_1, p_2, \ldots, p_{h-1}, i^{\uparrow}, p_h, \ldots, p_{k-2}, i^{\downarrow}, p_{k-1}, \ldots, p_m). \end{aligned} \tag{2}$$

For example, an insertion $I(s, i, h, k) = I((1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow}), 3, 2, 4)$ would produce the service sequence $(1^{\uparrow}, 3^{\uparrow}, 2^{\uparrow}, 3^{\downarrow}, 1^{\downarrow}, 2^{\downarrow})$. In other words, the numbers $h = 2$ and $k = 4$ mark the positions of pickup and delivery of the new customer 3 in the new service sequence. Furthermore, the insertion $I(\emptyset, 1, 1, 2)$ would correspond to the service sequence $(1^{\uparrow}, 1^{\downarrow})$.

By means of the above definition, we can now describe the structure of the advanced insertion algorithm specifically, see Algorithm 1. Briefly, the structure of the advanced insertion algorithm may be sketched as follows. At first, the set of feasible service sequences S_i is determined recursively for each customer $i \in \{1, \ldots, N\}$ by inserting the pickup and delivery nodes of customer i into each feasible service sequence with respect to customers 1, …, $i - 1$. In this way the algorithm produces the set S_N of all feasible routes with respect to customers 1, …, N. Then the solution to the static problem is obtained by choosing the sequence $s \in S_N$ with minimal cost $C(s)$. In the general form (1) of the cost function, this involves the calculation of waiting times and ride times for all customers for all feasible service sequences. If only route duration is minimized, we can simply choose the sequence for which the arrival time at the last node is smallest.

Algorithm 1: The advanced insertion algorithm

```
Set S_0 = {∅};        (S_i = set of feasible service sequences with
   respect to customers 1, ..., i)
for each i ∈ {1,...,N} do
  Set S_i = ∅;
  for each s ∈ S_{i-1} do
    for each h ∈ {1,...,|s| + 1} and k ∈ {h + 1, ..., |s| + 2} do
      Set r = I(s,i,h,k);      (I = the insertion function)
      if service sequence r is feasible then
        S_i = S_i ∪ {r};
      end if
    end for
  end for
  if S_i = ∅ then
    STOP: The problem has no solution;
  end if
endfor
for each s ∈ S_N
    Calculate cost C(s);
end for
```

In the following part, the verification of feasibility of sequences is examined.

2.2.1. Feasibility considerations

At each insertion, the feasibility of the new sequence $I(s, i, h, k) = (r_1, \ldots, r_m)$, where $s = (p_1, \ldots, p_{m-2})$, is verified. Since the precedence constraint is taken care of automatically in the insertion procedure, it is sufficient to check feasibility with respect to time and capacity constraints.

Time windows. The arrival time t_j at the jth node of the route should satisfy $t_j \leqslant l_{r_j}$, where l_{r_j} is the latest arrival time of node r_j and t_j is defined recursively by

$$t_j = \max(t_{j-1}, e_{r_{j-1}}) + t(r_{j-1}, r_j), \tag{3}$$

where $t_0 = 0$. For simplicity, the time needed for each customer to get on the vehicle or get off the vehicle is not taken into account in this study. However, unique service times $d_i, d_{N+i} \in \mathbb{R}$ for each customer's pickup and delivery may be incorporated in the model with minimal effort.

Maximum ride time constraints. If maximum ride time constraints are considered in addition to the fixed time windows, the feasibility is checked as follows. Let $T_i^{\max}$ denote the maximum ride time of customer i. When the arrival time t_j at pickup node r_j is calculated by means of formula (3), the latest delivery time of customer r_j is updated by means of the formula

$$l'_{N+r_j} = \min\left(l_{N+r_j}, \max(t_j, e_{r_j}) + T_{r_j}^{\max}\right), \tag{4}$$

where $\max(t_j, e_{r_j})$ denotes the pickup time of customer r_j. The arrival time t_k at each delivery node r_k should then satisfy $t_k \leqslant l'_{r_k}$.

Capacity. The total load on the vehicle Y_j just after it leaves the jth node of the route should satisfy $Y_j \leqslant C$, where C is the capacity of the vehicle and Y_j is defined recursively by

$$Y_j = Y_{j-1} + q_{r_j}, \tag{5}$$

where $Y_0 = 0$ and q_{r_j} is the load on node r_j.

The verification can be executed by means of the following procedure.

1. Set $t_j = \max\{t_{j-1}, e_{r_{j-1}}\} + t(j-1, r_j)$. If $t_j > l_{r_j}$, STOP: insertion $I(s, i, h, k)$ is infeasible.
2. If $j < k$: Set $Y_j = Y_{j-1} + q_{r_j}$. If $Y_j > C$, STOP: insertion $I(s, i, h, k)$ is infeasible.

If the insertion $r = I(s,i,h,k)$ passes the above procedure, the sequence $(r_1,\ldots,r_m)$ is added to the set of feasible service sequences S_i for the new customer i.

2.2.2. A priori clustering

In problems involving a large number of customers, assuming that the time windows are relatively narrow, a significant portion of service sequences can be eliminated before the actual insertion process by simply studying the mutual relationships between nodes, similarly as described in Dumas et al. (1991). More precisely, assume that the vehicle departs from a node i at the lower bound e_i of the time window and moves directly to node j. If the vehicle does make it in time to j, that is, if

$$e_i + t(i,j) > l_j, \tag{6}$$

it is said that the transition $i \to j$ is *a priori infeasible*. A priori infeasibility could be defined similarly for capacity constraints. However, assuming that the load associated to each customer is at most $C/2$, where C is the capacity of the vehicle, each transition is a priori feasible with respect to capacity. In other words, two customers i and j with loads satisfying $q_i \leqslant C/2$ and $q_j \leqslant C/2$ may be picked up and dropped of in any order without the capacity constraint being violated.

Note that if for any two nodes i and j, both transitions $i \to j$ and $j \to i$ are infeasible a priori, the entire problem is infeasible. Otherwise, either $i \to j$ or $j \to i$ or both are a priori feasible and the pickup and delivery nodes of customers can be divided among m preceding clusters by using the following rule.

Definition 1. Cluster C_k precedes cluster C_l if and only if the transition $x \to y$ is a priori infeasible for all $x \in C_l$ and $y \in C_k$. In this case, we shall use the notation $C_k \prec C_l$.

Clearly, assuming that there exists a feasible solution, the above precedence relation of clusters is a strict order: (i) $C \not\prec C$ for all clusters, (ii) $C_a \prec C_b$ implies $C_b \not\prec C_a$ and (iii) $C_b \prec C_c$ and $C_a \prec C_b$ implies $C_a \prec C_c$. In addition, note that if a single node i has an infinite time window $[-\infty,+\infty]$, there is only one cluster since all transitions $j \to i$ and $i \to j$ are feasible a priori.

In practice, the clusters can be determined by means of an adjacency matrix F for which $F_{ij} = 1$ if $i \to j$ is feasible and $F_{ij} = 0$ if $i \to j$ is infeasible a priori. By arranging the rows and columns suitably we get a block upper triangular matrix where each block corresponds to a cluster. Fig. 1 shows the adjacency matrix of a sample problem involving four customers. The rows and columns are arranged in a descending order with respect to row sums. From the matrix five clusters can be identified (blocks on the diagonal), namely $\{1^\uparrow\} \prec \{1^\downarrow,3^\uparrow,2^\uparrow\} \prec \{2^\downarrow\} \prec \{3^\downarrow,4^\uparrow\} \prec \{4^\downarrow\}$. Thus, since the positions of nodes $1^\uparrow$, $2^\downarrow$ and $4^\downarrow$ are fixed, the problem falls down to determining the optimal ordering of nodes $1^\downarrow$, $3^\uparrow$, $2^\uparrow$ and $3^\downarrow$, $4^\uparrow$. In general, by clustering the nodes a priori, a significant amount of insertions need not be checked for feasibility.

	$1^\uparrow$	$1^\downarrow$	$3^\uparrow$	$2^\uparrow$	$2^\downarrow$	$3^\downarrow$	$4^\uparrow$	$4^\downarrow$
$1^\uparrow$	1	1	1	1	1	1	1	1
$1^\downarrow$	0	1	1	1	1	1	1	1
$3^\uparrow$	0	1	1	1	1	1	1	1
$2^\uparrow$	0	1	1	1	1	1	1	1
$2^\downarrow$	0	0	0	0	1	1	1	1
$3^\downarrow$	0	0	0	0	0	1	1	1
$4^\uparrow$	0	0	0	0	0	1	1	1
$4^\downarrow$	0	0	0	0	0	0	0	1

Fig. 1. A priori adjacency matrix of a sample problem involving 4 customers. From the matrix five clusters can be identified, namely $\{1^\uparrow\} \prec \{1^\downarrow,3^\uparrow,2^\uparrow\} \prec \{2^\downarrow\} \prec \{3^\downarrow,4^\uparrow\} \prec \{4^\downarrow\}$.

2.3. Structure and complexity

The structure of the algorithm for the static case is as follows:

1. The set of feasible service sequences S_N with respect to all customers is determined.
2. The optimization part consists of the evaluation of the objective function for all potential sequences $s \in S_N$.

The number of insertions that are tested for feasibility for customer i is bounded above by the formula

$$n(i) \leqslant m_{i-1} \sum_{x=1}^{2i-1} x = \frac{m_{i-1}(2i)(2i-1)}{2}, \tag{7}$$

where m_{i-1} is the number of feasible service sequences with respect to customers $1,\ldots,i-1$. The inequality can be justified by the fact that not all possible insertions $I(s,i,h,k)$ have to be checked for feasibility. This is due to the fact that if a pickup point index h' proves to produce only infeasible solutions, there is no point in checking the feasibility of any of the combinations (h',k).

In the worst case, where capacity and time constraints are *not restrictive*, we get

$$m_1 = 1, \quad m_2 = (4\cdot 3)/2 = 6, \quad m_3 = 6(6\cdot 5)/2 = 90,$$

$$m_i = \frac{(2i)(2i-1)}{2}\,\frac{(2(i-1))(2(i-2)-1)}{2}\cdots\frac{4\cdot 3}{2} = \frac{(2i)!}{2^i}.$$

Thus the maximum number of insertions required in the solution of the static case is $\sum_{i=1}^{N}\frac{(2i)!}{2^i} \approx \sum_{i=1}^{N} 2^{1-i} i^{2i+\frac{1}{2}}\sqrt{\pi}$. Thus, in the worst case, the number of feasible solutions is of order $\mathcal{O}(\sqrt{N}(N^2/2)^N)$ and thus the computational complexity of the optimization part is also high, as all feasible routes have to be evaluated.

Compared to the computational complexity $\mathcal{O}(N^2 3^N)$ of the solution to the static DARP without time windows studied in Psaraftis (1980), these figures do not seem interesting. However, in this work it is assumed that the time constraints are relatively strict implying few feasible service sequences for each customer and thus the computational efficiency of the insertion algorithm is increased.

Suppose that, due to strict time (and capacity) constraints, the number of potential service sequences m_i for customers $1,\ldots,i$ is bounded by some function $m : \mathbb{N} \to \mathbb{N}$ such that $m(i) \leqslant \frac{(2i)!}{2^i}$. Then the number of insertions for customer i is bounded by $n(i) \leqslant \frac{m(i-1)(2i)(2i-1)}{2}$. The total number of insertions is bounded by $\sum_{i=1}^{N}\frac{m(i-1)(2i)(2i-1)}{2}$. For example, if $m(i) = kN$ for some constant k, the computational complexity of the screening phase is reduced to $\mathcal{O}(N^4)$. If $m(i) = k$, the computational complexity is reduced to $\mathcal{O}(N^3)$.

On the grounds of previous calculations, it can be stated that the advanced insertion algorithm will generally not be able to produce exact solutions efficiently in cases where the capacity and time constraints are not restrictive. However, if the number of feasible service sequences is bounded due to strict constraints, the algorithm will lead to an exact solution computationally inexpensively. It is evident that the computational effort of existing routing algorithms also decrease when the problem becomes highly restricted. However, no exact algorithms for the time constrained single-vehicle DARP with the generalized objective function (1) have previously been reported in the literature. Thus, the complexity can not be rightfully compared to existing exact algorithms for the single-vehicle dial-a-ride problem with time windows, since they are designed to solve a special case of the problem in which only the route duration is minimized.

In addition, the advanced insertion algorithm has a special property of being extendable to an adjustable heuristic, as described in the following subsection.

2.4. A heuristic extension

Even if the capacity and time constraints were not highly restrictive, the algorithm can be modified easily by bounding the size of the set S_i of service sequences, in which new customers are inserted, by including only a maximum of L service sequences for each customer i. More precisely, if after inserting customer i, the number of feasible service sequences with respect to customers $1, \ldots, i$ is larger than L, the set of feasible service sequences S_i with respect to customers $1, \ldots, i$ is narrowed by including only L service sequences, that seem to allow the insertion of remaining customers (see part Section 2.4.1). After the last customer has been inserted, the feasible service sequences are evaluated by means of the objective function (1).

This modification leads to a heuristic algorithm, in which the computational effort can be controlled by the parameter L, referred to as the *degree* of the heuristic. The resulting algorithm is somewhat sophisticated in a way that it produces globally optimal solutions for small sets of customers and when the number of customers is increased, the algorithm still produces locally optimal solutions with reasonable computational effort. In the special case where $L = 1$, the algorithm reduces to the classical insertion algorithm. If $L \geqslant \frac{(2N)!}{2^N}$, the heuristic coincides with the exact version of the algorithm as no routes are discarded.

2.4.1. Objective functions

In order to be able to efficiently make use of the above heuristic extension idea, the set of service sequences is narrowed by means of a certain heuristic objective function after the insertion of each customer. Since the main purpose of heuristics at the operational level is to always produce some implementable solutions very quickly, even if they were only locally optimal, such an objective function should be defined in a way that the algorithm is capable of producing *feasible* solutions even if the complexity of the problem was high.

Looking only at the cost defined in formula (1) may eliminate from consideration sequences that are marginally costlier but would easily allow the insertion of remaining customers in the route. Thus, more sophisticated criteria should be considered to help ensure that the heuristic will find a feasible solution when one exists.

In other words, the function should favor service sequences with enough *time slack* for those customers, that have not been inserted into the sequences. In this work, the following heuristic objectives are considered. Given the service sequence $s = (p_1, \ldots, p_m)$, we wish to optimize one of the functions

$$f_{rl}(s) = t_m, \quad \text{(Route duration)} \tag{8}$$

$$f_{ts}(s) = \sum_{j=1}^{m} l_j - t_j, \quad \text{(Total time slack)} \tag{9}$$

$$f_{min}(s) = \min_{j \in \{1,\ldots,m\}} l_j - t_j, \quad \text{(Max−min time slack)} \tag{10}$$

where $[e_j, l_j]$ is the time window and t_j is the calculated time of arrival at node p_j.

In general, each of the above objective functions aim to maximize the temporal flexibility of service sequences in different ways.

Route duration (8) favors service sequences in which the time to serve all customers is as small as possible. This objective can be justified by the fact that it is likely that new customers may be inserted at the rear of a route that is executed quickly.

Total time slack (9) stores sequences in which the sum of excess times (or the average excess time) at the nodes is maximized, that is, sequences which are likely to allow the insertion of a new customer before the last node.

Max–min (10) seeks sequences in which the minimum excess time at the nodes of the route is maximized. In other words, the sequences in which there is at least some time slack at each node are considered potential.

A simple example motivating the use of the above objective functions is presented in Fig. 2. In Section 4, the different objectives are compared by means of computational experiments.

2.4.2. Tuning

In order to ensure that the heuristic always produces a feasible solution when one exists, the parameter L can be tuned during runtime by using the following idea.

1. At first, the problem is solved by using an initial degree L_0.
2. Each time the algorithm is unable to find a feasible solution, the degree is increased and the problem is solved again.

At this point one might ask which initial value for L should be chosen and which tuning strategy would produce a feasible solution with least computational effort? These questions can be answered by studying the expected CPU time of the algorithm by means of the following model.

Let $P(S|L = l)$ denote the probability that a solution is found by using the degree $L = l$ and let $T(l)$ denote the average running time of the algorithm with $L = l$. The average CPU times of feasible runs are assumed equal to those occurring during infeasible runs, although infeasible runs are actually slightly less expensive. However, it can be seen that this approximation will not affect the results of the following examination. The expected CPU time for a given tuning strategy $(L_0, L_1, \ldots)$, where $L_0 < L_1 < L_2 < \ldots$, is given by the formula

$$T(L_0) + \sum_{i=1}^{\infty} T(L_i) \prod_{j=0}^{i-1} (1 - P(S|L = L_i)).$$

It should be noted that the above probabilities are actually conditional: if a solution is not found with some value of L, there is no point in solving the problem again with the same value. In addition, a relatively small increase in the value of L will not significantly affect the possibilities of finding a solution. Thus, the tuning approach used in the following examination is based on multiplying the value

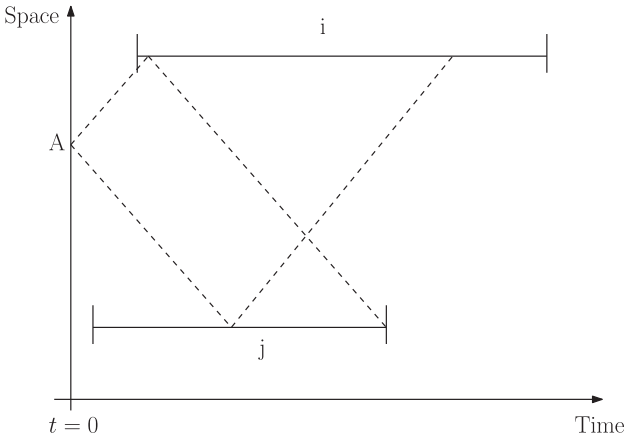


Fig. 2. Route flexibility. A vehicle is located at A at $t = 0$, and two customers are due to be picked up within the presented time windows at i and j. The dashed lines represent two possible routes for the vehicle. If the route duration were minimized, i should be visited before j. However, since there would be no "slack time" at j, this decision would a priori exclude the possibility that new customers could be inserted between i and j. On the other hand, if j were visited before i, there would be more possibilities for inserting new customers on the route before i. However, the route $A \to i \to j$ is shorter and thus it is more likely that customers can be inserted at the end of the sequence.

of L by a number K each time a feasible solution is not found. This method will be referred to as K-multiplying.

Assuming that the complexity grows linearly with l, that is $T(l) = l\,T(1)$, and that the probability of finding a solution is sufficiently high, it can be shown that the optimal initial degree for the K-multiplying method has to be relatively small. At first we will show that if $P(S|L \geqslant 1) > 1/2$, then any initial degree $L_0 = mK$, where $m \in \mathbb{N}$, is suboptimal.

Theorem 1. *Let $E(\tau(L_0))$ denote the expected CPU time of the K-multiplying method for the initial value L_0. If $T(l) = lT(1)$ for $l \in \mathbb{N}$ and $P(S|L \geqslant 1) > 1/2$, then*

$$E(\tau(L_0)) \leqslant E(\tau(KL_0))$$

for all $L_0 \in \mathbb{N}$ and $K \geqslant 2$.

Proof. Since it is clear that $E(\tau(l)) \geqslant T(l)$ for $l \in \mathbb{N}$, for any $K \geqslant 2$, we get

$$\begin{aligned} E(\tau(L_0)) &= T(L_0) + (1 - P(S|L = L_0))E(\tau(KL_0)) < T(L_0) + \frac{1}{2}E(\tau(KL_0)) \\ &= \frac{1}{k}T(KL_0) + \frac{1}{2}E(\tau(KL_0)) \leqslant \left(\frac{1}{K} + \frac{1}{2}\right)E(\tau(KL_0)) \\ &\leqslant E(\tau(KL_0)). \qquad \square \end{aligned}$$

Then, the following theorem states that if the probability of finding a solution is at least $1 - 1/K$, the optimal initial value is bounded above by the logarithm of the smallest natural number l for which $P(S|L = l) = 1$.

Theorem 2. *Let l' denote the smallest natural number l for which $P(S|L = l) = 1$. If $P(S|L \geqslant 1) > 1 - 1/K$, then*

$$E(\tau(L_0)) < T(L_0(1 + \log_K l'/L_0))$$

for all $L_0 \in \mathbb{N}$ and for all $K > 1$.

Proof. The expected CPU time is given by the formula

$$E(\tau(L_0)) = \sum_{i=0}^{\infty}(1 - P(S|L = K^i L_0))^i T(K^i L_0).$$

Since $(1 - P(S|L = K^i L_0)) = 0$ for $K^i L_0 > l'$, that is, $i > \log_K l'/L_0$, we get

$$\begin{aligned} E(\tau(L_0)) &= \sum_{i=0}^{\lfloor \log_K l'/L_0 \rfloor}(1 - P(S|L = K^i L_0))^i T(K^i L_0) \\ &\leqslant \sum_{i=0}^{\lfloor \log_K l'/L_0 \rfloor}(1 - (1 - 1/K))^i K^i T(L_0) \\ &\leqslant T(L_0)(1 + \log_K l'/L_0). \qquad \square \end{aligned}$$

In general, the two theorems imply that the expected CPU time is minimized when the initial degree L_0 is small. However, this is true only if the probability of finding a solution with small values of L is relatively high. For example, if $l' = 16$, $K = 2$ and $P(S|L \geqslant 1) > 0.5$, Theorem 2 states that $E(\tau(1)) < T(5)$. By Theorem 1 we know that the values 2 and 4 are suboptimal for $K = 2$. Thus, the optimal initial value is either 1 or 3 depending on the actual probabilities of finding solutions with different values of L. The fact that the optimal value of L_0 (with respect to complexity) is small is verified by the computational results presented in Section 4.

Finally, it should be noted that the optimal choice of K and L_0 will in general depend on the type of the problem to be solved and that by increasing the values, the optimality of the solution with respect to the objective of the problem is increased as well since more sequences are evaluated, as will be seen in Section 4.

3. The dynamic case

Let us discuss the extension of the predescribed algorithm to the dynamic case, in which new customers are appended to the route in real time.Generally, it can be seen that there is no need to compute a new route except when a new customer request occurs, a customer does not show up at the agreed pickup location, the vehicle is ahead or behind of schedule or the vehicle has lost its way.

A simple example, similar to the one presented in Psaraftis (1980), on the real-time updating of a vehicle route is shown in Fig. 3. Initially, customers 1 and 2 have been assigned pickup and delivery points and corresponding time windows. The vehicle located at A follows the tentative optimal route $(1^{\uparrow}, 2^{\uparrow}, 1^{\downarrow}, 2^{\downarrow})$, where "$\uparrow$"-symbols denote pickup nodes and "$\downarrow$"-symbols denote delivery nodes. (see Fig. 3a). At the time the vehicle is at B, the vehicle route is updated with respect to customers 1, 2 and 3. At this instant a new tentative optimal route shown in Fig. 3b is produced.

Note that the new route does not have B as origin, but a point B', slightly ahead of B. This is due to two facts: (i) It will take some time for the algorithm to process the new input and reoptimize, (ii) It will take some time for the driver of the vehicle to process the information regarding the new route. The distance between B and B' depends in general on the particular dial-a-ride service examined. For example, in a highly dynamic setting, in which the modification of the route is not allowed to take more than a few seconds, it may be assumed that the latter of the facts is more restrictive. Any dynamic dial-a-ride service should make use of a mechanism to update the point B' at sufficient time intervals. In this work, however, it is assumed that the point B' is known at each instant, as well as the estimated time of arrival T' at B'. The *vehicle checkpoint* (B', T') acts as the starting point for all service sequences.

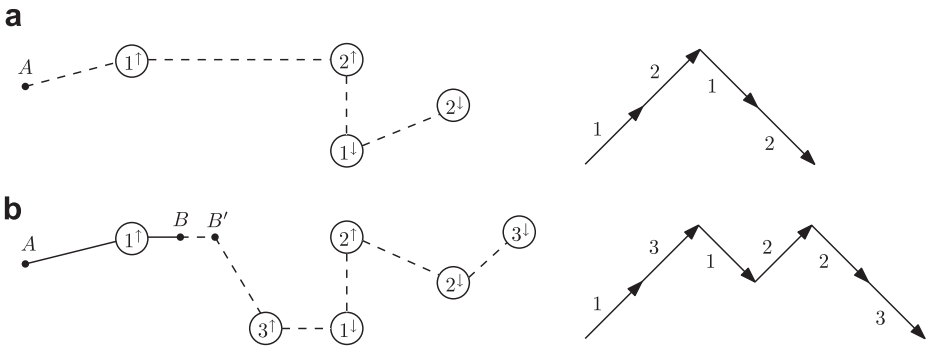


Fig. 3. Modifications in the vehicle route. The route is updated when the vehicle is at B and a new tentative optimal route beginning at B' is produced. The figures on the right show the routes as so-called labeled Dyck paths (Cori, 2009), in which each pickup $i^{\uparrow}$ precedes the corresponding drop-off $i^{\downarrow}$. At the time a new customer is added to the route, a new path is formed. Clearly, the "height" of the path shows the number of customers aboard in different parts of the route.

For customers that have already been picked up, the pickup point is not a part of the input of the problem. Thus, for such customers only the delivery point is considered in the insertion procedure. More precisely, given a service sequence $s = (p_1, \ldots, p_m)$, the insertion function $I'(s,i,k)$ is used for all $k \in \{1, \ldots, m+1\}$, where the insertion $I'(s,i,k)$ produces the service sequence $(p_1, \ldots, p_{k-1}, N+i, p_k, \ldots, p_m)$.

In dynamic models, the objective function used to find the solution over a rolling horizon has often little to do with the measures developed to evaluate the overall quality of a solution (Powell et al., 1995). However, the advanced insertion method is designed in a way that several objective functions, that are thought to be suitable for the dynamic problem, may be incorporated with minimal work. For example, if the vehicle route is subject to several modifications in short periods of time, one of the flexibility measures (10), (9) or forward time slack defined in Savelsbergh (1992) may be used as an objective function for the problem in order to be able to insert as many future customers in the route as possible. Generally, the choice of the objective function depends on the particular version of the dynamic routing problem and the performance of different objective functions may be sensitive to, for example, constraints, demand intensity or the number of vehicles. For a study related to choosing an objective function for the immediate-request DARP, the reader is referred to Hyytiä et al. (2010).

4. Numerical experiments

In the following paragraphs, a collection of computational results obtained by the advanced insertion method are proffered. The exact and heuristic versions of the algorithm were tested on a set of problems involving different numbers of customers and different time window widths determined by a travel time ratio R describing the maximum allowed ratio of travel time to direct ride time. The pickup and drop-off points of customers were chosen randomly from a square-shaped service area and the ride times between the points were modeled by euclidean distances.

At first, the complexity of the problem was studied with respect to three parameters, namely (i) the number of customers N, (ii) travel time ratio R and (iii) the average time interval μ between customer requests. The complexity of the exact algorithm was measured in terms of the average number of feasible sequences in *feasible problem instances*, that is, randomized problems for which at least one feasible solution was found. The number of feasible sequences refers to the number of feasible sequences after inserting the last customer, even though in some cases the number of feasible sequences for customers before the last one may be greater. However, using the final number of feasible sequences as a measure of complexity can be justified by the fact that the number of feasible sequences was seen to be an increasing function of the number of inserted customers in most studied cases (67% of feasible instances with $N = 20$, $R = 3$ and $\mu = 1800$). In addition, the measure is general in a sense that it is independent of the insertion order of customers. In fact, the final number of feasible sequences gives us an insight on the complexity of the problem itself, since any enumeration algorithm would have to evaluate the same number of feasible sequences.

Then, the performance of the heuristic with different objective functions and different degrees was evaluated. The complexity was measured in terms of the number of sequences evaluated by the heuristic.

A summary of performed experiments and the corresponding results is shown in Table 2.

4.1. Parameters

The parameters of the experiments are based on a demand-responsive transport service operating in a neighborhood of 100 km^2. For simplicity, a square-shaped service area is studied. The largest travel time ratio used in the experiments is 3, which describes a relatively low level of service. The number of customers assigned to a single vehicle at a certain instant is strongly dependent on the type of the dial-a-ride service. In highly dynamic services, in which most trips are requested not long before the trip is due to begin, the number of customers in the tentative route is relatively small (see Hyytiä et al., 2010). On the other hand, if trips are requested hours in advance, the planning horizon is significantly longer and thus the complexity of the problem is increased. In the experiments, no pre-order time limits are assumed. The demand is generated by means of a Poisson process, in a way that the single vehicle serves up to 10 customers per hour on average, which is considered a relatively high value for vehicle efficiency. The vehicles used in demand-responsive transport services are often small or medium sized and thus the number of customers assigned to a single-vehicle route is often limited to 10–20 customers. In the experiments, the vehicle capacity is set to $C = 10$ and the focus is on problem instances with up to 20 customers. In experiment 6, we study the robustness of the algorithm in unexpected situations, in which there may be up to 50 customers assigned to the vehicle.

4.2. Data generation

Letting N denote the number of customers, R denote the travel time ratio and μ denote the average time interval between customers, the data used in the following experiments was generated as follows. For each customer $i \in \{1, \ldots, N\}$:

1. Choose a pickup point u_i and a drop-off point u_{N+i} randomly in $[0, 10, 000] \times [0, 10, 000] \subset \mathbb{R}^2$ (unit = 1 meter).
2. Determine the earliest pickup time e_i by means of a Poisson process with intensity $\lambda = 1/\mu$.

Table 2
A summary of performed experiments.

Experiment number	Algorithm	Travel time ratio	Number of customers	Goal	Main result	Figure number
1	Exact	2, 2.5, 3	1 …, 20	Complexity with respect to number of customers	$\mathcal{O}(\exp(N))$	Fig. 4
2	Exact	1.5, …, 3	5, 10, 20	Complexity with respect to travel time index	$> \mathcal{O}(\exp(R))$	Fig. 5
3	Exact	3	10	Complexity with respect to time interval	$\mathcal{O}(\exp(-\mu))$	Fig. 6
4	Heuristic	3	20	Compare heuristic cost functions	Total time slack	Fig. 7
5	Tuned heuristic	3	20	Complexity and error with respect to L_0	Error decreases when L_0 is increased	Fig. 8
6	Tuned heuristic	3	2, …, 50	Robustness with respect to number of customers	Feasible solutions with little effort	Fig. 9

3. Determine the latest drop-off time l_{N+1} by means of the formula $l_{N+i} = e_i + Rt(u_i, u_{N+i})$, where $t(u_i, u_{N+i}) = \|u_i - u_{N+1}\|/40$ kilometers/hour is the direct ride time from the pickup point to the drop-off point.
4. Determine the latest pickup time by $l_i = l_{N+i} - t(u_i, u_{N+i})$ and the earliest drop-off time by $e_{N+i} = e_i + t(u_i, u_{N+i})$.

For simplicity, both pickup and drop-off points were considered for each customer in all experiments, even though the pickup points of customers on board need not be considered in the dynamic case. However, this property of the dynamic case indicates that enumerating all feasible sequences during the execution of a route requires less computational work than complete enumeration in the corresponding static case, in which pickup points of all customers are considered as well. Thus, the results to be presented act as an upper bound for the dynamic case.

4.3. Experiments

The following experiments were performed on a standard laptop computer with a 2.2 GHz processor. The CPU times and the number of evaluated sequences appeared to have a roughly linear relationship. A typical problem instance involving 20 customers could be solved up to optimality within less than a second.

4.3.1. Experiment 1: Number of customers

At first we study the complexity of the exact algorithm with respect to the number of customers. In practice, the number of customers assigned to a single vehicle is governed by the pre-order time of the dial-a-ride service: if customers may request service in advance, the planned vehicle routes are expected to be longer than in immediate-request services. Fig. 4 shows the average number of feasible sequences in feasible problem instances on a logarithmic scale, computed over 10,000 randomized instances for $N = 1, \ldots, 20$, $R = 2$, 2.5, 3 and $\mu = 1800$ seconds.

Referring to the figure, it can be seen that the complexity of the problem increases exponentially with respect to the number of customers with all studied values of the travel time ratio. In addition, the complexity is increased with the travel time ratio.

4.3.2. Experiment 2: Travel time ratio

Let us study the complexity of the exact algorithm as a function of travel time ratio. Fig. 5 shows the average number of feasible sequences in feasible problem instances on a logarithmic scale, computed over 10,000 randomized instances for $R = 1.5, \ldots, 3$, $N = 5$, 10, 20 and $\mu = 1800$ seconds.

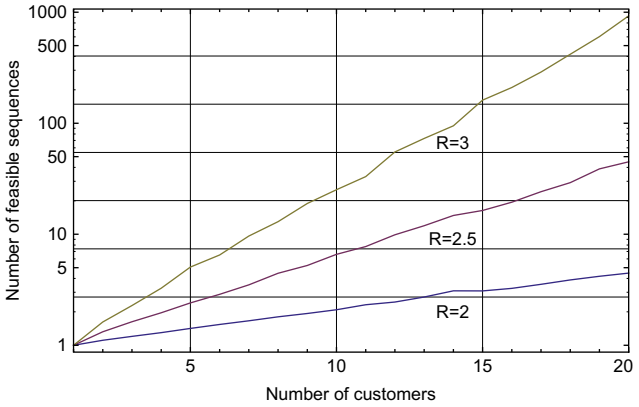


Fig. 4. Experiment 1. The complexity of the exact algorithm with respect to the number of customers on a logarithmic scale. The three curves represent, as a function of the number of customers, the average number of feasible sequences for $R = 2$, 2.5, 3 and $\mu = 1800$ seconds. Clearly, the complexity increases exponentially with respect to the number of customers.

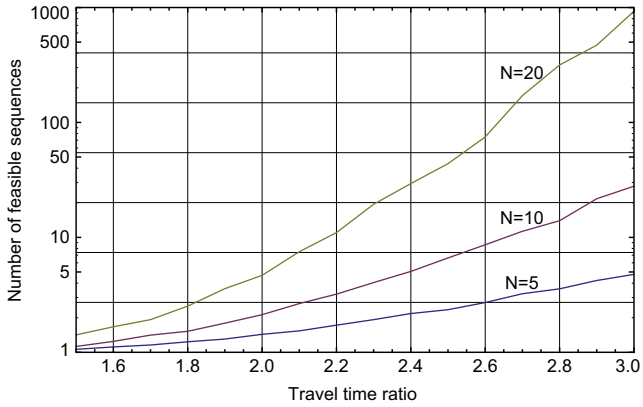


Fig. 5. Experiment 2. The complexity of the exact algorithm with respect to travel time ratio on a logarithmic scale. The three curves represent, as a function of the travel time ratio, the average number of feasible sequences for $N = 5$, 10, 20 and $\mu = 1800$ seconds.

The figure shows that the effect of the travel time ratio on the complexity of the problem is significant. The fact that the slopes of the curves increase with R on the logarithmic scale indicates that the relation between complexity and R is superexponential.

4.3.3. Experiment 3: Time interval

Let us conclude the study of the exact algorithm by examining complexity with respect to the average time interval between customer requests. The solid lines in Fig. 6 represent the average number of feasible sequences in (i) feasible problem instances and (ii) all problem instances, on a logarithmic scale, computed over 10,000 randomized instances for $N = 10$, $R = 3$ and $\mu = 6, \ldots, 40$ minutes. The dashed line corresponds to the fraction of problem instances, for which at least one feasible solution was found.

The figure indicates that the complexity of feasible problem instances decreases exponentially with respect to the average time interval μ. On the other hand, the probability of finding at least one feasible solution is increased with μ. By looking at the curve corresponding to the average complexity of all problem instances (including infeasible cases), it can be seen that the complexity is maximized at a certain time interval ($\mu = 24$ minutes in this case), in which both the probability of finding a feasible solution and the number of feasible sequences in feasible cases are relatively large.

At this point, it should be emphasized that the above results apply to randomized instances for the single-vehicle problem. In a dynamic multiple-vehicle setting, an effort is made to divide the customers among available vehicles optimally. Thus, it is suggested that a larger number of customers can be served by a single vehicle in less time than in the case in which the customers' pickup and drop-off locations are completely random. However, the above results give us an insight on how the complexity of the single-vehicle subroutine behaves with respect to the average time interval between the earliest pickup times of customers assigned to a single vehicle.

4.3.4. Experiment 4: Objective functions

Let us study the performance of the three different heuristic cost functions (8)–(10) as a function of the degree L of the heuristic. Fig. 7 shows the fraction of problem instances for which a feasible solution was found by the heuristic (compared to the exact algorithm), computed over 10,000 randomized instances. The number of customers was set to $N = 20$, the travel time ratio was 3 and the average time interval $\mu = 1800$ seconds was used.

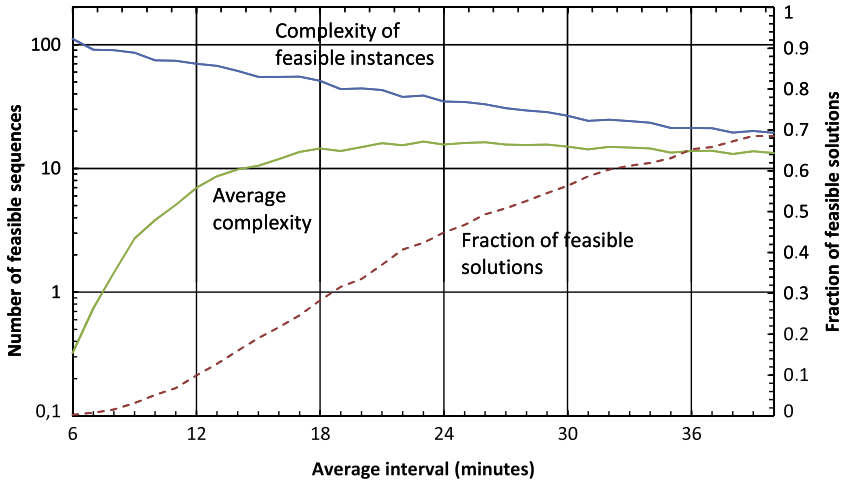


Fig. 6. Experiment 3. The complexity of the exact algorithm with respect to the average time interval between customers. The solid lines represent the average number of feasible sequences in feasible problem instances and all problem instances, on a logarithmic scale for $N = 10$ and $R = 3$. The dashed line corresponds to the fraction of problem instances, for which at least one feasible solution was found.

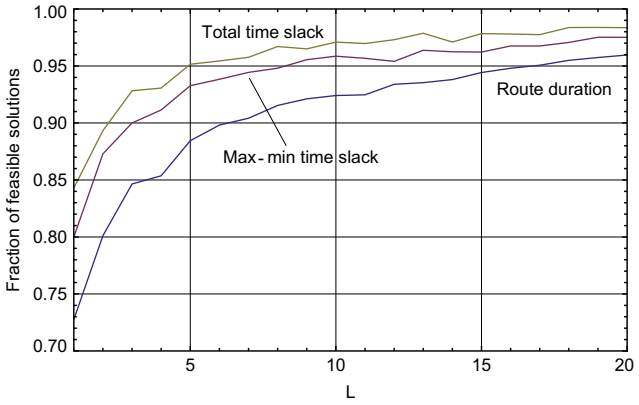
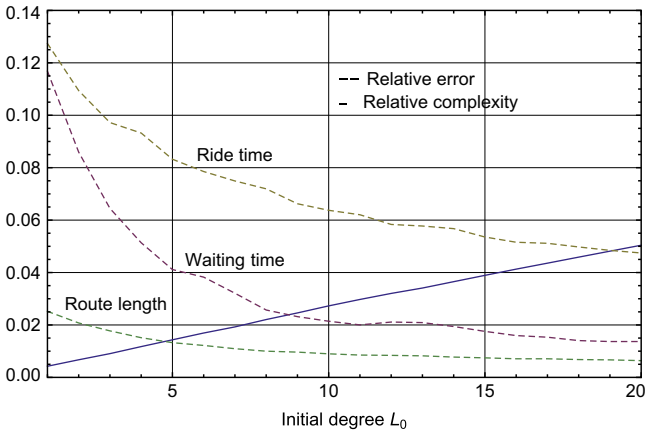


Fig. 7. Experiment 4. The performance of three different heuristic objective functions as functions of degree L. The curves represent the fractions of instances for which a feasible solution was found by the objective functions (compared to the exact algorithm), for $N = 20$, $R = 3$ and $\mu = 1800$ seconds. The total slack time objective function outperforms the other two in all studied cases.



Relative error in route duration							
L_0	1	2	3	4	5	6	7
Relative error (10^{-5})	9.8	4.9	2.4	2.4	2.4	2.4	0

Fig. 8. Experiment 5. Relative complexity and relative error of the tuned heuristic as a function of the initial degree L_0. The solid line shows the relative complexity of the tuned heuristic, compared to the exact algorithm and the dashed lines correspond to the relative error in total ride time and total waiting time of customers and in route length, compared to the exact algorithm for $N = 20$ and $R = 3$. The table below the figure shows the corresponding relative error in route duration. Clearly, the heuristic is capable of finding short routes with little effort. As for more complex objectives, such as total ride time and total waiting time, more work is needed to control the error.

Referring to the figure, it can be seen that the total time slack cost function (9) is capable of finding a feasible solution to randomized problems most often, while the performance of the route duration cost function (8) is worst of the three algorithms. Note that as the degree L is increased, the fraction of feasible solutions converges to 1 for any heuristic cost function, since whenever $L \geqslant \frac{(2N)!}{2^N}$, the heuristic coincides with the exact algorithm regardless of the studied problem.

4.3.5. Experiment 5: Relative error

Let us study the complexity and relative error of the tuned heuristic with respect to the initial degree L_0. The solid line in Fig. 8 shows the relative complexity of the tuned heuristic, compared to the exact algorithm and the dashed lines correspond to the relative error in total ride time, total waiting time and route length, compared to the exact algorithm, calculated over 10,000 randomized runs. The table below the figure shows the corresponding relative error in route duration.

The relative complexity values were computed by means of the formula $\sum_{k\in F} h_k / \sum_{k\in F} H_k$, where h_k and H_k denote the number of sequences evaluated by the tuned heuristic and the exact algorithm in feasible problem instance k and F denotes the set of feasible problem instances. The relative error values for total ride time were computed by means of the formula $\sum_{k\in F} r_k / \sum_{k\in F} R_k - 1$, where r_k and R_k denote the sum of ride times $\left(\sum_{i=1}^{N} T_i^R\right)$ of N customers in feasible problem instance k. The relative error for total waiting time, route length and route duration were calculated in a similar fashion.

As in the previous experiment, the number of customers was set to $N = 20$, the travel time ratio was 3 and the average time interval $\mu = 1800$ seconds was used. The multiplier K of the tuned heuristic was set equal to 2, that is, each time a feasible solution was not found, the degree of the heuristic was doubled and the problem was solved again.

By looking at the figure, it can be seen that the complexity of the tuned heuristic is a linear function of the initial degree L_0. On the

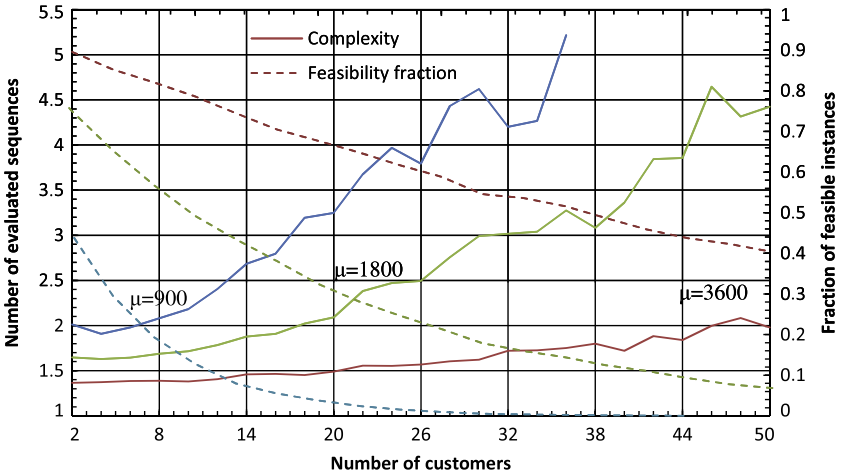


Fig. 9. Experiment 6. Robustness of the tuned heuristic as a function of the number of customers. The solid lines show the complexity of the tuned heuristic in terms of the average number of evaluated sequences in feasible instances and the dashed lines correspond to the fraction of instances with at least one feasible solution for $\mu \in \{900, 1800, 3600\}$, $R = 3$ and $L_0 = 1$. The heuristic is capable of finding feasible routes for a large number of customers with little effort. The complexity of feasible instances is increased as the time interval between customers is decreased, but the fraction of feasible problems is decreased with μ.

other hand, the error in both total ride time and total waiting time is decreased when L_0 is increased. The error in total waiting time is relatively smaller than the corresponding error in total ride time. By looking at the curve corresponding to route length and the table below the figure it can be seen that the heuristic is capable of finding short routes close to the optimal solution with little effort. As for more complex objectives, such as total ride time and total waiting time, more work is needed to find good quality sequences. Thus, it can be suggested that using a large initial degree L_0 is well motivated only with complex objective functions.

4.3.6. Experiment 6: Robustness

Finally, we examine the robustness of the tuned heuristic as a function of the number of customers. The solid lines in Fig. 9 show the complexity of the tuned heuristic in terms of the average number of evaluated sequences in feasible instances and the dashed lines correspond to the fraction of problem instances with at least one feasible solution, computed over 10,000 randomized runs for $\mu \in \{900, 1800, 3600\}$, $R = 3$ and $L_0 = 1$. As in the previous experiment, the multiplier $K = 2$ was used.

Clearly, the heuristic is capable of finding feasible routes for a large number of customers with little effort. While the complexity of feasible instances is increased as the average time interval between customers is decreased, the fraction of problems with at least one feasible solution decreases with μ. By looking at the dashed curves, it can be seen that the probability of finding a feasible solution tends to zero as the number of customers is increased. The above observations imply that with randomized data, problems for which the solution is difficult to find are relatively uncommon.

As a conclusion, a significant advantage of using the heuristic algorithm can be identified as the possibility of being able to control the computational effort smoothly. While the case $L = 1$ coincides with the classical insertion algorithm in which the complexity and the probability of finding a feasible solution is relatively low, by increasing the value of L, the CPU time, as well as the overall quality of the solution, is increased. In addition, the heuristic may also be used to always produce a solution whenever one exists by tuning it during run-time (see K-multiplying method, part Section 2.4.2).

5. Conclusions

In this work, an exact optimization procedure is developed to solve the static and dynamic versions of the single-vehicle dial-a-ride problem with time windows. Using complete enumeration to solve the problem with respect to a generalized objective function is motivated by the dynamic nature of online demand-responsive transport services, in which looking only at the tentative route duration, as in existing algorithms for the problem, may decrease the possibilities of serving future customers. In addition, an adjustable heuristic extension to the algorithm is introduced, in order to be able to control the CPU times: if the problem size is reasonable, the proposed solution method produces globally optimal solutions. If the problem size is increased, the algorithm adjusts itself to produce locally optimal solutions, closing the gap between the classical insertion heuristic and the exact solution and thus making the algorithm applicable to any static or dynamic dial-a-ride problem.

Acknowledgments

This work was partly supported by the Finnish Funding Agency for Technology and Innovation, Finnish Ministry of Transport and Communications, Helsinki Metropolitan Area Council and Helsinki City Transport. The author is most indebted to Dr. Harri Hakula for his assistance and would also like to thank Dr. Aleksi Penttinen, Dr. Esa Hyytiä and three anonymous referees for their insightful comments to improve the quality of the paper.

References

Attanasio, A., Cordeau, J.-F., Ghiani, G., Laporte, G., 2004. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Computing 30, 377–387.

Bent, R., Van Hentenryck, P., 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research 33 (4), 875–893.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. European Journal of Operational Research 202, 8–15.

Bianco, L., Mingozzi, A., Ricciardelli, S., Spadoni, M., 1994. Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. INFOR 32, 19–31.

Cordeau, J.-F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Operations Research 54, 573–586.

Cordeau, J.-F., Laporte, G., 2003a. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research B 37, 579–594.

Cordeau, J.-F., Laporte, G., 2003b. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. 4OR: A Quarterly Journal of Operations Research 1, 89–101.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: Models and algorithms. Annals of Operations Research 153, 29–46.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., Savelsbergh, M., 2007. Transportation on demand. In: Transportation. North-Holland, Amsterdam, pp. 429–466.

Cori, R., 2009. Indecomposable permutations, hypermaps and labeled dyck paths. Journal of Combinatorial Theory, Series A 116 (8), 1326–1343.
Coslovich, L., Pesenti, R., Ukovich, W., 2006. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. European Journal of Operational Research 175, 1605–1615.
Desrosiers, J., Dumas, Y., Soumis, F., 1986. A dynamic programming solution of the large-scale single vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences 6, 301–325.
Diana, M., Dessouky, M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transportation Research Part B 38, 539–557.
Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7–22.
Garaix, T., Artigues, C., Feillet, D., Josselin, D., 2010. Vehicle routing problems with alternative paths: An application to on-demand transportation. European Journal of Operational Research 204, 62–75.
Hernández-Pérez, H., Salazar-González, J.-J., 2009. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. European Journal of Operational Research 196, 987–995.
Horn, M., 2002. Fleet scheduling and dispatching for demand-responsive passenger services. Transportation Research Part C 10, 35–63.
Hunsaker, B., Savelsbergh, M.W.P., 2002. Efficient feasibility testing for dial-a-ride problems. Operations Research Letters 30, 169–173.
Hyytiä, E., Häme, L., Penttinen, A., Sulonen, R., 2010. Simulation of a large scale dynamic pickup and delivery problem. In: Third International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010).
Jaw, J., Odoni, A., Psaraftis, H., Wilson, N., 1986. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation Research Part B 20, 243–257.
Madsen, O., Ravn, H., Rygaard, J., 1995. A heuristic algorithm for the a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. Annals of Operations Research 60, 193–208.
Melachrinoudis, E., Ilhan, A.B., Min, H., 2007. A dial-a-ride problem for client transportation in a healthcare organization. Computers & Operations Research 34, 742–759.
Papadimitriou, C., 1977. The euclidean travelling salesman problem is np-complete. Theoret. Comput. Sci. 4, 237–244.
Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 37, 1129–1138.
Powell, W.B., Jaillet, P., Odoni, A., 1995. Stochastic and dynamic networks and routing. Network Routing, vol. 8. North-Holland, Amsterdam, pp. 141–295.
Psaraftis, H., 1980. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. Transportation Science 14, 130–154.
Psaraftis, H., 1983. An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. Transportation Science 17, 351–357.
Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40 (4), 455–472.
Savelsbergh, M., 1992. The vehicle routing problem with time windows: Minimizing route duration. ORSA Journal on Computing 4, 146–154.
Sexton, T., 1979. The single vehicle many-to-many routing and scheduling problem. Ph.D. Dissertation, SUNY at Stony Brook.
Sexton, T., Bodin, L.D., 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. Transportation Science 19, 378–410.
Sexton, T., Bodin, L.D., 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: Ii routing. Transportation Science 19, 411–435.
Toth, P., Vigo, D., 1997. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science 31, 60–71.
Wong, K., Bell, M., 2006. Solution of the dial-a-ride problem with multi-dimensional capacity constraints. International Transactions in Operational Research 13, 195–208.
Xiang, Z., Chu, C., Chen, H., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. European Journal of Operational Research 174, 1117–1139.
Xiang, Z., Chu, C., Chen, H., 2008. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. European Journal of Operational Research 185, 534–551.

Publication II

Esa Hyytiä, Lauri Häme, Aleksi Penttinen, Reijo Sulonen. Simulation of a Large Scale Dynamic Pickup and Delivery Problem. In *SIMUTools*, Malaga, Spain, March 2010.

Simulation of a Large Scale Dynamic Pickup and Delivery Problem

Esa Hyytiä, Lauri Häme, Aleksi Penttinen and Reijo Sulonen
Aalto University, School of Science and Technology
PO Box 11000, FI-00076 Aalto, Finland
{esa.hyytia,lauri.hame,aleksi.penttinen,reijo.sulonen}@tkk.fi

ABSTRACT

We study a variant of dynamic vehicle routing problem with pickups and deliveries where a vehicle is allocated to each service (i.e., trip) request immediately upon the arrival of the request. Solutions to this problem can be characterized as dynamic *policies* that define how each customer is handled by operating a fleet of vehicles. Evaluation of such policies is beyond the grasp of analytical studies and requires extensive simulations. We present an efficient and modular simulation tool developed for studying the performance of a large scale system with different policies under given trip arrival process. Numerical and analytical observations on the model are utilized to provide guidelines for solving the routing problem efficiently, and to support the validation of the simulation results. Application of the developed framework is demonstrated by several numerical examples, e.g., policy parameter optimization, which all give insight on the viability of this type of transportation system.

Categories and Subject Descriptors

G.4 [**Mathematical software**]: *algorithm design and analysis*; I.6.3 [**Simulation and modeling**]: Applications

General Terms

Experimentation,Algorithms,Performance

Keywords

Vehicle routing, dial-a-ride problem

1. INTRODUCTION

The dynamic vehicle routing problem with pickups and deliveries (VRPPD) involves the dispatching of a fleet of vehicles in real time in order to serve customers requesting transportation of goods or passengers from one location to another. In general, vehicles can serve more than one request at the same time. The main applications of this problem include the transportation of handicapped and elderly people in urban areas (dial-a-ride problem, DARP) and the transportation services of letters and parcels performed by courier companies (urban courier service problem) [4].

In this type of dynamic routing problems, customer requests are revealed gradually in the course of time and thus the vehicle routes are subject to modifications as they are executed. Therefore, solutions to dynamic problems are often characterized as *policies* specifying the actions in different situations. Policies attempt to manage the well-known *dichotomy* between the work the vehicle fleet conducts and the service the passengers obtain, e.g., between the kilometers the vehicles drive and the mean traveling time. Indeed, an efficient policy seeks to optimize either of the two aspects, or some balanced combination of them (cf., Pareto optimality).

In this work, we consider a *large scale dynamic* vehicle routing problem with pickups and deliveries where each customer is assigned to a vehicle immediately at the arrival of the request. In this case the solution to the dynamic VRPPD can be decomposed into two subpolicies: vehicle allocation, and vehicle routing. For each customer the system decides, in on-line fashion, which vehicle takes the customer after which the route of the corresponding vehicle is updated. The vehicle routing part is inherently combinatorial by nature (cf., the traveling salesman problem) and, in many cases, needs to be solved for all vehicles actually *before* a vehicle allocation can be made. Furthermore, as we focus on systems with at least several hundreds of vehicles under high transportation demand, even a simple policy tends to induce extensive computations. Thus, evaluation of the performance of different policies requires an efficient simulation platform tailored particularly for this purpose.

We have developed a modular simulator that is capable of handling various capacity and time constraints in a large system at fast speed. Our goal is two-fold. First, we use the simulator to study this complex transportation system in general in order to understand, e.g., the trade-off between the system's work and the service the passengers experience. Secondly, we apply the simulator to develop efficient policies for solving the problem. In this paper we describe the simulator design, make some fundamental observations on the dynamics of the problem, and demonstrate how the simulator can be used to improve the heuristic vehicle routing and allocation policies by tuning policy parameters.

Rest of the paper is organized as follows. In Section 2, the main concepts of the model are introduced. Vehicle routing and allocation are studied in Section 3. Section 4 describes the simulator. In Section 5, a collection of simulation results are given, and Section 6 concludes the paper.

SIMUTools 2010 March 15–19, Torremolinos, Malaga, Spain.

1.1 Related work

Most studies related to vehicle routing problems focus on the static case and relatively few results for dynamic problems have been reported. The main solution concepts related to the dynamic VRP, some of which apply to the dynamic VRPPD, are examined in [20, 11, 8]. Typically a special case of the problem, namely the online dial-a-ride problem (OlDARP) without time constraints, is considered [6]. Only a few analytical results for this problem have been reported.

In [5], it is shown that the competetive ratio[1] of any deterministic algorithm is at least 2 for minimizing route duration and at least $1+\sqrt{2}$ for minimizing the sum of service times. Additionally, an algorithm for the first objective with competetive ratio 2 is given for the special case of infinite capacity vehicles, An improved algorithm achieving the competetive ratio of 2 with an arbitrary capacity is given in [1]. In [13], a special case of the OlDARP is studied, in which only the pickup point is initially revealed. The authors prove that the competitive ratio for this problem is at least 3 and also give an algorithm which achieves this.

The first exact algorithm for the dynamic single vehicle DARP was introduced in [19]. Because of the inherent complexity of the problem, most of the recent studies related to the dynamic VRPPD resort to heuristic approaches. In [23], lower bounds and constant-factor policies for the uncapacitated multiple vehicle version of the problem are presented. In [21], an algorithm for a real-life multiple vehicle VRPPD with capacity and time constraints is described. A specially tailored objective function for a dynamic environment is used instead of the objective function of the static problem. Motivated by courier services, a heuristic for a dynamic uncapacitated VRPPD with time windows is proposed in [17]. The authors report that the total distance traveled may be reduced by an appropriate waiting strategy. An improved algorithm is given in [16]. In [7], a tabu search algorithm for a similar problem is presented and applied to problems with up to 33 requests. In [22], an algorithm for a capacitated dynamic VRPPD is proposed. Computational results indicate that the performance of the algorithm can be improved by up to 22% when information on future requests is taken into account. Finally, in [9], two algorithms for an uncapacitated dynamic VRPPD are described. Computational tests show that by estimating future requests, an improvement of 11-69% can be achieved for both algorithms.

As already stated, this paper focuses on a large scale dynamic VRPPD involving at least several hundreds of vehicles. We introduce a general solution concept particularly tailored for this type of environment and different policies for handling customers with a sub-second interarrival time. The effect of the rate of trip requests and the choice of the solution policy on travel time, distance driven per customer and waiting time are evaluated by means of computational experiments. In addition, several analytical observations related to the level of service and system performance in a dynamic large scale transportation system are presented.

2. MODEL

We consider an abstract model where *trip requests* arrive according to a Poisson process with rate λ [trip/s], and that for each trip request both the pickup and delivery locations are uniformly distributed in a finite convex region with area A (cf. Poisson point process). We have *a fleet of n vehicles* each with c passenger seats in order to support the given transportation demand in online fashion. We assume Euclidean distances between any two points and thus each vehicle uses *direct path* between the waypoints that define the route. When a trip request arrives, it is immediately assigned to a single vehicle. The chosen vehicle then, at some point of time, picks up the passenger for delivery to the corresponding destination. With $c > 1$ several passengers can share a vehicle in time, which allows one to combine trips and decrease the effort per passenger.

For simplicity, we assume a constant velocity v (e.g., 36 km/h) and a constant (minimum) stop time of t_{st} (e.g., 30 s). After the stop time, passengers can enter and exit the vehicle until the vehicle starts moving again. The stop time is assumed to include deceleration and acceleration of the vehicle. Note that in our trip demand model (Poisson point process), each stop corresponds to exactly one passenger entering or exiting the vehicle at a time, i.e., passengers do not share the stops (cf., door-to-door vs. stop-to-stop service).

We study the performance of such a model when the number of vehicles is large and demand is high with the goal of developing efficient ways to operate the vehicle fleet. Although the model is presented here in the passenger transportation context, the problem itself covers also other applications such as on-demand parcel or message delivery [23].

2.1 Heuristic policies

Let α denote *the policy*, which (i) decides on which vehicle a new customer is allocated to, and (ii) decides on the vehicle's route in dynamic fashion based on the current state of the system. The optimal way to operate a fleet of n vehicles is a very difficult problem, and to start with, requires defining a satisfactory balance between the two conflicting goals: the level of service and the system's efforts.

In this paper, we limit our approach to policies that can be expressed in terms of a cost function $f(\cdot)$ in the following way. A cost function can be evaluated for any given route ξ, which is defined by a sequence of waypoints, i.e., pickup and delivery locations. Also existing state information can be included in the cost function, e.g., in the form of the existing route of a vehicle, denoted by ξ_{old}. Different routes can be evaluated for each vehicle. The new passenger is allocated to the vehicle with the lowest cost route. The particular vehicle also switches to the new route immediately, i.e., the routes are constantly changing in response to the trip requests.

We investigate the performance of the system with three simple but intuitive heuristic policies. Let $t(\xi)$ be the route length (in time) and let $d(\xi)$ be the total remaining travel time of all customers assigned to the vehicle (waiting for pickup or traveling in the vehicle) according to the planned route. For each trip request, the new trip is added to the route of a vehicle with respect to the following criteria:

- `min-RD` (minimum route duration) assigns the new trip to the vehicle which can deliver both the new and its existing passengers fastest:

$$f(\xi, \xi_{\text{old}}) = t(\xi).$$

- `min-ΔRD` (minimum difference in route duration) chooses the vehicle (and route) which can serve the new trip

[1]Online algorithm has a competetive ratio α if its performance for any input is at most α times worse than that of an optimal offline algorithm.

request with the smallest additional effort (in time):

$$f(\xi, \xi_{\text{old}}) = t(\xi) - t(\xi_{\text{old}}).$$

- `min-ΔST` (minimum difference in system times) represents the difference between the sum of the passengers' system (sojourn) times before and after inclusion of the new trip request:

$$f(\xi, \xi_{\text{old}}) = d(\xi) - d(\xi_{\text{old}}).$$

Note that global information on the system is utilized only in the allocation phase; the route evaluation can be done locally for each vehicle. Note also that `min-RD` implements a some kind load balancing between the vehicles.

2.2 Performance

2.2.1 Passenger service level

Given a policy, the stochastic system is, in principle, well-defined and one can study its various statistics. In a stable system, each passenger first waits until the assigned vehicle arrives, and then travels in the vehicle to her destination, possibly via an indirect zig-zag route due to the other passengers assigned to the same vehicle. Let the random variable $W_i = W_i(\alpha)$ denote the *waiting time* of customer i, i.e., the length of the time interval from the request until the passenger enters the vehicle. Similarly, let $T_i = T_i(\alpha)$ denote the time customer i spends inside a vehicle (ride time), and $X_i = X_i(\alpha)$ the total distance she travels. Now,

$$\begin{aligned} W_i &> t_{\text{st}} \quad \text{with probability of 1,} \\ T_i &\geq X_i/v + t_{\text{st}}, \\ S_i &= W_i + T_i, \end{aligned}$$

where, $S_i = S_i(\alpha)$ is the sojourn time or *latency* of customer i. We are interested in the mean values, $\mathrm{E}[W]$, $\mathrm{E}[T]$, $\mathrm{E}[S]$, and $\mathrm{E}[X]$ as they describe how efficiently the system works from the passenger point of view. In particular, the mean latency $\mathrm{E}[S]$ is important as it defines the *transportation capability* of the system as observed by the customers. Note that according to the Little's result [14], e.g., the mean number of passengers in the system is $\lambda \cdot \mathrm{E}[S]$.

2.2.2 System performance

In addition to the above metrics, one can study the situation from the vehicles' point of view. In our model, the vehicle is either moving at velocity v, or stopped. Thus, its activity corresponds to an ON/OFF process. Let $B_i^{(j)}$ denote the duration of the ith leg of vehicle j, and $I_i^{(j)}$ the succeeding stopping time, which is at least t_{st}. Assuming policy α treats all vehicles equally, we can basically focus on the mean values $\mathrm{E}[B]$ and $\mathrm{E}[I]$, which provide us all the long term (average) quantities. Firstly, in a stable system the average customer flow in and out are equal:

$$\lambda = \frac{(1/2)n}{\mathrm{E}[B] + \mathrm{E}[I]},$$

where n denotes the number of vehicles and the factor of $1/2$ is due to the fact that only every second stop corresponds to a departing customer. Note that the above assumes that an empty vehicle stays where the last customer is delivered. Substituting $\mathrm{E}[I] \geq t_{\text{st}}$ and $\mathrm{E}[B] \geq 0$ in the above yields a *capacity constraint*,

$$\lambda_{\max} \leq \frac{n}{2\,t_{\text{st}}}. \tag{1}$$

Similarly, the *mean number of moving vehicles* is given by

$$\mathrm{E}[\text{moving}] = \frac{n\,\mathrm{E}[B]}{\mathrm{E}[B] + \mathrm{E}[I]}. \tag{2}$$

A key performance metric for the system is the *mean distance driven per passenger* and denoted by τ. The average collective velocity of the fleet is $n\mathrm{E}[B]/(\mathrm{E}[B]+\mathrm{E}[I])\cdot v$ [m/s]. Passengers rate was λ [1/s], and thus

$$\tau = \frac{n\,v\,\mathrm{E}[B]}{\lambda\,(\mathrm{E}[B] + \mathrm{E}[I])}, \tag{3}$$

i.e., the amount of work (in metres) the fleet conducts in order to fulfil a single trip on average. Note that the τ is constrained not only by the speed of the vehicles but also by the time the vehicle is stopped. A useful way of characterizing the "resource" of the system is to consider the *effective speed* of a vehicle. Assume that the vehicle is under such a demand that it never stays stopped longer than the minimum time for a passenger to enter or exit the vehicle. Each passenger requires a pickup and delivery resulting in a delay of $2t_{\text{st}}$. Given that each vehicle serves on average λ/n requests per unit time, we have for the effective speed,

$$v_{\text{eff}} \leq v\left(1 - \frac{\lambda 2 t_{\text{st}}}{n}\right).$$

Consequently, the distance driven per passenger is constrained by the *effective speed bound*:

$$\tau = \frac{n\,v_{\text{eff}}}{\lambda} \leq \left(\frac{n}{\lambda} - 2\,t_{\text{st}}\right) v. \tag{4}$$

3. ROUTING AND ALLOCATION

Recall that when a trip is ordered from the system it is immediately allocated a vehicle that will handle the trip. This decomposes the policy into two subpolicies, vehicle allocation and routing. In this paper we limit ourselves to policies where allocation and routing decisions are based on cost functions: When a trip request arrives all vehicles compute a candidate route based on a cost function and then the vehicle with the lowest cost route is selected.

In this case the allocation is simple and scales linearly to the number of vehicles, but computing the candidate route is somewhat more demanding. The problem is inherently combinatorial in nature; when a new trip request arrives the vehicle needs to combine the new pickup and delivery with the previously routed requests, i.e., with the existing route. In principle, this requires enumeration of possible routes and evaluating the cost function in each one of them. We assume that the route of a vehicle is simply defined as the order of (pickup and delivery) waypoints that the vehicle passes by. Possible routes are limited only by the fact that a pickup must take place before the corresponding delivery.

In this section we describe two experiments that aim to shed light on two important questions that arise from our approach. First, by locking the vehicle and the trip already at the arrival of the request we achieve several benefits; (i) decomposition - the vehicles can independently solve their routing problem without consulting with other vehicles, (ii) the solution space becomes significantly smaller thus facilitating the computational burden, and (iii) customer can be immediately notified the identity of the vehicle that handles the request. However, these benefits come with a performance cost. This question will be elaborated in Section 3.1.

In the literature, this kind of routing problems are often solved by *insertion heuristics*, in which a new trip is added in such a way that the relative order of the existing waypoints is preserved. Although such an approach enables a polynomial running time, the associated performance loss seems inevitable. Somewhat surprisingly, for large systems the performance loss appears to be neglible, as will be motivated in Section 3.2 and evaluated by simulations in Section 5.4.

3.1 Immediate allocation

A routing and allocation policy, in which a vehicle is fixed for each customer at the release time of the request, will not generally perform as well as a policy in which customers may be exchanged between vehicles at any instant. This is due to the fact that the appearance of a new customer may render some of the existing customer-vehicle assignments suboptimal. In the following examination, we will estimate the performance decrement due to immediate allocation policy.

For simplicity, let us study a static vehicle routing problem involving two vehicles, in which a single service point chosen randomly from the unit square is associated to each customer. The goal is to assign the customers to the two vehicles and generate routes for the vehicles in a way that the total route length is minimized. We will compare the difference between (i) the optimal solution, in which all possible partitionings of customers among the two vehicles are considered and (ii) a solution based on sequential allocation, in which the customers are assigned to the vehicles one by one. In the second method used to model the immediate allocation policy, the vehicle for which the increase in the route length is minimized, is selected for each customer. Comparing the two solutions gives us an idea on how the immediate allocation method might perform in a dynamic setting, compared to a solution method in which customers may be exchanged between vehicles at any instant.

The relative increase in the total route length of the two vehicles obtained by the sequential allocation method compared to the exact solution is shown in Figure 1. The solid line represents the sequential allocation method in which the customers are assigned in a random order. The relative deviations were computed by means of the formula $\sum_k r_k / \sum_k e_k - 1$, where $\sum_k r_k$ and $\sum_k e_k$ denote the sums of total route lengths acquired by the random order allocation and exact offline algorithms in 500 runs. The dotted and dashed lines represent the corresponding relative deviations obtained by choosing the best and worst assignment orders (out of 100 randomized orders for each run) of customers.

The curves in the figure indicate that the difference between the exact solution and the sequential allocation method increases with the number of customers. The worst ordering produces an increase of approximately 10% on the total route length for the problem involving seven customers. On the other hand, with the best ordering of customers, the increase compared to the static solution is less than 0.5% for each of the studied problems. Generally, it can be stated that the order in which the customers are assigned to vehicles has a substantial effect on the performance of the sequential allocation method. However, even if the customers were assigned in the worst possible order, the increase compared to the optimal solution is limited to relatively small values.

In this paper we focus on modeling services in which each customer is given an instant response and the allocation has

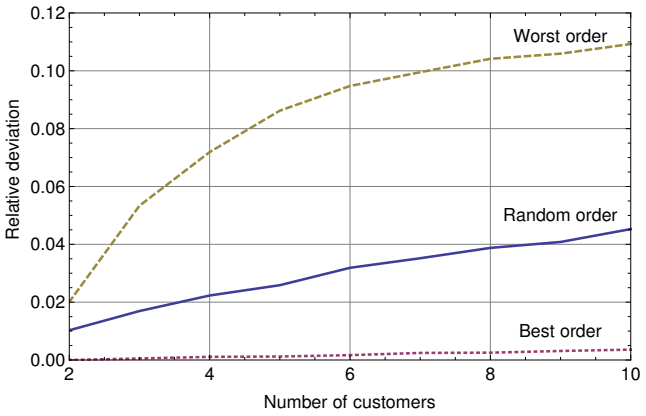


Figure 1: Difference between the solutions to a vehicle routing problem with two vehicles obtained by the exact and sequential allocation methods. The solid line represents the mean relative increase in route length, when the customers are assigned to vehicles in a random order, over 500 runs. The dotted and dashed lines represent the corresponding increase produced by the best and worst assignment orders of customers (out of 100 orders for each run).

to be executed immediately upon a request. Thus, the allocation order is defined by the arrival process and cannot be optimized. Nevertheless, the above results suggest that by freely exchanging customers among vehicles, only a relatively small improvement in performance can be achieved. In other words, immediate allocation can be considered as a relatively efficient method for solving dynamic problems.

3.2 Insertion vs. enumeration

Clearly, in addition to the allocation order, the quality of the solutions produced by the immediate allocation procedure is strongly dependent on the algorithm used to solve the route for each vehicle. A complete enumeration algorithm, as used in the previous experiment, will in general produce the best possible results with respect to a given routing and vehicle selection policy. In some situations, however, the use of such an algorithm may not be feasible since it would require too much computational work. In the following experiment, the difference between the solutions produced by an exact single vehicle algorithm and the intuitive insertion algorithm is evaluated.

More specifically, we will study the effect of problem size on the difference between the performance of the insertion algorithm and the exact algorithm. The algorithms are tested on a static vehicle routing problem involving 1 to 5 vehicles and 1 to 10 customers per vehicle. Similarly as in the previous experiment, the immediate allocation policy is used. The total route length obtained by the two algorithms with respect to the number of customers per vehicle is shown in Figure 2. Table 1 shows the relative increase in the total route length acquired by using the insertion heuristic with 10 customers per vehicle.

In general, the results indicate that the performance of the insertion heuristic decreases as the number of customers per vehicle increases. Since the curves representing the insertion and enumeration algorithms converge when the problem size

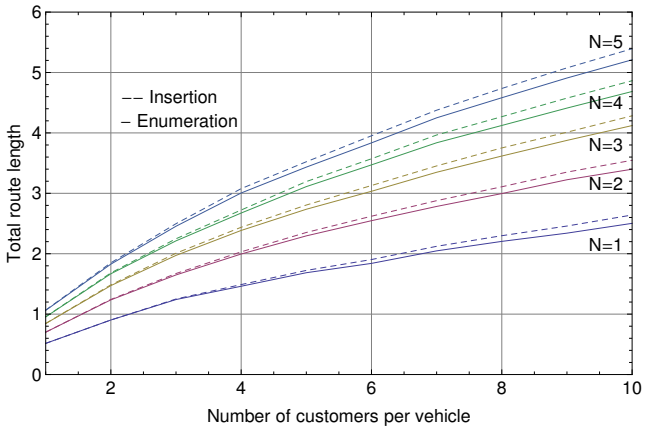


Figure 2: Difference between the total route length obtained by the enumeration and insertion algorithms. The difference between the algorithms increases with the number of customers per vehicle.

customers per vehicle = 10					
number of vehicles	1	2	3	4	5
increase (%)	5.5	4.3	3.9	3.7	3.5

Table 1: Relative increase in the total route length obtained by comparing the insertion to the enumeration with 10 customers per vehicle. Performance loss decreases as the number of vehicles increases.

is decreased, the use of the classical insertion heuristic can be seen to be well motivated in problems in which the number of customers assigned to a single vehicle is sufficiently small at any instant. This observation is in line with the fact that the insertion algorithm is asymptotically optimal. Indeed, when the number of customers assigned to a single vehicle is limited to 2 or less, the insertion algorithm goes through all possible permutations (even if both a pickup and a delivery node were associated to each customer).

Furthermore, by looking at Table 1, it can be seen that the gap between the two algorithms is slightly decreased as the number of vehicles is increased. Thus, it may be suggested that the insertion algorithm will perform well in problems in which the number of vehicles is large. We will return to this question with simulation results in Section 5.4.

4. SIMULATOR DESIGN

In this section we will briefly describe the design of our simulator tool and some important implementation decisions that provide us reasonably fast simulation times. As mentioned, our goal is to study a system with a large number of vehicles n (e.g., $n = 500$) and a high demand of trips. This means a relatively complicated system with a huge number of state information and internal constraints that must be checked constantly throughout the simulation. Thus, even if the policy deciding on vehicle assignment and routes was computationally lightweight, simulating the system with constant parameters for a sufficiently long time period can easily take infeasible amount of real time if the simulator is not implemented efficiently.

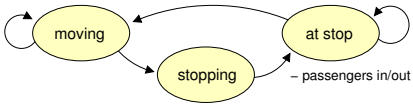


Figure 3: State of each vehicle follows the depicted pattern. The "stopping" state is deterministic delay that models braking, acceleration and delay due to the boarding and alighting passengers. Self-transitions correspond to plan changes.

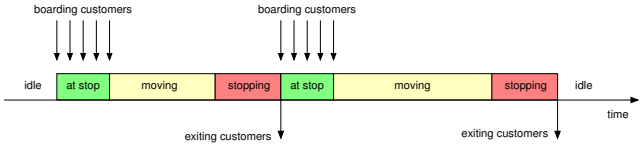


Figure 4: Realization of vehicle's state as a function of time. The duration of "stopping" state is a constant, while "at stop" state can have a zero duration.

However, the policies are hardly computationally light. The core decision each vehicle must do is to decide on route if a new trip were assigned to it. Unfortunately, the number of feasible routes (at each point in time) can be very large. For example, if a vehicle is full and has, say, $c = 10$ passengers and they can be dropped in $10! = 3628800$ different orders. Adding the new trip and the passengers already waiting for a pickup (and delivery) increases the complexity of the problem even further. Considering that each vehicle must evaluate the routes for all arriving trip requests, even simulating performance of one policy at a single load level can be an overwhelming task. Some policies may also contain parameters to be optimized, which sets even higher requirements for the simulation speed.

In order to have a maximal simulation speed, we decided to implement our simulator from scratch using standard C. This approach has also other benefits: (i) The simulator is relatively compact as there are no unnecessary features. (ii) The architecture can be tailored to match our objectives. (iii) Standard C implementation makes the simulator highly portable (we are using it in both Linux and Windows systems). On the downside, we had to re-implement some standard components available in almost any simulation library such as pseudo random number generation and event based scheduling. This, however, is a straightforward task, e.g., by following the steps laid out in [15, 2, 12]. Next we will describe the essential features of our simulator architecture.

4.1 State Description

4.1.1 Passengers

The life of a passenger in our system is very similar to jobs in a queueing system. Upon arrival a passenger is either (i) *accepted* and assigned to some vehicle, or (ii) *rejected*. The accepted customers first (iii) *wait for the pickup*. Next the customer (iv) *enters the vehicle* and the system starts to actually process her transportation need with the difference that here it is also possible to conduct "negative work" by moving the passenger further away from her destination. Finally, at some point in time, the vehicle reaches the destination and passenger (v) *exits the system*.

4.1.2 Vehicles

The number of vehicles in our model is assumed to be a constant n, and each vehicle is in one of the following three states: (i) *At stop* waiting for (more) passengers, (ii) *Moving* towards the next waypoint, and (iii) *Stopping* phase after a transition.

This is illustrated in Figures 3-4. Note that the state "stopping" includes braking, acceleration and other unavoidable delays per stop. The state "at stop" means the vehicle is free to go at any moment. Also the bookkeeping of boarding and alighting passengers occurs in this state. In particular, the duration of the stoppping state can be zero, except when the vehicle extends the stay for some reason, e.g., when it is idle. In addition to this, each vehicle maintains an ordered list of waypoints (worklist), which defines its current route. This list can be modified in response to each new customer with exception of the first waypoint in case the vehicle has started the corresponding stopping state. This is illustrated with the self-transitions in Figure 3.

4.2 Policy Architecture

As discussed previously, we define policy α by means of cost functions. That is, each vehicle is basically asked (to estimate) how much it costs if a new trip request is assigned to it. In order to estimate the vehicle specific cost, one generally needs to consider some set of possible routes. To this end, the simulator provides an unified interface which separates the route enumeration and the cost function evaluation. Such a cost function f then defines policy α if we always choose the vehicle and the route which yields the lowest cost at that point in time. This decision chain is illustrated in Figure 5. We observe a modular design, where each block can be replaced without modifying the others.

4.2.1 Arrival process

Arrival process module is responsible for generating the trip requests. In this paper, we assume a Poisson point process in a circular area, while the simulator design allows any other more specific arrival process and area.

4.2.2 Policy handler

This module collects the information (cost estimates) from the vehicles and then assigns the new trip to the most suitable vehicle. That is, the *vehicle allocation* occurs at this point. Note that the example policies considered in this paper are all such that the global information is only available at this point. That is, each vehicle independently evaluates the cost incurred if the given trip is assigned to it. However, the modular design of the simulator allows that also the vehicles can take into account global information if seen relevant in the particular case.

4.2.3 Route enumeration

The simulator provides several options to define the subset of all routes to be considered upon a new trip request:

1. *taxi*: New trip is inserted in such a way that no two trips share the vehicle at the same time.
2. *insertion*: New trip is inserted in such a way that the relative order of the existing waypoints is unchanged.
3. *all*: Consider all orders of the waypoints, exhaustive and thus the number of potential routes may be huge.

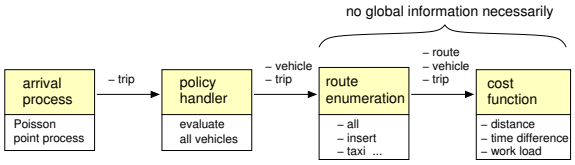


Figure 5: Modular decision chain from the request generation to evaluation of the cost function.

Clearly, $taxi \subset insertion \subset all$. The *taxi* subset limits the performance severely. However, in large scale system with a large number of vehicles, as we discussed previously and will also show in numerical experiments, the *insertion* and *all* provide a similar performance level.

With *insertion* and *all*, it is important to prune infeasible routes efficiently. To this end, we have two types of constraints: (i) *time constraints* that are typically deadlines for arriving to each waypoint, and (ii) *capacity and order constraints*, i.e., a vehicle may not pickup more passengers than its capacity allows, and pickup must be before the corresponding delivery. The time constraints can be given either "external", e.g., one can require that each customer must be picked up within 10 minutes from the request, or they can be based on the currently known best route when evaluating the routes (policy specific).

Assume that while enumerating the routes at some stage we have fixed the first k waypoints and the task is to choose the next. If some remaining waypoint x cannot be reached in time anymore, then all routes with this initial sequence are infeasible and can be excluded. The same does not hold for capacity or order constraint. Unlike time, the occupation can still (and will) decrease. Similarly, the pickup can be scheduled before the corresponding delivery. In summary, multiple ways to prune the set of routes efficiently exist, of which the (potential) time constraints are more "definitive" due to the nature of time.

4.2.4 Cost function

This function returns some real number corresponding to the relative cost of the given routing decision. As discussed previously, with the `min-RD` policy the idea was to minimize the planning horizon. Thus, the time instance $t(\xi)$ representing the time when given vehicle becomes empty is returned when the cost function is called. Note that the modular design enables fast prototyping of new policies without a need to re-implement, e.g., the route enumeration repeatedly.

4.3 Running the simulator

For each simulation run the user must specify (i) the area and passenger arrival rate, (ii) the number of vehicles, their capacity, velocity and the stopping time, and (iii) the fleet operating policy. Additionally one must also decide on the simulation and warm-up periods. All these parameters can be conveniently tuned from a command line interface.

By default the simulator outputs a summary report, which includes all the main performance quantities (essentially totals and mean values). However, when necessary, one can also enable various log-files to which more detailed information during the simulation run is written. The most important are perhaps the vehicle log file (actions taken by the vehicles) and trip log file (per trip information). Based on these log-files one can obtain, e.g., waiting and travelling

trip request rate:	2/s
area:	disk with 5km radius
vehicles:	500, each with 10 seats
velocity:	10 m/s
stopping time:	30 s

Table 2: Basic simulation parameters.

time distributions, or study how the direct trip length affects the realized trip length and the number of additional stops. This type of information is vital when one is, e.g., developing better policies.

4.3.1 Validation

Before proceeding further with performance evaluation, it is important to ensure that the simulator works correctly. Simulation results to this end are in good agreement, e.g., with the following analytical observations, which support the validity of our implementation:

- Mean trip request length can be computed analytically (cf., random waypoint mobility model [3, 18, 10]), which for Table 2 disk area gives about 4527m. This is used to assess the module generating the trip request.
- Simulating a single vehicle when $\lambda \to 0$ converges to a system where each arriving customer observes the single vehicle idle. Thus, the work the single vehicle does, assuming a work conservative policy, is two times higher than the direct trips.
- Similarly, with a low demand, a huge number of vehicles, and a policy that assigns the trip to the nearest (idle) vehicle yields a system where the ratio of vehicle kilometres to passenger kilometres approaches one.
- With a high demand and efficient policies the driven kilometers per passenger approach the bound (4).

4.3.2 Simulation speed

Simulation speed turned out to be more than satisfactory for our purposes. For example, it takes only about 5 minutes of real time to simulate a 10 hour time interval with Table 2 parameters using a standard PC. This already corresponds to a rather high load as an average 72000 passengers are processed. Further speed improvements can be obtained by code optimization and multi-threading. Indeed, both our problem and the design of the simulator lend themselves well to parallel computation. For example, the vehicle specific relative costs can be computed in parallel.

5. NUMERICAL ANALYSES

In this section we demonstrate the simulator and investigate the performance of the defined policies. We use the simulation parameters defined in Table 2, unless otherwise mentioned. We omit the specific results of the `min-ΔRD` policy as it turns out to perform particularly badly in this case resulting in mean travel times of several hours, but use it instead as a component of a parameterized policy.

5.1 Warm-up period

When the simulation is started from a state where all vehicles are empty, the initial state has a strong but transient effect on the behavior of the system. In order to get some idea about the length of the initial transient we next ran 10000 experiments using the `min-RD` policy. The result is shown in Figure 6, where the initially highest curve corresponds to the *passenger arriving rate*, the middle curve to the *passenger pickup rate*, and the lowest to the *passenger delivery rate*. The pickup rate catches the arrival rate relatively fast, while it takes a somewhat longer time before the system's output rate (alighting passengers) stabilizes. In particular, we observe that at least a 2 hour warm-up period should be used in this case. In order to be on a safe side, in the following experiments we use 10 hour warm-up period.[2] Moreover, note that the area between the passenger arriving rate and the passenger pickup rate curves corresponds to the mean number of waiting customers in the equilibrium, $\mathrm{E}[N_\mathrm{w}]$. Similarly, the area between the pickup and delivery rates is equal to the mean number of customers in the vehicles, $\mathrm{E}[N_\mathrm{v}]$. Thus,

$$\mathrm{E}[N_\mathrm{w}] = \int_0^\infty \lambda - p(t)\,dt, \text{ and } \mathrm{E}[N_\mathrm{v}] = \int_0^\infty p(t) - d(t)\,dt,$$

where $p(t)$ denotes the expected pickup rate at time t, and $d(t)$ the mean delivery rate at time t. Obviously, $\mathrm{E}[N_\mathrm{v}]$ is bounded from the above by the total capacity of the fleet.

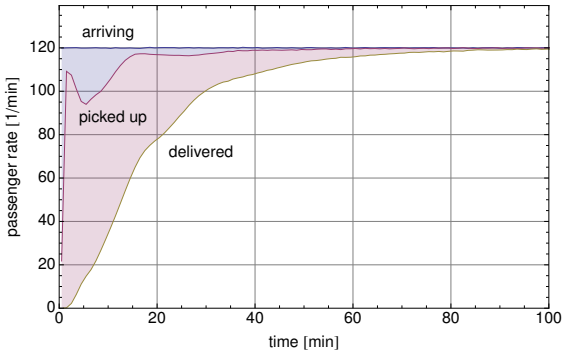


Figure 6: Mean rate of arriving, picked up and delivered passengers over 10000 simulation runs starting from an empty system. The initial transient period is order of 2 hours.

5.2 Waiting time distribution

The simulator provides various log-files, from which one can extract more detailed statistics when necessary. Figures 7-8 illustrate the empirical waiting time distribution as observed by the passengers when the fleet is operated according to the `min-RD` and `min-ΔST` policies. We observe that the `min-RD` policy picks up the passengers somewhat quicker than the `min-ΔST`. Passenger arrival rate $\lambda = 2/\mathrm{s}$ is already a rather high demand for the given $n = 500$ vehicles to handle (see Section 5.4). Despite of this, the waiting times are actually reasonable and most passengers are picked up within 5 minutes, and only very few have to wait more than 15 minutes, as shown in Table 3.

5.3 Latency vs. trip distance

Waiting time does not necessary depend on the trip distance. In contrast, the latency, i.e., the time from the re-

[2]Even though in real-life the daily 24 hour rhythm implies that the trip demand will not be constant for 10 hours.

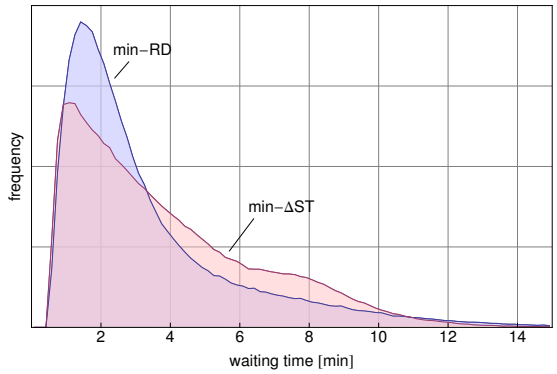


Figure 7: Waiting time distribution for `min-RD` and `min-ΔST` policies with Table 2 parameters. Most passengers wait only few minutes before the pickup.

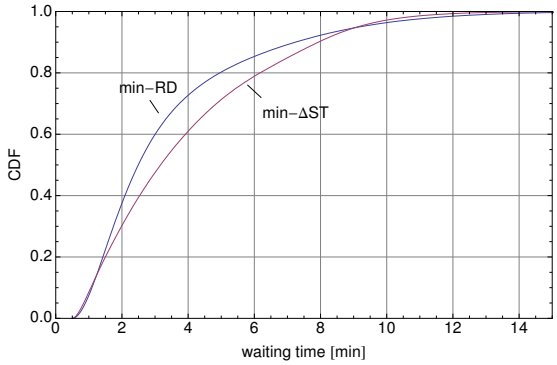


Figure 8: CDF of the empirical waiting time distribution for `min-RD` and `min-ΔST` policies with Table 2 parameters.

quest until the delivery in general correlates strongly with the trip distance due to the finite velocity of the vehicles. Figure 9 illustrates how the mean waiting time and latency depend on the trip distance. The x-axis corresponds to the direct distance in kilometres. On the y-axis we have both the waiting time and the latency in minutes. The dashed line depicts the latency assuming private cars. The `min-ΔST` policy clearly outperforms the `min-RD`. Also note that `min-RD` gives a higher priority to long trips as such trips tend to extend the planning horizon further than the short trips. This, however, may not always be an optimal strategy, and indeed, with `min-ΔST` the situation appears to be more fair.

5.4 Insertion vs. enumeration

In Section 3.2 we motivated that in a large system with many vehicles there may not be need to enumerate all routes but a simple insertion heuristic would perform nearly as well. Figure 10 represents the mean travel time as a function of load, computed for both the insertion heuristic (dashed curves) and the full enumeration (markers) with `min-RD` and `min-ΔST` policies. As expected, in this large system with 500 vehicles, the difference between the insertion and enumeration methods is almost negligible. Looking at the distance driven per passenger metric we observe the same result. We omit the illustration for brevity.

policy	mean	tail P$\{W > t\}$		
	E[W]	5min	10min	15min
`min-RD`	3min 25sec	19.8%	3.8%	0.4%
`min-ΔST`	3min 54sec	28.8%	2.9%	0.1%

Table 3: Tail probabilities of the waiting time.

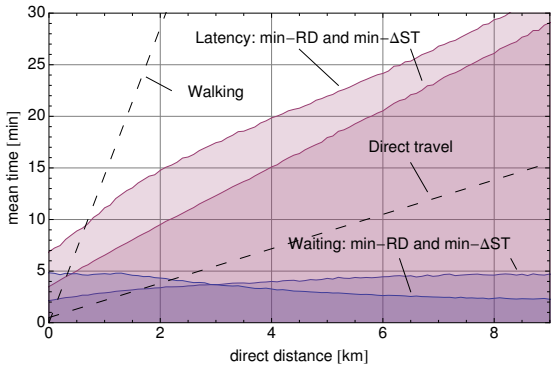


Figure 9: Empirical mean waiting time and latency (sojourn time) with `min-RD` and `min-ΔST` policies conditioned on the direct distance of the trip. Walking time with 70m/min, and the direct driving time with a private car are illustrated with dashed lines.

5.5 Policy optimization

Viability of this kind of transportation system depends on the performance of the applied policies. Simulations are needed for experimenting with different policies. In this example we evaluate the performance of the system with `min-RD` and `min-ΔST`. In addition, we consider a parameterized policy, where the cost function for a trip request is defined as

$$f(\xi, \xi_{\text{old}}, p) = p \cdot \big(t(\xi) - t(\xi_{\text{old}})\big) + (1-p)\big(d(\xi) - d(\xi_{\text{old}})\big),$$

where the policy parameter p defines the weighting between the two objectives (i.e., the policy has $100p\%$ of `min-ΔRD` and $100(1-p)\%$ of `min-ΔST`) and is subject to optimization. Clearly, this policy represents a balance between the driven distance and passenger travel time; `min-ΔRD` attempts to add the new trip request so that the additional work is as small as possible, whereas `min-ΔST` greedily minimizes the sum of all travel times in the system. Note that `min-ΔRD` by itself causes excessive delays in this setting (of the order of several hours) and is not considered here. In optimizing a policy parameter an efficient simulator is invaluable as each parameter adds a new dimension to the parameter space.

Figure 11 shows the mean travel time and distance driven per passenger as a function of load for different values of the policy parameter p. We observe that even a small increase in the mean travel time can be efficiently converted into saved distance and in some cases also the travel time decreases even below the `min-ΔST` profile (shown as an opaque box in the figure). Including an `min-ΔRD` component into the policy allows us to take into account the additional work induced by the new trip request. By allocating the trip to a vehicle that can handle the trip with little additional time the overall performance can be improved.

To show the results in more traditional form, let us select the policy with the parameter value $p = 0.4$ and study its

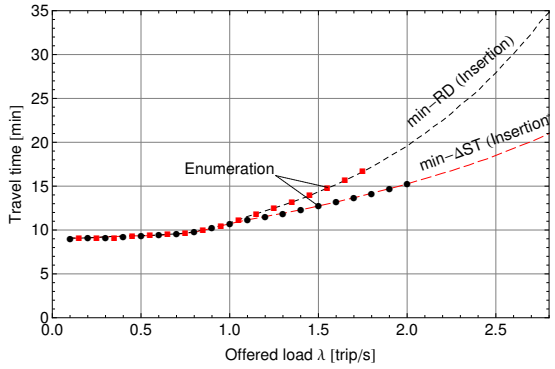


Figure 10: Travel time as a function of load. Insertion heuristic (lines) provide a scalable way of solving the routing subproblem without almost any difference in performance compared to the full enumeration (markers). Full enumeration becomes computationally infeasible at large load values.

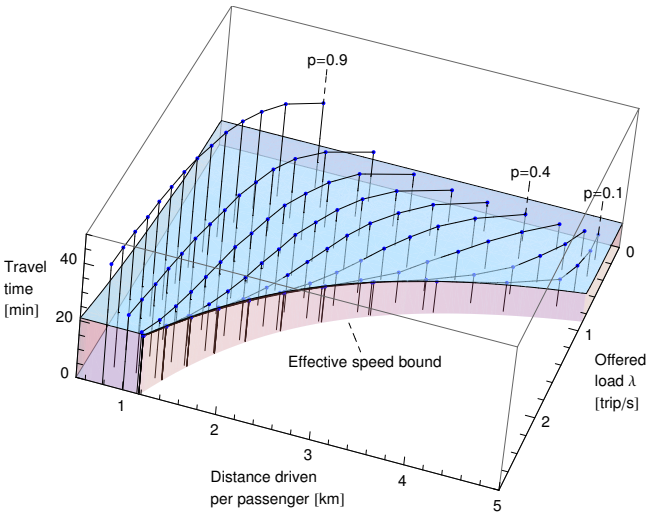


Figure 11: Performance of the system with different values of policy parameter p (cf. bound (4)).

performance in detail. Figure 12 shows the distance driven per passenger as a function of load and compares it to that of `min-ΔST` and `min-RD`. Whereas these comparison heuristics tend to start moving the vehicles even at low loads our parameterized policy $p = 0.4$ performs clearly more efficiently in this respect. The difference between mean direct distance and distance driven per passenger is coined as *the distance gain*, cf. the figure, and reflects the amount of kilometers saved per passenger by using this transportation system instead of serving the trip requests by private vehicles. All policies utilize the available kilometers quite quickly and afterwards follow the effective speed bound (4) very closely.

Figure 13 depicts the travel time (including waiting time) profiles of the compared policies. It can be seen that as soon as the distance approaches its upper bound the delays start to increase. `min-ΔST` shows rather moderate increase in the travel time, but `min-RD` performs significantly worse. The parameterized heuristic $p = 0.4$ is reasonably good especially

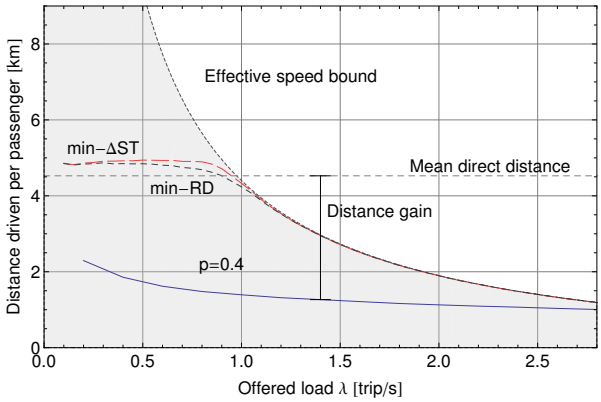


Figure 12: Driven distance per passenger as a function of offered load. The mean length of requests is 4.527km, which equals to the driven kilometers per passenger if private vehicles. Shaded area corresponds to the feasible region defined by (4).

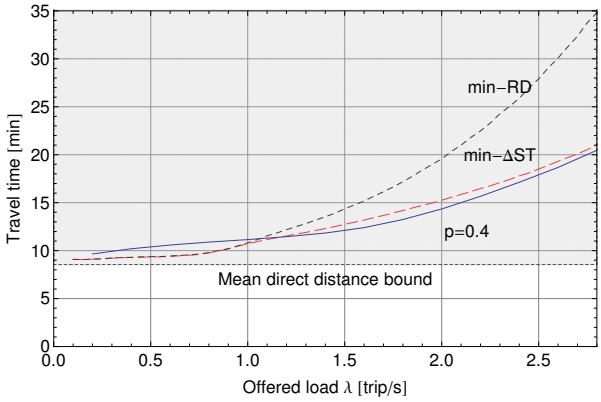


Figure 13: Mean travel time. Mean direct distance bound corresponds to the travel time in a system where all trips are driven in private vehicles.

at higher loads, which makes it a good compromise between the two conflicting objectives.

6. CONCLUSIONS

In this paper we have considered a dynamic behicle routing problem with pickups and deliveries. Our focus was on systems where a large number of vehicles are needed to support the transportation demand. As a particular feature of our system, a vehicle is assigned to each passenger immediately upon the trip request. In this context, we have described a specifically tailored simulator framework, that can be used to evaluate the transportation system and to rapidly prototype new operating policies for the vehicle fleet. The main objectives have been efficiency and modularity.

One of the most important design choices for a vehicle routing policy is the set of routes considered for each trip request. We have shown, both by means of analysis and simulation experiments, that in our context it is typically sufficient, without any significant loss in performance, to consider the insertion approach for route enumeration, where

the relative order of the earlier waypoints is always kept the same. That is, it is not necessary to enumerate *all* the feasible orders of waypoints per trip request and per vehicle, which indeed can take some time in a large system.

On the other hand, we have also demonstrated the viability of this type of transportation system. In general, there is a well-known trade-off between the work conducted (driven kilometers) and the level of the service (e.g., mean waiting times). However, our experiments suggest that if the passengers are willing to accept even a small average delay for their trips, in form of waiting time and/or a longer route, then the amount of work can be reduced considerably. That is, the transportation cost per trip can be reduced significantly.

7. ACKNOWLEDGMENTS

This work was conducted in Metropol project that is supported by the Finnish Funding Agency for Technology and Innovation, Finnish Ministry of Transport and Communications, Helsinki Metropolitan Area Council and Helsinki City Transport. The authors would like to thank Dr. Samuli Aalto, Mr. Teemu Sihvola and Prof. Jorma Virtamo for their invaluable comments while preparing this paper.

Correction notes

This is the corrected version of our Simutools 2010 paper. In the version that appears in Simutools 2010, the preliminary names of the heuristic control policies `min-RD` (`MRT`) and `min-`Δ`ST` (`MPTT`) are mistakenly used in Figures 7-10 and 12-13.

8. REFERENCES

[1] N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In *STACS 2000 Lecture Notes in Computer Science*, volume 1770, pages 639–650. Springer, Berlin, 2000.

[2] J. Banks, J. S. C. II, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Prentice-Hall International Series in Industrial and Systems Engineering. Prentice-Hall, third edition, 2001.

[3] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Trans. on Mobile Computing*, 2(3):257–269, Jul.–Sept. 2003.

[4] J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M. Savelsbergh. Transportation on demand. In *Transportation*, pages 429–466. Amsterdam: North-Holland, 2007a.

[5] E. Feuerstein and L. Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268:91–105, 2001.

[6] G.Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 2009. In press.

[7] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C*, 14:157–174, 2006.

[8] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151:1–11, 2003.

[9] G. Ghiani, E. Manni, A. Quaranta, and C. Triki. Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E*, 45:96–106, 2009.

[10] E. Hyytiä, P. Lassila, and J. Virtamo. Spatial node distribution of the random waypoint mobility model with applications. *IEEE Trans. on Mobile Computing*, 5(6):680–694, June 2006.

[11] A. Larsen. The dynamic vehicle routing problem. *PhD thesis, Technical University of Denmark*, 2000.

[12] P. L'Ecuyer. Random number generation. In *Handbook of Computational Statistics*, chapter 2, pages 35–70. Springer-Verlag, 2004.

[13] M. Lipmann, X. Lu, W. E. de Paepe, R. A. Sitters, and L. Stougie. On-line dial-a-ride problems under restricted information model. *Algorithmica*, 40:319–329, 2004.

[14] J. D. C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, (9):383/387, 1961.

[15] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1), Jan. 1998.

[16] S. Mitrovic-Minic, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38:669–685, 2004.

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

[18] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *IEEE Trans. on Mobile Computing*, 3(1):99–108, Jan-Mar 2004.

[19] H. Psaraftis. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–154, 1980.

[20] H. Psaraftis. Dynamic vehicle routing: status and prospects. *Annals of Operations Research*, 61, 1995.

[21] M. Savelsbergh and M. Sol. DRIVE: Dynamic routing of independent vehicles. *Operations Research*, 46, 1998.

[22] D. Sáez, C. Cortés, and A. Núñez. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers and Operations Research*, 35:3412–3438, 2008.

[23] H. Waisanen, D. Shah, and M. Dahleh. A dynamic pickup and delivery problem in mobile networks under information constraints. *IEEE Trans. on Automatic Control*, 53:1419–143, 2008.

Publication III

Lauri Häme, Jani-Pekka Jokinen, Reijo Sulonen. Modeling a competitive demand-responsive transport market. In *Kuhmo Nectar Conference on Transport Economics*, Stockholm, Sweden, June-July 2011.

Modeling a competitive demand-responsive transport market

Lauri Häme[1,*]

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Jani-Pekka Jokinen

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Reijo Sulonen

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Abstract

We present a model for a demand-responsive transport (DRT) service in which customers request trips from specified origins to specified destinations and a fleet of competing vehicles serves the customers. The trips are requested without pre-order times and the vehicle routes are generated in real time. Customers seek rides to minimize travel time and competing vehicles aim to maximize profit. We describe the demand and movement of vehicles by logit models and define the network equilibrium as a state in which the demand meets the supply of DRT trips. We show that an equilibrium always exists and provide an equilibration algorithm. Finally, we study the long-run behavior of competitive DRT by varying the number of vehicles and price. Examples suggest that (i) with no price regulation, free entry results in a taxi-type service and (ii) the optimal equilibrium from the customers' perspective is achieved by price regulation and free entry.

Keywords: Demand-responsive transport, Competitive market, Network equilibrium, Transportation network

1. Introduction

Demand responsive transport (DRT) is an advanced, user-oriented form of public transport involving flexible routing of small or medium sized vehicles between the pick-up and drop-off locations of customers. DRT can be seen as a passenger transport service between bus and taxi ranging from less formal community transportation to area-wide service networks (Mageean and Nelson, 2003). The major difference to taxi services is that customers with different pick-up and drop-off locations may share the same vehicle. The major difference to bus services is that

*Corresponding author

Email addresses: Lauri.Hame@tkk.fi (Lauri Häme), Jani-Pekka.Jokinen@tkk.fi (Jani-Pekka Jokinen), Reijo.Sulonen@tkk.fi (Reijo Sulonen)

[1]Tel.: +358 40 576 3585

the vehicle routes and schedules are generated in real time according to the trips requested by customers. Earlier studies suggest that DRT has potential to become a socially and economically remarkable form of public transportation (Mulley and Nelson, 2009; Sihvola et al., 2010; Jokinen et al., 2011). In addition, DRT is seen to be more effective than a fixed route service in minimizing emissions of pollutants (Diana et al., 2007).

Most existing DRT services provide door-to-door transportation for elderly or handicapped people and require customers to book trips at least one hour in advance (Cordeau et al., 2007a; Helsinki Region Traffic, 2010). Conventionally, the trips are organized centrally via *travel dispatch centers*, which have the capability of assigning customers to vehicles and optimizing the routes (Mageean and Nelson, 2003). In contrast to such centralized DRT services, we consider a competitive form of DRT in which each driver providing service attempts to maximize his/her profit. That is, the movement of vehicles is governed by the decisions of individual drivers, instead of a travel dispatch center controlled by a single transport operator. This market structure is in fact similar to conventional taxi-markets, which have been extensively studied, see for example (Hackner and Nyberg, 1995; Arnott, 1996; Cairns and Liston-Heyes, 1996; Flores-Guri, 2003; Lagos, 2003; Wong et al., 2005; Matsushima and Kobayashi, 2006; Fernandez et al., 2006; Moore and Balaker, 2006; Yang et al., 2002, 2005, 2010),

We introduce a network model to characterize the behavior of customers and vehicles in a competitive DRT market. The drivers providing service seek routes that serve as many customers per unit time as possible, and customers seek rides to minimize subjective trip price, defined as a combination of ticket price and travel time. Given these vehicle-customer behavior models, we calculate the *network equilibrium*, defined as a state in which the demand matches the supply of trips in each part of the network.

This approach is somewhat similar to the taxi model proposed by Yang et al. (2010). However, DRT is to some extent a more complex transportation service to model than a taxi service. The main difference is that in a taxi service, customers are delivered to their destinations directly, whereas in a DRT service, a vehicle can serve several customers simultaneously and therefore a customer's trip from an origin to a destination is not necessarily a direct one. From modeling perspective, this means that optimal decisions of customers and vehicles cannot be based only on customer waiting times and vehicle searching times as in the taxi equilibrium model of Yang et al. (2010). Basically, a DRT vehicle can make more profits by serving more customers per unit time, but the number of served customers also increases average travel times, which reduces customers' satisfaction and demand for the DRT service. Therefore, in our DRT model, the level of service of a trip is determined by the total travel time of the trip, which depends on the vehicles' routing decisions.

This work is partially motivated by a demand-responsive transport (DRT) service currently being planned to operate in Helsinki. Helsinki Region Transport board has approved a plan under which the trial period of the service takes place from 2012 to 2014. Similarly as the current flexible service routes (Helsinki Region Traffic, 2010), the new DRT service is designed to operate on a demand-responsive basis, that is, vehicle routes can be modified in real time in order to meet the demand efficiently. The main difference to existing services is that no pre-order times for trips are required and the trips can be booked "on the fly" by means of an interactive user interface.

The rest of this paper is organized as follows. In Section 2, we describe the behavior of

customers and the movement of vehicles. In Section 3, we define the network equilibrium for our model and present an algorithm for determining the equilibrium. In Section 4, we study the long-run market behavior as a function of the number of vehicles and price. The conclusions of this work are given in Section 5.

2. Model

Our model of competitive DRT is governed by the following preliminary assumptions.

1. There are N competing vehicles that produce trips between origins and destinations in a specific operating zone. Different trips may have different prices and travel times. A single vehicle can simultaneously serve several customers.
2. The subjective price of a trip is defined as a combination of ticket price and travel time. Customers choose between trips provided by DRT and a virtual mode by comparing the subjective prices of different trips.
3. Customers are assigned to vehicles by means of an electronic booking service. For analysis, we assume that the DRT customers that choose trip r are divided equally[2] among vehicles that produce trip r.

In the following, we will study the behavior of customers (Section 2.1) and drivers providing service (Section 2.2) in more detail.

2.1. Demand

Let us consider a set of *nodes* I representing the origins and destinations of customers in a specific operating zone. Each pair of nodes $(i, j) \in I \times I$ is associated with a specific *direct ride time* t_{ij}. Generally, the direct ride times are not symmetric, that is, t_{ij} and t_{ji} are not necessarily equal for $i, j \in I$.

A *trip* from an origin $i_0 \in I$ to a destination $i_d \in I$ is defined as an acyclic sequence of nodes $(i_0, i_1, \ldots, i_{d-1}, i_d)$ in I, that is, each node in the sequence appears in the sequence exactly once. When a customer takes a trip $(i_0, i_1, \ldots, i_{d-1}, i_d)$, the customer enters a vehicle at i_0, which visits the nodes $i_1, \ldots, i_{d-1}$ before the drop-off of the customer at i_d. That is, the trip denotes the path of the vehicle that transports the customer from i_0 to i_d. For example, a *direct* trip from $i_0 \in I$ to $i_d \in I$, denoted by (i_0, i_d), describes a trip in which a customer enters a vehicle at i_0 and the vehicle drives directly to i_d without stopping between i_0 and i_d. The set of all trips in I is denoted by $\mathcal{R}$ and the set of trips from $i \in I$ to $j \in I$ is denoted by $\mathcal{R}_{ij}$.

2.1.1. Subjective price

Each trip $r \in \mathcal{R}$ has a specific ticket price p_r. The ticket price may be defined equal for all trips with the same origin and destination, that is, for any $i, j \in I$ we have $p_r = p_{r'}$ for all $r, r' \in \mathcal{R}_{ij}$. This pricing idea is used in the numerical examples in Section 3.2. At this point, however, we consider the general case in which the prices of trips with the same origin and destination may be

[2] cf. mean value analysis (Reiser and Lavenberg, 1980).

different. Similarly as in (Yang et al., 2010), each customer seeks a trip to minimize the *subjective price*, defined as a combination of p_r and *travel time*.

The *ride time* of a trip $(i_0, \ldots, i_d) \in \mathcal{R}_{i_0 i_d}$ refers to the time spent inside a vehicle and is defined by

$$q^{\mathrm{DRT}}_{(i_0,\ldots,i_d)} = \sum_{k=1}^{d} t_{i_{k-1} i_k}. \tag{1}$$

Note that since the direct ride times t_{ij} are transitive, the ride time from i_0 to i_d is minimized in the direct trip (i_0, i_d).

For any pair of nodes $i, j \in I$, the demand responsive transport service produces different trips from i to j at different intervals. That is, there may be different *waiting times* for different trips. For example, if a direct trip (i, j) from i to j is produced two times per hour and a trip (i, k, j) is produced three times per hour, we expect that the average waiting times for the trips (i, j) and (i, k, j) satisfy $w^{\mathrm{DRT}}_{(i,j)} > w^{\mathrm{DRT}}_{(i,k,j)}$ (see Section 2.2.5 for a more detailed discussion on waiting times).

The average *travel time* for a trip $r \in \mathcal{R}_{ij}$ is defined as the sum of waiting time and ride time, that is,

$$t^{\mathrm{DRT}}_r = w^{\mathrm{DRT}}_r + q^{\mathrm{DRT}}_r. \tag{2}$$

The *subjective price* of a DRT trip $r \in \mathcal{R}_{ij}$ is given by

$$g^{\mathrm{DRT}}_r = p_r + \beta t^{\mathrm{DRT}}_r, \tag{3}$$

where p_r is the ticket price for the trip r and β is the customers' monetary value of unit travel time.

2.1.2. Demand for DRT

The choices of customers are determined by a logit model, similar to the one in (Yang et al., 2010), as follows. The subjective trip price of a trip from i to j provided by the *virtual mode*[3] is denoted by $\bar{g}_{ij}$. The probability that a customer traveling from i to j chooses a DRT trip $r \in \mathcal{R}_{ij}$ is defined by

$$P^{\mathrm{DRT}}_r = \frac{\exp(-\theta g^{\mathrm{DRT}}_r)}{\exp(-\theta \bar{g}_{ij}) + \sum_{r' \in \mathcal{R}_{ij}} \exp(-\theta g^{\mathrm{DRT}}_{r'})}, \tag{4}$$

where θ is a nonnegative parameter describing the uncertainty in transport services and demand from the perspective of customers. Note that when $\theta = 0$, the choice probability is equal for all alternatives. When $\theta \to \infty$, each customer chooses the option with the lowest subjective price with probability 1. Clearly, the above logit model has the property of independence from irrelevant alternatives, that is, the ratio $P_r / P_{r'}$ depends on the subjective prices of trips r and r' but not on the subjective prices of other trips (Small and Verhoef, 2007).

[3]The virtual mode represents alternatives for the DRT service.

The *total demand* from i to j is denoted by Q_{ij} and describes the number of customers willing to travel from i to j per unit time. The corresponding demand for a trip $r \in \mathcal{R}_{ij}$ is given by

$$Q_r^{\mathrm{DRT}} = Q_{ij} P_r^{\mathrm{DRT}}. \tag{5}$$

The *demand for DRT* describing the expected number of customers that choose to travel from i to j by DRT per unit time is given by

$$Q_{ij}^{\mathrm{DRT}} = \sum_{r \in \mathcal{R}_{ij}} Q_r^{\mathrm{DRT}}. \tag{6}$$

2.2. Vehicle movements

We assume that there are N vehicles available for transporting customers. At any point in time, each vehicle follows a specific *route* $(i_0, i_1, \ldots, i_m)$ determined by a sequence of nodes in the transportation network. The vehicle starts at the first node i_0 and proceeds by visiting the other nodes i_k for $k = 1, \ldots, m$ in the order determined by the route. Each node corresponds to a stop during which customers may enter or exit the vehicle. We assume that each route is acyclic and consists of a minimum of two nodes, that is, none of the vehicles are idle at any time. Since the routes are acyclic, each route consists of a maximum of $|I|$ nodes.

The total time needed to complete a route is given by

$$d^{\mathrm{route}}_{(i_0,i_1,\ldots,i_m)} = \sum_{h=1}^{m} t_{i_{h-1} i_h} \tag{7}$$

and the total cost of a route equals

$$c^{\mathrm{route}}_{(i_0,i_1,\ldots,i_m)} = \sum_{h=1}^{m} c_{i_{h-1} i_h}, \tag{8}$$

where $c_{i_{h-1} i_h}$ is the cost associated with leg (i_{h-1}, i_h).

2.2.1. Trip production

When a vehicle following a route $(i_0, i_1, \ldots, i_m)$ arrives at stop i_k, customers waiting at i_k to be transported to any of the stops $i_{k+1}, \ldots, i_m$ may enter the vehicle. In other words, we say that the vehicle *produces* trips from i_k to the nodes $i_{k+1}, \ldots, i_m$, see Figure 1. Formally, the set of produced trips from the current stop i_k to the remaining stops $i_{k+1}, \ldots, i_m$ is given by

$$\mathcal{R}_{(i_k,\ldots,i_m)} = \bigcup_{h=k+1}^{m} \{(i_k, i_{k+1}, \ldots, i_h)\}. \tag{9}$$

During the execution of the route $(i_0, i_1, \ldots, i_m)$, the vehicle consecutively arrives at stops i_k, where $k = 0, \ldots, m$. The set of produced trips during the execution of the route equals

$$\mathcal{R}^{\mathrm{route}}_{(i_0,\ldots,i_m)} = \bigcup_{k=0}^{m-1} \mathcal{R}_{(i_k,\ldots,i_m)}, \tag{10}$$

that is, the set of subsequences of $(i_0, \ldots, i_m)$. This idea of producing trips is in fact similar as in traditional public transport: a bus with a given route produces trips that are subsequences of the route (see Figure 1).

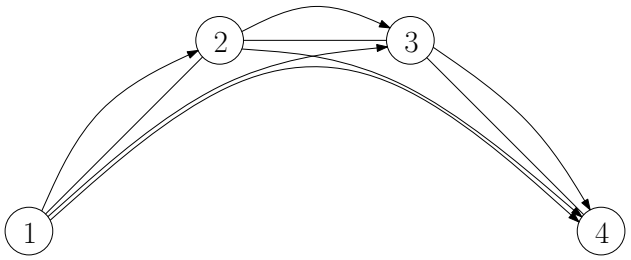


Figure 1: Trip production. A vehicle follows the route $(1, 2, 3, 4)$. Upon arrival at each stop, the vehicle produces trips to the other stops on the route. The set of produced trips during the execution of the route is equal to the set of subsequences of $(1, 2, 3, 4)$, that is, $\mathcal{R}^{\text{route}}_{(1,2,3,4)} = \{(1, 2), (1, 2, 3), (1, 2, 3, 4), (2, 3), (2, 3, 4), (3, 4)\}$.

2.2.2. Route modifications

The most distinguishing feature of demand-responsive transport, which differentiates it from traditional public transport, is that the vehicle routes may be modified in real time in order to meet the demand for transportation efficiently. In this work, we consider the following type of route modifications.

Suppose a vehicle follows a route $(i_0, \ldots, i_m)$ and arrives at node i_k. Naturally, the vehicle produces trips from i_k to all remaining nodes $i_{k+1}, \ldots, i_m$. In order to serve additional customers, the driver may choose to offer transportation from i_k to *destinations that are not included in the current route*. That is, the driver may *extend*[4] the current route by adding a sequence of nodes $i_{m+1}, \ldots, i_q$ to the end of the route (see Figure 2). In this case, the vehicle produces the set of trips $\mathcal{R}_{(i_k,\ldots,i_q)}$ from i_k to the stops $(i_{k+1}, \ldots, i_q)$, see Figure 2c.

When a vehicle following a route $(i_0, \ldots, i_m)$ arrives at i_k, where $0 \leq k < m$, there are two options:

- The route is *extended*. The vehicle transfers to a new route beginning at i_k, namely, $(i_k, \ldots, i_q)$.
- The route is *not extended*. The vehicle proceeds by executing the current route $(i_0, \ldots, i_m)$.

By definition, the minimum number of nodes in a route is two. Thus, when the vehicle arrives at i_m, the route is automatically extended.

In the following, we discuss the route extensions and production of trips in more detail by representing routes as sequences of *states*.

2.2.3. States

The *state* of a vehicle describes which part of the route the vehicle is currently executing. Each time a vehicle following a route $(i_0, \ldots, i_m)$ arrives at stop i_k, we say that the vehicle *transfers* to a new state. Similarly as routes, the states are defined as sequences of nodes. The difference between

[4]In some services, it might be reasonable to think that the route of a vehicle could be modified *during* the trip of a customer by adding nodes to the sequence before the drop-off point of the customer. In this work, however, we assume that when a customer enters a vehicle, the route to the customer's destination is fixed and the possible route extensions affect the part of the route after the customer's destination.

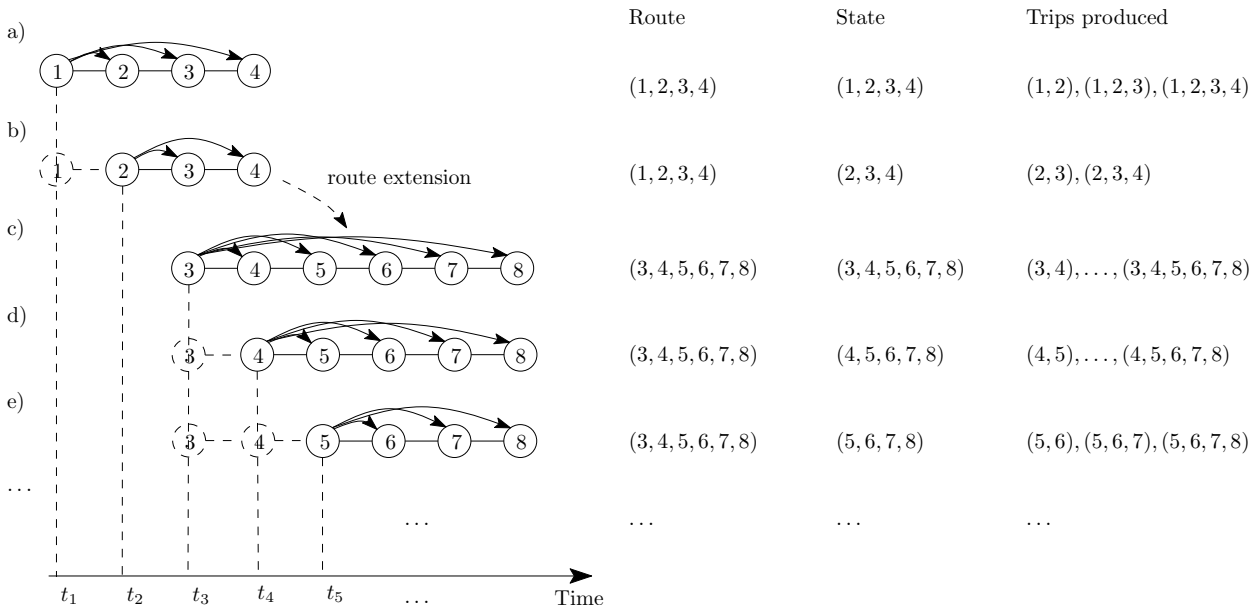


Figure 2: Routes, states and trips. a) At t_1, a vehicle follows the route $(1, 2, 3, 4)$ consisting of four nodes. Customers are transported from 1 to 2, 3 and 4. b) At time t_2, the vehicle arrives at node 2. At this instant, customers waiting at 2 to be transported to 3 or 4 may enter the vehicle, that is, the vehicle produces trips $(2, 3)$ and $(2, 3, 4)$. The route of the vehicle remains unchanged. c) At t_3, the vehicle arrives at node 3. In addition to node 4, the driver chooses to offer transportation from node 3 to nodes $5, \ldots, 8$. In this case, the vehicle produces trips $(3, 4), (3, 4, 5), (3, 4, 5, 6), (3, 4, 5, 6, 7), (3, 4, 5, 6, 7, 8)$. The route of the vehicle is extended, and the new route is $(3, 4, 5, 6, 7, 8)$. The execution of the new route is represented as a sequence of states, namely $(3, 4, 5, 6, 7, 8), (4, 5, 6, 7, 8), \ldots, (7, 8)$, see Figures c,d,e.

states and routes is that the state of a vehicle corresponds to the *remaining part* of its current route. That is, the vehicle transfers to a new state each time it arrives at a stop, even if the route remains unchanged. During the execution of a route $(i_0, \ldots, i_m)$, the vehicle successively transfers to states $(i_0, \ldots, i_m), (i_1, \ldots, i_m), (i_2, \ldots, i_m), \ldots, (i_{m-1}, i_m)$. For example, in Figure 2, the vehicle follows the same route in both 2a and 2b, but the state in 2a is $(1, 2, 3, 4)$ and the state in 2b is $(2, 3, 4)$. In 2c, the vehicle transfers to state $(3, 4, 5, 6, 7, 8)$.

Generally, the set of states is defined by

$$\mathcal{S} = \{(i_0, i_1, \ldots, i_m) \mid m \geq 1,\ (h \neq g \Rightarrow i_h \neq i_g),\ i_h, i_g \in I \text{ for all } h, g \in \{0, \ldots, m\}\}.$$

The total number of states equals $|\mathcal{S}| = \sum_{l=2}^{|I|} \binom{|I|}{l} l!$. Note that the set of states is equal to the set of trips, that is, $\mathcal{S} = \mathcal{R}$. Clearly, the number of states grows rapidly with the size of the network. In practice, it might be reasonable to consider only a subset of all possible states in a network due to computational issues. However, since the modification of our model to this approximate case is straightforward, we will focus on the general case in which the state space is complete.

When a vehicle arrives at a stop, the route of the vehicle may be extended by adding a sequence of nodes to the end of the route, as dicussed in Section 2.2.2. In other words, the set of states *to*

which a transfer from state $s = (i_0, \ldots, i_m)$ is possible is defined by

$$\mathcal{S}_s = \{(s_1, \ldots, i_m, i_{m+1}, \ldots, i_q) \in \mathcal{S} \mid i_{m+1}, \ldots, i_q \in I\}. \tag{11}$$

This set will be referred to as the *successor set* of state s. Note that if the route is not extended, we have $q = m$ in the above equation.

The first two nodes i_0, i_1 of a state $(i_0, \ldots, i_m)$ determine the leg the vehicle is currently traversing. Since the state of the vehicle is modified at the time the vehicle arrives at i_1, the duration of a state $(i_0, \ldots, i_m)$ is given by

$$d_{(i_0,\ldots,i_m)} = t_{i_0 i_1}. \tag{12}$$

Note that the duration of a state $(i_0, \ldots, i_m)$ is generally different from the total time (7) needed to complete the route.

2.2.4. Arrival rate and production rate

The *arrival rate* of vehicles at state $s \in \mathcal{S}$ describes the number of vehicles arriving at state s per unit time and is denoted by T_s. During a finite time period $[0, t]$, we require that the total service time of vehicles at different states equals Nt, that is,

$$\sum_{s\in\mathcal{S}} tT_s d_s = Nt \quad \Leftrightarrow \quad \sum_{s\in\mathcal{S}} T_s d_s = N, \tag{13}$$

where d_s is the duration of state s defined by Equation (12).

Let us consider the *production rate* T_r^p for a trip $r \in \mathcal{R}$, that is, how many times the trip r is produced per unit time. Using the notation of (9), we note that a state s *produces* the set of trips $\mathcal{R}_s$, that is, the set of all subsequences of s beginning from the first node of s. Correspondingly, for any trip $r \in \mathcal{R}$, we define the set of states that produce trip r by

$$\mathcal{S}_r^p = \{s \in \mathcal{S} \mid r \in \mathcal{R}_s\},$$

where $\mathcal{S}$ is the set of all states. That is, a state s is included in S_r^p if r is a subsequence of s beginning from the first node in s.

Thus, the production rate T_r^p for a trip r is given by the total arrival rate of vehicles at states $s \in \mathcal{S}_r^p$ that produce trip r, that is,

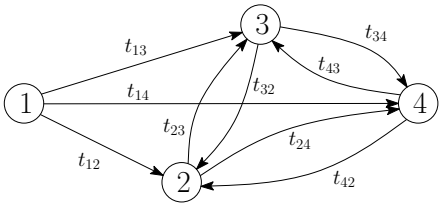
$$T_r^p = \sum_{s\in\mathcal{S}_r^p} T_s, \tag{14}$$

where T_s is the arrival rate of vehicles at state s (see Figure 3).

2.2.5. Average travel time

Let us then consider the *average travel time* t_r^{DRT} for a trip $r \in \mathcal{R}$, defined as the sum of waiting time and ride time, see Equation (2).

The ride time q_r^{DRT} for a trip r is given by Equation (1). The average waiting time w_r^{DRT} for a trip r depends on the production rate T_r^p of the trip, that is, the arrival rate of vehicles at states S_r^p



Trip $r \in \mathcal{R}_{14}$	Ride time q_r	States S_r^p that produce r
$(1,4)$	t_{14}	$(1,4),(1,4,2),(1,4,3),$ $(1,4,2,3),(1,4,3,2)$
$(1,2,4)$	$t_{12}+t_{24}$	$(1,2,4),(1,2,4,3)$
$(1,3,4)$	$t_{13}+t_{34}$	$(1,3,4),(1,3,4,2)$
$(1,2,3,4)$	$t_{12}+t_{23}+t_{34}$	$(1,2,3,4)$
$(1,3,2,4)$	$t_{13}+t_{32}+t_{24}$	$(1,3,2,4)$

Figure 3: The production rate for different trips between two nodes. Let us consider the rate of vehicles producing trips from node 1 to node 4 in the example network presented in the figure. The trips $r \in \mathcal{R}_{14}$ from 1 to 4 are presented in the first column of the table next to the figure. The second column of the table shows the ride time q_r for the trip r and the third column shows the set of states $\mathcal{S}_r^p$ that produce the trip r. The production rate T_r^p for a trip r is defined as the sum of the arrival rates T_s of vehicles at states $s \in \mathcal{S}_r^p$ that produce trip r. The average travel time for a trip r is given by $t_r^{\mathrm{DRT}} = \frac{1}{T_r^p} + q_r$.

that produce trip r. In addition to the production rate T_r^p, the average waiting time w_r^{DRT} depends on the *distribution* of the arrival times of vehicles at states S_r^p.

Given the production rate T_r^p, the average waiting time for trip r is minimized when the vehicles arrive at *equal time intervals*. In this case, the average waiting time equals $\frac{1}{2T_r^p}$. This idea is widely used in traditional public transport: If vehicles arrive every n minutes, the average waiting time is $n/2$ minutes. In addition to minimizing average waiting time, equal time intervals divide the customers evenly among vehicles providing service.

However, in our competitive DRT model, a driver does not know the exact locations of other vehicles and thus, it may not be reasonable to think that the inter-arrival times of vehicles were equal. In the absence of accurate information, we assume that the vehicles arrive according to a Poisson process, in which arrivals occur continuously and independently of one another (Ross, 1995). In this case, the expected waiting time for a customer departing at a random instant is equal to the expected time interval between arrivals, that is, $w_r^{\mathrm{DRT}} = \frac{1}{T_r^p}$. The average travel time is thus given by

$$t_r^{\mathrm{DRT}} = \begin{cases} \frac{1}{T_r^p} + q_r^{\mathrm{DRT}} & \text{if } T_r^p > 0, \\ \infty & \text{otherwise.} \end{cases} \tag{15}$$

For instance, if during one hour there are three vehicles that produce a trip r for which the *ride time* is 10 minutes, the average *travel time* for that trip is given by adding the average *waiting time*, that is, $t_r^{\mathrm{DRT}} = \frac{1}{\frac{3}{60}} + 10 = 30$ minutes.

Given the production rate T_r^p, the average waiting time in the Poisson model is twofold compared to the equal time intervals model. In other words, the arrival time distribution has a significant effect on the waiting time. This is seen to be a major difference between centralized and competitive transport services[5].

[5]Assuming that the competing divers had some information on the locations of other vehicles, the inter-arrival times could even out since it would result in a better level of service. In practice, we expect that the average waiting time is between $\frac{1}{2T_r^p}$ and $\frac{1}{T_r^p}$.

Note that if the production rate T_r^p for a trip $r \in \mathcal{R}$ is zero, the average travel time for r is infinite. In this case, the demand for the trip is zero, see Equation (4).

2.2.6. Expected profit rate

The drivers attempt to maximize profit by offering transportation to as many customers per unit time as possible. More precisely, the drivers attempt to transfer to states at which the *expected profit rate* is maximized. By using the expected profit rate as a utility measure we can compare the profitability of routes of different lengths.

For example, a long route including a large number of nodes produces more trips than a short one and thus, we expect that the expected profit earned during a long route is greater than in a short one. However, the time needed to execute a long route is also greater. That is, in some cases it may be profitable to execute many short routes successively instead of a single long route, as in the example presented in Figure 4.

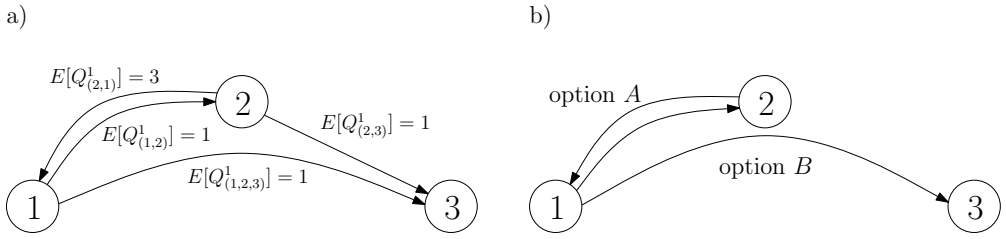


Figure 4: Expected profit rate example. Figure a) shows a simplified example network with three nodes and the demand for different trips. The cost and time of each leg is assumed to be one, that is, $c_{ij} = t_{ij} = 1$ for all $i, j \in \{1, 2, 3\}$. Figure b) shows two alternative route options for a vehicle located at node 1. Option A consists of two short routes $(1, 2)$ and $(2, 1)$, whereas option B consists of a single route $(1, 2, 3)$. By choosing option B, the vehicle produces the trips $(1, 2), (1, 2, 3), (2, 3)$, thus serving three customers in two time units. By choosing option A, the vehicle produces the trips $(1, 2), (2, 1)$ and serves four customers in the same amount of time.

Given that a single vehicle arrives at state $s \in \mathcal{S}_r^p$ that produces a trip $r \in \mathcal{R}_{ij}$ from i to j, let us study the expected number $E[Q_r^1]$ of DRT customers traveling from i to j that enter the vehicle. We determine $E[Q_r^1]$ by dividing the demand $E[Q_r^{\mathrm{DRT}}]$ for the trip r by the total production rate T_r^p of the trip, that is,

$$E[Q_r^1] = \frac{E[Q_r^{\mathrm{DRT}}]}{T_r^p}. \tag{16}$$

This equation is justified by the assumption that the customers that choose the trip r are divided equally among vehicles that produce the trip r. For instance, if during one hour there are three vehicles that produce a trip r and the demand for the trip r is six customers/hour, the expected number of DRT customers for a single vehicle equals $E[Q_r^1] = \frac{6 \text{ customers/h}}{3 \text{ vehicles/h}} = 2$ customers/vehicle.

Let us then consider the expected profit rate for a vehicle arriving at state $s = (i_1, \ldots, i_m)$. For analysis, we calculate the expected profit rate by assuming that the route corresponding to s is not extended during its execution. That is, the vehicle executes the route by successively transferring to states $(i_1, \ldots, i_m), (i_2, \ldots, i_m), \ldots, (i_{m-1}, i_m)$. In this case, the set of trips produced during the

execution equals $\mathcal{R}_s^{\text{route}}$ defined by Equation (10), that is, the set of all subsequences of s (see Figure 1).

The expected number of customers that enter the vehicle during the execution of the route is given by $\sum_{r\in\mathcal{R}_s^{\text{route}}} E[Q_r^1]$ and the corresponding expected revenue is given by

$$\sum_{r\in\mathcal{R}_s^{\text{route}}} E[Q_r^1]p_r,$$

where p_r is the ticket price for the trip r.

The corresponding *expected revenue rate* $E[R(s)]$ is given by dividing the expected revenue by the total time (7) needed to execute the route, that is,

$$E[R(s)] = \frac{\sum_{r\in\mathcal{R}_s^{\text{route}}} E[Q_r^1]p_r,}{t_s^{\text{route}}}. \tag{17}$$

The corresponding *cost rate* $C(s)$ of a route s is given by

$$C(s) = \frac{c_s^{\text{route}}}{t_s^{\text{route}}}, \tag{18}$$

where c_s^{route} denotes the total cost of the route s defined by (8).

By combining equations (17) and (18), the expected profit rate at state $s \in \mathcal{S}$ is given by

$$U(s) = E[R(s)] - C(s). \tag{19}$$

By assuming a logit model similar to the one in Equation (4), the probability that a vehicle at state s transfers to state $s' \in \mathcal{S}$ is given by

$$P_{s,s'} = \begin{cases} \frac{\exp(\theta^d U(s'))}{\sum_{s''\in\mathcal{S}_s}\exp(\theta^d U_{s''})}, & \text{if } s' \in \mathcal{S}_s \quad (\mathcal{S}_s = \text{ successor set of } s), \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

where θ^d is a nonnegative parameter reflecting the uncertainty on demand and DRT services from the perspective of drivers.

Finally, we state that the arrival rate of vehicles at state $s \in \mathcal{S}$ satisfies

$$T_s = \sum_{s'\in\mathcal{S}} T_{s'}P_{s',s}. \tag{21}$$

By using matrix notation, we can write equation (21) in the form

$$T = T \cdot P, \tag{22}$$

where T is a $1 \times |\mathcal{S}|$ row vector containing the arrival rates T_s of vehicles at different states and P is a $|\mathcal{S}| \times |\mathcal{S}|$ matrix consisting of the transition probabilities (20) between states. Note that the arrival rate vector T satisfying Equation (22) is an eigenvector of the matrix P corresponding to eigenvalue 1. However, the calculation of T is not straightforward since the matrix P depends on the entries of T, see Equation (20). The calculation of T is described in Section 3.1.

3. Network equilibrium

The drivers attempt to maximize profit rate by transporting as many customers per unit time as possible. Thus, we expect that the drivers prefer detours instead of direct routes in order to serve more customers, see Equation (17). In some cases, however, producing only direct trips may be more profitable, as in Figure 4.

We also expect that the customers prefer direct trips instead of detours, see Equation (3). However, if many vehicles produce non-direct trips, the travel times in non-direct trips may be smaller than in direct trips due to small waiting times.

The movement of vehicles is described by means of the *arrival rates* T_s of vehicles at different states $s \in \mathcal{S}$ and the movement of customers is described by means of the demands Q_r^{DRT} for different DRT trips $r \in \mathcal{R}$.

A *network equilibrium* denotes the following combination of arrival rates T_s^* at states $s \in \mathcal{S}$ and demands $Q_r^{\mathrm{DRT}*}$ for trips $r \in \mathcal{R}$:

- The set of arrival rates $\{T_{s_1}^*, \ldots, T_{s_{|\mathcal{S}|}}^*\}$ of vehicles at states $s_i \in \mathcal{S}$ generates the set of demands $\{Q_{r_1}^{\mathrm{DRT}*}, \ldots, Q_{r_{|\mathcal{R}|}}^{\mathrm{DRT}*}\}$ for trips $r_i \in \mathcal{R}$.
- The set of demands $\{Q_{r_1}^{\mathrm{DRT}*}, \ldots, Q_{r_{|\mathcal{R}|}}^{\mathrm{DRT}*}\}$ for trips $r_i \in \mathcal{R}$ generates the set of arrival rates $\{T_{s_1}^*, \ldots, T_{s_{|\mathcal{S}|}}^*\}$ of vehicles at states $s_i \in \mathcal{S}$.

Formally, we define a network equilibrium T^* as a $1 \times |\mathcal{S}|$ vector, containing the arrival rates T_s of vehicles at different states $s \in \mathcal{S}$, that satisfies Equations (22) and (13). This definition is justified by the fact that any arrival rate vector T generates a unique demand vector Q containing the demands for trips $r \in \mathcal{R}$ (see Equations (15) and (5)) and the demand vector Q produces unique transition probabilities (20). Thus, if an equilibrium exists, it is uniquely defined by the arrival rate vector T^*. The following theorem is related to the existence of such an equilibrium.

Theorem 1. *For any finite transportation network I, there exists a network equilibrium T^*.*

Proof. Brouwer's fixed point theorem states that every continuous function f from a convex compact subset of a Euclidean space to itself has a fixed point, that is, a point x_0 for which $f(x_0) = x_0$.

Let $\mathcal{T}$ denote the set of arrival rate vectors that satisfy Equation (13). Clearly, since $\mathcal{T}$ is a plane in $R^{|\mathcal{S}|}$ restricted by $T_s \geq 0$ for $s \in \mathcal{S}$, we see that $\mathcal{T}$ is convex. In addition, since $\mathcal{T}$ is closed and bounded, we know by the Heine-Borel theorem that $\mathcal{T}$ is compact.

Equations (1 - 5) and (15) define a continuous function $g : \mathcal{T} \to Q^{\mathrm{DRT}}$, where Q^{DRT} is the set of demand vectors Q^{DRT} that contain the demands for trips $r \in \mathcal{R}$. Equations (19 - 22) define a continuous function $h : Q^{\mathrm{DRT}} \to \mathcal{T}$. Thus, since $f = g \circ h$ is a continuous function from $\mathcal{T}$ to $\mathcal{T}$, there exists an arrival rate vector T^* for which $f(T^*) = T^*$. The corresponding equilibrium demand is given by $Q^{\mathrm{DRT}*} = g(T^*)$ □.

3.1. Algorithm

If the transition probability matrix P were independent of the arrival rate vector T, the solution to Equation (22) would be achieved by means of a straightforward power method by successively computing $T^{k+1} = T^k \cdot P$ for $k = 1, \ldots$ and normalizing T^{k+1} after each multiplication, as described

in (Meyer, 2000). However, since the transition probabilities (20) depend on the arrival rates, the matrix P needs to be updated after each step of the power method. In other words, the idea for finding a steady-state equilibrium is to solve both the customer choice subproblem and the vehicle movement subproblem iteratively until a convergence criterion is met, similarly as in (Yang et al., 2010).

The outline of the procedure is presented in Algorithm 1.

Algorithm 1: Equilibration of DRT.

Data: Set I of nodes, direct ride time t_{ij}, cost c_{ij}, demand Q_{ij}, ticket price p_r, cost of virtual mode $\bar{g}_{ij}$ for all pairs $i, j \in I$ of nodes, number of vehicles N, customers' monetary value for unit time β and the uncertainty parameters θ, θ^d.

Result: Equilibrium arrival rates T_s of vehicles at states $s \in \mathcal{S}$.

Initialize the arrival rates T_s of vehicles by dividing the vehicles evenly among all states $s \in \mathcal{S}$, $T_s = \frac{N}{d_s|\mathcal{S}|}$;

repeat

  Customer demand updating: Calculate $E[Q_r^{\mathrm{DRT}}]$ for all $r \in \mathcal{R}$ given by equation (6) ;

  Vehicle movements: Solve T_s for all $s \in \mathcal{S}$ by equation (21) ;

until *convergence* ;

3.2. A three node example

Let us demonstrate the calculation of the network equilibrium in a simple case in which the network consists of three nodes denoted by $1, 2, 3$. The number of vehicles is $N = 30$. The direct ride times (in minutes) are defined by $t_{12} = t_{21} = 3$, $t_{13} = t_{31} = 4$ and $t_{23} = t_{32} = 5$. The distances between the nodes (in kilometers) are equal to the direct ride times, as shown in Figure 5.

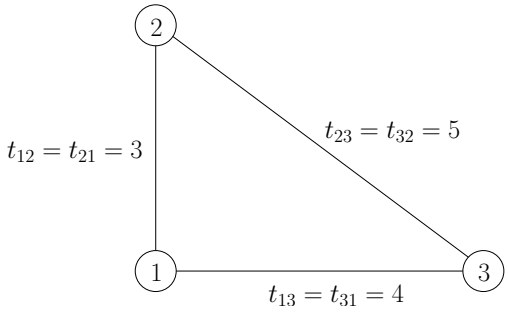


Figure 5: A three node example network. The distances between the nodes (in kilometers) are equal to the direct ride times t_{ij}.

The cost of leg (i, j) is defined by $c_{ij} = \frac{1}{2}t_{ij}$. The customers' monetary value of unit time is set to $\beta = 1$ and the values of the uncertainty parameters are set to $\theta = \theta^d = 1$. For simplicity, the ticket price of all trips from i to j is $p_{ij} = 0.3t_{ij}$, that is, the price per kilometer (in EUR) is 0.30. The subjective cost of the virtual mode is $\bar{g}_{ij} = 2t_{ij}$. Finally, the total demand is $Q_{ij} = 3$ for all $i, j \in I$. The parameters values are presented in Table 1.

Table 1: Parameter values of the three node example.

Leg ij	t_{ij}	c_{ij}	p_{ij}	$\bar{g}_{ij}$	Q_{ij}
12 and 21	3	1.5	0.90	6	3
13 and 31	4	2	1.20	8	3
23 and 32	5	2.5	1.50	10	3

The number of states equals $|\mathcal{S}| = \sum_{l=2}^{3} \binom{3}{l} l! = 3 \cdot 2 + 1 \cdot 6 = 12$ and the set of states $\mathcal{S}$ is given by

$$\mathcal{S} = \{(1,2),(1,3),(2,1),(2,3),(3,1),(3,2),(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}.$$

The set of trips is equal to the set of states, that is, $\mathcal{R} = \mathcal{S}$. The successor sets $\mathcal{S}_s$ of the states are given by

$$\begin{aligned}
\mathcal{S}_{(1,2)} &= \mathcal{S}_{(3,2)} = \{(2,1),(2,3),(2,1,3),(2,3,1)\} \\
\mathcal{S}_{(1,3)} &= \mathcal{S}_{(2,3)} = \{(3,1),(3,2),(3,1,2),(3,2,1)\} \\
\mathcal{S}_{(2,1)} &= \mathcal{S}_{(3,1)} = \{(1,2),(1,3),(1,2,3),(1,3,2)\} \\
\mathcal{S}_{(1,2,3)} &= \{(2,3),(2,3,1)\} \quad \mathcal{S}_{(1,3,2)} = \{(3,2),(3,2,1)\} \\
\mathcal{S}_{(2,1,3)} &= \{(1,3),(1,3,2)\} \quad \mathcal{S}_{(2,3,1)} = \{(3,1),(3,1,2)\} \\
\mathcal{S}_{(3,1,2)} &= \{(1,2),(1,2,3)\} \quad \mathcal{S}_{(3,2,1)} = \{(2,1),(2,1,3)\}.
\end{aligned}$$

For example, a transition from state $(1,3,2)$ to state $(2,3)$ is not possible since $(2,3)$ is not included in the successor set of $(1,3,2)$.

The set of states $\mathcal{S}_r^p$ that produce trip r are given by

$$\begin{aligned}
&\mathcal{S}^p_{(1,2)} = \{(1,2),(1,2,3)\} && \mathcal{S}^p_{(1,3)} = \{(1,3),(1,3,2)\} && \mathcal{S}^p_{(2,1)} = \{(2,1),(2,1,3)\} \\
&\mathcal{S}^p_{(2,3)} = \{(2,3),(2,3,1)\} && \mathcal{S}^p_{(3,1)} = \{(3,1),(3,1,2)\} && \mathcal{S}^p_{(3,2)} = \{(3,2),(3,2,1)\} \\
&\mathcal{S}^p_{(1,2,3)} = \{(1,2,3)\} && \mathcal{S}^p_{(1,3,2)} = \{(1,3,2)\} && \mathcal{S}^p_{(2,1,3)} = \{(2,1,3)\} \\
&\mathcal{S}^p_{(2,3,1)} = \{(2,3,1)\} && \mathcal{S}^p_{(3,1,2)} = \{(3,1,2)\} && \mathcal{S}^p_{(3,2,1)} = \{(3,2,1)\}.
\end{aligned}$$

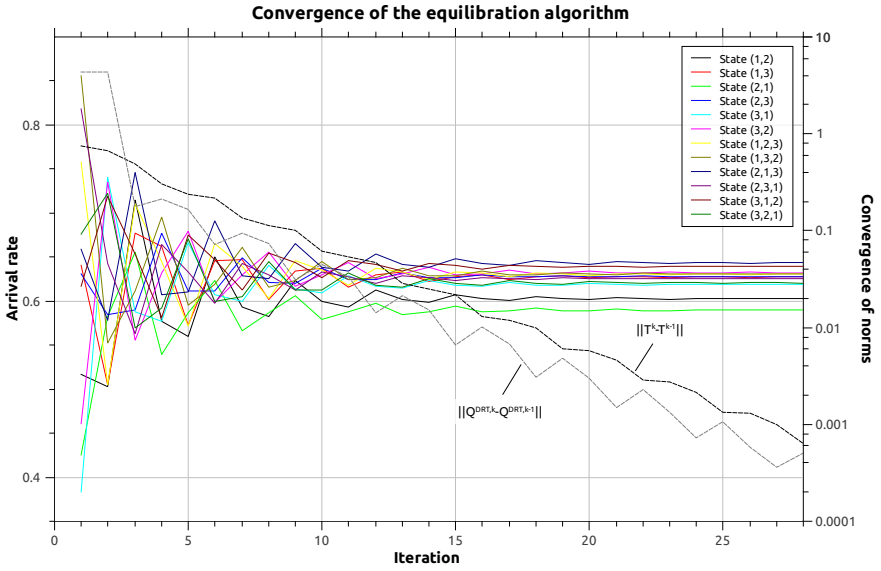
For example, trip $(2,3)$ is produced by states $(2,3),(2,3,1)$ whereas each trip consisting of three nodes is produced by a single state equal to the trip.

Assuming that a vehicle arriving at state $s \in \mathcal{S}$ executes the route corresponding to s without route extensions, the set of trips $\mathcal{R}_s^{\text{route}}$ produced during the execution of routes corresponding to states s are given by

$$\begin{aligned}
&\mathcal{R}^{\text{route}}_{(1,2)} = \{(1,2)\} && \mathcal{R}^{\text{route}}_{(1,3)} = \{(1,3)\} && \mathcal{R}^{\text{route}}_{(2,1)} = \{(2,1)\} \\
&\mathcal{R}^{\text{route}}_{(2,3)} = \{(2,3)\} && \mathcal{R}^{\text{route}}_{(3,1)} = \{(3,1)\} && \mathcal{R}^{\text{route}}_{(3,2)} = \{(3,2)\} \\
&\mathcal{R}^{\text{route}}_{(1,2,3)} = \{(1,2),(1,2,3),(2,3)\} && \mathcal{R}^{\text{route}}_{(1,3,2)} = \{(1,3),(1,3,2),(3,2)\} && \mathcal{R}^{\text{route}}_{(2,1,3)} = \{(2,1),(2,1,3),(1,3)\} \\
&\mathcal{R}^{\text{route}}_{(2,3,1)} = \{(2,3),(2,3,1),(3,1)\} && \mathcal{R}^{\text{route}}_{(3,1,2)} = \{(3,1),(3,1,2),(1,2)\} && \mathcal{R}^{\text{route}}_{(3,2,1)} = \{(3,2),(3,2,1),(2,1)\}.
\end{aligned}$$

Note that the routes consisting of three nodes produce more trips than the short routes consisting of two nodes. However, in some cases it may be profitable to execute short routes instead of long ones, as in the example presented in Figure 4.

With the above sets $\mathcal{S}_s, S_r^p$ and $\mathcal{R}_s^{\text{route}}$, we calculated the network equilibrium by using Algorithm 1. The solid lines in Figure 6 show the arrival rates T_s of vehicles at different states $s \in \mathcal{S}$ after each step of the algorithm. The black dashed line shows on a logarithmic scale the convergence of the norm $\|T^k - T^{k-1}\|$ of the difference between two successive arrival rate vectors T^k and T^{k-1}. The grey dashed line shows the corresponding convergence of the demand vector Q^{DRT}. After 28 iterations, the norm $\|T^k - T^{k-1}\|$ was less than 0.001. The equilibration was continued until the norm was less than 0.00001.

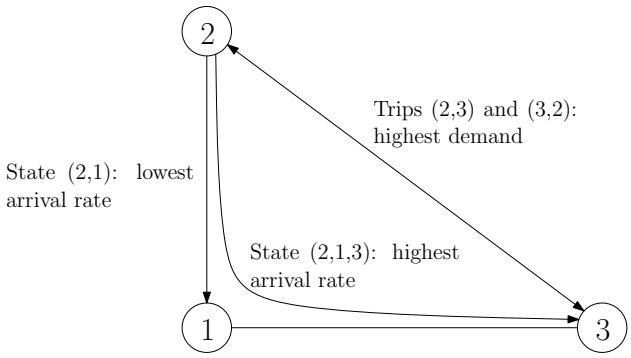


Arrival rates at states in the network equilibrium T^*												
State s	(1,2)	(1,3)	(2,1)	(2,3)	(3,1)	(3,2)	(1,2,3)	(1,3,2)	(2,1,3)	(2,3,1)	(3,1,2)	(3,2,1)
Arrival rate T_s	0.603	0.627	0.590	0.628	0.619	0.632	0.631	0.631	0.644	0.626	0.639	0.621

Figure 6: Convergence of Algorithm 1 in the three node example. The solid lines in show the arrival rates T_s of vehicles at different states $s \in \mathcal{S}$ after each step of the algorithm. The black and grey dashed lines show on a logarithmic scale the convergence of the arrival rate vector T and the demand vector Q^{DRT}, respectively. After 28 iterations, the norm $\|T^k - T^{k-1}\|$ was less than 0.001. The equilibration was continued until the norm was less than 0.00001. The network equilibrium is defined by the arrival rates at different states shown in the table below the figure.

By looking at the solid lines, we see the oscillatory nature of the arrival rates: When the arrival rate of vehicles in a specific state s increases during the equilibration, the number of customers available for a single vehicle in that state decreases. This causes the drivers to choose other states instead of s. When the arrival rate at state s decreases, it becomes more profitable for individual vehicles and results in drivers choosing state s more often. In the network equilibrium, the arrival rate is highest at state $(2, 1, 3)$ and lowest at state $(2, 1)$.

Referring to the dashed lines, which are roughly straight lines on a logarithmic scale, we see that the norms of the arrival rate and demand vectors converge exponentially with respect to the



Trip	(1,2)	(1,3)	(2,1)	(2,3)	(3,1)	(3,2)	(1,2,3)	(1,3,2)	(2,1,3)	(2,3,1)	(3,1,2)	(3,2,1)
Demand	2.350	2.625	2.350	2.653	2.625	2.655	0.022	0.003	0.169	0.003	0.167	0.021
Travel time ratio	1.270	1.199	1.270	1.160	1.199	1.160	2.396	3.528	1.711	3.533	1.713	2.403
Waiting time	0.811	0.795	0.811	0.798	0.795	0.798	1.586	1.585	1.554	1.598	1.564	1.611
Subjective price	4.711	5.995	4.711	7.298	5.995	7.298	10.786	11.485	10.054	11.498	10.064	10.811
- virtual mode	6	8	6	10	8	10	8	6	10	6	10	8

Figure 7: Equilibrium statistics. The table shows the demand, travel time ratio, waiting time, subjective price and the subjective price of the virtual mode in the network equilibrium for all trips in the three-node example.

number of iterations. Decreasing the difference $\|T^k - T^{k-1}\|$ by a factor of $F \in \mathbb{R}$ requires $G \in \mathbb{N}$ iterations. That is, $\|T^k - T^{k-1}\| = \frac{1}{F^K}\|T^{k-GK} - T^{k-1-GK}\|$ for $K \in \mathbb{N}$. By looking at the slope of the dark dashed line, we see that the difference $\|T^k - T^{k-1}\|$ decreases by a factor of 10 in approximately 10 iterations, that is, $\|T^k - T^{k-1}\| = \frac{1}{10^K}\|T^{k-10K} - T^{k-1-10K}\|$ for $K \in \mathbb{N}$.

The trip statistics of the example are presented in the table in Figure 7. Generally, the demand for detours is low due to the fact that the ride times are long compared to the direct ride times. For example, the ride time in trips $(1, 3, 2), (2, 3, 1)$ is $4 + 5 = 9$ and the direct ride time is $t_{12} = t_{21} = 3$. Thus, the subjective price is at least $9 + 0.30 \cdot 3 = 9.90$. Since the subjective price of the virtual mode is $\bar{g}_{12} = \bar{g}_{21} = 6$, the demand for these DRT trips is low. The most popular detours are $(2, 1, 3)$ and $(3, 1, 2)$, since the ratio of ride time (= 7) to direct ride time $t_{13} = t_{31} = 5$ is relatively small. This can be seen also by looking at the arrival rates at states $(2, 1, 3), (3, 1, 2)$ in Figure 6, which are the highest among all states.

4. Long-run market equilibria

In a competitive demand responsive market with no entry limits, we expect that the number of vehicles increases as long as the profit rate of vehicles is positive. Regulating the number of vehicles and ticket price could improve the service from the customers' point of view as well as from the perspective of drivers. In the following, we study different characteristics of the competitive DRT service as a function of the number of vehicles N and average price per kilometer p. We identify four *long-run market equilibria* (24), (26), (28), (27), defined by combinations of N and p.

4.1. Free entry equilibrium

In the absence of entry limits, new vehicles arrive as long as the total profit rate $\Pi(N, p)$ of vehicles is positive[6]. Thus, the feasible set for the free entry equilibrium is given by

$$\mathcal{F} = \{(N, p) \mid \Pi(N, p) \geq 0\}. \tag{23}$$

In the free entry market, we assume that the drivers may agree to modify the ticket price of the service. Thus, when the number of vehicles increases and the total profit rate approaches zero with a given price, the price is modified in order to keep the service profitable for the drivers. These type of modifications result in ticket prices that attract additional vehicles to the market. Thus, we define the free entry equilibrium by means of the condition

$$(N^{\mathrm{f}}, p^{\mathrm{f}}) \in \{(N, p) \in \mathcal{F} \mid N \geq N' \text{ for all } (N', p) \in \mathcal{F}\}. \tag{24}$$

That is, both the number of vehicles and price per kilometer are chosen in a way that the number of vehicles is maximized.

4.2. Customer welfare equilibrium

Public authorities might consider regulating the DRT service in a way that it would improve the service provided to customers as much as possible, compared to existing transport services. We define the *customer welfare effect* as the difference between the subjective price of the virtual mode and the subjective price of DRT. Formally, the customer welfare effect is given by

$$W(N, p) = \sum_{(i,j)\in I\times I} \left(Q_{ij} - Q_{ij}^{DRT}\right) \bar{g}_{ij} + \sum_{(i,j)\in I\times I} \sum_{r\in\mathcal{R}_{ij}} Q_r^{DRT} g_r = \sum_{(i,j)\in I\times I} \sum_{r\in\mathcal{R}_{ij}} Q_r^{DRT} (\bar{g}_{ij} - g_r), \tag{25}$$

where Q_{ij}^{DRT} is the total demand for DRT trips from i to j defined by Equation (6), $Q_{ij} - Q_{ij}^{DRT}$ is the demand for the virtual mode from i to j, $\mathcal{R}_{ij}$ is the set of DRT trips from i to j, g_r is the subjective price of trip r and $\bar{g}_{ij}$ is the subjective price of the virtual mode from i to j.

Similarly as in the free entry equilibrium, we require that the total profit rate of vehicles is positive. Thus, the customer welfare equilibrium is defined by

$$(N^{\mathrm{W}}, p^{\mathrm{W}}) = \arg \max_{(N,p)\in\mathcal{F}} W(N, p). \tag{26}$$

In this case, the number of vehicles and price are regulated in a way that the customer welfare effect is maximized.

4.3. Maximum profit rate

Maximizing the *total profit rate* of vehicles results in the equilibrium

$$(N^{\mathrm{tot}}, p^{\mathrm{tot}}) = \arg \max_{(N,p)\in\mathcal{F}} \Pi(N, p). \tag{27}$$

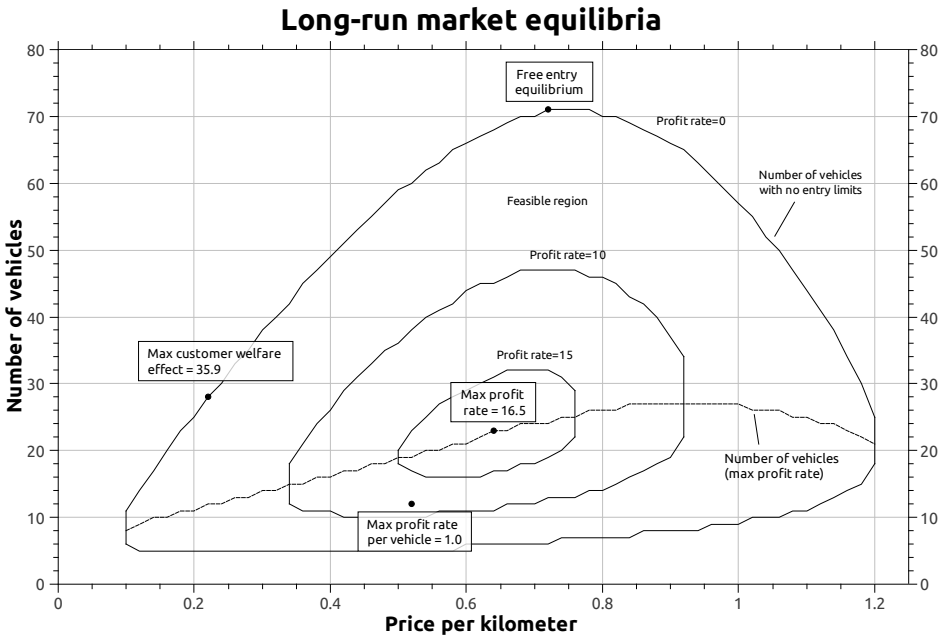
However, from the perspective of individual drivers, the optimal combination of the number of vehicles and price is such that the *profit rate per vehicle* is maximized, that is

$$(N^1, p^1) = \arg \max_{(N,p)\in\mathcal{F}} \frac{\Pi(N, p)}{N}. \tag{28}$$

[6]We assume that the profits are divided equally among vehicles on average.

4.4. A three node example

We determined the long-run market equilibria described above for the three node example case introduced in Section 3.2. The network equilibrium was calculated for different numbers of vehicles N and prices per kilometer p by means of Algorithm 1. For each combination of N and p, we calculated the total profit rate, profit rate per vehicle and customer welfare effect. The results are shown in Figure 8.



Equilibrium	Number of vehicles	price per kilometer	Demand for DRT (customers/min)	Tot. profit rate / per vehicle (EUR/min)	Welfare effect (EUR/min)	Travel time ratio	Average occupancy
Max. customer welfare effect	28	0.22	16.1	0.13 / 0.005	35.9	1.24	2.4
Maximum total profit rate	23	0.64	10.8	16.5 / 0.7	3.9	1.28	1.9
Maximum profit rate per vehicle	12	0.52	8.4	12.6 / 1.0	0.2	1.50	3.0
Free entry equilibrium	71	0.72	12.4	0.1 / 0.001	8.7	1.11	0.7

Figure 8: Long-run market equilibria for the three node example. The first column shows the four studied equilibrium points. The second and third columns show the number of vehicles and price per kilometer with which the equilibrium is achieved. The remainder of the columns show the total demand for DRT, total profit rate and profit rate per vehicle, customer welfare effect, the average ratio of travel time to direct travel time and the average number of customers in a single vehicle.

The solid curves represent contour lines in which the total profit rate $\Pi(N, p)$ of vehicles is equal ($= 0, 10, 20$). In particular, the outermost contour line corresponding to $\Pi(N, p) = 0$ encloses the *feasible region*, that is, the area in which the service is profitable for drivers. The dashed curve shows the number of vehicles for different prices for which the total profit rate is maximized. The four points represent the long-run market equilibria defined in the previous section.

The table in Figure 8 shows the demand for DRT, total profit rate, profit rate per vehicle, customer welfare effect, the average ratio of travel time to direct travel time and the average number of customers in a single vehicle in the four equilibrium points.

By looking at the figure, we note that there is a significant difference between the four equilibria. In the free entry equilibrium, the number of vehicles is significantly higher than in the other points. The average travel time ratio and the average occupancy are extremely low. This indicates that with no regulation, the DRT service would approach a taxi-type service, in which all customers are transported privately and all trips are direct trips.

The difference in price between the free entry equilibrium and the point in which the total profit rate is maximized is small. In addition, the number of served customers (demand for DRT) is only slightly smaller in the maximum profit rate case compared to the free entry case. However, maximizing the profit rate of vehicles would decrease the customer welfare effect and decrease the average level of service, as can be seen by looking at the average travel time ratio and average occupancy.

The profit rate per vehicle is maximized with an extremely small number of vehicles. In this case, only a small number of customers could be served and the level of service would be poor.

The customer welfare effect is maximized by using a significantly lower ticket price than would be optimal from the perspective of drivers. This is mainly due to the fact that the low price results in a high demand for DRT. Moreover, we note that the customer welfare equilibrium is achieved by using price regulation exclusively. That is, the results suggest that the optimal solution from the customers' point of view would be to regulate price and allow free entry.

5. Conclusions

In this work, we present a model for a competitive demand-responsive transport (DRT) market. In our model, customers seek trips to minimize travel times and drivers attempt to maximize profit rate by dynamically choosing routes that serve a large number of customers. We define the network equilibrium as a state in which the demand for trips matches the supply of trips: the choices of drivers do not change if the demand remains constant and the choices of customers do not change if the distribution of the vehicles at different routes remains constant. We show that such an equilibrium always exists in finite transportation networks and provide an algorithm for determining the equilibrium. In addition, we study long-run market equilibria for competitive DRT by varying the number of vehicles and price per kilometer.

Numerical examples conducted on a simple network suggest that with no entry limits or price regulation, the DRT service approaches a taxi-type service providing private transport and direct trips. The optimal equilibrium from the customers' perspective is achieved by regulating price and allowing free entry. This type of regulation could substantially increase the service rate.

The proposed model can be used to simulate the operations of a DRT service in a wide range of scenarios, from paratransit services for the elderly and disabled to large-scale urban DRT services designed to compete with private car traffic. Such calculations can provide valuable information to public authorities and planners of transportation services, regarding, for example, regulation and the magnitude of investments.

Acknowlegments

This work was partly supported by the Finnish Funding Agency for Technology and Innovation, Finnish Ministry of Transport and Communications and Helsinki Region Transport. The authors would like to thank Dr. Esa Hyytiä and Dr. Harri Hakula for their insightful comments to improve the quality of the paper.

References

Arnott, R., 1996. Taxi travel should be subsidized. Journal of Urban Economics 40 (3), 316–333.

Cairns, R., Liston-Heyes, C., 1996. Competition and regulation in the taxi industry. Journal of Public Economics 59 (1), 1–15.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., Savelsbergh, M., 2007a. Transportation on demand. In: Transportation. Amsterdam: North-Holland, pp. 429–466.

Diana, M., Quadrifoglio, L., Pronelloa, C., 2007. Emissions of demand responsive services as an alternative to conventional transit systems. Transportation Research Part D: Transport and Environment 12 (3), 183–188.

Fernandez, J., De Cea, J., Briones, J., 2006. A diagrammatic analysis of the market for cruising taxis. Transportation Research Part E 42 (6), 498–526.

Flores-Guri, D., 2003. An economic analysis of regulated taxicab markets. Review of Industrial Organization 23 (3-4), 255–266.

Hackner, J., Nyberg, S., 1995. Deregulating taxi services: a word of caution. Journal of Transport Economics and Policy 29 (2), 195–207.

Helsinki Region Traffic, Oct. 2010. Jouko neighbourhood routes and service routes.
URL `http://www.hsl.fi/EN/passengersguide/ServiceroutesandJoukoroutes/Pages/default.aspx`

Jokinen, J.-P., Sihvola, T., Hyytiä, E., Sulonen, R., June-July 2011. Why urban mass demand responsive transport? In: IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS). Vienna, Austria, to appear.

Lagos, R., 2003. An analysis of the market for taxicab rides in new york city. International Economic Review 44 (2), 423–434.

Mageean, J., Nelson, J. D., 2003. The evaluation of demand responsive transport services in europe. Journal of Transport Geography 11 (4), 255–270.

Matsushima, K., Kobayashi, K., 2006. Endogenous market formation with matching externality: an implication for taxi spot markets. In: Structural Change in Transportation and Communications in the Knowledge Economy. Edward Elgar, pp. 313–336.

Moore, A., Balaker, T., 2006. Do economists reach a conclusion on taxi deregulation? Econ Journal Watch 3 (1), 109–132.

Mulley, C., Nelson, J. D., 2009. Flexible transport services: A new market opportunity for public transport. Research in Transportation Economics 25 (1), 39–45.

Reiser, M., Lavenberg, S. S., 1980. Mean value analysis of closed multi-chain queueing networks. JACM 27.

Ross, S. M., 1995. Stochastic Processes. Wiley. ISBN 978-0-471-12062-9.

Sihvola, T., Häme, L., Sulonen, R., 2010. Passenger Pooling and Trip Combining Potential of High-Density Demand Responsive Transport. Presented at the Annual Meeting of the Transportation Research Board, Washington D.C.

Small, K. A., Verhoef, E. T., 2007. The Economics of Urban Transportation. Routledge, 2 Park Square, Milton Park, Abingdon, OX14 4RN.

Wong, K., Wong, S., Bell, M., Yang, H., 2005. Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing markov chain approach. Journal of Advanced Transportation 39 (1), 81–104.

Yang, H., Leung, C. W. Y., Wong, S. C., Bell, M. G. H., 2010. Equilibria of bilateral taxi-customer searching and meeting on networks. Transportation Research Part B 44, 1067–1083.

Yang, H., Wong, S., Wong, K., 2002. Demand-supply equilibrium of taxi services in a network under competition and regulation. Transportation Research Part B 36, 799–819.

Yang, H., Ye, M., Tang, W. H.-C., Wong, S. C., 2005. A multiperiod dynamic model of taxi services with endogenous service intensity. Operations Research 53 (3), 501–515.

Publication IV

Jani-Pekka Jokinen, Lauri Häme, Esa Hyytiä, Reijo Sulonen. Simulation Model for a Demand Responsive Transportation Monopoly. In *Kuhmo Nectar Conference on Transport Economics*, Stockholm, Sweden, June-July 2011.

Simulation Model for a Demand Responsive Transportation Monopoly

Jani-Pekka Jokinen[1,*]

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Lauri Häme

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Esa Hyytiä

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Reijo Sulonen

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Abstract

In this paper, we consider alternative regulation policies for a demand responsive transportation (DRT) monopoly, where public transportation is provided by a fleet of vehicles capable of quickly adapting to the constantly changing demand. On one hand, we are interested in the profit of the monopoly, and on the other hand, the level of service and the surplus the customers obtain. A combination of these, referred to as social welfare, serves as a well-defined one-dimensional objective to be maximized by means of regulation policies. We propose a new type of real-time regulation policy, enabled by fully automated vehicle dispatching, yielding a significantly higher social welfare than the considered traditional regulation policies. Our main research question is: How regulation policies affect the DRT monopoly?

Keywords: Demand responsive transportation, Monopoly, Regulation policy, Simulation

1. Introduction

Demand responsive transportation (DRT) is a form of public transportation involving flexible routing and scheduling of small or medium sized vehicles, somewhere between bus and taxi which covers a wide range of transportation services ranging from less formal community transportation

*Corresponding author

Email addresses: Jani-Pekka.Jokinen@tkk.fi (Jani-Pekka Jokinen), Lauri.Hame@tkk.fi (Lauri Häme), Esa.Hyytia@tkk.fi (Esa Hyytiä), Reijo.Sulonen@tkk.fi (Reijo Sulonen)

[1]Tel.: +358 40 576 3585

through to area-wide service networks[2] (Mageean and Nelson, 2003). Earlier work suggests that this type of transportation services has potential to become a socially and economically remarkable form of public transportation, see for example (Mulley and Nelson, 2009; Sihvola et al., 2010; Jokinen et al., 2011). Additionally, DRT service is more effective than a fixed route service in minimizing emissions of pollutants (Diana et al., 2007).

In this work, we study a sophisticated form of DRT service operating in an urban area with a high demand. The studied service is an instant-response DRT service, which means that customers are given *trip offers* immediately after requesting service. The bidding of trip offers and vehicle routing is fully automated, which is necessary in order to provide DRT service efficiently for a high demand market. An urban area is interesting for several reasons. Firstly, DRT services are often motivated by problems arising from the congestion of urban areas caused by the increasing number of private cars, i.e., DRT could be a competitive transportation mode for private cars and taxis in terms of quality of service and price, and thereby reduce the number of private cars (see for example Cortes and Jayakrishnan, 2001). Secondly, *high demand* in urban areas can enable considerable business opportunities for transport operators and economic benefits for society. Thirdly, *high demand density*[3] in urban areas enables effective use of vehicles in DRT, since the possibilities for trip combining without notable travel time increase are significantly improved (Sihvola et al., 2010).

As the present technology enables implementing the described advanced DRT service, the pilot projects are becoming reality soon. For our knowlege, the first pilot for highly automated public DRT service starts at Helsinki in 2012.

One fundamental question related to DRT is, how regulation policies affect an advanced DRT service with a real-time trip trading and vehicle routing. Can traditional regulation policies improve the *social welfare*? We define social welfare as the sum of profit and customer surplus, similarly as in, for example, (Yang et al., 2002) and (Yang et al., 2005). In this work, we use a simulation approach to study and compare different regulation policies for a monopoly of instant-response DRT service.

The DRT is studied as complementary service to conventional bus and taxi services, i.e., customers can always choose between these transportation modes. Demand for DRT is modeled as a linear function of price and level of service, describing potential customers' willingness to pay for different types of trips. We are interested in customers' travel mode choice decisions between DRT and other modes. As in widely used random utility choice models (see for example Polak and Heertje, 2001), customers are assumed to choose the utility-maximizing alternative. For simplicity, we assume that potential DRT customers always have an alternative transportation mode where the surplus is zero, which means that a customer chooses the DRT service for a given trip if the expected surplus is positive.

Moreover, we consider total cost of service as a sum of fixed cost and variable cost. The fixed cost is a function of the number of vehicles and the variable cost is a function of total vehicle mileage. The quality of service is modeled by means of the ratio of the realized travel time to the direct ride time of the trip (more detailed definitions are given in Section 2.1).

[2] area-wide service = a transit service which gives a reasonably uniform level of service throughout an area.

[3] Demand density = number of trip requests per hour per square kilometer (Sihvola et al., 2010).

The rest of the paper is organized as follows. Section 2 describes the model of an instant response DRT service, describes the decision making process of customers and vehicle operators, and defines the studied regulation policies. Section 3 describes the simulation model and used parameters, and presents the simulation results. Section 4 concludes the paper with a discussion about future research directions.

2. Model

The model of an instant response DRT service studied in this work is governed by the following preliminary assumptions.

1. There are K vehicles available to transport customers requesting service within a certain operating zone A. For each pair of points a and b in A, the distance $d(a, b)$ and direct ride time $t(a, b)$ are known and equal for all vehicles.
2. At each moment, each vehicle is assigned a certain set of customers and a tentative route passing through all unvisited pick-up and drop-off points associated with these customers. In this work, we assume that the vehicles follow the shortest route with respect to known customers.
3. At any moment, a new customer may request a trip from a specific pick-up point a to a specific drop-off point b constituting a request for trip (a, b). In addition, each request is associated with a unit load (1 passenger/request).
4. As an instant response to each customer request, each vehicle formulates at most one proposal for transportation by means of the following procedure: A new route is determined, passing through the pick-up and drop-off points associated with already assigned customers and the new customer. Expected pick-up and drop-off times are calculated by means of this route, as depicted in Figure 1.
5. The monopolist compares all available proposals formulated by vehicles, and offers the customer a single proposal maximizing the expected profit[4]. If an offer is accepted, the customer is assigned to the corresponding vehicle and its route is modified appropriately (see Figure 1).
6. There is a fixed fare structure. The price R of a trip is assumed to be dependent on the direct trip length $d(a, b)$ exclusively by means of the formula

$$R(a, b) = p \cdot d(a, b), \tag{1}$$

where p is the price per kilometer in relation to direct trip length[5]. The customers know the fare structure and the direct trip lengths of their trips beforehand.

[4]Calculating the expected profit for a trip proposal in a real-time DRT service is a complex task. See Section 3 for a description of the algorithm used in the simulation model.

[5]In this work we define the price per kilometer in relation to direct trip length, even though the realized trip length is normally greater than the direct trip. In practice, it might be reasonable to consider a sum of fixed initial price and a price per kilometer for each trip. For simplicity, we approximate this type of pricing by using a single price parameter.

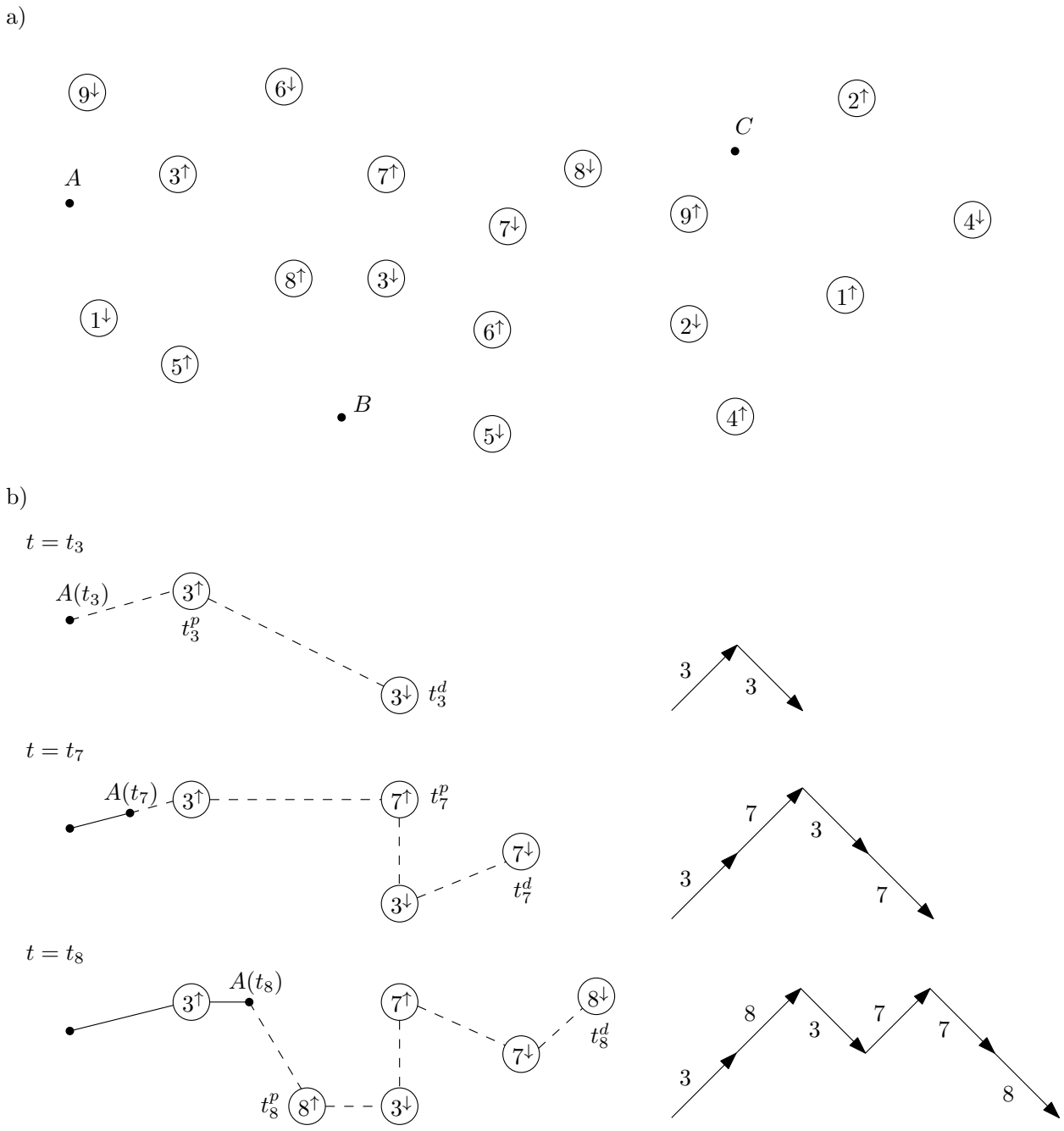


Figure 1: Formulating proposals. The top figure (a) shows the pick-up points (denoted by ↑-signs) and drop-off points (↓-signs) of nine customers and the initial locations of vehicles A, B and C. Each customer i requests a trip from $i^{\uparrow}$ to $i^{\downarrow}$ at time $t = t_i$, where $t_1 < t_2 < \ldots < t_9$. The bottom figure (b) shows the modifications in the route of vehicle A. At the time customer 3 requests a trip ($t = t_3$), the vehicle is located at $A(t_3)$. A new route for the vehicle, namely $(3^{\uparrow}, 3^{\downarrow})$, beginning from $A(t_3)$ is calculated and the expected pick-up and drop-off times, t_3^p and t_3^d, are determined by means of the new route. Customer 3 accepts the proposal and the vehicle route is updated. Customers 7 and 8 are added to the vehicle route in a similar fashion. The figures on the right show the routes as so-called labeled Dyck paths (Cori, 2009; Häme, 2011), in which each pick-up $i^{\uparrow}$ precedes the corresponding drop-off $i^{\downarrow}$. After each step, a new path is formed due to the addition of a new customer. Clearly, the "height" of the path shows the number of customers aboard in different parts of the route.

Note that due to the dynamic nature of the problem, the realized pick-up and drop-off times may differ from the expected pick-up and drop-off times due to the assignment of future customers. This means that the realized level of service may be worse than the expected in some situations. In this work, however, we are only interested in cases in which the average surplus of customers is positive (see Section 2.1). In other words, the realized level of service is better than what the customers are willing to accept most of the time.

In the following, we will elaborate the modeling of customer behavior, DRT monopoly and regulation policies in more detail.

2.1. Behavior of customers

The level of service provided by various transportation modes is normally characterized in terms of travel time and its components (Talvitie and Dehghani, 1980). In this work, for simplicity, we model the level of service by means of *travel time ratio*, which describes the ratio of travel time to direct ride time. In other words, a travel time ratio equal to one corresponds to the best possible level of service and a larger travel time ratio corresponds to a lower level of service.

Since the fare structure for trips is assumed to be fixed, the only question a customer faces after requesting service is whether or not to accept the best transportation offer in terms of the *offered level of service*. The level of service offered to a customer traveling from a to b is defined by means of *expected travel time ratio*

$$\tau = \frac{t_d - t_r}{t(a,b)}, \tag{2}$$

where t_d is the expected drop-off time[6], t_r is the release time of the request and $t(a,b)$ is the direct ride time from a to b. In other words, τ describes the ratio of the expected travel time of a given proposal to direct ride time. Clearly, since $t_d - t_r \geq t(a,b)$, we have $\tau \geq 1$, and $\tau = 1$ corresponds to the best possible offered level of service.

Due to modifications in the vehicle routes, the offered level of service may be different from the final outcome of the service. We define *realized travel time ratio* τ' by means of the formula

$$\tau' = \frac{t'_d - t_r}{t(a,b)}, \tag{3}$$

where t'_d is the realized drop-off time, that is, the time the customer actually reaches the destination.

As previously stated, we consider the DRT as complementary service to conventional taxi and fixed-route bus services. We assume that potential DRT customers have always alternative transportation mode where surplus is zero. Thus, a customer rejects DRT trip proposal if the expected surplus is negative. That is, a customer that rejects a DRT offer is assumed to travel by some other means. In order to attract customers from these transportation modes, it is assumed that the offered level of service τ should be better than that of the bus, and the price per kilometer p

[6]The calculation of the expected drop-off time in a real-time DRT service is a non-trivial task, since the route may be modified during the trip. Here we assume that the service provider can estimate the expected drop-off time with sufficient accuracy.

should be lower than the price per kilometer of a taxi. In addition, we associate with each customer a certain *willingness to pay*, which describes the maximum price the customer accepts for a certain level of service. The assumptions that hold for all customers and thus define the demand model can be summarized as follows.

1. Denote the average level of service provided by a conventional taxi by τ_T and the price per kilometer[6] of a taxi by p_T. If the travel time ratio offered by the DRT service is greater than τ_T and the price per kilometer is greater than p_T, no customer accepts the offer.
2. Denote the average level of service provided by traditional bus transportation by τ_B and the price per kilometer by p_B. We assume that $\tau_B > \tau_T$ and $p_B < p_T$. That is, the average level of service and price per kilometer of traditional bus transportation are lower than those of a taxi. If the travel time ratio offered by the DRT service is greater than τ_B and the price per kilometer is greater than p_B, no customer is willing to accept the offer.
3. Finally, if the travel time ratio offered by DRT is less than τ_T and the price per kilometer is less than p_B, each customer accepts the offered trip.

Since the service studied in this work is hypothetical, no accurate information on customer behavior is available. We use a linear approximation, in which the fraction of customers that are willing to accept a certain level of service for a certain price increases linearly from (τ_T, p_T) to (τ_T, p_B) and from (τ_B, p_B) to (τ_T, p_B). In other words, the three points

$$(\tau_T, p_T, 0), (\tau_B, p_B, 0) \text{ and } (\tau_T, p_B, 1)$$

define a linear demand model as a plane in $\mathbb{R}^3$, Figure 2.

The demand model can thus be expressed by means of a probability distribution $\mathrm{P}(\text{accept} \mid \tau, p)$, which denotes the conditional probability that an arbitrary customer accepts a proposal with expected travel time ratio τ and price per kilometer p. The proposed acceptance probability function $\mathrm{P}(\text{accept} \mid \tau, p)$ is formally given by the equation

$$\mathrm{P}(\text{accept} \mid \tau, p) = \min\left(\max\left(1 - \frac{p - p_B}{p_T - p_B} - \frac{\tau - \tau_T}{\tau_B - \tau_T}, 0\right), 1\right). \tag{4}$$

An example of the acceptance probability function with $\tau_T = 1.5, \tau_B = 3, p_B = 0.4$ and $p_T = 2.3$ is illustrated in Figure 2. The relation to alternative transportation modes is clarified in Figure 3, which shows the basic idea behind the demand model.

2.1.1. Willingness to pay and surplus

Since our demand model (4) is linear, we may equivalently think that with a certain price per kilometer $p \in [p_b, p_T]$, the worst level of service that an arbitrary customer is willing to accept follows a uniform distribution. Similarly, for a certain offered level of service τ, each customer is willing to pay a certain maximum price per kilometer, which is also uniformly distributed. The upper bound $p(\tau)$ for the maximum price per kilometer over all customers is a straight line in the

[6] In this work we define the price per kilometer in relation to direct trip length for all transportation modes.

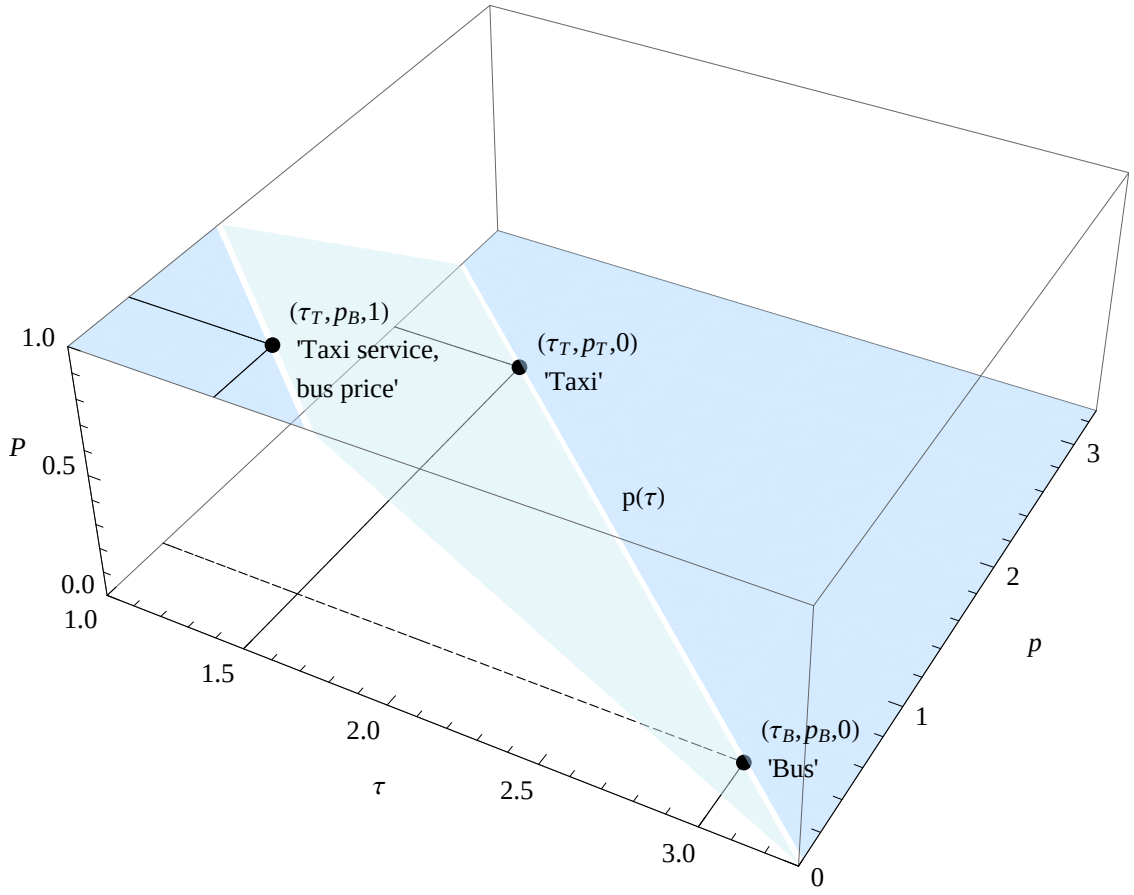


Figure 2: The acceptance probability function P(accept | τ, p) determined by $\tau_T = 1.5$, $\tau_B = 3$, $p_T = 2.3$ and $p_B = 0.4$. The linear function is specified by means of three points representing conventional bus and taxi services and a taxi-type service for the price of a bus.

plane $P = 0$, see figure 2. The equation of $p(\tau)$ can be written in terms of the two points (τ_T, p_T) and (τ_B, p_B), namely

$$p(\tau) = p_T - \frac{\tau - \tau_T}{\tau_B - \tau_T}(p_T - p_B).$$

Clearly, $p(\tau_T) = p_T$ and $p(\tau_B) = p_B$. We define the *willingness to pay* with level of service τ as a real-valued random variable $W(\tau)$ by means of the uniform distribution

$$f_{W(\tau)} = U\Big(p(\tau) - (p_T - p_B), p_\tau\Big). \tag{5}$$

The random variable $W(\tau)$ describes the price per kilometer that an arbitrary customer is willing to pay, when the level of service τ is known. For a trip (a, b), an arbitrary customer is willing to pay the maximum price $W(\tau) \cdot d(a, b)$.

The acceptance probability with given service level τ and price per kilometer p is equal to the

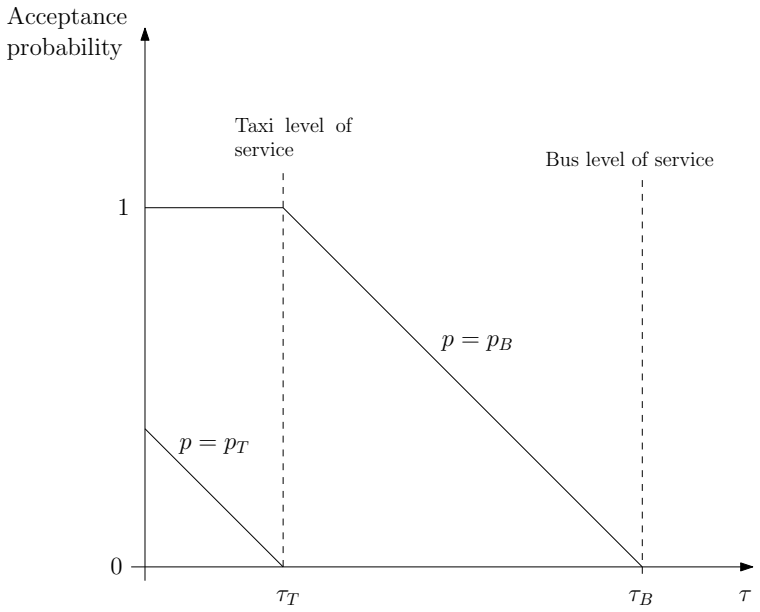


Figure 3: The acceptance probability function related to alternative transportation modes. If the level of service is as good as that of a normal taxi ($\tau = \tau_T$) and the price is equal to the price of a bus ($p = p_B$), any customer is likely to accept the DRT service. The fraction of customers willing to accept the service decreases with the level of service. If the price and level of service are equal to the taxi ($p = p_T$ and $\tau = \tau_T$), the acceptance probability is zero.

corresponding cumulative distribution function

$$\mathrm{P}(\text{accept} \mid \tau, p) = \mathrm{P}(W(\tau) \geq p) = \int_p^\infty f_{W(\tau)}(x)\, dx. \tag{6}$$

For each realized trip (a, b) with the price per kilometer p and realized travel time ratio τ', the *realized surplus* S' is defined as the difference between the willingness to pay $W(\tau') \cdot d(a, b)$ for the realized level of service and the actual price paid for the trip $R(a, b) = p \cdot d(a, b)$. If the customer does not accept the trip, the realized surplus is zero. The *expected surplus*, denoted by $\mathrm{E}[S \mid \tau, p]$, is thus given by

$$\mathrm{E}[S \mid \tau, p] = \mathrm{P}(W(\tau) \geq p) \cdot \Big(d(a, b) \cdot \mathrm{E}[W(\tau) \mid W(\tau) \geq p] - R(a, b)\Big), \tag{7}$$

where $\mathrm{E}[W(\tau) \mid W(\tau) \geq p]$ is the expected willingness to pay on condition that an offer with expected level of service τ and price per kilometer p is accepted. Note that the conditional expected willingness to pay satisfies $\mathrm{E}[W(\tau) \mid W(\tau) \geq p] \geq \mathrm{E}[W(\tau)]$.

2.2. Market mechanisms and regulation policies

First, in Section 2.2.1 we describe the long-run profit maximization decisions, i.e., decisions about the number of vehicles and the price per kilometer. The aim of the long-run decisions is to maximize the expected profit for each day, but these decisions cannot be changed during the day.

In Section 2.2.2 we describe daily decisions, i.e., the trip offering decisions which are made during the day.

We consider two types of traditional regulation policies which are focused on the long-run decisions and used simultaneously: 1. price regulation and 2. vehicle number regulation. In addition, we consider a novel new regulation policy which is focused on the monopolist's daily trip offer-making decisions. The new regulation policy is explained more detailed in Section 2.2.2. In all regulation policies the aim is to maximize social welfare defined as the sum of profit and surplus.

2.2.1. Long-run decisions

A monopoly operator controls all available vehicles. Number of vehicles, denoted by K, and price per kilometer (length of trip is measured from direct trip), denoted by p, are determined in a way that the daily profit is maximized. The optimal number of vehicles and price are given by

$$(K^*, p^*) = \arg\max_{K,p} D_s\bar{R} - KC_F - C_V, \tag{8}$$

where D_s is the number of sold trips ($D_s <$ potential demand), $\bar{R}$ is the average price of a trip, C_F is the fixed cost of vehicle and C_V is the variable cost of vehicles (which depend on driven kilometers), during one day. $D_s, \bar{R}$ and C_V are functions of K and p.

2.2.2. Daily decisions

At any instant, each vehicle is restricted to follow the route that minimizes the total route length with respect to the customers assigned to the vehicle. In addition, a trip offer for a customer x is generated by means of the shortest tour with respect to x and the existing customers assigned to the vehicle. The shortest tour is chosen for two reasons: Firstly, it is the most efficient way of providing service for known customers. Secondly, it guarantees a certain fairness of service since all customers are treated equally (no customer may be favored by proposing an unreasonably good trip at the expense of other customers). In this work, we study two different methods for generating proposals at the time a customer request is released:

1. Monopoly market mechanism: In order to maximize the total revenue, the monopolist aims to formulate a single proposal in a way that the additional cost of providing the service is minimized and the probability that the customer accepts the proposal is maximized. In other words, the monopolist chooses for each trip (a, b) the vehicle in a way that the expected profit

$$\mathrm{E}[\pi] = \mathrm{P}(\text{accept} \mid \tau, p) \cdot \Big(R(a, b) - \mathrm{E}[\Delta C]\Big) \tag{9}$$

is maximized, where $\mathrm{P}(\text{accept} \mid \tau, p)$ denotes the acceptance probability with level of service τ and price per kilometer p. $\mathrm{E}[\Delta C]$ denotes the expected increase in the cost of the vehicle route caused by the new potential customer.

2. Real time regulation for monopolist: As in the former case, a single proposal is presented for each customer requesting service. Instead of expected profit, the vehicle is chosen in a way that the expected social welfare

$$\mathrm{E}[L] = \mathrm{P}(\text{accept} \mid \tau, p) \cdot \Big(R(a, b) - \mathrm{E}[\Delta C]\Big) + \mathrm{E}[S \mid \tau, p] \tag{10}$$

is maximized, where $\mathrm{E}[S \mid \tau, p]$ is the expected surplus defined by (7).

3. Simulation

We consider a model for DRT adopted from (Hyytiä et al., 2010), where trip requests occur within a bounded region in a plane. Each trip request is defined as a triple, (t_i, a_i, b_i), where t_i denotes the time instance (release time) of the ith request, a_i and b_i denote the origin and the destination of the trip, respectively.

We assume that trip requests arrive according to a Poisson process with rate λ [trip/s], and that for each trip request both the pick-up and drop-off locations are uniformly distributed in a finite convex region with area A (i.e., the trip request arrive according to a Poisson point process).

There are K vehicles each with c passenger seats to support the given transportation demand in online fashion. We assume Euclidean distances between any two points and thus each vehicle uses the direct path between the waypoints that define the route. In addition, as already stated, the ordering of waypoints is determined in a way that the route length is minimized. When a trip request arrives, it is immediately assigned to a single vehicle with probability P(accept $\mid \tau, p$) as discussed in section 2.1. The chosen vehicle then, at some point of time, picks up the passenger for delivery to the corresponding destination point. With $c > 1$, several passengers can share a vehicle, which allows the system to combine trips and decrease the effort per passenger.

For analysis, we assume a constant velocity v (e.g., 36 km/h) and a constant minimum stop time of t_{st} (e.g., 30 s) for each vehicle. After the minimum stop time, passengers can enter and exit the vehicle until it starts moving again. The minimum stop time is assumed to include the time needed for deceleration and acceleration. Note that in our trip demand model (Poisson point process), each stop corresponds to exactly one passenger entering or exiting the vehicle at a time, i.e., passengers do not share the stops (cf., door-to-door vs. stop-to-stop service). The basic parameters used in the simulation are presented in table 1.

We simulate a period of time representing one day. The simulation starts with all vehicles idle and ends when the last customer is dropped off at the destination. The simulation time in table 1 corresponds to the period of time during which customers request trips. After the simulation time has expired, the remaining accepted customers are still processed.

3.1. Real-time optimization

Calculating the additional cost of providing service for a new customer in a real-time transportation service is generally seen to be a challenging task (Psaraftis, 1995). In this work, the expected increase in the cost of the vehicle route $\mathrm{E}[\Delta C]$ in (9) and (10) is estimated by taking into account i) the immediate cost increase caused by the increase in route length and ii) the expected additional cost due to the allocated vehicle capacity. The latter is calculated by means of the average realized profit rate of a seat in a vehicle as follows. Letting t_0 denote the beginning of the simulation period and t_i denote the release date of a new customer i, the average realized profit rate R_{t_i} of a seat in a vehicle at t_i is calculated by dividing the total price of realized trips until t_i by the total vehicle seat hours until t_i, namely $Kc(t_i - t_0)$. The expected additional cost due to the allocated vehicle capacity is given by $r_i \cdot R_{t_i}$, where r_i is the tentative ride time of the customer. In other words, each served customer occupies a single seat in a vehicle between the pick-up and drop-off. Thus, when customer x accepts a trip offer, the possibilities of making profit in the future are reduced since the seat occupied by x cannot be used to transport new customers when customer x is in the vehicle.

trip request rate:	1/s
simulation time:	12 hours
area:	disk with 5 km radius
velocity of vehicles:	10 m/s
capacity of vehicles:	10
stop time:	30 s
Fixed costs:	200 euros per vehicle per day
Variable costs:	0.5 euros per vehicle kilometer
τ_T:	1.5
τ_B:	3
p_B:	0.4 euros
p_T :	2.3 euros

Table 1: Basic simulation parameters. The values for the costs and the prices describe roughly the Finnish price and cost level.

3.2. Long-run optimization

The simulator is used to find the optimal number of vehicles K and price per kilometer p in three cases: 1) monopoly, 2) regulated monopoly, 3) real-time regulated monopoly. In non-regulated monopoly, the objective is to maximize profit as defined in Equation (8). In the regulated cases 2 and 3, we seek to maximize social welfare defined as the sum of surplus and profit. The optimal values K^* and p^* are determined for each case by running the simulator with different values of K and p and choosing the best combination with respect to the objective. We use increment 5 for the number of vehicles and 0.01 for the price per kilometer.

3.3. Confidence analysis

Since we are interested in daily operations of a DRT service, the simulation time is fixed for each run. In order to eliminate noise from the results, we calculate the mean values of quantities over several runs using the same parameter values. Let us study the width of the 95% confidence interval for mean social welfare as a function of the number of simulation runs. More precisely, we examine the relative margin of error m defined by means of the formula $m = r/\bar{l}$, where r is the radius of the confidence interval and $\bar{l}$ is the observed mean social welfare. In other words, the difference between the observed mean from the simulation model and and the true mean is at most r with probability 95%. The relative margin of error expresses the ratio of r to the observed mean. Figure 4 shows the relative margin of error for mean social welfare in the monopoly mechanism as a function of the number of runs with $K = 210$ and $p = 1.65$.

Referring to Figure 4, we note that the margin of error is relatively small even if only a few runs were performed. With 10 runs, the margin of error is approximately 0.8%. With 100 runs, the margin of error is only order of 0.2%. As will be seen in the following section, the accuracy achieved by 100 runs is sufficient to identify the main differences between the studied regulation policies.

We determine the optimum (K^*, p^*) for each mechanism by means of a local search as follows: At first, an approximate optimum (K_1^*, p_1^*) is determined by performing an initial run for each

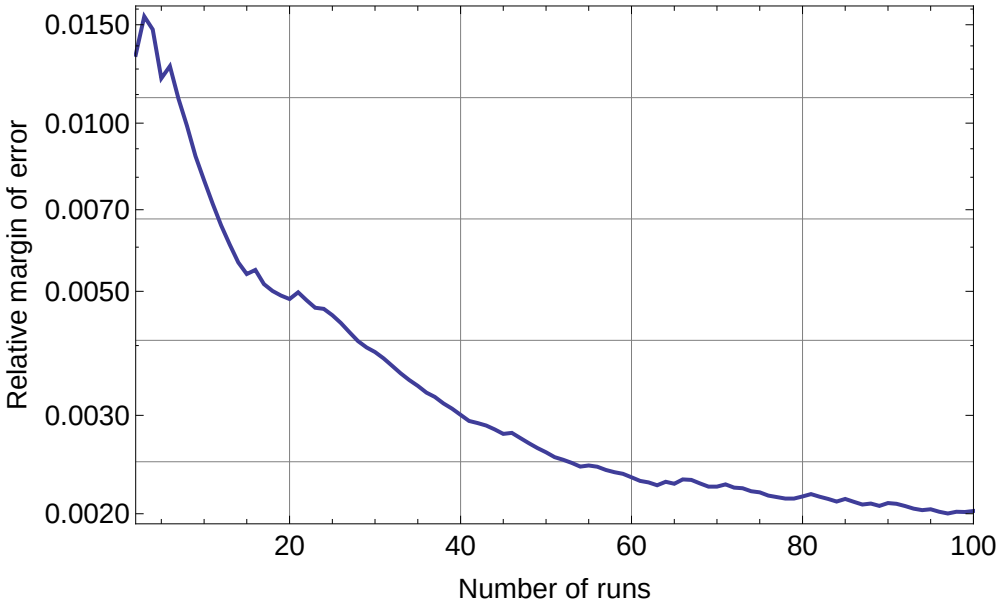


Figure 4: Relative margin of error at a 95% confidence level for the mean social welfare in the monopoly mechanism as a function of the number of runs, on a logarithmic scale, with $K = 210$ and $p = 1.65$.

$(K, p) \in \{100, 110, \ldots, 800\} \times \{0.4, 0.45, \ldots, 2.3\}$ (see figure 5). Then, a second optimum (K_2^*, p_2^*) is obtained by calculating the average over 10 runs for each point in the neighborhood grid of (K_1^*, p_1^*) consisting of 81 points, namely $\{K_1^* - 40, \ldots, K_1^*, \ldots K_1^* + 40\} \times \{p_1^* - 0.2, \ldots, p_1^*, \ldots, p_1^* + 0.2\}$. The new optimum (K_3^*, p_3^*) is given by calculating the average over 100 runs for nine points in the neighborhood of (K_2^*, p_2^*). If $(K_3^*, p_3^*) \neq (K_2^*, p_2^*)$, we set $(K_2^*, p_2^*) = (K_3^*, p_3^*)$ and repeat the last step until the equality is valid. Since our search space is finite, the above local search procedure obviously converges to some point.

3.4. Results

The results of the simulations are summarized in Table 2. The table shows the optimal price per kilometer and the optimal number of vehicles for each market mechanism together with the corresponding total profit, realized customer surplus and social welfare, i.e. the sum of the profit and surplus. In addition, the number of served customers, average realized travel time ratio and relative driven distance are given for each case. The average travel time ratio is determined by dividing the total travel time of customers by the sum of direct trip ride times. Thus, it describes the average level of service experienced by the customers. The relative driven distance is calculated by dividing the total distance driven by vehicles by the sum of direct trip lengths. Consequently, it describes how efficiently the vehicles are utilized. The average values in the table were calculated over 100 runs with the parameter values of Table 1. The average number of requested trips was 43253 in each case.

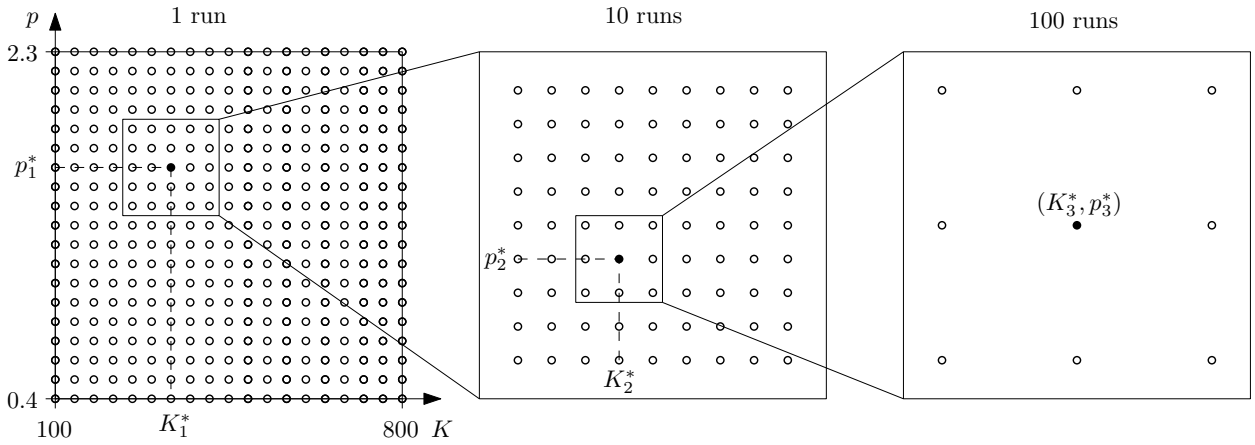


Figure 5: Determining optimal number of vehicles K and price per kilometer p by means of simulation. At first, an initial run is executed for each $(K, p) \in \{100, 110, \ldots, 800\} \times \{0.4, 0.45, \ldots, 2.3\}$. Then, the optimum (K_1^*, p_1^*) is updated by narrowing the parameter space to 81 points chosen from the neighborhood of the optimal point and performing 10 runs for each point, yielding (K_2^*, p_2^*). Finally, a new optimum is determined by performing 100 runs for nine points in the neighborhood of the previous optimum. The last step is repeated until convergence.

Market mechanism	Number of vehicles K	Price/ km p	Surplus	Profit	Social welf.	Served cust.	Travel time ratio	Relative dist.
1) Monopoly	210	1.65	417	94535	95429	21213	1.688	0.613
2) Regulated monopoly	240	1.65	5089	91799	96888	22180	1.642	0.626
3) Real time reg. monopoly	320	1.40	31162	81488	112650	29387	1.585	0.660

Margin of error at a 95% confidence level					
Surplus	Profit	Social welf.	Served cust.	Travel time ratio	Relative dist.
150	180	200	20	0.0011	0.00029

Table 2: Simulation results. The first columns of the upper table show the optimal number of vehicles K and price per kilometer p for the three studied cases. The remaining columns show the corresponding average values of surplus, profit, social welfare, number of served customers, realized travel time ratio and relative driven distance calculated over 100 runs with the parameter values of table 1. The lower table shows the margin of error of the mean of the studied quantities at a 95% confidence level.

3.4.1. Monopoly

From table 2, one can observe that while the monopoly (1) yields the highest profit, focusing only on profit will wither the total customer surplus. By regulating the kilometer price and the number of vehicles of the monopoly in order to maximize social welfare (2), the total customer surplus, number of served customers, average level of service and number of vehicles increase a bit, but the optimal price remains the same. It seems that price regulation of DRT monopolist is relatively ineffective compared to vehicle regulation, because the monopolist can react to regulated lower price by dropping service level in daily dispatching decisions. Referring to the relative driven distance, it can be seen that the monopoly without regulation is slightly more efficient than regulated monopoly.

We studied scale economies of DRT by simulating the monopoly market mechanism with different levels of potential demand. The results in Table 3 indicate significant economies of scale,

Demand (requests / s)	Number of vehicles K	Price/km p	Surplus	Profit	Social welfare	Served customers	Travel time ratio	Relative distance
0.25	65	1.57	46	17808	17854	5356	1.746	0.757
0.5	120	1.58	433	41960	42393	11089	1.724	0.672
0.75	165	1.63	267	68301	68569	16528	1.717	0.627
1	210	1.65	416	94535	95414	21213	1.688	0.613

Table 3: Economics of scale. The table shows the outcome of the monopoly mechanism with different demands (0.25,0.5,0.75,1 requests /second).

which can be seen from decreasing relative distance due to the higher demand. This means that on a higher demand level, less vehicle kilometers are needed to produce the same number of trips. This is a natural consequence of improved possibilities to combine trips. The other interesting result is the positive effect of scale on the level of service, which can be seen from the decreasing travel time ratio.

3.4.2. Real-time regulated monopoly

The last market mechanism in comparison, i.e. real-time regulated monopoly market mechanism (3), is a novel attempt to transform the monopolist from profit maximizer to social welfare maximizer. The monopolist's regulation is focused both on long run decisions (price and vehicle number) and on daily trip offering decisions to maximize expected social welfare of each trip request, which results in clearly higher social welfare than the other mechanisms. This is a very interesting result for two reasons. Firstly, this mechanism is a "winner" of the comparison, because it leads to highest social welfare with pretty fair allocation of utility between profits and customers surplus. Secondly, the result suggests that this type of new ICT enabled transportation service, where the vehicle routes are not fixed beforehand and routing decisions are delegated from the driver to the computer and trip offers are also automated, enables the social planner to use new regulation policies that can be much more efficient than traditional regulation policies such as price- and entry-regulations.

3.5. Discussion on the validity of the results

In this paper, we have compared various market mechanisms for the DRT service with automated trip proposals and fixed fare structure by means of simulations. This includes designing an appropriate model for the task that captures the important phenomena and neglects the rest. One can identify simplifications related (i) to the DRT service model, and (ii) to the anticipated DRT market as we see it. In the following we try to justify the modeling assumptions and argue that they have negligible effect to the validity of the comparison.

The used model for DRT service is elementary, i.e., trip requests arrive according to a Poisson process with a constant rate, the pick-up and drop-off locations are uniformly distributed, each trip request corresponds to exactly one passenger, there is no road network, and vehicles move at a constant velocity. Poisson process is a widely used model, e.g., for telephone calls and generally it models well many activities a large human population generates. However, some caution is still in place. For example, a packet arrival process in communication networks is typically considerably more bursty than what a Poisson process suggests (Paxson and Floyd, 1995). Similarly, spatially

the trips in reality are hardly uniformly distributed, e.g., some destinations are more popular at certain time of day than others. We believe that such differences affect mainly the absolute level of performance, i.e., the relative performance between the market mechanisms remains the same. Similarly, the road network and velocity aspects are unnecessary details in this respect. Also, the assumption of single passenger per trip request is merely a scaling factor, as long as the occupancy in the vehicles remains below the capacity (seats per vehicle), as typically was the case in our experiments.

Moreover, there are three simplifying assumptions on the customer behavior: (i) Customers may accept a trip proposal immediately after receiving the proposal, (ii) Customers perceive quality of service by means of travel time index (Section 2.1), (iii) Customers accept a trip proposal if expected surplus is positive (Section 2.1). The first assumption is obviously unrealistic, bacause a customer typically needs at least few seconds to make desicion. However, one can expect that "thinking times" have negligible influence to the comparison of the market mechanisms, and thus can be omitted. Regarding (ii), we acknowledge that the perceived quality of service is a complicated issue itself, but argue that the travel time is the most important factor with this respect. The third assumption ignores the fact that not only the expected travel time but uncertainty about travel time, i.e., the variance of travel time is also an important criterion for passengers, see for example (Small and Verhoef, 2007). However, the variance of travel time is quite similar in all compared market mechanisms. Thus, the third simplifying assumption is harmless in the context of the study.

4. Conclusions and future research

In this paper, we have analyzed and compared alternative regulation policies for a monopoly of a DRT service by using a simulation model. In particular, we studied three versions of monopoly market mechanisms, i.e., non-regulated, price and vehicle number regulated, and real-time regulated monopoly. Social welfare is highest in the real-time regulated monopoly mechanism. Moreover, our results indicate that DRT service has positive economies of scale and a positive effect of scale on the level of service.

We studied market mechanisms with fixed fare structure. Fixed fare structures are a widely used form of pricing in transportation services. Therefore, the studied mechanisms are relevant alternatives for the demand responsive transportation services in practice. A fixed fare structure is beneficial for the customer, because she knows the price of the trip before the trip request. However, market mechanisms where fares are not fixed beforehand can result in a much higher social welfare, because trip proposing is less limited and therefore a customer has a higher probability to get an offer with a positive expected surplus. For the vehicle operator, a market mechanism with variable fares means that it is reasonable to make many offers for the customer, i.e., fast trip offers at a higher price and slower trip offers at a lower price, because the operator usually has no information on customers' preferences and acute needs beforehand. We see that this is an important future direction of research.

The other important research topic related to the market mechanisms of DRT is externalities. As we mentioned earlier, DRT is often motivated by problems arising from the congestion of urban areas caused by the increasing number of private cars. There are positive externalities if private

car users or taxi users change their transportation mode to the demand responsive transportation, whereas change from bus to the DRT can have negative externality. Studying relationships between market mechanisms, pricing, regulation policies and externalities requires comprehensive data on consumers' transportation needs, preferences and a more detailed modeling of consumers' trip mode decisions.

Understanding the effects and possibilities of market mechanisms with real time pricing presumably provide means to improve DRT service and externalities of DRT naturally must be estimated and taken into account in regulation policies. The regulation policies considered in this study provide relevant baselines for the other regulation policies and market mechanisms and the results point out for policymakers that traditional regulation policies alone are not optimal for a modern demand responsive transportation service.

Acknowledgements

This work was conducted in Metropol project that is supported by the Finnish Funding Agency for Technology and Innovation, Finnish Ministry of Transport and Communications, Helsinki Metropolitan Area Council and Helsinki City Transport. The authors would like to thank Aleksi Penttinen, Juha Savolainen and Teemu Sihvola for their useful comments and ideas during the preparation of this paper.

References

Cori, R., 2009. Indecomposable permutations, hypermaps and labeled dyck paths. Journal of Combinatorial Theory, Series A 116 (8), 1326–1343.

Cortes, C. E., Jayakrishnan, R., 2001. Design and operational concepts of a high coverage point-to-point transit system. Center for Traffic Simulation Studies.

Diana, M., Quadrifoglio, L., Pronelloa, C., 2007. Emissions of demand responsive services as an alternative to conventional transit systems. Transportation Research Part D: Transport and Environment 12 (3), 183–188.

Häme, L., 2011. An adaptive insertion algorithm for the single vehicle dial-a-ride problem with narrow time windows. European Journal of Operational Research 209, 11–22.

Hyytiä, E., Häme, L., Penttinen, A., Sulonen, R., 2010. Simulation of a large scale dynamic pickup and delivery problem. In: Proceedings of SIMUTools 2010.

Jokinen, J.-P., Sihvola, T., Hyytiä, E., Sulonen, R., June-July 2011. Why urban mass demand responsive transport? In: IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS). Vienna, Austria, to appear.

Mageean, J., Nelson, J. D., 2003. The evaluation of demand responsive transport services in europe. Journal of Transport Geography 11 (4), 255–270.

Mulley, C., Nelson, J. D., 2009. Flexible transport services: A new market opportunity for public transport. Research in Transportation Economics 25 (1), 39–45.

Paxson, V., Floyd, S., Jun. 1995. Wide area traffic: the failure of poisson modeling. IEEE/ACM Transactions on Networking 3 (3), 226–244.

Polak, J. B., Heertje, A., 2001. Analytical Transport Economics: An International Perspective. Edward Elgar Publishing Ltd.

Psaraftis, H., 1995. Dynamic vehicle routing: status and prospects. Annals of Operations Research 61, 143–164.

Sihvola, T., Häme, L., Sulonen, R., 2010. Passenger Pooling and Trip Combining Potential of High-Density Demand Responsive Transport. Presented at the Annual Meeting of the Transportation Research Board, Washington D.C.

Small, K. A., Verhoef, E. T., 2007. The Economics of Urban Transportation. Routledge, 2 Park Square, Milton Park, Abingdon, OX14 4RN.

Talvitie, A., Dehghani, Y., 1980. Models for transportation level of service. Transportation Research Part B: Methodological 14 (1-2), 87–99.

Yang, H., Wong, S., Wong, K., 2002. Demand-supply equilibrium of taxi services in a network under competition and regulation. Transportation Research Part B 36, 799–819.

Yang, H., Ye, M., Tang, W. H.-C., Wong, S. C., 2005. A multiperiod dynamic model of taxi services with endogenous service intensity. Operations Research 53 (3), 501–515.

Publication V

Teemu Sihvola, Lauri Häme, Reijo Sulonen. Passenger-Pooling and Trip-Combining Potential of High-Density Demand Responsive Transport. In *Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2010.

Passenger-Pooling and Trip-Combining Potential of High-Density Demand Responsive Transport

Teemu Sihvola*
Helsinki University of Technology
Software Business and Engineering Institute
P.O.Box 9210
FI-02015 TKK, Finland
Telephone: +358 40 735 8885
Fax: +358 9 451 4958
teemu.sihvola@tkk.fi

Lauri Häme
Helsinki University of Technology
Networking Laboratory
P.O. Box 3000
FI-02015 TKK, Finland
lauri.hame@tkk.fi

Reijo Sulonen
Helsinki University of Technology
Software Business and Engineering Institute
P.O.Box 9210
FI-02015 TKK, Finland
reijo.sulonen@tkk.fi

* Corresponding author

A Paper Accepted for Presentation at the Annual Meeting of the Transportation Research Board
Washington, D. C. January 2010

Word Count: 5,294 + 3 tables + 5 figures = 7,294

ABSTRACT

The aim of transport services is to find a good balance between efficiency and quality of service. Passenger-pooling and trip-combining increase efficiency but in most cases require passengers to compromise on the quality. In this paper, we analyze the passenger-pooling potential (PPP) and trip-combining potential (TCP) in Demand Responsive Transport (DRT) in high demand density situations. The studied DRT system operates on a real-time basis without pre-order times. Simulation study is used to analyze the effect of demand density on PPP and TCP, without relying on any particular control and routing algorithms. We also examine the effect of spatial and temporal dimensions of a DRT system to the potential of pooling and combining. The results indicate that both PPP and TCP have a strong positive correlation with demand density. With low demand densities no distinct potential for trip-combining and large-scale passenger-pooling is possible except with large walking distances. Increasing the demand density increases the possibilities of producing more efficient DRT services with better quality of service. The presented results argue for further research of DRT in high demand density situations, even though DRT system has conventionally been seen applicable mainly for low demand density situations.
Keywords*: demand responsive transport, passenger-pooling potential, trip-combining potential, high demand density*

INTRODUCTION

In a transport system, passengers and vehicles have to connect each other so that the passenger perceives a benefit from the ride. In practice, the connection means that the passenger and the vehicle have to be in the same place at the same time to enable passenger boarding. In addition, the vehicle must transport the passenger sufficiently near the destination. The connection is enabled by the temporal and spatial flexibility of a vehicle or a passenger or both. The flexibility of passengers is connected to a group of service quality attributes, such as walk, wait and ride time, that passengers value and accept differently (*1; 2*). Similarly, the flexibility of vehicles is connected to the efficiency of the transport service, as vehicle route kilometreage is typically increased as the flexibility increases.

The aim of a *Demand Responsive Transport* (DRT) system is to operate without fixed routes and timetables, thus enabling flexible service based on the passenger's trip requests. However, in order to achieve the efficiency of the system, passengers also need to be flexible in most cases by walking to and from stops and adjusting departure times based on the vehicle schedule. A central concept here is passenger-pooling. Passengers are pooled so that many passengers can either board to or alight from the vehicle at one stop. Trips are combined using the same vehicle so that the total vehicle kilometreage is decreased from the sum of direct trip lengths, but at the same time each passenger may have to travel more than the shortest distance to his or her destination.

Conventionally, DRT is seen as a transport service for special user groups or situations where the demand is too low for other public transport services (*3-6*). Low demand density, however, means that the trips are dispersed temporally and spatially. This results in low trip-combining ratio and low average occupancy (*3; 7*). Demand density has a significant impact on DRT system efficiency (*8*). The presumption we have made is that with state-of-the-art technology the DRT system can be implemented in high demand density situations so that the average occupancy is high even though the trips are ordered without pre-ordering times. One proposal for this kind of system is presented by Cortés and Jayakrishnan (*9*).

Few studies and implementations of DRT have addressed passenger-pooling. The probable reason for neglecting passenger-pooling is that the majority of current implementations have been focusing on door-to-door systems for special user groups. Imagine a system with an average of 10 passengers picked up per vehicle-hour, average ride times of 0.25 hour excluding stop times, and stop times of 30 seconds including deceleration, service at stop and acceleration. A straightforward calculation shows that if the *average number of passengers boarding or alighting the vehicle per stop* (ρ) is one, the stop times increase the ride time by 15%. With $\rho=2$ the increase in ride time is 7%: and with $\rho=3$ it is 4%. These calculations demonstrate the significance of passenger-pooling in a high demand density situation and show that the passenger-pooling is one essential part of the development of dispatching strategies.

The presented paper is a part of ongoing research project developing and studying new concepts and dispatching strategies for DRT in metropolitan areas. Our aim is to analyze the potential for passenger-pooling and trip-combining under reasonable conditions without relying on any particular control and routing algorithms under intensive study in our project. The main goal of this paper is to understand whether passenger-pooling and trip-combining are significant enough factors for more detailed studies in the DRT systems operated in the high demand density situations.

HIGH-DENSITY DEMAND RESPONSIVE TRANSPORT (HD-DRT)

In this paper, we study DRT operated in a high demand density situation. The concept is labeled as a *High-Density Demand Responsive Transport* (HD-DRT). The HD-DRT has at least the following characteristics: the *service region* is pre-defined and fixed, no timetables and predefined routes are used, passengers are picked up from stops and delivered to stops (*stop-to-stop*), service is offered

from any stop to any stop within the service region (*many-to-many*) and no pre-ordering is required. We consider an HD-DRT system where the dispatching system selects the stops for customers.

The *demand density* DD is the number of trip requests per hour per square kilometer. This study considers demand density to be high when there is at least ten trip requests per hour per square kilometer. Of note is that demand density can be a misleading measure in cases where the demand density varies greatly within different parts of the service region (*10*). The presumption is that the inhomogeneous demand distribution gives more potential for passenger-pooling and trip-combining as trips take place between high-density "clusters". If the area of the service region is A, an average *trip request time interval* $\lambda = 1 / (DD*A)$.

Jitneys are one form of DRT that operate along fixed routes. Jitneys are the mainstay of the transit network in many parts of the developing world with high numbers of served customers (*3; 11*). The systems could have millions of daily users and the demand density is well over fifty. Truly flexible DRT systems with many-to-many service have not been implemented with high demand densities (*4; 7; 12-14*). The largest systems are typically offering services for special user groups such as individuals with disabilities in the metropolitan areas. The highest number of passengers is a few million per year so that the demand density remains below or just over one.

A HD-DRT could be achieved by offering an open-for-all service mainly to current private car users. For instance, the land area of the Helsinki metropolitan area is 769 square kilometers. The number of daily private car trips in the area is 1.3 million, and the trip count during a peak-hour is approximately 10% of the daily trip count. The peak-hour demand density of the private car trips in the Helsinki metropolitan area is thus 169 trips per hour per square kilometer. The trip count and demand densities of the daytime traffic are about one-half of the peak-hour values. The demand density of ten trips per hour per square kilometer would be obtained, if 6% of the peak-hour private car trips or 12% of the daytime private car trips shift to HD-DRT.

TEMPORAL AND SPATIAL PARAMETERS

The following temporal and spatial parameters might be specific for each trip. However, in this paper it is not necessary to indicate the trip specificity. Thus, the indexes representing that parameters are referring to the *ith* trip are not shown.

Every trip has an *origin a* and a *destination b*. The *Direct trip length s* is the shortest path from *a* to *b*. The HD-DRT system serves customers from a *pickup point u* to a *delivery point v*. The *Actual trip length s'* is the length of the vehicle route between *u* and *v*. It is possible, but not usual, that the pickup and delivery points of a customer are the same as his or her origin and destination.

The *order time* t_O is the time a trip request is made known. If a customer is picked up from the origin, the *earliest pickup time* t_{EP} of the trip is the same as its order time. Otherwise, the earliest pickup time is set by the walking time between the origin and the pickup point. For purposes of this study, walking times are ignored thus defining that the earliest pickup time is the same as order time for all trips.

The HD-DRT system responds immediately to a trip request by setting a *pickup time* t_P. If the pickup time is delayed from the earliest pickup time, the customer has to adjust his or her departure time from the origin. Thus, we define an *adjustment time AT* to be the time difference between pickup time and earliest pickup time. The nature and weight of adjustment time is not discussed in this paper. The customer arrives at the delivery point at the *delivery time* t_D. *Ride time RT* is the time difference between pickup time and delivery time. In reality, the HD-DRT application will have additional system flexibility for the pickup so that the customer is not always picked up exactly at the declared pickup time. *Waiting time WT* is the time difference between actual pickup time and declared pickup time. This study considers all customers to be picked up at the pickup time and therefore ignores waiting times.

Passenger-pooling and trip-combining mean scheduling and routing new trips so that the decisions made for existing trips and routes are exploited as much as possible. Pooling is possible at

both the pickup points and the delivery points. If a pickup of a customer cannot be pooled to any of existing stops, a new vehicle stop must be scheduled. The same applies to the delivery of a customer. If a trip cannot be combined to the existing routes, a new route must be created for the trip. We use separate studies for analyzing passenger-pooling and trip-combining. For the studies, we define the following system parameters:

WK_{max}, *the maximum walking distance* defines maximum distance between the origin and the pickup point as well as between the destination and the delivery point.

AT_{max}, *the maximum adjustment time* defines the time difference between earliest pickup time and *latest pickup time* t_{LP}

RTI_{max}, *the maximum ride time index* defines the maximum ratio of *maximum ride time* RT_{max} and *direct ride time* RT_{dir}, where maximum ride time is the time difference between pickup time and *latest delivery time* t_{LD}, and direct ride time is the ride time along the shortest path from *a* to *b*.

In a study of passenger-pooling, we define *pooling base trips* to be trips that cannot be pooled at the moment they are ordered. The pooling base trips will be picked up from the origin so that *u=a* and delivered to the destination so that *v=b*. The pickup time of the pooling base trips is delayed from the earliest pickup time for improving passenger-pooling. Thus we define that:

DF, *the delay factor* describes how much the pickup times are delayed from the *ATmax*.

For example, values *DF=0.5* and AT_{max} *=30 min* would mean that for all pooling base trips the pickup time is set 15 minutes after the earliest pickup time.

Possibility for passenger-pooling of a trip is examined so that the trip is picked up at the same time from the same pickup point as the existing pooling base trips. A trip that can be pooled is called a *pooled trip*. In this study, we define that each pooled trip is pooled only to one pooling base trip. If a pooled trip can be pooled to more than one pooling base trips, the one where the walking distance for the pooled trip is the shortest is chosen. *Passenger-pooling potential (PPP)* is defined as the average number of pooled trips that are pooled to the same pickup point with pooling base trips under reasonable assumptions. The value of *PPP* states what the achievable maximum is for passenger-pooling for dispatching strategies. The efficiency of dispatching strategies determines how well the potential is utilized.

In a study of trip-combining, we define *combining base trips* to be trips that cannot be combined at the moment they are ordered. Similarly than with the pooling base trips, the combining base trips will be picked up from the origin, delivered to the destination and picked up at the time defined by the earliest pickup time, AT_{max} and *DF*.

Possibility for trip-combining of a new trip is examined so that such route is created that would serve both the new trip and the combining base trip. A trip that can be combined is called a *combined trip*. In this study, we define that each combined trip is combined to only one combining base trip. If combined trip can be combined to more than one combining base trip, the one with the largest *kilometreage reduction* from the sum of direct trip lengths is chosen. *Trip-combining Potential (TCP)* is defined as the average number of combined trips that are combined with combining base trips under reasonable assumptions.

The temporal and spatial parameters in the before described studies are presented in Figure 1. Even though, the temporal and spatial parameters are used with the above mentioned simplifications, it is reasonable to argue that the *PPP* and *TCP* values represent the potential for passenger-pooling and trip-combining with sufficient detail. The values have to be noted only as indicative, so that detailed studies in the future could concentrate on the right issues based on the results of this paper.

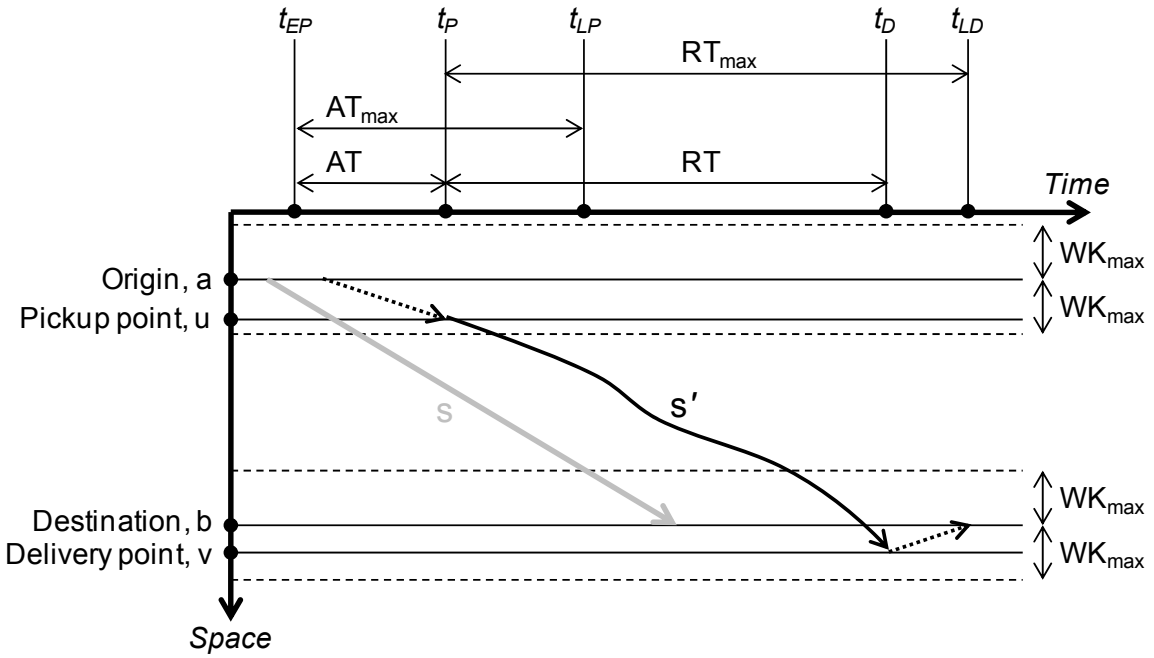


FIGURE 1 Spatial and temporal parameters in the examined HD-DRT system.

METHODOLOGY

The effect of demand density and system parameters was studied using a MATLAB simulation study. The simulations assumed a square service region with sides of 20 km and area of 400 km^2. The area was divided into 2*2 km *grids* numbered consecutively from left to right and down to top. Simulations were run with three demand distributions:

1. *Homogenous demand.*
2. *Peak-hour demand.*
3. *Daytime demand.*

The peak-hour demand and daytime demand were generated based on the unpublished traffic survey data of the Helsinki Metropolitan Area Council (YTV). The 20*20 km square study area was located in the Helsinki metropolitan area (Figure 2). Only those trips that were inside the study area using a private car, van, motorcycle or bus as their main travel mode were included. Peak-hour demand included trips with departure hour 7 (the trips between 7:00 and 8:00). Daytime demand included trips with departure hour between 8 and 16. The sample included 467 morning peak trips and 2,301 daytime trips that fulfilled the criteria. After the sample expansion, they represent a total of 89,133 and 429,598 trips. Origin-Destination (OD) matrix between grids was formed by means of the origin and destination coordinates. Origin grid and destination grid were drawn based on the OD matrix so that the probability for trip from grid *i* to grid *j* was $P_{i,j}=s_{i,j}/\Sigma s_{i,j}$, where $s_{i,j}$ is the trip count in the expanded traffic survey data from grid *i* to *j*. The origin of a generated trip is a random point inside the origin grid and destination a random point inside the destination grid.

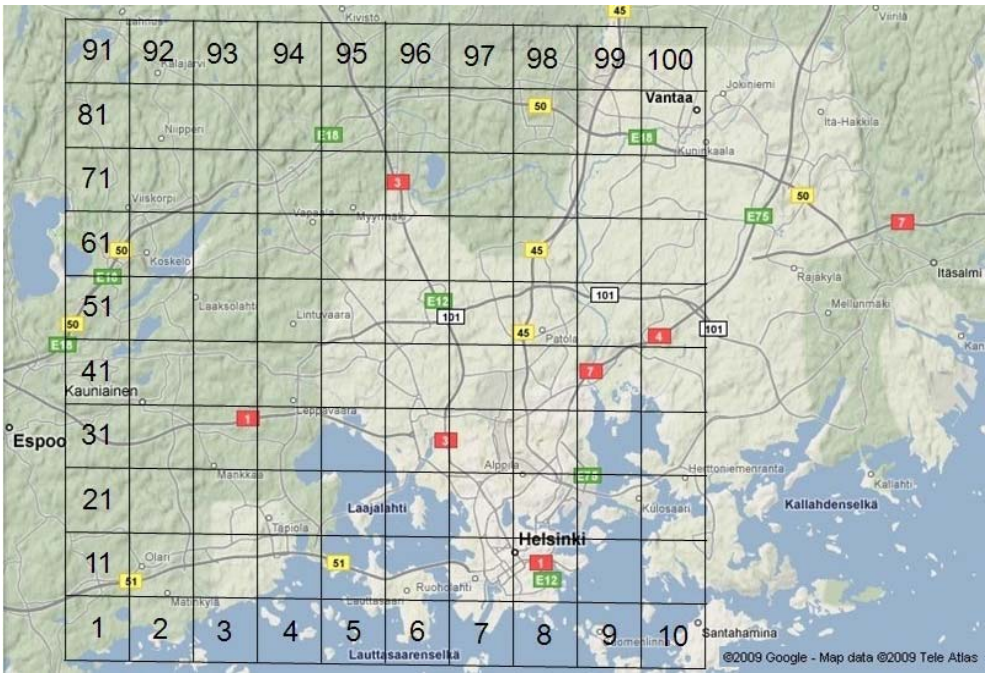


FIGURE 2 Study area in the Helsinki metropolitan area.

In the case of homogeneous demand, the origin was a random point inside the service region. The destination was determined based on the direct trip length distribution of the peak-hour demand simulation using Euclidean distances. Destination point generation was based on the origin and direct trip length. The generation was repeated until the destination was inside the service region.

Trip requests were generated so that $\lambda \sim \text{Exp}(DD*A)$. The trip request generation time became the order time for each trip.

Each simulation included a set of system parameters, which were constant for all trip requests (Table 1). Simulations studying passenger-pooling varied parameters AT_{max}, DF and WK_{max}. Simulations studying trip-combining varied parameters AT_{max}, DF and RTI_{max}. System parameters were varied one at a time so that default parameter values for the others were used. Each set of system parameters were simulated with four levels of demand density, from which three were regarded as high demand density.

TABLE 1 Used System Parameters*

Parameter				
Demand density (DD), no. of trips/(hour*km^2)	5	10	50	100
Maximum adjustment time (AT_{max}), min	10	**15**	20	30
Delay factor (DF)	0.25	**0.50**	0.75	1.00
Maximum walking distance (WK_{max}), km	0.1	**0.3**	0.5	0.8
Maximum ride time index (RTI_{max})	1.25	**1.50**	1.75	2.00

** Default parameter values have been bolded*

The first generated trip request in the passenger-pooling simulations became the first pooling base trip. We define that new trip can be pooled with pooling base trip if three criteria are fulfilled. Firstly, the distance from the origin of new trip to the pickup point of the pooling base trip cannot be more than WK_{max}. Secondly, the earliest pickup time of new trip has to be before the pickup time of the pooling base trip. Thirdly, the time difference between the earliest pickup time of new trip and the pickup time of the pooling base trip cannot be more than AT_{max}. If the new trip cannot be pooled with any of the pooling base trips, it becomes a new pooling base trip.

The first generated trip request in the trip-combining simulations became the first combining base trip. We define that new trip can be combined with combining base trip if a route can be created so that four criteria are fulfilled. Firstly, the pickup time for both trips must be before the delivery time. Secondly, the length of the route has to be less than the sum of direct trip lengths. In other words, the kilometreage reduction has to be larger than zero. Thirdly, the time difference between the earliest pickup time and pickup time of new trip cannot be more than AT_{max}. Fourthly, the ratio of ride time and direct ride time of the trips cannot be more than RTI_{max}. If the new trip cannot be combined with any of the combining base trips, it becomes a new combining base trip.

In addition to passenger-pooling potential and trip-combining potential, some average values were reported. Average walking distance between the origin and the pickup point and average adjustment time were reported from pooled trips. Average adjustment time, average ride time index and average kilometreage reduction of routes were reported from combined trips. These are not directly applicable for estimating the quality of service or efficiency of HD-DRT as the simulations didn't model the actual HD-DRT transport but the passenger-pooling potential and trip-combining potential within it.

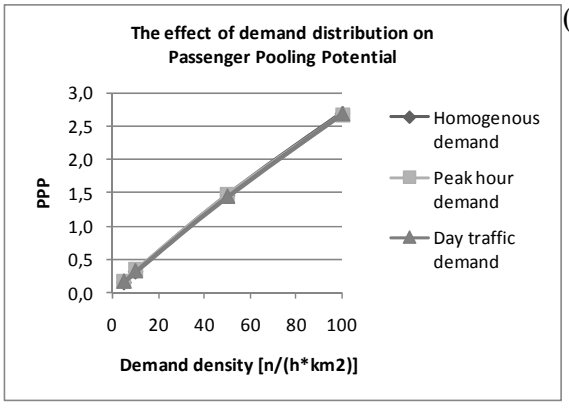
The simulation system was tested to determine needed warm-up period and reporting interval. Tests indicated that the model needs at least $DF * AT_{max}$ for warming up and AT_{max} to empty. Therefore, a half-hour warm-up period was set at the beginning of the simulation time and a half-hour period at the end of simulation when no results were reported.

Trip generation includes randomness that has an effect on the results. The longer the simulation time, the less randomness had an effect on the results. The effect of randomness was studied with test simulations varying simulation times. Test simulations used default parameter values, DD=10, and the above-mentioned warm-up and reporting periods. The results of test simulations indicated that the averages and standard deviations of *PPP* and *TCP* were converged as the simulation time was increased. Simulations with a 6-hour and longer simulation times had results where no significant variation based on the simulation time was observed. A 7-hour simulation time was used on the basis of the test simulations.

RESULTS

We presumed that the inhomogeneous demand distribution would lead to larger values of *PPP* and *TCP* than the homogeneous distribution. However, simulations do not support this presumption. Demand distributions appeared to have an insignificant effect on *PPP* and *TCP* as shown in Figure 3, even though the significance was not tested with statistical methods.

(a)



(b)

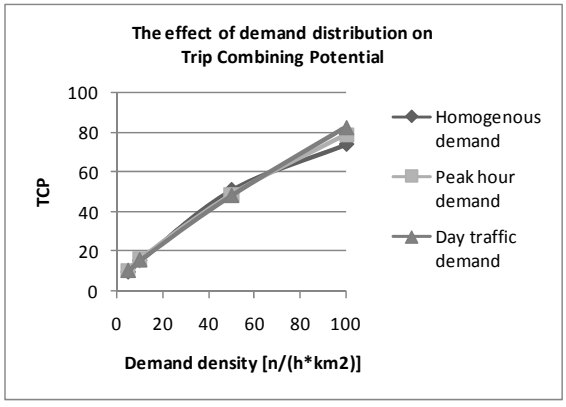


FIGURE 3 The effect of demand distribution to (a) Passenger-pooling Potential and (b) Trip-combining Potential with default parameter values.

One explanation for the insignificant difference between the demand distributions is that origins and destinations in inhomogeneous demand distributions were drawn so that origins and destinations within one grid were homogeneously distributed. This did not bring out the "trip clusters". For example, a large shopping market inside a grid would gather a large portion of origins and destinations to a small area inside the grid. The traffic survey data we used was not detailed enough to enable simulations in address detail. Based on this, only results with homogenous demands are presented in this paper.

Simulation results indicate a positive linear correlation existing between demand density and *PPP*. With default parameter values, the dependence is $PPP = 0.0274 * DD$ ($R^2 = 0.9964$). This means that *PPP* value over 1 must have DD=36.5 and *PPP* value over 5 results in a *DD=182* (Figure 4). The results also indicate a correlation existing between *PPP* and AT_{max}, between *PPP* and *DF* as well as between *PPP* and square of WK_{max}.

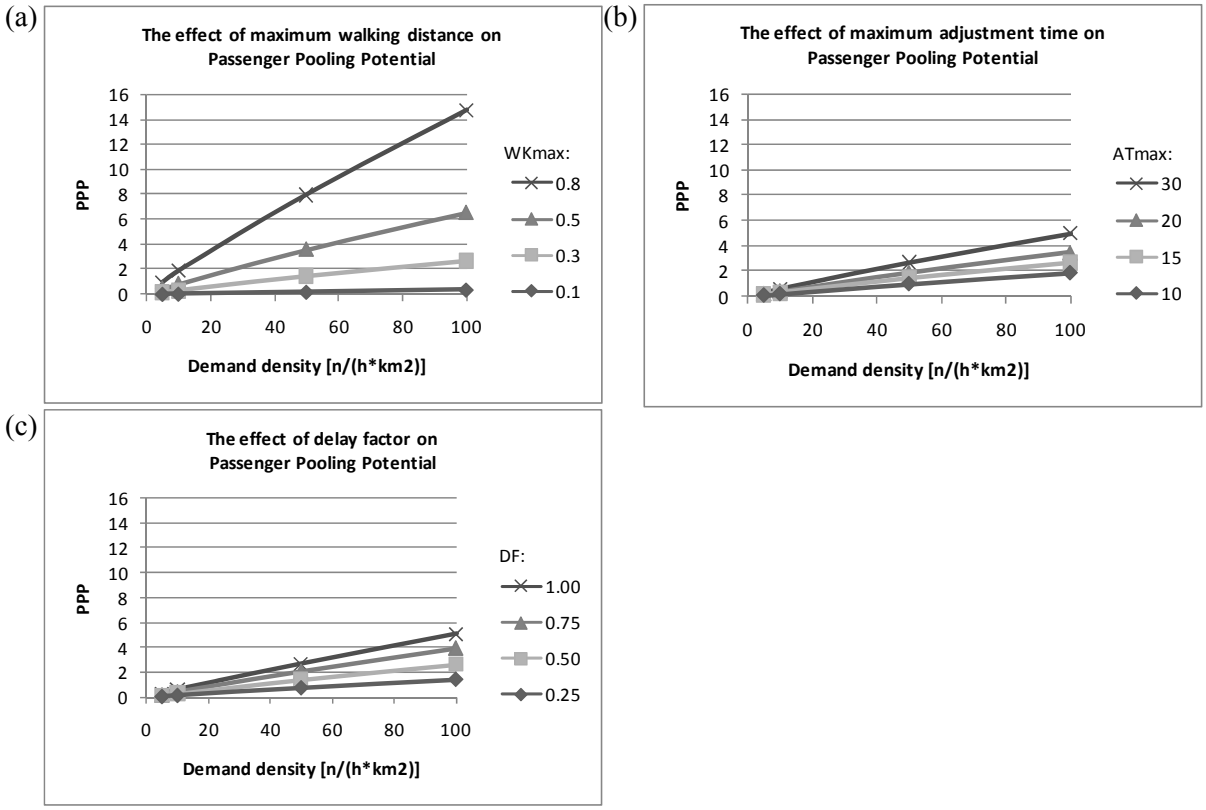


FIGURE 4 Passenger-pooling Potential with various values of (a) maximum walking distance, (b) maximum adjustment time and (c) delay factor on homogenous demand distribution.

The increase of WK_{max} had evident effect on average walking distances (Table 2). Average walking distances were 55% to 66% of the maximum walking distances. The *DF* and AT_{max} had no effect on average walking distances on the results. The effect of demand density on average walking distance was low.

Demand density and WK_{max} had no effect on average adjustment time. However, the effects of *DF* and AT_{max} were evident. The adjustment times with *DF=1.00* were over three times as large as with *DF=0.25*. Similarly, the adjustment times with AT_{max} *=30 min* were about two times as large as with AT_{max} *=10 min*. Average adjustment times with *DF=0.5* were 24% to 26% of the maximum AT_{max} values.

TABLE 2 The Effect of Maximum Walking Distance, Delay Factor, Maximum Adjustment Time and Demand Density on Average Walking Distance and Average Adjustment Time

Average walking distance (km)	Max. walking distance (km)				Delay factor				Max. adjustment time (min)			
Demand density (n/km^2/h)	0.1	0.3	0.5	0.8	0.25	0.50	0.75	1.00	10	15	20	30
5	0.07	0.19	0.32	0.50	0.19	0.19	0.20	0.20	0.19	0.19	0.20	0.20
10	0.07	0.20	0.32	0.49	0.20	0.20	0.19	0.19	0.20	0.20	0.19	0.19
50	0.07	0.19	0.30	0.45	0.19	0.19	0.19	0.18	0.19	0.19	0.19	0.18
100	0.07	0.18	0.29	0.44	0.19	0.18	0.18	0.18	0.19	0.18	0.18	0.18
Average adjustment time (min)	Max. walking distance (km)				Delay factor				Max. adjustment time (min)			
Demand density (n/km^2/h)	0.1	0.3	0.5	0.8	0.25	0.50	0.75	1.00	10	15	20	30
5	3.8	3.7	3.7	3.8	1.8	3.7	5.8	7.7	2.4	3.7	4.9	7.6
10	3.9	3.8	3.8	3.8	1.9	3.8	5.7	7.6	2.5	3.8	5.0	7.6
50	3.8	3.8	3.8	3.8	1.9	3.8	5.7	7.6	2.5	3.8	5.0	7.6
100	3.8	3.8	3.8	3.8	1.9	3.8	5.7	7.6	2.5	3.8	5.0	7.6

Simulation results indicate that the demand density has evident impact on *TCP*. With default parameter values, the *TCP* is just below ten with *DD=5* but more than fifty with *DD=50* (Figure 5). Each simulation parameter had an effect on *TCP*, even though the effect of *DF* was not as significant as the effect of RTI_{max} and AT_{max}. The smaller the value of RTI_{max} was the bigger effect its increase had on *TCP*. Enlargement of AT_{max} increased the value of *TCP* evenly. The values of *TCP* with AT_{max} *=30 min* were 50% to 88% larger than the values with AT_{max} *=10 min*. Simulations with *DF=1.00* gave 11% to 35% larger *TCP* values than simulations with *DF=0.25*.

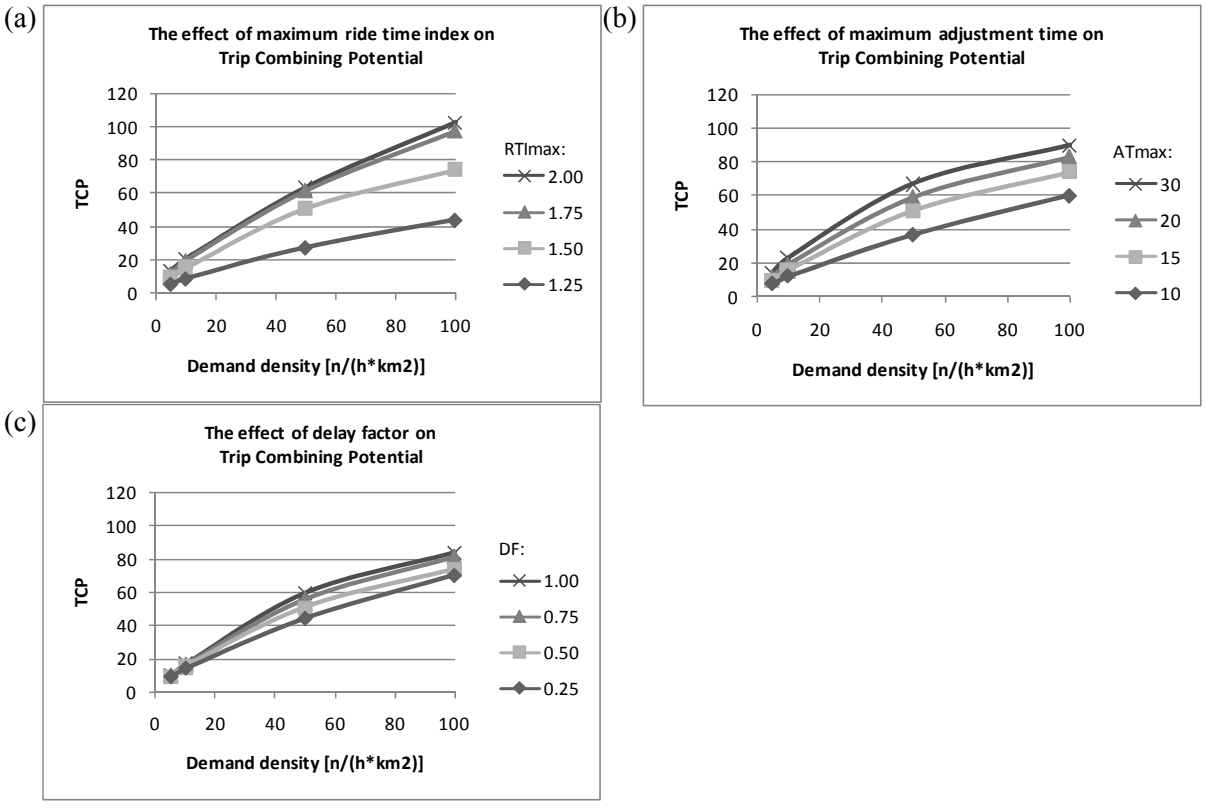


FIGURE 5 Trip-combining Potential with various values of (a) maximum ride time index, (b) maximum adjustment time and (c) delay factor on homogenous demand distribution.

Demand density, *DF* and AT_{max} had no effect on average ride time index (Table 3). However, the effect of the RTI_{max} was evident.

The increase of demand density decreased the average adjustment times. Average adjustment times with *DD=100* were 11% to 23% smaller than with *DD=5*. All system parameters had some effect on average adjustment times. The AT_{max} had the most evident effect. Average adjustment times were 22% to 43% of the AT_{max}.

The average kilometreage reduction was decreased as the demand density was increased. This is explained by the aspect that when the demand density was increased, the probability of finding a suitable combining base trip for the longer trip requests was increased more than the probability for shorter trip requests. Thus, the share of short trips among combining base trips was increased and the average length of the combining base trips was decreased as the demand density increased. RTI_{max} had the largest effect of the simulation parameters on average kilometreage reduction. The reduction with RTI_{max} *=2.00* was 29% to 32% smaller than with RTI_{max} *=1.25*. AT_{max} and *DF* had no significant effect on average kilometreage reduction until the demand density was high and the parameters had loose values. With loose parameter values, the criteria enabled trips to be combined with a combining base trip even though the kilometreage reduction was small. With tighter parameter values, the combining was not possible and the trip became a new combining base trip. Therefore, with tight parameter values, the combined trips more appropriately generate larger average kilometreage reductions, on average.

TABLE 3 The Effect of Maximum Ride time index, Delay Factor, Maximum Adjustment Time and Demand Density on Average Ride time index, Average Adjustment Time and Average Kilometreage Reduction

Average ride time index	Max. ride time index				Delay factor				Max. adjustment time (min)			
Demand density (n/km^2/h)	1.25	1.50	1.75	2.00	0.25	0.50	0.75	1.00	10	15	20	30
5	1.16	1.31	1.43	1.54	1.31	1.31	1.31	1.32	1.31	1.31	1.31	1.31
10	1.16	1.31	1.43	1.55	1.31	1.31	1.31	1.32	1.31	1.31	1.31	1.31
50	1.16	1.31	1.43	1.55	1.31	1.31	1.31	1.32	1.31	1.31	1.31	1.31
100	1.16	1.31	1.43	1.55	1.31	1.31	1.31	1.32	1.31	1.31	1.31	1.32
Average adjustment time (min)	Max. ride time index				Delay factor				Max. adjustment time (min)			
Demand density (n/km^2/h)	1.25	1.50	1.75	2.00	0.25	0.50	0.75	1.00	10	15	20	30
5	5.3	5.5	5.6	5.7	4.8	5.5	6.1	6.4	4.1	5.5	6.7	8.5
10	5.0	5.3	5.5	5.5	4.6	5.3	5.9	6.3	3.9	5.3	6.4	8.2
50	4.5	4.8	5.1	5.2	4.2	4.8	5.4	6.0	3.6	4.8	5.8	7.4
100	4.3	4.6	4.9	4.9	4.1	4.6	5.3	5.7	3.5	4.6	5.5	6.6
Average kilometreage reduction (km)	Max. ride time index				Delay factor				Max. adjustment time (min)			
Demand density (n/km^2/h)	1.25	1.50	1.75	2.00	0.25	0.50	0.75	1.00	10	15	20	30
5	5.28	4.37	3.92	3.57	4.34	4.37	4.39	4.32	4.43	4.37	4.37	4.26
10	5.06	4.45	3.98	3.49	4.31	4.45	4.44	4.17	4.50	4.45	4.39	4.34
50	4.80	4.20	3.73	3.39	3.97	4.20	4.04	3.79	4.12	4.20	4.28	4.05
100	4.64	3.88	3.62	3.19	3.87	3.88	3.95	2.98	4.06	3.88	3.82	3.16

DISCUSSION AND CONCLUSIONS

Although, the used model for determining *PPP* and *TCP* contains some simplifications, it is nevertheless sufficient to say that *PPP* and *TCP* represent the potential for passenger-pooling and trip-combining. Previous DRT related studies have generally concentrated on some particular control or routing algorithms. The model used in this study is more general but anyhow connects the main temporal and spatial parameters to the functionality of HD-DRT systems.

The results indicate that both *PPP* and *TCP* have a strong positive correlation with demand density. The question is, how well the potentials for passenger-pooling and trip-combining can benefit HD-DRT systems? Due to their highly approximative nature, the significance of *PPP* and *TCP* is only indicative. Nonetheless, the results yield new information on the significance of passenger-pooling and trip-combining in HD-DRT. The results argue for further research of HD-DRT, even though DRT systems have conventionally been seen applicable mainly for low demand density situations.

PPP does not pay attention to the direction of trip requests. Counting only trip requests going in the same direction might have potentially resulted in at least four times smaller PPP values. The *PPP* treats only passengers boarding the vehicle at the pickup point. The other side of passenger-pooling at delivery points has left outside the study. The pooling at delivery points might have potentially doubled the *PPP* values. High *PPP* values give hope to large-scale passenger-pooling. Simulation results indicate that no distinct potential for passenger-pooling emerges until high demand density. With low demand densities the large-scale passenger-pooling is possible only with large walking distances.

The criteria for viable routes in *TCP* do not take into consideration route decisions made for other combined trips. Thus, the actual possibilities for route modifications are more limited than when calculating the *TCP* values. Small *TCP* values would most likely mean that trip-combining happens rarely and that the empty kilometreage is high. High *TCP* values enable trip-combining in "tighter packages" so that the average occupancy increases while the quality of service remains high. The simulation results indicate that with demand densities between five and ten the distinct potential for trip-combining is emerging. With low demand densities, no distinct potential exists. When the demand density is fifty, the potential for trip-combining is very high, even with high quality of service.

In this paper, we do not discuss what system parameters values are acceptable to user. The valuations of parameters such as walking distance and waiting time vary largely according to user and trip type. It is a task of the operator of the HD-DRT system to address the trade-off between the offered quality of service and system efficiency.

The presumption was that the inhomogeneous demand distribution would give higher *PPP* and *TCP* values than the homogeneous demand distribution as the origins and the destinations are more concentrated. The used methodology proved to be unsuitable for bringing out the difference between homogeneous and inhomogeneous demand distributions. Thus, only results with homogenous demands are presented in this paper.

The delay factor was introduced to increase *PPP* and *TCP* in dynamic DRT operation with short adjustment times. The effect of the delay factor was more evident on *PPP* than on *TCP*. This can be explained by the possibility of combining trips with the combining base trip even though the combining base trip has already been picked up. If and when the aim of the HD-DRT is to obtain high average occupancy levels, the share of pooling base trips and combining base trips approximates zero. Thus, the importance of delay factor decreases as the demand density increases.

The results of this paper argue for further research of HD-DRT systems. New concepts and dispatching strategies combined with state-of-the-art technology could utilize a large portion of the potential presented in this study.

ACKNOWLEDGEMENTS

This research is part of the Metropol-project, which is partly funded by the Finnish Funding Agency for Technology and Innovation, the Ministry of Transport and Communications, Helsinki Metropolitan Area Council and Helsinki City Transport.

REFERENCES

1. Wardman, M. Public transport values of time. *Transport Policy,* Vol. 11, Issue 4, 2004, pp. 363-377.

2. Wardman, M. A review of British evidence on time and service quality valuations. *Transportation Research Part E*, Vol. 37, Issues 2-3, 2001, pp. 107-128.

3. Black, A. *Urban mass transportation planning*. McGraw-Hill, New York, 1995.

4. Enoch, M., S. Potter, G. Parkhurst and M. Smith. *Intermode: Innovations in Demand Responsive Transport*. Intermode Final Report. Department for Transport and Greater Manchester Passenger Transport Executive, Manchester, 2004.

5. Finn, B. *Analysis of User Needs for Demand Responsive Transport Services*. Deliverable no. D3, SAMPO project - TR1046, European Commission, 1996.

6. Bakker, P. Large scale demand responsive transit systems - A local suburban transport solution for the next millennium. Presented at European Transport Conference, Cambridge, UK, 1999.

7. Round, A. and R. Cervero. *Future ride: Adapting new technologies to paratransit in the United States*. Working Paper 306. University of California Transportation Center, UC Berkeley, CA, 1996.

8. Fu L., J. Yang and J. Casello. Cortés C.E., and R. Jayakrishnan. Quantifying Technical Efficiency of Paratransit Systems by Data Envelopment Analysis Method. In *Transportation Research Record: Journal of the Transportation Research Board*, No. 2034, Transportation Research Board of the National Academies, Washington, D.C., 2007, pp. 115–122.

9. Cortés C.E., and R. Jayakrishnan. Design and Operational Concepts of High-Coverage Point-to-Point Transit System. In *Transportation Research Record: Journal of the Transportation Research Board*, No. 1783, Transportation Research Board of the National Academies, Washington, D.C., 2002, pp. 178–187.

10. Lave, R. E., R. Teal and P. Piras. *A Handbook for Acquiring Demand-Responsive Transit Software*. TCRP Report 18. Transportation Research Board, 1996.

11. Cervero R. *The Transit Metropolis*. Island Press, Washington, DC, 1998.

12. Quadrifoglio L, Dessouky MM, Ordóñez F. A simulation study of demand responsive transit system design. *Transportation Research Part A*, Vol. 42, Issue 4, 2008, pp. 718-737.

13. Westerlund Y., A. Ståhl, J. Nelson and J. Mageean. Transport Telematics for elderly users: Successful Use of Automated Booking and Call-back for Demand Responsive Transport Services in Gothenburg. Presented at 7th ITS World Congress, Turin, Italy, 2000.

14. Palmer K., M. Dessouky, T. Abdelmaguid. Impacts of management practices and advanced technologies on demand responsive transit systems. *Transportation Research Part A*, Vol. 38, Issue 7, 2004, pp. 495-509.

Publication VI

Lauri Häme, Harri Hakula. Dynamic journeying under uncertainty. Submitted to *Under review for publication in European Journal of Operational Research*, 20.12.2011 .

Dynamic Journeying under Uncertainty

Lauri Häme*,[1]

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Harri Hakula

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Abstract

We introduce a journey planning problem in multimodal transportation networks under uncertainty. The goal is to find a journey, possibly involving transfers between different transport services, from a given origin to a given destination within a specified time horizon. Due to uncertainty in travel times, the arrival times of transport services at public transport stops are modeled as random variables. If a transfer between two services is rendered unsuccessful, the commuter has to reconsider the remaining path to the destination. The problem is modeled as a Markov decision process in which states are defined as paths in the transport network. The main contribution is a backward induction algorithm that generates an optimal policy for traversing the public transport network in terms of the probability of reaching the destination in time. By assuming history independence and unconditionality of successful transfers between services we obtain approximate methods for the same problem. Analysis and numerical experiments suggest that while solving the path dependent model requires the enumeration of all paths from the origin to the destination, the proposed approximations may be useful for practical purposes due to their computational simplicity. In addition to on-time arrival probability, we show how travel and overdue costs can be taken into account, making the model applicable to freight transportation problems.

Key words: Stochastic processes, Transportation, Stochastic shortest path problem, Itinerary planning problem, Markov decision processes

1. Introduction

The urban itinerary planning problem involves determining a path, possibly involving transfers between different transport modes, from a specified origin to a similarly specified destination in a transport network. Common criteria used for evaluating itineraries include the total duration, number of transfers and cost (Androutsopoulos and Zografos, 2009).

We study a new objective for journey planning in scheduled public transport networks, motivated by uncertainty in travel times of transport services (buses, trams, trains, ferries, ...). Our

*Corresponding author

Email addresses: Lauri.Hame@tkk.fi (Lauri Häme), Harri.Hakula@tkk.fi (Harri Hakula)

[1]Tel.: +358 40 576 3585, Fax: +358 9 470 23016

goal is to maximize the reliability[2] of an urban journey. In contrast to existing itinerary planning algorithms designed for scheduled public transport networks, where the path is a priori optimized with respect to an objective, for example, (Androutsopoulos and Zografos, 2009), we take into consideration the fact that the realized journey may differ from the original plan.

Real-time information on the status of transport services is available via mobile devices with location-based capabilities. This makes it possible for a commuter to dynamically modify the planned journey in case of a delay or cancellation. For example, if a transfer from a transport service to another is unsuccessful due to a delay, the commuter may reconsider the remaining path to the destination. Our approach is to design the journey in a way that the probability of reaching the destination in time is maximized, even if some transfers are rendered unsuccessful in the course of time. Clearly, the importance of reliability is emphasized when the number of transfers between different transport services is increased.

Taking into account the uncertainty in travel times is particularly important in difficult weather conditions when delays are common. In addition to traditional public transport with fixed schedules, uncertainty in travel times should be given special attention in flexible transport services without fixed routes (Mulley and Nelson, 2009). If the vehicle routes are modified in real time, the estimation of travel times between subsequent stops is more difficult than in the case of fixed routes.

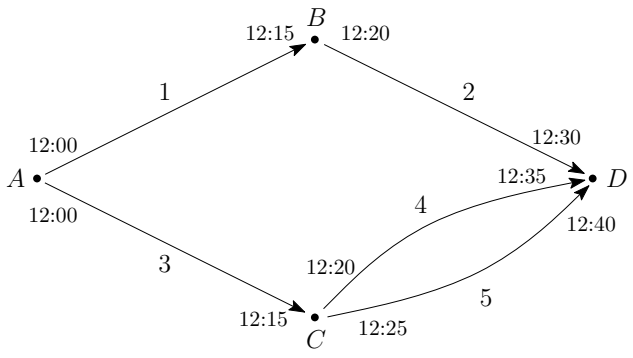


Figure 1: The difference between stochastic and deterministic journey planning for a commuter traveling from A to D. The four points represent public transport stops (A, B, C, D) and the arrows between them represent public transport services $(1, \ldots, 5)$. Initially, there are three possible journeys from A to D: $(1, 2), (3, 4)$ and $(3, 5)$. If the commuter initially chooses service 1, the success of the journey is dependent of the success of the transfer from 1 to 2 at stop B. If the commuter chooses service 3 first, the destination is reached if one of the transfers $3 \rightarrow 4$ or $3 \rightarrow 5$ is successful at stop C.

A simplified example clarifying the main difference between stochastic and deterministic journey planning is shown in Figure 1. The four nodes represent public transport stops (A, B, C, D) and the arrows between them represent scheduled public transport services $(1, \ldots, 5)$ operating

[2]In the scheduling of an activity of random duration in general and in traveling under congested conditions in particular, the value of reliability is seen to be significant (Fosgerau and Karlström, 2010). Brownstone and Small (2005); Small et al. (2005) argue that there is substantial heterogeneity in the valuation of reliability among motorists. In an empirical study on commuter behavior in California, where commuters chose between a free and a variably tolled route, the model in (Lam and Small, 2001) suggested that the average value of reliability is \$15.12 per hour for men and \$31.91 for women (in 1998 US dollars).

within a specific time horizon. Each transport service has a specific schedule determined by the scheduled departure and arrival times shown next to the arrows.

(i) Let us first consider the deterministic case where the *realized* departure and arrival times of services are assumed to be equal to the *scheduled* departure and arrival times. For a commuter traveling from A to D, there are three feasible journeys: $(1, 2), (3, 4)$ and $(3, 5)$. In order to reach D as fast as possible, the commuter should follow the path $(1, 2)$.

(ii) In the stochastic case, the realized departure and arrival times of services are not necessarily equal to the scheduled times. A transfer from a service to another may fail due to a delay, even if the transfer was feasible according to the deterministic schedule. For example, if the commuter initially chooses service 1, the success of the journey from A to D is dependent of the success of the transfer from 1 to 2 at stop B. Assuming that services 1 and 3 are equally likely to be delayed, it might be reasonable to initially choose service 3: It is more probable that one of the transfers $3 \rightarrow 4$ and $3 \rightarrow 5$ is successful than that $1 \rightarrow 2$ is successful.

More generally, in the model presented in this paper, a commuter wishes to travel from an origin node v_o to a destination node v_d within a time horizon $[0, T]$ using different transport services. Each transport service is represented as a sequence of *legs*. Each leg is associated with a *start node* and *end node*, as well as a random *start time* and *end time*. Adjacent nodes in the network are connected with similarly defined walking legs.

A path from the origin to the destination is represented as a sequence of legs, in which the start node of each leg is equal to the end node of the previous leg. We assume that during the execution of a leg, the commuter receives information on which services have already visited the end node and which are yet to arrive. In other words, the customer "sees" the available successor legs of the current leg and may choose to (i) stay in the vehicle, (ii) transfer to another vehicle or (iii) get off the vehicle and start walking towards a nearby stop (or the destination). Our approach is to define an optimal policy specifying the actions that are executed in different situations in order to maximize the probability of reaching the destination before T.

In summary, the problem can be characterized as a dynamic and stochastic path finding problem. Such problems are often modeled as *Markov decision processes* (Psaraftis and Tsitsiklis, 1993; Polychronopoulos and Tsitsiklis, 1996), in which the *actions* of a decision maker at a given *state* are independent of all previous actions and states. We first present a conditional Markov model, in which the path history is included in each state by defining states as sequences of legs in the transport network. That is, the current state is determined by the path taken so far. This model is further approximated by means of history independent models, in which the current state is defined as the current leg.

This work is partially motivated by a demand-responsive transport (DRT) service currently being planned to operate in Helsinki. Helsinki Region Transport board has approved a plan under which the trial period of the service takes place from 2012 to 2014. Similarly as the current flexible service routes (Helsinki Region Traffic, 2010a), the new DRT service is designed to operate on a demand-responsive basis, that is, vehicle routes are modified according to the demand situation. The main difference to existing services is that no pre-order times for trips are required and the trips can be booked "on the fly" by means of an interactive user interface. This type of new service calls for a journey planner that is capable of communicating with flexible services as well as traditional public transport, thus combining the benefits of both transport modes.

In addition to public transport, a similar journey planning problem arises in freight transporta-

tion by for-hire carriers. For this purpose, we show how travel and overdue costs can be incorporated in the model.

The remainder of this document is organized as follows: The journey planning problem under uncertainty is formalized in Sections 2, 3 and an algorithm that generates an optimal policy for the conditional Markov decision process is presented in Section 4. In Section 5, we approximate the conditional solution by assuming history independence and compare the solutions by analysis. The solution methods are evaluated by numerical experiments in Section 6.

1.1. Related work

There is a vast literature devoted to the deterministic path-finding problem in a transit network. Zografos and Androutsopoulos (2008) classify the approaches into the following types of formulations: 1) the headway-based model, in which a constant headway for each transit line is assumed (Wong and Tong, 1998) and 2) the schedule-based model, which assumes a fixed route and timetable for each transit line.

Our approach stems from model 2, for which most existing solution approaches are based on label correcting, label setting or branch-and bound, see for example (Zografos and Androutsopoulos, 2008; Peng and Huang, 2000; Modesti and Siomachen, 1998; Huang and Peng, 2001, 2002; Horn, 2003; Tong and Richardson, 1984; Tong and Wong, 1999; Ziliaskopoulos and Wardell, 2000; Ziliaskopoulos and Mahmassani, 1993; Brub et al., 2006; Cooke and Halsey, 1966; Cai et al., 1997; Chabini, 1998; Kostreva and Wiecek, 1993; Hamacher et al., 2006; Androutsopoulos and Zografos, 2009). Heuristic solutions, that are useful when the fast solution of the problem is essential, are presented in (Bander and White, 1991) and (Tan et al., 2007).

In addition to the above-mentioned itinerary planning models, most of which are deterministic, our approach is closely related to the stochastic shortest path problem (SSPP). There are many different versions of the problem considered in the literature, each with a different meaning for the optimal path (Murthy and Sarkar, 1997). Early studies related to the problem defined the optimal path to be the one that maximizes the decision maker's expected utility. This objective is motivated by the Von Neumann-Morgenstern approach of preference judgments under uncertainty (Loui, 1983). Bard and Bennett (1991) present heuristic methods involving Monte-Carlo simulation to solve the SSPP with a general non-increasing utility function. An exact algorithm for the SSPP with a quadratic utility function is presented in (Mirchandani and Soroush, 1985).

Recent studied objectives for the stochastic shortest path problem include (i) the maximization of the probability that the length of the path does not exceed a threshold value or finding the path that maximizes the probability of arriving on time (Fan et al., 2005a; Nikolova et al., 2006b; Nie and Wu, 2009), (ii) finding the path with the greatest probability of being the shortest (Sigal et al., 1980; Kamburowski, 1985) and (iii) the minimization of expected cost, distance or travel time (Jaillet, 1992; Waller and Ziliaskopoulos, 2002; Fan et al., 2005b; Nikolova et al., 2006a; Thomas and White, 2007; Peer and Sharma, 2007).

Most studies related to the stochastic shortest path problem are considered *static*, since they call for the a priori selection of a fixed path optimizing an appropriate objective. In contrast, solutions to *dynamic and stochastic* shortest path problems[3] can be characterized as policies specifying the

[3]The dynamic and stochastic shortest path problem is also referred to as the *stochastic shortest path problem with recourse* (Waller and Ziliaskopoulos, 2002).

appropriate actions for each particular real-time scenario (Psaraftis and Tsitsiklis, 1993). The first version of the dynamic problem, in which the travel-time distributions of arcs are known and time-dependent, was presented in (Hall, 1986). A heuristic search algorithm for a similar problem is presented in (Bander and White III, 2002). The model is extended in (Fu and Rillet, 1998) to the case of continuous-time random arc costs. Miller-Hooks and Mahmassani (2000) provide an algorithm for finding the least expected cost path in a discrete-time dynamic problem.

Polychronopoulos and Tsitsiklis (1996) consider a dynamic shortest path problem in which arc costs are randomly distributed, and the arc costs are realized once the vehicle arrives at the arc. Iterative and adaptive algorithms for a similar problem are presented in (Cheung, 1998; Cheung and Muralidharan, 2000). Psaraftis and Tsitsiklis (1993) study a variation of the dynamic problem in which the distributions on arc travel times evolve over time according to a Markov process and the changes in the status of an arc are not observed until the vehicle reaches the arc. A genetic algorithm for dynamic re-routing in a similar problem setting is presented in (Davies and Lingras, 2003). Kim et al. (2005a) extend the model to include real-time information and present results regarding optimal departure times from a depot and optimal routing policies. A state-space reduction technique that significantly improves computation time for the problem is presented in (Kim et al., 2005b). Azaron and Kianfar (2003) extend the analytical results presented in (Psaraftis and Tsitsiklis, 1993) for the case where the states of the current arc and immediately adjacent arcs are known. Ferris and Ruszczynski (2000) model a problem in which arcs can fail and become unusable as an infinite-horizon Markov decision process and give an example of the behavior of an optimal policy. Again, using a Markov decision process, Thomas and White (2007) consider the problem of constructing a minimum expected total cost route from an origin to a destination that anticipates and then responds to changes in congestion.

Datar and Ranade (2000) examine a public transport model in which bus arrival times at each stop are exponentially and independently distributed and present an algorithm to generate travel plans for such conditions. Boyan and Mitzenmacher (2001) extend this approach by characterizing an optimal policy for traversing a bus network in which bus-arrival times are distributed according to a probability distribution satisfying the increasing failure rate property.

In addition to the shortest path problem, stochastic travel times have been incorporated in studies related to the traveling salesman problem (Kao, 1978) and its generalization, the vehicle routing problem. Kenyon and Morton (2003) study a vehicle routing problem with multiple vehicles and stochastic travel times, where the objective is to maximize the probability that the duration of each vehicle tour is less than a specified maximum time. Jula et al. (2006) consider a traveling salesman problem with time windows and stochastic travel times, in which the goal is to find the route with minimum expected cost. Problems with up to 80 customers are solved by means of the proposed dynamic programming solution. Russell and Urban (2007) present a similar problem with multiple vehicles in which an additional costs are incurred if the time windows are violated. By using a tabu search algorithm, problems involving 100 customers are solved.

In this work we examine a problem similar to the ones studied in (Datar and Ranade, 2000; Boyan and Mitzenmacher, 2001). The major difference is that we consider a more detailed version of the problem by assuming that each public transport service has a specific schedule, but the arrival times of the services at stops are defined as random variables. In addition, we consider the possibility of walking between adjacent stops, as in (Zografos and Androutsopoulos, 2008).

2. Model

Let $\mathcal{V}$ denote a set of *nodes* representing public transport stops in a specific area and let $\mathcal{K} \subset \mathbb{N}$ denote a set of public transport *services* operating in this area, indexed by natural numbers.

Each service $k \in \mathcal{K}$ follows a *route*, represented as a sequence of nodes $(v_1^k, \ldots, v_m^k)$ in $\mathcal{V}$. Note that it is possible for a service to visit the same node more than once. For example, if node v_i^k is included in the route twice, there exists an index $j \neq i$ for which $v_i^k = v_j^k$. Each service departs at node v_1^k at a specific time and proceeds to nodes $v_2^k, \ldots, v_m^k$ in the order determined by the route. Due to uncertainty in departure and travel times, we define the arrival times of services at nodes as real-valued random variables. Letting τ_j^k denote the *random arrival time* of service k at node v_j^k, a service k can be represented as a sequence of deterministic nodes and random arrival times $((v_1^k, \tau_1^k), \ldots, (v_m^k, \tau_m^k))$, see Figure 2. Any two arrival times τ_j^k and τ_l^h of distinct services k and h are assumed to be independent, whereas the arrival times τ_j^k and τ_l^k of a single service k at stops v_j^k and v_l^k are not necessarily independent (see Section 5.1). In this section, we examine the arrival times as arbitrarily distributed real random variables. In the numerical experiments discussed in Section 6, the arrival times are defined as gamma distributed random variables, similarly as in (Russell and Urban, 2007).

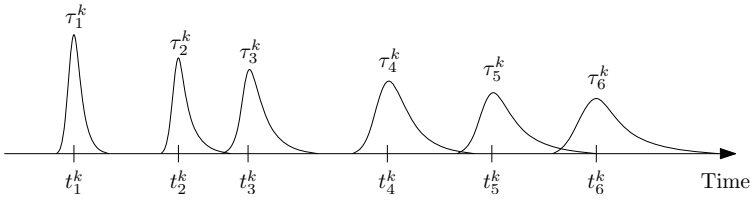


Figure 2: The schedule of a transport service. The real numbers t_j^k represent the scheduled arrival times of a transport service at six subsequent stops. The curves represent the distributions (gamma distributions in this example) of random variables τ_j^k that are used to model the actual arrival times of the service at stops v_j^k, $j \in \{1, \ldots, 6\}$.

2.1. Service legs

Each service $k \in \mathcal{K}$ can be decomposed as a set of scheduled *legs* between subsequent stops. That is, each leg has a start node, end node, start time and end time. By this decomposition, we can represent any path in the transport network as a sequence of legs indexed by natural numbers. For example, consider a transport network consisting of ten legs numbered from 1 to 10. Then, the sequences $(5, 2, 7)$ and $(5, 8, 3, 7)$ define two paths in the network involving two and three *transfers*, respectively. Each transfer between two legs has a specific probability of success. Transfers between subsequent legs of the same service are assumed to be successful with probability 1.

Formally, letting $\mathcal{L} \subset \mathbb{N}$ denote the set of indices of legs, a leg corresponding to index $i \in \mathcal{L}$ is defined by a pair of node-arrival time pairs and a service number $((v_i, \tau_i), (v_i', \tau_i'), k_i)$, where v_i is the *start node*, v_i' is the *end node*, τ_i is the *start time*, τ_i' is the *end time* and the *service number* k_i is the index of the service that executes leg i.

By using this definition, a service $((v_1^k, \tau_1^k), \ldots, (v_m^k, \tau_m^k))$ is represented by a set of legs $\{i_1^k, \ldots, i_{m-1}^k\} \subset \mathcal{L}$, where leg i_h^k is determined by

$$\left((v_{i_h^k}, \tau_{i_h^k}), (v'_{i_h^k}, \tau'_{i_h^k}), k_{i_h^k}\right) = \left((v_h^k, \tau_h^k), (v_{h+1}^k, \tau_{h+1}^k), k\right) \tag{1}$$

for $h \in \{1, \ldots, m-1\}$. Note that the end time $\tau'_{i_h^k}$ of leg i_h^k refers to the same specific random variable as the start time $\tau_{i_{h+1}^k}$ of leg i_{h+1}^k, that is, $\tau'_{i_h^k} = \tau_{i_{h+1}^k} = \tau_{h+1}^k$. To keep track which legs are subsequent legs of the same service, we define for each leg i_h^k the *set of immediate successors* $I_{i_h^k} = \{i_{h+1}^k\}$ for $h \in \{1, \ldots, m-1\}$.

2.2. Walking legs

After each leg traversed, a commuter may choose to continue the journey by foot. Similarly as the service legs defined above, each *walking leg* $i \in \mathcal{L}$ is determined by a start node v_i, end node v'_i, start time τ_i, end time τ'_i and service number k_i. For all walking legs, we choose $k_i < 0$ in order to have a distinction between walking legs and service legs.

Walking legs are added by associating with each service leg $((v_i, \tau_i), (v'_i, \tau'_i), k_i)$ a set $L_i \subset \mathcal{L}$ of walking legs beginning at v'_i and ending at a stop within a specific *maximum walking distance* $d_w^{\max}$ from v'_i, see Figure 3a. Such walking legs $j \in L_i$ are of the form $\Big((v_j, \tau_j), (v'_j, \tau'_j), k_j\Big) = \Big((v'_i, \tau'_i), (v'_j, \tau'_j), -k_i\Big)$, where the walking distance d_w satisfies $d_w(v'_i, v'_j) \leq d_w^{\max}$. Note that the end time τ'_i of service leg i and the start time τ_j of a walking leg $j \in L_i$ refer to the same specific random variable. For each service leg i, the walking legs are added to the set of immediate successors I_i of i, that is, $I_i \leftarrow I_i \cup L_i$.

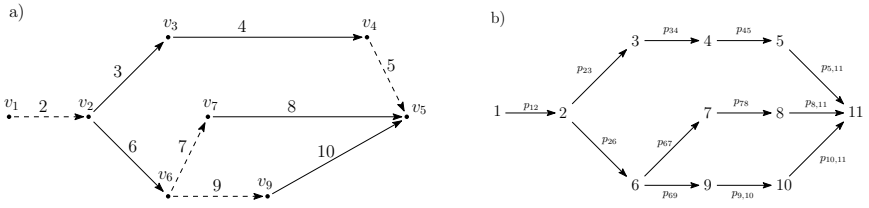


Figure 3: a) A sample transport network with eight stops and five transport services consisting of a single leg (solid arrows $3, 4, 6, 8, 10$). Adjacent stops are connected with walking legs (dashed arrows $2, 5, 7, 9$). b) A directed graph representing the relations of legs in Figure 3a. The origin and destination are represented by legs 1 and 11. The prior transfer probability from leg i to leg j is denoted by p_{ij}. Note that $p_{12} = p_{45} = p_{67} = p_{69} = 1$, since $2, 5, 7$ and 9 are walking legs.

The numerical experiments presented in Section 6 are restricted to the case in which there are no sequential walking legs, that is, no walking leg is followed by another walking leg. However, sequential walking legs can be added to the network iteratively: First, walking legs (indexed with service number $-k_i$) are added to the end of each each service leg i (step 1). Then, walking legs (indexed with service number k_j) are added to the end of each walking leg j (step 2). Step 2 is repeated until the desired amount of walking legs is obtained. The number of walking legs included in the model is thus controlled by two parameters: The maximum walking distance $d_w^{\max}$ and the maximum number of sequential walking legs.

2.3. Transfer probability

A transfer from leg i to leg j is possible only if leg i ends at the node from which leg j begins, that is, $v'_i = v_j$ (see Figure 3b). Letting $\mathcal{L}$ denote the set of legs and $S_i = \{j \in \mathcal{L} \mid v'_i = v_j\}$ denote

the *successor set* of leg i, the *prior transfer probability* from leg i to leg j is defined by

$$p_{ij} = \begin{cases} P(\tau'_i \leq \tau_j), & \text{if } j \in S_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Note that if j is a walking leg beginning at the end of leg i or if i and j are successive legs of the same service, by definition we have $p_{ij} = 1$, since τ'_i and τ_j refer to the same random variable. The legs and prior transfer probabilities form a directed graph (Figure 3b).

More generally, since the start and end time of a leg are not necessarily independent, the transfer probability from i to j depends on the *path* taken to get to i. A path can be represented as an acyclic sequence of legs $(i_1, \ldots, i_m)$ satisfying $i_{h+1} \in S_{i_h}$ for all $h \in \{1, \ldots, m-1\}$, see Figure 4. The path is *successful*, if $\tau'_{i_h} \leq \tau_{i_{h+1}}$ for all $h \in \{1, \ldots, m-1\}$. Note that if i_{h+1} is an immediate successor of i_h, that is, $i_{h+1} \in I_{i_h}$, we have $\tau'_{i_h} = \tau_{i_{h+1}}$ since the start time of $i_{h+1} \in I_{i_h}$ refers to the same specific random variable as the end time of i_h.

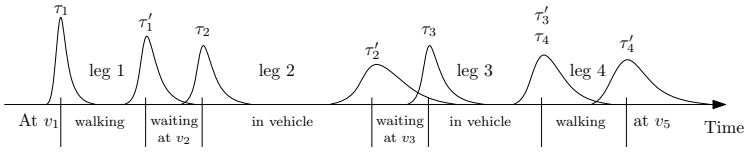


Figure 4: Deterministic and stochastic representations of a path. The ticks on the time axis represent the schedule of a deterministic itinerary from v_1 to v_5. The curves represent the distributions (gamma distributions in this example) of random variables τ_i, τ'_i that are used to model the start and end times of four legs that define the corresponding stochastic path. (1) A commuter starts walking at τ_1 from the origin v_1, and arrives at stop v_2 at τ'_1. (2) A transport service departs at v_2 at τ_2 and travels to stop v_3, arriving at τ'_2. (3) A transport service departs at v_3 at τ_3 and arrives at stop v_4 at τ'_3. (4) Immediately at $\tau_4 = \tau'_3$, the commuter continues by foot to the destination v_5, arriving at τ'_4. Note that the path is successful with probability $P(\tau'_1 \leq \tau_2 \cap \tau'_2 \leq \tau_3) = P(\tau'_1 \leq \tau_2) \cdot P(\tau'_2 \leq \tau_3 \mid \tau'_1 \leq \tau_2)$.

3. Problem formulation

The dynamic journeying problem is defined as follows. Let $[0, T] \subset \mathbb{R}$ be the time horizon of the problem, let v_1 denote the origin node and v_d denote the destination node. The *origin leg* is defined by $((v_1, \tau_1), (v_1, \tau_1), 0)$, where $P(\tau_1 = 0) = 1$, and the *destination leg* is defined by $((v_d, \tau_d), (v_d, \tau_d), 0)$, where $P(\tau_d = T) = 1$. Although it is possible to reach the destination node before T, the start time of the destination leg equals T with probability 1. By this definition, we can represent the entire journey as legs, including the origin and the destination. If a journey ends at a successful transfer to the destination leg, we know that the commuter has reached the destination node before T or at T.

With no loss of generality, all legs for which the start or end time is outside the time horizon $[0, T]$ with probability 1 and all legs for which the start node is the destination node (except the destination leg) are excluded from the problem. The cropped set of legs is defined by

$$\mathcal{L}_{[0,T]} = \{i \in \mathcal{L} \mid P(0 \leq \tau_i \leq T) > 0 \text{ and } P(0 \leq \tau'_i \leq T) > 0\} \setminus \{i \in \mathcal{L} \setminus \{d\} \mid v_i = v_d\}.$$

For clarity, we define $n := |\mathcal{L}_{[0,T]}|$ and the legs in $\mathcal{L}_{[0,T]}$ are numbered from 1 to n, where the origin leg is indexed by 1 and the destination leg is indexed by n.

The problem is modeled as a finite-state Markov decision process $(S, A, P_\cdot(\cdot,\cdot), R_\cdot(\cdot,\cdot))$, where the parameters are defined as follows.

3.1. States

The set of *states* S consists of paths $(1, i_1, \ldots, i_m)$ of legs, where $i_h \in \{1, \ldots, n\}$ for $h \in \{0, \ldots, m\}$, beginning from the origin leg 1. The state (1) is referred to as the *origin state*. Each state $s = (i_0, i_1, \ldots, i_m) \in S$, where $i_0 = 1$, is associated with a set of successor states $S_s = \{(i_0, i_1, \ldots, i_m, j) \in S \mid j \in S_{i_m}\}$, where S_{i_m} is the successor set of leg i_m. Similarly, the set of immediate successors of state s is defined by $I_s = \{(i_0, i_1, \ldots, i_m, j) \in S \mid j \in I_{i_m}\}$, where I_{i_m} is the immediate successor set of leg i_m. Formally, the set of states is defined by

$$S = \{(i_0, \ldots, i_m) \mid i_0 = 1 \text{ and } i_{h+1} \in S_{i_h} \text{ for } h \in \{0, \ldots, m-1\}\}. \tag{3}$$

The above definition of a state includes travel history in the form of a path, see Figure 5. The history of performed actions is not taken into account in our model.

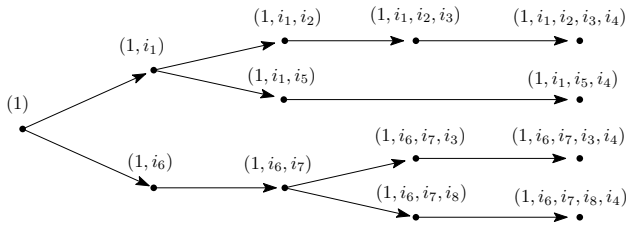


Figure 5: States of the markov decision process. A state is defined as a path consisting of legs, beginning from the origin leg 1. The states s in the figure are connected to successor states $s' \in S_s$ by arrows. Since the travel history is included in the definition of a state, the state space has a tree structure. In this example, the destination leg is denoted by i_4 and the set of destination states equals $\mathcal{D} = \{(1, i_1, i_2, i_3, i_4), (1, i_1, i_5, i_4), (1, i_6, i_7, i_3, i_4), (1, i_6, i_7, i_8, i_4)\}$.

The set of *destination states*, that is, the set of states $(1, \ldots, i_m) \in S$ for which the last leg i_m is the destination leg, is defined by $\mathcal{D} = \{(1, \ldots, i_m) \in S \mid i_m = n\}$. Since legs beginning from the destination node are not included in the problem, we have $S_s = I_s = \emptyset$ for all $s \in \mathcal{D}$.

The start and end times of state s are denoted by the random variables τ_s and τ'_s, respectively. The distributions of τ_s and τ'_s, where $s = (i_0, i_1, \ldots, i_m)$ and $i_0 = 1$, are defined by

$$f_{\tau_s}(x) = f_{\tau_{i_m}}(x \mid \cap_{h=0}^{m-1} \tau'_{i_h} \leq \tau_{i_{h+1}}), \tag{4}$$

$$f_{\tau'_s}(x) = f_{\tau'_{i_m}}(x \mid \cap_{h=0}^{m-1} \tau'_{i_h} \leq \tau_{i_{h+1}}), \tag{5}$$

for $x \in \mathbb{R}$. Note that if state s' is an immediate successor of state s, that is, $s' \in I_s$, the end time of state s and the start time of state s' refer to the same specific random variable $\tau'_s = \tau_{s'}$. In addition, if i_m is the first leg in the path executed by service k_{i_m}, that is, $k_{i_h} \neq k_{i_m}$ for all $h \in \{0, \ldots, m-1\}$, we have $f_{\tau_s}(x) = f_{\tau_{i_m}}(x)$ due to the fact that the arrival times of different services are assumed to be independent.

The service number of state $s = (i_0, i_1, \ldots, i_m)$ is defined by $k_s = k_{i_m}$.

3.2. Actions

The set of *actions* A consists of sets A_s of actions available at states $s \in S$. An action $a \in A_s$ is defined as a *preference order* of the successor states $s' \in S_s$, that is, a bijection $a : S_s \to \{1, \ldots, |S_s|\}$, where $a(s')$ denotes the ranking of the successor state $s' \in S_s$ in the preference order. The successor states of s ranked by the preference order a are denoted by $s^{a,a(s')}$. Given the the sorted successor states $s^{a,1}, \ldots, s^{a,|S_s|} \in S_s$ of s, the commuter transfers to state $s^{a,k}$ if (i) the transfer to $s^{a,g}$ is unsuccessful for $1 \leq g < k$ and (ii) the transfer to $s^{a,k}$ is successful.

3.3. Transition probabilities

$P_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at step t will lead to state $s' \in S_s$ at step $t + 1$. Given the preference order $s^{a,1}, \ldots, s^{a,|S_s|} \in S_s$ defined by action $a \in A_s$, we have

$$P_a(s, s^{a,k}) = P\left(\tau'_s \le \tau_{s^{a,k}} \cap \bigcap_{g=1}^{k-1} \tau'_s > \tau_{s^{a,g}}\right). \tag{6}$$

If the service numbers of all pairs of distinct successor states $s', s'' \in S_s$ of state s satisfy $k_{s'} \neq k_{s''}$, we have $P_a(s, s^{a,k}) = \int_{-\infty}^{\infty} f_{\tau'_s}(x) P\left(x \le \tau_{s^{a,k}}\right) \prod_{g=1}^{k-1} P\left(x > \tau_{s^{a,g}}\right)\, dx$, since the start times of legs with different service numbers are independent. Note that since the end time τ'_s of state s and the start time of an immediate successor state $s' \in I_s$ refer to the same random variable, that is, $\tau'_s = \tau_{s'}$ for all $s' \in I_s$, the transfer probability (6) satisfies

$$P_a(s, s^{a,k}) = \begin{cases} 0, & \text{if } \{s^{a,1}, \ldots, s^{a,k-1}\} \cap I_s \neq \emptyset\,, \\ P\left(\bigcap_{g=1}^{k-1} \tau'_s > \tau_{s^{a,g}}\right), & \text{if } \{s^{a,1}, \ldots, s^{a,k-1}\} \cap I_s = \emptyset \text{ and } s^{a,k} \in I_s\,, \\ P\left(\tau'_s \le \tau_{s^{a,k}} \cap \bigcap_{g=1}^{k-1} \tau'_s > \tau_{s^{a,g}}\right), & \text{otherwise.} \end{cases} \tag{7}$$

For example, let $s^{a,1}, s^{a,2}, s^{a,3}$ denote three successor states of state s, sorted in the preference order defined by action a. If $s^{a,2}$ is an immediate successor of s, that is, $s^{a,2} \in I_s$, and $s^{a,1} \notin I_s$, we have $P_a(s, s^{a,1}) = P\left(\tau'_s \le \tau_{s^{a,1}}\right)$, $P_a(s, s^{a,2}) = P\left(\tau'_s > \tau_{s^{a,1}}\right)$ and $P_a(s, s^{a,3}) = 0$.

3.4. Rewards

$R_a(s, s')$ is the expected immediate reward received after transition from state $s \in S$ to state $s' \in S$ with transition probability $P_a(s, s')$. Since the objective is to arrive at a destination state $s \in \mathcal{D}$, we define $R_a(s, s')$ by

$$R_a(s, s') = \begin{cases} 1, & \text{if } s' \in \mathcal{D}\,, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Note that $R_a(s, s')$ is independent of the action a. Thus, for the remainder of this document, we will use the notation $R(s, s') = R_a(s, s')$.

While the main focus of this paper is on on-time arrival probability, travel and overdue costs can be included with minimial effort as follows. Instead of maximizing the probability of reaching the destination leg, the goal is to maximize the expected profit gained by arriving at the destination node.

Let u denote the revenue (or utility) received upon arrival at the destination node and $c(s)$ denote a real valued positive function representing the cost of state s. For example, $c((i_0, \ldots, i_m))$ could be defined as the sum of the *leg costs* c_i, that is, $c((i_0, \ldots, i_m)) = \sum_{h=0}^{m-1} c_{i_h}$. Let C_o denote the *overdue cost*, that is, the cost of arriving at the destination node after T. By introducing an additional *late destination leg* $((v_d, \tau_l), (v_d, \tau_l), 0)$, where $P(\tau_l = \infty) = 1$, and the set of *late destination states* $\mathcal{D}'$, for which the last leg is the late destination leg, the reward function is written in the general form

$$R'(s, s') = \begin{cases} u - c(s'), & \text{if } s' \in \mathcal{D}\,, \\ u - c(s') - C_o, & \text{if } s' \notin \mathcal{D} \text{ and } s' \in \mathcal{D}'\,, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

4. Problem solution

The solutions to Markov decision processes are characterized as policies, that is, functions π that specify the action $a(s)$ that the commuter chooses when in state s. The goal is to find a policy π that maximizes the expected reward. Generally, the calculation of an optimal policy requires two arrays indexed by state: value V, which contains real values, and policy π which contains actions. In our problem, $V(s)$ corresponds to the probability of reaching a destination state $s' \in \mathcal{D}$ from state s by following a policy that maximizes the probability of reaching a destination state. Note that the value $V((1))$ of the origin state (1) is strictly positive if and only if there exists a state $s \in S$ for which $S_s \cap \mathcal{D} \neq \emptyset$ and $P(\tau_s \leq T) > 0$.

Similarly as in (Bellman, 1957), the value $V(s)$ is defined by

$$V(s) := \max_{a \in A_s} \left\{ \sum_{s' \in S_s} P_a(s, s') \left(R(s, s') + V(s') \right) \right\} \tag{10}$$

for all $s \in S$. Note that since $V(s)$ represents a probability, we have $0 \leq V(s) \leq 1$ for all $s \in S$. In addition, $V(s) = 0$ for all $s \in \mathcal{D}$, since $S_s = \emptyset$ for all $s \in \mathcal{D}$. Since $R(s, s') = 0$ for all $s' \notin \mathcal{D}$ and $R(s, s') = 1$ for all $s' \in \mathcal{D}$, we have $0 \leq R(s, s') + V(s') \leq 1$ for all $s \in S$.

An optimal policy is characterized as follows: When at state s, the available transfer options $X \subset S_s$ to successor states are revealed. The commuter transfers to a state $s' \in X$ for which $R(s, s')+V(s')$ is maximized. Formally, an optimal action a at state s is determined by the following theorem.

Theorem 1. *Let $s \in S$ be a state. An action $a \in A_s$ is optimal satisfying Equation* (10)*, if*

$$R(s, s^{a,1}) + V(s^{a,1}) \geq R(s, s^{a,2}) + V(s^{a,2}) \geq \ldots \geq R(s, s^{a,|S_s|}) + V(s^{a,|S_s|}), \tag{11}$$

where the successor states ranked by action a are denoted by $s^{a,a(s')}$ for all $s' \in S_s$.

Proof. Let $S_s = \{s^1, \ldots, s^{|S_s|}\}$ denote the successor set of state s. Given an arbitrary realization $\{\tau'_s = t'_s, \tau_{s^1} = t_{s^1}, \ldots, \tau_{s^{|S_s|}} = t_{s^{|S_s|}}\}$ of the end time of state s and the start times of states $s^k \in S_s$, the conditional value of s satisfies

$$V(s \mid t'_s, t_{s^1}, \ldots, t_{s^{|S_s|}}) = \max_{\{k \in \{1,\ldots,|S_s|\} \mid t'_s \leq t_{s^k}\}} \left\{ R(s, s^k) + V(s^k) \right\}.$$

Note that $t_{s^k} = t'_s$ for all $s^k \in I_s$. Let a be an action satisfying Equation (11). Since the conditional transition probability is given by

$$P_a(s, s^{a,k} \mid t'_s, t_{s^{a,1}}, \ldots, t_{s^{a,|S_s|}}) = \begin{cases} 1, & \text{if } t'_s \leq t_{s^{a,k}} \text{ and } t'_s > t_{s^{a,g}} \text{ for } 1 \leq g < k, \\ 0, & \text{otherwise,} \end{cases}$$

we have

$$V(s \mid t'_s, t_{s^{a,1}}, \ldots, t_{s^{a,|S_s|}}) = \sum_{k=1}^{|S_s|} P_a(s, s^{a,k} \mid t'_s, t_{s^{a,1}}, \ldots, t_{s^{a,|S_s|}}) \left(R(s, s^{a,k}) + V(s^{a,k}) \right).$$

Integrating over $\mathbb{R}^{|S_s|+1}$ yields

$$V(s) = \sum_{k=1}^{|S_s|} P_a(s, s^{a,k}) \left(R(s, s^{a,k}) + V(s^{a,k}) \right). \tag{12}$$

□

Theorem 1 gives an optimal action for a state s, given that the values $V(s')$ of its successor states are known. In the following, we present an algorithm for calculating the values for all states.

4.1. Backward induction algorithm

The values $V(s)$ of states can be determined by means of backward induction, as shown in Algorithm 1. By executing Prob((1)), the program recursively calculates values and optimal actions for all states s that are reachable from the origin state (1).

Initially, we only know the values of destination states, that is, $V(s) = 0$ for all $s \in \mathcal{D}$. Thus, the first states for which the value can be calculated are the ones that precede a destination state. The algorithm then proceeds backwards until the value $V((1))$ of the origin state is calculated.

Algorithm 1: A recursive function Prob(s) for calculating the value $V(s)$ for state s.

forall $s' \in S_s$ *(successor set of s)* **do**
    $V(s') \leftarrow \text{Prob}(s')$;
end
Determine an optimal action a by sorting the states $s' \in S_s$ in descending order of $R(s, s') + V(s')$;
$V(s) \leftarrow \sum_{k=1}^{|S_s|} P_a(s, s^{a,k}) \left(R(s, s^{a,k}) + V(s^{a,k}) \right)$ ($V(s) = 0$ for destination states);
Return $V(s)$;

5. Analysis

In the following examination, we present theoretical results related to optimal policies described in the previous section. For clarity, walking legs are not considered in this section. However, since walking legs and service legs are defined similarly apart from the service number, the results are extended to handle walking legs with minimal effort.

We first prove an intuitive result: The probability of reaching the destination from a given state s by following an optimal policy is always greater than or equal to the probability of success of any predetermined path from s to the destination.

Lemma 2. *Let $(i_0, i_1, \ldots, i_{d-1}, i_d)$, where $i_0 = 1$, be a destination state. Let p_{s_h} denote the probability of success of the path $(i_h, \ldots, i_d)$ from leg i_h to the destination leg i_d, given that the commuter is at state $s_h = (i_0, i_1, \ldots, i_{h-1}, i_h)$. Then, $V(s_h)$ defined by Equation* (12) *satisfies $V(s_h) \geq p_{s_h}$ for all $h \in \{0, \ldots, d-1\}$.*

Proof. For clarity, we will prove the case in which the service numbers of distinct successor states $s', s'' \in S_s$ satisfy $k_{s'} \neq k_{s''}$. Clearly, $p(s_h)$ satisfies $p(s_h) = P(\tau'_{s_h} \leq \tau_{s_{h+1}}) p(s_{h+1})$ for all $h \in$

$\{0, \ldots, d-1\}$. Note that for an optimal action a, we can choose $s_{d-1}^{a,1} = s_d$ since $R(s_{d-1}, s_d) + V(s_d) = 1$. Thus,

$$V(s_{d-1}) = \sum_{k=1}^{|S_{s_{d-1}}|} P_a(s_{d-1}, s_{d-1}^{a,k}) \left(R(s_{d-1}, s_{d-1}^{a,k}) + V(s_{d-1}^{a,k})\right) \geq P_a(s_{d-1}, s_{d-1}^{a,1}) \left(R(s_{d-1}, s_{d-1}^{a,1}) + V(s_{d-1}^{a,1}\right)$$

$$= P_a(s_{d-1}, s_d) \overbrace{(R(s_{d-1}, s_d) + V(s_d))}^{=1} = P(\tau'_{s_{d-1}} \leq \tau_{s_d}) = p(s_{d-1}).$$

The proof is completed by showing that $V(s_{h+1}) \geq p(s_{h+1})$ implies $V(s_h) \geq p(s_h)$ for all $h \in \{0, \ldots, d-2\}$.

Let $s_h \in \{s_0, s_1, \ldots, s_{d-2}\}$ be an arbitrarily chosen state and assume that $V(s_{h+1}) \geq p(s_{h+1})$ holds. Let $a(s_{h+1})$ denote the ranking of s_{h+1} in the ordered successor states of s_h, that is, $s_h^{a,a(s_{h+1})} = s_{h+1}$. Then, $V(s_h^{a,k}) \geq V(s_{h+1})$ for all $k \in \{1, \ldots, a(s_{h+1})\}$ and

$$V(s_h) \geq \sum_{k=1}^{a(s_{h+1})} V(s_h^{a,k}) \int_{-\infty}^{\infty} f_{\tau'_{s_h}}(x) P(x \leq \tau_{s_h^{a,k}}) \prod_{g=1}^{k-1} P(x > \tau_{s_h^{a,g}})\, dx$$

$$\geq V(s_{h+1}) \sum_{k=1}^{a(s_{h+1})} \int_{-\infty}^{\infty} f_{\tau'_{s_h}}(x) P(x \leq \tau_{s_h^{a,k}}) \prod_{g=1}^{k-1} P(x > \tau_{s_h^{a,g}})\, dx$$

$$= V(s_{h+1}) \int_{-\infty}^{\infty} f_{\tau'_{s_h}}(x) \left(1 - P(x > \tau_{s_h^{a,a(s_{h+1})}}) \overbrace{\prod_{g=1}^{a(s_{h+1})-1} P(x > \tau_{s_h^{a,g}})}^{\leq 1} \right) dx$$

$$\geq V(s_{h+1})(1 - P(\tau'_{s_h} > \tau_{s_{h+1}})) = P(\tau'_{s_h} \leq \tau_{s_{h+1}}) V(s_{h+1}) \geq P(\tau'_{s_h} \leq \tau_{s_{h+1}}) p(s_{h+1}) = p(s_h).$$

□

Note that Lemma 2 is true for any path that maximizes the probability of success for reaching the destination. By using the above result, we show that the ratio of probabilities $V(s_h)/p(s_h)$ is an increasing function of the distance from the destination.

Theorem 3. *Using the notation of Lemma 2, we have* $V(s_h)/p(s_h) \geq V(s_{h+1})/p(s_{h+1}) \geq 1$ *for all* $h \in \{0, \ldots, d-1\}$.

Proof. Given that the commuter is at state $s_h = (i_0, \ldots, i_h)$, the probability of success of the path $(i_h, \ldots, i_d)$ satisfies $p(s_h) = P(\tau'_{s_h} \leq \tau_{s_{h+1}}) p(s_{h+1})$. Similarly as in the proof of Lemma 2, we see that $V(s_h) \geq P(\tau'_{s_h} \leq \tau_{s_{h+1}}) V(s_{h+1})$ and thus

$$\frac{V(s_h)}{p(s_h)} \geq \frac{P(\tau'_{s_h} \leq \tau_{s_{h+1}}) V(s_{h+1})}{P(\tau'_{s_h} \leq \tau_{s_{h+1}}) p(s_{h+1})} = \frac{V(s_{h+1})}{p(s_{h+1})} \geq 1.$$

□

Theorem 3 establishes a fundamental difference between dynamic and static journey planning: The importance of being able to reconsider the remaining path is emphasized when the number of transfers is increased.

5.1. Leg duration models

Thus far we have considered the general case where the start and end times of legs are defined as arbitrary real-valued random variables. For a more specific analysis, we present two alternative ways of modeling the durations of legs.

5.1.1. Independent durations

Perhaps the most intuitive way of defining the start and end times of a leg of a public transport service is to assume that the *duration* of each leg i is a real-valued random variable D_i with a strictly positive distribution and that the durations of legs are independent. Given a service consisting of legs $i_1, \ldots, i_r$ and the start time τ_{i_1}, the end time of leg i_h and the start time of leg i_{h+1} satisfy $\tau'_{i_h} = \tau_{i_{h+1}} = \tau_{i_1} + \sum_{h=1}^{r} D_{i_h}$. Note that in this model the uncertainty of start and end times are greater at the end of the route (see Figure 2). In addition, the conditional transfer probabilities are smaller than the prior transfer probabilities: For example, let us consider a transfer $j \to l$ with prior transfer probability $p_{jl} = P(\tau'_j \le \tau_l) = P(\tau_j + D_j \le \tau_l)$. Given that the commuter has already successfully transferred from i to j, where $P(\tau'_i \le \tau_j) < 1$, the transfer $j \to l$ is generally less likely to be successful, that is, $P(\tau_j + D_j \le \tau_l \mid \tau'_i \le \tau_j) \le P(\tau_j + D_j \le \tau_l)$. In this model, the start and end time distributions of states are characterized by the following theorem.

Theorem 4. *Let $s = (i_0, \ldots, i_m) \in S, i_0 = 1$ be a state for which all legs are executed by different services, that is, $g \neq h \Rightarrow k_{i_g} \neq k_{i_h}$ for all $g, h \in \{0, \ldots, m\}$. Then, the density function $f_{\tau_s}(x)$ of the start time τ_s of state s is given by $f_{\tau_s}(x) = f_{\tau_{i_m}}(x)$ and the cumulative distribution function $F_{\tau_s}(x)$ of the end time τ'_s of state s is given by*

$$F_{\tau'_s}(x) = \int_{-\infty}^{\infty} f_{\tau_{i_0}+D_{i_0}}(x'_0) \int_{x'_0}^{\infty} f_{\tau_{i_1}}(x_1) \int_{x_1}^{\infty} f_{x_1+D_{i_1}}(x'_1) \cdots \int_{x'_{m-1}}^{\infty} f_{\tau_{i_m}}(x_m) \int_{x_m}^{x} f_{x_m+D_{i_m}}(x'_m) dx'_m dx_m \ldots dx'_0. \tag{13}$$

Proof. Since the start and end times of legs with different service numbers are independent, by definition (4) we get $f_{\tau_s}(x) = f_{\tau_{i_m}}(x \mid \cap_{h=0}^{m-1} \tau'_{i_h} \le \tau_{i_{h+1}}) = f_{\tau_{i_m}}(x)$.

The cumlative distribution function of τ'_s is defined by $F_{\tau'_s}(x) = P(\tau'_s \le x)$. Let us use the notation $s_h = (i_0, \ldots, s_h)$ for all $h \in \{0, \ldots, m\}$. Clearly, the density function of τ'_{s_0} is given by $f_{\tau'_{s_0}}(x) = f_{\tau_{i_0}+D_{i_0}}(x)$. Furthermore,

$$F_{\tau_{s_{h+1}}}(x) = \int_{-\infty}^{\infty} f_{\tau'_{s_h}}(x'_h) \int_{x'_h}^{x} f_{\tau_{i_{h+1}}}(x_{h+1}) dx_{h+1} dx'_h \quad \text{and}$$

$$F_{\tau'_{s_{h+1}}}(x) = \int_{-\infty}^{\infty} f_{\tau_{s_{h+1}}}(x_{h+1}) \int_{x_{h+1}}^{x} f_{x_{h+1}+D_{i_{h+1}}}(x'_{h+1}) dx'_{h+1} dx_{h+1}.$$

Since $f_{\tau_{s_{h+1}}}(x_{h+1}) = \frac{d}{dx} F_{\tau_{s_{h+1}}}(x)$, combining the equations yields a recursive equation

$$F_{\tau'_{s_{h+1}}}(x) = \int_{-\infty}^{\infty} f_{\tau'_{s_h}}(x'_h) \int_{x'_h}^{\infty} f_{\tau_{i_{h+1}}}(x_{h+1}) \int_{x_{h+1}}^{x} f_{x_{h+1}+D_{i_{h+1}}}(x'_{h+1}) dx'_{h+1} dx_{h+1} dx'_h.$$

□

Theorem 4 gives us a formula for calculating the end time distribution of a state defined as a sequence of legs, given that the legs are executed by different services. For a state $s = (i_0, \ldots, i_m)$ for which some subsequent legs $i_g, \ldots, i_h$ are immediate successors, equation (13) can be used by joining these legs to produce a single leg with duration $\sum_{k=g}^{h} D_{i_k}$.

In practice, the start and end time distributions of states can be determined by means of random sampling, as discussed in Section 5.2.

5.1.2. Independent start and end times

In some cases it might be reasonable to think that the driver of a transport service is capable of adapting to the situation: If the service is behind schedule, the driver makes an effort to reach the remaining stops on the route in time by increasing the pace. That is, the durations of subsequent legs of a service are not necessarily independent. This scenario can be approximated by assuming that the start and end times of all legs are independent random variables. In this case, the transition probabilities between legs are conditionally independent of all previous states and actions.

Since the travel history is ignored, the states are definded as legs, and the state space is defined by

$$S = \mathcal{L}_{[0,T]} = \{1, \ldots, n\}, \tag{14}$$

where state 1 denotes the orgin leg and state n denotes the destination leg, see Figure 6. Furthermore, the start and end times of states coincide with the unconditional start and end times of legs, in contrast to Equations (4) and (5), where the start and end times are defined as conditional random variables.

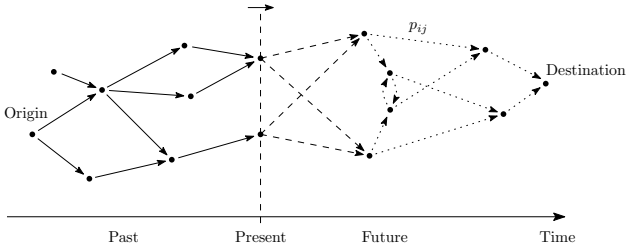


Figure 6: A timeline of the history independent markov decision process. The points denote states (=legs) and the arrows between the points denote transfers between states. The solid arrows denote successful *past transfers*, which form an acyclic graph. The dashed arrows denote *present transfers*, which are realized at the present time. The dotted arrows denote *future transfers* $i \to j$, each of which is successful with prior probability p_{ij}. Note that there may exist states i, j for which both $p_{ij} > 0$ and $p_{ji} > 0$. Over time, future transfers become present transfers and present transfers become past transfers.

We suggest that the history independent model (14) can be used to approximate to the conditional case (3). The following theorem characterizes the relation between the two models.

Theorem 5. *Let i, j and k denote three legs and let D_j denote the random duration of leg j. Given that the transfer $i \to j$ is successful and assuming that the durations of legs are independent, the conditional probability $P(\tau'_j \leq \tau_k \mid \tau'_i \leq \tau_j)$ of a successful transfer $j \to k$ satisfies*

$$P(\tau'_j \leq \tau_k) \geq P(\tau'_j \leq \tau_k \mid \tau'_i \leq \tau_j) \geq \frac{P(\tau'_i \leq \inf \operatorname{supp} f_{\tau_j})}{P(\tau'_i \leq \tau_j)} P(\tau'_j \leq \tau_k),$$

where $\inf \operatorname{supp} f_{\tau_j}$ *denotes the greatest real number* t *for which* $f_{\tau_j}(t) = 0$ *and* $t \leq x$ *for all* x *satisfying* $f_{\tau_j}(x) > 0$.

Proof. Defining $t = \inf \operatorname{supp} f_{\tau_j}$, the probability that both transfers $i \to j$ and $j \to k$ are successful satisfies

$$\begin{aligned} P(\tau'_j \leq \tau_k \cap \tau'_i \leq \tau_j) &= \int_{-\infty}^{\infty} f_{\tau'_i}(x'_i) \int_{x'_i}^{\infty} f_{\tau_j}(x_j) P(x_j + D_j \leq \tau_k)\, dx_j\, dx'_i \\ &\geq \int_{-\infty}^{t} f_{\tau'_i}(x'_i) \int_{t}^{\infty} f_{\tau_j}(x_j) P(x_j + D_j \leq \tau_k)\, dx_j\, dx'_i = P(\tau'_i \leq t) P(\tau'_j \leq \tau_k). \end{aligned}$$

□

Theorem 5 states that if the distributions of τ'_i and τ_j are relatively far apart, the conditional probability $P(\tau'_j \leq \tau_k \mid \tau'_i \leq \tau_j)$ of a successful transfer from j to k is close to the unconditional probability $P(\tau'_j \leq \tau_k)$. In this case, the two leg duration models discussed above converge and the independent durations model can be approximated by assuming independence of start and end times.

5.2. Algorithms for the independent durations model

For the independent leg durations model (Section 5.1.1), the start and end time distributions can be calculated by means of random sampling as follows. For each leg $i \in \{1, \ldots, n\}$, we calculate $R \in \mathbb{N}$ realizations for the start and end times, that is, sets $T_i = \{t_i(1), \ldots, t_i(R)\}$ and $T'_i = \{t'_i(1), \ldots, t'_i(R)\}$, which approximate the distributions of τ_i and τ'_i, respectively. The random sampling algorithm method is described in Algorithm 2.

Algorithm 2: A recursive function Sample(i, t, q) for calculating the start and end time distributions of legs. The function Sample(i, t_i, q) is called for all $q \in \{1, \ldots, R\}$ and for all initial legs i of services (see Equation (1)), where t_i is a random number from the distribution τ_i.

$t_i(q) \leftarrow t$;
Choose a random number d according to distribution D_i;
$t'_i(q) \leftarrow t + d$;
forall $i' \in I_i$ *(immediate successor set of* i*)* **do**
    Sample$(i', t'_i(q), q)$;
end

Given the random samples of the start and end times of legs, the approximate start and end time distributions $T_s = \{t_s(1), \ldots, t_s(Q)\}$ and $T'_s = \{t'_s(1), \ldots, t'_s(Q)\}$ of states $s \in S$ are determined by repeating the procedure described in Algorithm 3 for all $q \in \{1, \ldots, R\}$.

5.3. Algorithms for the history independent model

Equation (14) suggests an approximation for the conditional markov decision process (3) by assuming history independence. Since no information on already visited states is considered, the history independent probabilities can be computed more efficiently than the conditional values given by Algorithm 1.

In particular, since the state is defined by the current leg, that is, $S = \{1, \ldots, n\}$, the number of states is in most cases significantly smaller than in the history dependent model. Letting $V_H(s)$

Algorithm 3: A recursive function Sample2(s, q) for calculating the start and end time distributions of states. The function Sample2$((1), q)$ is called for all $q \in \{1, \ldots, R\}$, where (1) is the origin state.

$i \leftarrow$ last leg of s;
$T_s \leftarrow T_s \cup \{t_i(q)\}$;
$T'_s \leftarrow T'_s \cup \{t'_i(q)\}$;
forall $s' \in S_s$ *(successor set of s)* **do**
  $i' \leftarrow$ last leg of s';
  if $t'_i(q) \leq t_{i'}(q)$ **then**
    Sample2(s', q);
  end
end

Algorithm 4: A recursive function Prob(s) for calculating an optimal policy for the history independent model. Initially, set $V_H(s) \leftarrow -1$ for all $s \in \{1, \ldots, n-1\}$ and $V(n) \leftarrow 0$.

if $V_H(s) \geq 0$ *($V_H(s)$ has already been determined)* **then**
  Return $V_H(s)$;
end
forall $s' \in S_s$ *(successor set of s)* **do**
  $V_H(s') \leftarrow$ Prob(s');
end
Determine an optimal action a by sorting the states $s' \in S_s$ in descending order of $R(s, s') + V_H(s')$;
$V_H(s) \leftarrow \sum_{k=1}^{|S_s|} P_a(s, s^{a,k}) \left(R(s, s^{a,k}) + V_H(s^{a,k})\right)$;
Return $V_H(s)$;

denote the history independent value of state s, we may initially define $V_H(s) = -1$ for all $s \in \{1, \ldots, n-1\}$ and $V_H(n) = 0$ in order to keep track of which states have already been visited by the algorithm. The history independent version of Algorithm 1 is presented in Algorithm 4.

While the recursive solution presented in Algorithm 4 produces the value for all states that can be reached from a given initial state s, the values can be efficiently computed for all states simultaneously by using a power method as follows.

Let us first write Equation (12) in matrix form. Let $s^{a,1}, \ldots, s^{a,|S_s|} \in S_s$ denote the ordered successor states of s and let J denote a $(n \times n)$-matrix defined by

$$J_{ss'} = \begin{cases} 1, & \text{if } s = s' = n, \\ P_a(s, s'), & \text{if } s \neq s' \text{ and } s' \in S_s, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Element $J_{ss'}$ represents the probability that state s' succeeds state s given that the commuter follows a policy determined by action a.

Letting $V_H = (V_H(1), \ldots, V_H(n-1), 1)^T$ denote the *value vector*, Equation (12) can be written in the form $V_H = JV_H$. Note that the last element $V_H(n)$ is replaced by 1 due to the fact that $(V_H(n) + R(s, n)) = 1$ for all $s \in S$.

In other words, V_H is the dominant eigenvector of J, that is, the eigenvector corresponding to eigenvalue 1. However, the calculation of V_H is not straightforward since the elements $J_{ss'}$ depend on the descending order of the elements $V_H(s^{a,k})$.

We propose a variation of the power method (Meyer, 2000) for determining J as well as the dominant eigenvector V_H, see Algorithm 5. The main idea is that the vector V_H is consecutively updated by multiplying matrix J by V_H. After each update, the successor states s' of each state s are sorted in descending order of value $V_H(s')$.

Algorithm 5: A variation of the power method for calculating J and the dominant eigenvector V_H.

```
Set V_H = (1, ..., 1);
repeat
    forall s ∈ {1, ..., n} do
        Determine action a by sorting the states s' ∈ S_s in descending order of R(s, s') + V_H(s');
        forall s^{a,k} ∈ S_s do
            J_{s s^{a,k}} ← P_a(s, s^{a,k});
        end
    end
    V_H ← JV_H;
until convergence ;
```

5.4. Unconditional transfers

The history independent model defined by Equation (14) can be further approximated by assuming that successful transfers from a state s to its successor states are independent events, which is not generally true. However, if the spreads of the time distributions are relatively small, we obtain a reasonable approximation by assuming independence of successful transfers. From a practical viewpoint, since we are particularly interested in the choice for the commuter in each situation, the values are not as interesting as the *descending order* of the values, that is, the action at each state. We suggest that even if approximate models are used, the actions of the commuter are in most cases similar as with more accurate models since the descending order of the probabilities is preserved.

Since transfers between legs are assumed to be independent events, the transition probability function (6) satisfies

$$P_a(s, s^{a,k}) = P\left(\tau'_s \leq \tau_{s^{a,k}} \cap \bigcap_{g=1}^{k-1} \tau'_s > \tau_{s^{a,g}}\right) = \sum_{k=1}^{h} p_{s s^{a,k}} \prod_{g=1}^{k-1} (1 - p_{s s^{a,g}}), \tag{16}$$

where $p_{s s^{a,k}}$ denotes the prior transfer probability from s to $s^{a,k}$, as defined in Equation (2). Since the state space is equal to the state space of the history independent model, Algorithms 4 and 5 can be used to calculate the unconditional values .

Theorem 6. *The history independent value $V_H(s)$ defined by model* (14) *and the unconditional value $V_U(s)$ defined by* (16) *satisfy $V_H(s) \leq V_U(s)$ for all $s \in \{1, \ldots, n\}$.*

Proof. Letting a denote an optimal action, by Equation (12) we get

$$\begin{aligned}
V_H(s) &= \sum_{k=1}^{h} (R(s, s^{a,k}) + V_H(s^{a,k})) \int_{-\infty}^{\infty} f_{\tau'_s}(x) P(x \leq \tau_{s^{a,k}}) \prod_{g=1}^{k-1} P(x > \tau_{s^{a,g}})\, dx \\
&\leq \sum_{k=1}^{h} (R(s, s^{a,k}) + V_H(s^{a,k})) \int_{-\infty}^{\infty} f_{\tau'_s}(x) P(x \leq \tau_{s^{a,k}})\, dx \cdot \prod_{g=1}^{k-1} \int_{-\infty}^{\infty} f_{\tau'_s}(x) P(x > \tau_{s^{a,g}})\, dx \\
&= \sum_{k=1}^{h} (R(s, s^{a,k}) + V_H(s^{a,k})) P(\tau'_s \leq \tau_{s^{a,k}}) \prod_{g=1}^{k-1} P(\tau'_s > \tau_{s^{a,g}}) \\
&= \sum_{k=1}^{h} (R(s, s^{a,k}) + V_H(s^{a,k})) p_{s s^{a,k}} \prod_{g=1}^{k-1} (1 - p_{s s^{a,g}}).
\end{aligned}$$

Noting that $V_H(n) = V_U(n) = 0$ for the destination state n, the proof is completed by backward induction. □

Note in particular that if the transfer probabilities satisfy $P(\tau'_s \leq \tau_{s^{a,k}}) \in \{0, 1\}$ for all successor states $s^{a,k} \in S_s$, we have $V_H(s) = V_U(s) = V_U(s^{a,k^*})$, where k^* is the smallest index for which $P(\tau'_s \leq \tau_{s^{a,k}}) = 1$. In this case, the history independent and unconditional coincide. In other words, the approximation given by (16) is seen to be most accurate when the variances of the arrival times are small, which means that the transfer probabilities are likely to be close to zero or one.

Furthermore, if each state has a single successor state, the history independent and unconditional models yield the same result. We suggest that the accuracy of the unconditional model is greatest when the average number of successor states is small.

5.5. Summary of models

In summary, the conditional (3), history independent (14) and unconditional (16) models described above can be characterized by means of two types of conditionality: (i) *History dependence* refers to the fact that the executed part of a journey affects the probability of success of the remaining part of the journey. (ii) *Transfer conditionality* refers to the fact that the success of different transfers at a given stop are not independent events.

The conditional model takes into account both types of conditionality whereas in the unconditional model, neither of the conditionalities are considered. The relations between the conditionalities and the three models are presented in the following table.

Type of conditionality \ Model	Conditional model (3)	History independent model (14)	Unconditional model (16)
History dependence	Yes	No	No
Transfer conditionality	Yes	Yes	No

6. Numerical experiments

In the following, we present computational results obtained by the conditional model (3), history independent model (14) and the unconditional model (16). In the conditional model, the durations of legs are assumed to be independent as described in Section 5.1.1 and in the history independent and unconditional models, the start and end times of legs are assumed to be independent, as described in Section 5.1.2.

First, the computational performance of the different models is compared in randomized and real-life instances (Section 6.1). Then, we study the probability of success of journeys in regular networks as a function of transfer margin and the number of transfer options at each stop, as well as the number of transfers (Section 6.2).

In all experiments, the travel times between stops are defined as shifted gamma distributed random variables similarly as in (Russell and Urban, 2007; Chiang et al., 1980). More precisely, given the average travel time t_{uv} between stops u and v, we define the travel time τ_{uv} as a random variable $\tau_{uv} \sim \text{Gamma}(\alpha t_{uv}, \beta, \delta t_{uv})$, where the $\text{Gamma}(\alpha, \beta, \delta)$ distribution is defined by the probability density function

$$f(x) = \frac{(x-\delta)^{\alpha-1} e^{-(x-\delta)/\beta}}{\beta^{\alpha}\Gamma(\alpha)} \quad \text{for } x > \delta \geq 0. \tag{17}$$

The mean of a Gamma(α, β, δ) distribution is $\alpha\beta + \delta$, that is, by choosing α, β and δ such that $\alpha\beta + \delta = 1$, we have $E(\tau_{uv}) = t_{uv}$.

We assume that at the beginning of the time horizon at $t = 0$, the locations of all vehicles providing service are known. Thus, the deterministic schedules $((v_1, t_1), \ldots, (v_p, t_p))$ of services are converted into random schedules $((v_1, \tau_1), \ldots, (v_p, \tau_p))$ by defining $\tau_1 \sim$ Gamma$(\alpha t_1, \beta, \delta t_1)$ and $\tau_i = \tau_{i-1} + \tau_{v_{i-1}v_i}$ for $i \in \{2, \ldots, p\}$.

The values of the α, β and δ parameters are $(1, 0.25, 0.75)$, which corresponds to the coefficient of variation $\sigma/\sqrt{\mu} = 0.25$ (Russell and Urban, 2007).

The algorithms were implemented in Mathematica and the tests were performed on a 2.2 GHz Dual Core Intel PC.

6.1. Random and real-life networks

In this section we present computational results of tests conducted on randomized instances and real-life data. The real-life instances are based on the tram schedules of Helsinki and the parameters of the randomized instances are based on the size of the public transport network of Helsinki including trams, buses and subways, which is approximately ten times the size of the tram network (Helsinki City Transport, 2011).

The real-life tram network consists of ten tram lines, operated by 50 trams, and 154 stops. In the randomized instances, there are 100 tram lines (and a subway line), operated by 500 trams (and 10 subway vehicles), and approximately 1500 stops. The instances are available at `http://math.tkk.fi/~lehame/dju`.

In order to be able to compare the proposed solution methods, we focus on problems which can be solved by all three models (conditional, history independent and unconditional), even though the approximate methods could be used to solve larger instances in resonable computation times.

6.1.1. Real-life instances

The tram schedules of Helsinki Region Transport available at (Helsinki Region Traffic, 2010b). In each instance, the origin and destination nodes v_o and v_d are chosen randomly from the set of 154 tram stops. The departure time is set equal to 9:00 and the length of the time horizon is defined by $T = 1.2d(v_o, v_d)$, where $d(v_o, v_d)$ is the expected duration of the shortest path from v_o to v_d in the tram network. The expected duration of the shortest path is computed by assuming that each edge between adjacent nodes in the tram network (see Figure 7) is associated with a positive duration, extracted from the timetables.

Each service k operating within the time horizon is determined by a *scheduled route*, that is, a sequence $((v_0^k, t_0^k), \ldots, (v_p^k, t_p^k))$ of nodes, where t_i^k is the mean arrival time at node v_i^k for $i = 0, \ldots, p$. The tram network consisting of ten tram lines, operated by 50 trams (=services), is shown in Figure 7. The maximum walking distance equals $d_w^{\max} = 0.25$, that is, all pairs of nodes, the distance between which is less than 0.25 km, are connected by walking legs. The speed of each walking leg equals $v_w = 5$ km/h and the duration of walking legs is modeled similarly as the duration of service legs, see Equation (17).

By means of the above procedure, ten instances marked with 'r' were generated by converting the deterministic travel times to gamma distributed stochastic travel times, as described in the beginning of this section. In each case, the legs of the tram services, for which the start and end time is within the time horizon with strictly positive probability, were included. In addition, all

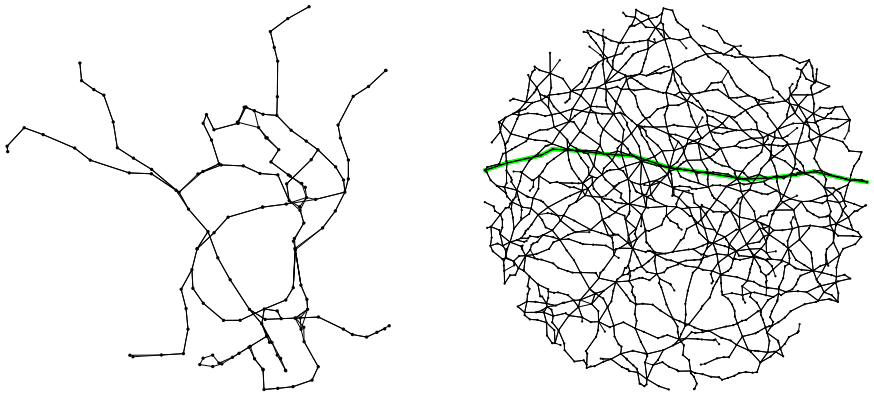


Figure 7: The real-life and randomized networks. The left figure shows the tram network of Helsinki consisting of ten tram lines and 154 stops. The right figure shows a randomized network consisting of 100 tram lines, one subway line (highlighted) and 1511 stops.

legs i for which $P(\tau'_i + \tau(v'_i, v_d) \leq T) = 0$, where $\tau(v'_i, v_d) \sim \text{Gamma}(\alpha d(v'_i, v_d), \beta, \delta d(v'_i, v_d))$ is the random duration of the shortest path from v'_i to v_d, were excluded.

6.1.2. Randomized instances

The randomized instances are divided into two sets: in the first set marked with 'd', the network consists of 100 lines operated by 500 services. In the second set marked with 'dm', there is an additional subway line operated by ten services. The set 'd' represents an extended tram network. The subway line is included in the set 'dm' due to the fact that in large urban areas, long journeys benefit from the multimodal nature of the network, resulting in relatively small travel times for long journeys.

First, n nodes are distributed randomly and uniformly in a disk in $\mathbb{R}^2$ with radius r (unit = 1 km). Then, the edges (roads) between the nodes are generated by Delaunay triangulation. Then, in order to remove the long edges on the border of the triangulation, the network is cropped by including the nodes within radius $0.9r$ from the center of the disk and the edges between them. The radius r of the disk is chosen such that the expected Delaunay edge length $E(L)$ is equal to the average distance (0.46 km) between adjacent stops in the tram network (Section 6.1.1). Since $E(L)$ is given by $E(L) \approx 1.132\lambda^{-\frac{1}{2}}$, where $\lambda = n/\pi r^2$ is the spatial density of nodes (Ritzerveld, 2007), we have $r = \sqrt{\frac{n}{\pi}} E(L)/1.132$.

The number of tram services in each instance is 500 and the services are divided equally among 100 lines. The lines are defined as a cycles in the Delaunay graph as follows.

1. The start node v_0^k of line k is chosen randomly from the set of all nodes.
2. The end node v_p^k of line k is chosen randomly from the set of nodes that define the convex hull of the network.
3. The shortest path $v_0^k, v_1^k, \ldots, v_{p-1}^k, v_p^k$ from the start node v_0^k to the end node v_p^k is determined. Line k is defined by the cycle $v_0^k, v_1^k, \ldots, v_{p-1}^k, v_p^k, v_{p-1}^k \ldots, v_1^k$, that is, a round-trip path between v_0^k and v_p^k.

Each line k is operated by five services h, whose starting points $v_0^{k,h}$ are chosen at equal intervals on the cycle that defines line k. Each service h operating on line k is determined by a scheduled route,

that is, a sequence $(v_0^{k,h}, t_0^{k,h}), \ldots, (v_q^{k,h}, t_q^{k,h})$ of nodes and scheduled arrival times, where subsequent nodes $v_i^{k,h}, v_{i+1}^{k,h}$ are subsequent nodes of the cycle. The scheduled arrival time at node $v_0^{k,h}$ equals $t_0^{k,h} = 0$ and the scheduled arrival time at node $v_i^{k,h}$ is defined by $t_i^{k,h} = t_{i-1}^{k,h} + \|v_{i-1}^{k,h} - v_i^{k,h}\|/\nu$ for $i = 1, \ldots, q$, where $\nu = 17$ km/h.

In the subway instances, the subway line and services are generated similarly as the tram lines and services except that the start and end nodes of the subway line are chosen from the opposite sides of the convex hull of the network (see Figure 7). In addition, 2/3 of the nodes in the shortest path are removed due to the fact that the average distance between adjacent subway stops is approximately three times the distance between adjacent tram stops (Helsinki Region Traffic, 2010b). Furthermore, the speed of the subway services equals $\nu_m = 40$ km/h.

Similarly as in the real-life instances, all pairs of nodes, the distance between which is less than 0.25 km, are connected by walking legs.

The origin and destination nodes v_o and v_d are chosen randomly from the union of the nodes included in the tram (and subway) lines. The length of the time horizon is defined by $T = 1.2 \cdot d(v_o, v_d)$, where $d(v_o, v_d)$ is the expected duration of the shortest path from v_o to v_d in the network. The expected duration of the shortest path is determined by assuming that each edge (v_i, v_j) in the randomized network (see Figure 7) is associated with a positive duration ($\|v_i - v_j\|/\nu$ for tram edges, $\|v_i - v_j\|/\nu_m$ for subway edges).

Since not all nodes in the initial Delaunay triangulation are included in any of the tram or subway lines in the above procedure, an initial set of tests were conducted in order to determine the number n of nodes such that the number of nodes included in the tram and subway lines was approximately 1500. As a result of these tests, the initial number of nodes was set to $n = 2800$.

20 instances were generated by means of the above procedure by including in each case the legs of the services for which the end time is within the time horizon with strictly positive probability. In addition, all legs i for which $P(\tau_i' + \tau(v_i', v_d) \leq T) = 0$, where $\tau(v_i', v_d) \sim \text{Gamma}(\alpha d(v_i', v_d), \beta, \delta d(v_i', v_d))$ is the random duration of the shortest path from v_i' to v_d, were excluded.

6.1.3. Results

Table 1: Computational results of real-life instances, 99.9% screening. The decision time for each model is given by the sum of screening time and algorithm time.

Instance	Number of legs	Time horizon T (min)	Screening time (s)	Conditional model		History independent model		Unconditional model	
				Algorithm time (s)	Probability $V((1))$	Algorithm time (s)	Probability $V_H(1)$	Algorithm time (s)	Probability $V_U(1)$
r1	745	20.4	455.29	12.42	0.989	0.96	0.989	0.54	0.989
r2	559	18.	474.83	62.67	0.693	5.68	0.693	3.99	0.693
r3	90	9.6	36.33	4.15	0.482	0.19	0.5	0.14	0.504
r4	355	15.6	1419.24	898.54	0.839	2.64	0.855	2.08	0.859
r5	558	24.	234.2	5.94	0.181	0.49	0.202	0.28	0.203
r6	704	22.8	1646.43	98.31	0.677	3.8	0.678	3.04	0.678
r7	615	19.2	1642.85	30.05	0.555	3.27	0.59	2.66	0.602
r8	562	21.6	1028.88	20.94	0.295	1.42	0.445	1.08	0.455
r9	508	15.6	787.51	883.16	0.588	1.84	0.669	1.43	0.677
r10	85	10.8	37.57	4.64	0.311	0.59	0.312	0.49	0.312

The computational results of the real-life instances are shown in Tables 1, 2, 3, and the results of the randomized instances are shown in Tables 4, 5. In all tables, the first column shows the name of the instance, the second column shows the number of legs included the instance and the third column shows the length of the time window in minutes.

In order to be able to compute the probability in the conditional model, which requires the enumeration of all paths, the set of paths from the origin state to destination states was initially

Table 2: Computational results of real-life instances, 90% screening. The decision time for each model is given by the sum of screening time and algorithm time.

Instance	Number of legs	Time horizon T (min)	Screening time (s)	Conditional model		History independent model		Unconditional model	
				Algorithm time (s)	Probability $V((1))$	Algorithm time (s)	Probability $V_H(1)$	Algorithm time (s)	Probability $V_U(1)$
r1	745	20.4	1.41	8.53	0.989	0.72	0.989	0.43	0.989
r2	559	18.	1.03	9.52	0.692	0.64	0.693	0.53	0.693
r3	90	9.6	0.13	1.3	0.473	0.1	0.487	0.08	0.487
r4	355	15.6	2.59	51.79	0.835	1.07	0.853	0.69	0.856
r5	558	24.	0.77	2.88	0.178	0.36	0.185	0.13	0.185
r6	704	22.8	2.9	14.16	0.676	1.1	0.676	0.8	0.676
r7	615	19.2	2.91	4.98	0.555	0.57	0.578	0.42	0.591
r8	562	21.6	1.68	4.58	0.301	0.4	0.41	0.22	0.419
r9	508	15.6	2.05	65.03	0.588	1.5	0.669	1.16	0.677
r10	85	10.8	0.17	2.98	0.31	0.34	0.312	0.3	0.312

Table 3: Computational results of real-life instances, heuristic screening. The decision time for each model is given by the sum of screening time and algorithm time.

Instance	Number of legs	Time horizon T (min)	Screening time (s)	Conditional model		History independent model		Unconditional model	
				Algorithm time (s)	Probability $V((1))$	Algorithm time (s)	Probability $V_H(1)$	Algorithm time (s)	Probability $V_U(1)$
r1	745	20.4	1.14	6.53	0.988	0.62	0.989	0.35	0.989
r2	559	18.	0.48	8.59	0.686	0.62	0.693	0.54	0.693
r3	90	9.6	0.11	2.09	0.473	0.15	0.498	0.12	0.498
r4	355	15.6	1.46	33.06	0.834	0.84	0.853	0.56	0.854
r5	558	24.	0.38	1.18	0.162	0.2	0.173	0.04	0.173
r6	704	22.8	1.06	10.28	0.676	0.83	0.676	0.58	0.676
r7	615	19.2	0.53	4.14	0.548	0.39	0.577	0.27	0.589
r8	562	21.6	0.9	4.7	0.295	0.42	0.41	0.22	0.419
r9	508	15.6	1.07	18.88	0.586	0.66	0.641	0.47	0.643
r10	85	10.8	0.09	2.19	0.31	0.34	0.312	0.3	0.312

narrowed by excluding paths for which the probability of success is less than a minimum probability p with confidence level $1 - p$. The duration of this screening phase (in seconds) is shown in the third column of the tables.

The remaining columns show the algorithm times for the different models and the corresponding probability values for the origin state. The *decision time* for each model is given by the sum of screening time and algorithm time. The decision time corresponds to the time needed to produce an optimal policy in each instance, that is, the time needed to suggest a decision for the commuter.

Table 1 shows the results for the ten real-life instances described in Section 6.1.1. In the screening phase, paths for which the probability of success is less than 0.1% with a 99.9% confidence level were excluded. This screening phase was executed by performing $R = 6905$ random tests, in each of which all paths beginning from the origin state were enumerated (see Algorithms 2 and 3). For all states not excluded during the screening phase, the distributions of the start and end times in the conditional model were approximated by random sampling with $R = 10000$ samples.

Table 2 shows the corresponding results obtained by excluding paths for which the probability of success is less than 10% with a 90% confidence level. In this case, the screening phase involved $R = 22$ random tests and the conditional distributions were approximated by $R = 1000$ samples.

Table 3 shows the results obtained by using the 90% accuracy described above and a heuristic screening function: All legs $((v_i, \tau_i), (v'_i, \tau'_i))$, for which the shortest distance $d(v'_i, v_d)$ from the end node v'_i to the destination node v_d is greater than the shortest distance $d(v_i, v_d)$ from the start node v_i to the destination, were excluded from the search.

The corresponding results of the randomized instances are shown in Tables 4 and 5. Due to the size of the randomized networks, the 99.9% screening phase could not be executed in reasonable computing times, and thus only results with 90% confidence level are reported.

By looking at the screening times in Tables 1 and 2, we see that the time required for obtaining a 99.9% confidence level requires typically 200 . . . 600 times as much computational effort as obtaining a 90% confidence level. This is justified by the fact that the number of random tests re-

Table 4: Computational results of randomized instances, 90% screening. The instances marked with 'd' correspond to the extended tram network and the instances marked with 'dm' include a subway line. The decision time for each model is given by the sum of screening time and algorithm time.

Instance	Number of legs	Time horizon T (min)	Screening time (s)	Conditional model		History independent model		Unconditional model	
				Algorithm time (s)	Probability $V((1))$	Algorithm time (s)	Probability $V_H(1)$	Algorithm time (s)	Probability $V_U(1)$
d1	4152	31.7	580.9	5.24	0.064	0.43	0.125	0.13	0.125
d2	4926	30.8	159.7	34.58	0.553	2.06	0.553	1.73	0.553
d3	362	15.3	1.9	355.32	0.892	1.64	0.899	1.36	0.92
d4	1309	21.4	76.5	61.5	0.362	2.27	0.407	1.59	0.407
d5	5927	39.1	5308.4	7.22	0.539	0.61	0.539	0.1	0.539
d6	2455	28.7	729.6	54.19	0.135	1.13	0.179	0.58	0.185
d7	5283	35.6	1922.2	59.11	0.565	3.9	0.596	2.96	0.609
d8	1874	21.	445.5	3.37	0.364	1.05	0.769	0.76	0.769
d9	4437	38.4	381.	9.94	0.337	1.22	0.433	0.44	0.433
d10	4896	35.	10974.5	55.12	0.605	2.59	0.639	1.82	0.639
dm1	171	9.4	0.1	2.3	0.615	0.26	0.632	0.12	0.667
dm2	3119	29.5	611.9	11.22	0.468	1.76	0.61	1.34	0.61
dm3	1576	23.3	21.6	7.98	0.886	0.79	0.894	0.45	0.894
dm4	3719	25.7	562.8	34.07	0.425	3.17	0.505	2.37	0.572
dm5	1264	19.1	6.7	1.95	0.826	0.31	0.83	0.07	0.83
dm6	2195	21.4	3.9	1.8	0.102	0.28	0.125	0.07	0.125
dm7	1683	17.5	60.4	3.47	0.4	1.26	0.4	0.92	0.4
dm8	1526	18.7	4.	7.11	0.829	0.92	0.838	0.62	0.838
dm9	656	17.1	0.9	1.46	0.356	0.22	0.358	0.06	0.358
dm10	622	14.1	1.8	4.69	0.699	1.1	0.703	0.81	0.703

Table 5: Computational results of randomized instances, heuristic screening. The instances marked with 'd' correspond to the extended tram network and the instances marked with 'dm' include a subway line. The decision time for each model is given by the sum of screening time and algorithm time.

Instance	Number of legs	Time horizon T (min)	Screening time (s)	Conditional model		History independent model		Unconditional model	
				Algorithm time (s)	Probability $V((1))$	Algorithm time (s)	Probability $V_H(1)$	Algorithm time (s)	Probability $V_U(1)$
d1	4152	31.7	0.87	2.6	0.064	0.45	0.125	0.13	0.125
d2	4926	30.8	0.6	4.62	0.553	0.99	0.553	0.59	0.553
d3	362	15.3	0.14	3.25	0.866	0.68	0.875	0.5	0.913
d4	1309	21.4	0.33	8.54	0.083	0.54	0.208	0.26	0.22
d5	5927	39.1	0.69	7.61	0.525	0.62	0.539	0.1	0.539
d6	2455	28.7	2.34	14.1	0.116	0.8	0.162	0.36	0.162
d7	5283	35.6	2.96	34.6	0.557	2.76	0.595	2.05	0.609
d8	1874	21.	3.15	3.4	0.363	1.06	0.769	0.75	0.769
d9	4437	38.4	3.7	12.09	0.337	1.22	0.433	0.42	0.433
d10	4896	35.	4.51	20.28	0.597	2.45	0.639	1.62	0.639
dm1	171	9.4	0.07	2.36	0.615	0.27	0.632	0.12	0.667
dm2	3119	29.5	2.4	11.98	0.443	3.74	0.61	2.86	0.61
dm3	1576	23.3	0.86	1.92	0.867	0.3	0.87	0.08	0.87
dm4	3719	25.7	3.63	12.59	0.425	2.53	0.505	1.93	0.572
dm5	1264	19.1	0.68	1.99	0.82	0.31	0.83	0.07	0.83
dm6	2195	21.4	0.37	1.86	0.099	0.28	0.125	0.07	0.125
dm7	1683	17.5	0.44	3.53	0.4	1.24	0.4	0.94	0.4
dm8	1526	18.7	0.21	1.46	0.826	0.22	0.833	0.06	0.833
dm9	656	17.1	0.11	1.48	0.33	0.22	0.358	0.06	0.358
dm10	622	14.1	0.36	3.	0.688	0.43	0.703	0.13	0.703

quired are 6905 and 22, respectively. The screening time is significantly greater in the randomized instances, as can be seen by looking at Table 4. Even obtaining the 90% confidence level requires three hours screening time in the worst case (instance d10). However, the screening time can be decreased by using a heuristic function (Tables 3,5). In some cases, for example dm1, the heuristic screening is relatively accurate compared to 90% screening. However, there are also cases in which the heuristic screening produces a significant error (for example d4).

By looking at the algorithm times in the tables, we see that the unconditional and history independent algorithms are significantly, typically of order ten times, faster than the conditional model. Furthermore, in a part of the instances, the difference between the results obtained by the three models is insignificant. For example, in instance r6, the probability of reaching the destination state is equal with three-digit precision for all studied models (Tables 1, 2). However, the results also indicate that the modeling error in the history independent and unconditional models is significant in some cases. For example, in instance d8, the conditional probability is 0.364 and the probability obtained by the history independent and unconditional models is 0.769. The dif-

ference between the history independent model and the unconditional model is relatively small in all cases.

As a conclusion, we state that even though the algorithm time of the conditional model is significantly greater compared to the history independent and unconditional models, the conditional model is justified by the fact that it may be used to estimate the error in approximate methods. By heuristic screening, the decision time (screening time + algorithm time) in the studied instances is half a minute at most in the conditional model, and a few seconds at most in the other models.

6.2. Regular networks

The purpose of the following experiments is to compare the different models with respect to three parameters, namely, transfer margin, number of transfer options at each node and the number of transfers. While these experiments do not represent any particular real-life scenario, the results give us an idea of the main differences of the proposed models. Results on more realistic cases are presented in Section 6.1.

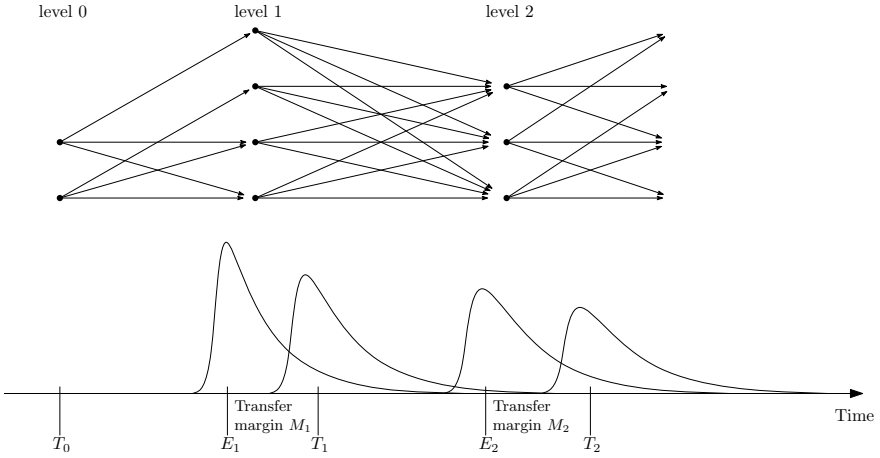


Figure 8: An example of a journey involving two transfers ($N = 2$). In this example there are two nodes at level 0, four nodes at level 1 and three nodes at level 2. For each node h at level $i \in \{0, 1, 2\}$ there are $K = 3$ services starting at h and ending at a node at level $i + 1$. The expected start time for each service starting at level i is denoted by T_i and the expected end time for each service ending at level $i + 1$ is given by $E_{i+1} = T_i + 1$. The expected start time for each service starting at level $i + 1$ is defined by $T_{i+1} = E_{i+1} + M_{i+1}$, where M_{i+1} is the studied transfer margin at level $i + 1$.

We consider the following class of scheduled networks: There are m nodes arranged in $N + 1$ *levels* numbered by $0, 1, \ldots, N$, (see Figure 8). The commuter departs at level 0 and the goal is to successfully transfer between two services at each level $\{1, \ldots, N\}$. For each node h at level $i \in \{0, \ldots, N - 1\}$, there are K services, each consisting of a single leg, which starts at node h at level i and ends at a node at level $i + 1$. The mean start time of each service starting at level i is denoted by T_i and the mean end time of each service ending at level i is denoted by E_i. The mean start time at level 0 satisfies $T_0 = 0$. The mean end time E_i satisfies $E_i = T_{i-1} + 1$ for all $i \in \{1, \ldots, N\}$. The mean start time T_i is defined by $E_i + M_i$ for all $i \in \{1, \ldots, N\}$, where M_i is the *transfer margin* at level i.

The start times of services starting at level i are independently and identically distributed according to the distribution Gamma$(\alpha T_i, \beta, \sigma T_i)$. In the conditional model, the arrival time τ'_k of

each service k is defined by $\tau'_k = \tau_k + D_k$, where τ_k is the start time of service k and D_k is a random variable defined by $D_k \sim \text{Gamma}(\alpha, \beta, \delta)$. Note that the realization of τ_k affects the distribution of τ'_k.

In the history independent and unconditional models, the start and end times of a leg are assumed to be independent and thus the end time τ'_k of transport service k ending at level i is defined by $\tau'_k \sim \text{Gamma}(\alpha E_i, \beta, \delta E_i)$. Note that this coincides with the prior distribution of $\tau_i^{k'}$ in the conditional model.

Table 6: A summary of performed experiments.

Experiment number	Number of transfers N	Transfer margin M ($\cdot\sigma$)	Number of transfer options K	Studied parameter	Difference between models	Figure number
1	2	$-2\ldots,3$	1, 3	Transfer margin M	Greatest with $-1 < M < 2$	9
2	2	0.5, 1.5	$1,\ldots,10$	Number of transfer options K	Increases with K	10
3	$1,\ldots,10$	1.5	1, 2, 3	Number of transfers N	Increases with N	11

A summary of performed experiments and the corresponding results is shown in Table 6.

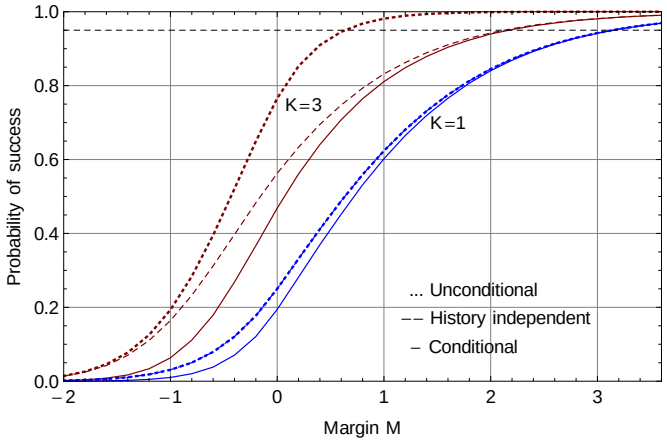


Figure 9: Experiment 1: Comparison of three probability models as a function of the transfer margin. The solid, dotted and dashed lines represent the probability of reaching the destination in the conditional model, history independent model and the unconditional model, respectively, with $N = 2$ and $K = 1, 3$. The history independent and unconditional models coincide when $K = 1$. With $K = 3$, the difference between the unconditional model and the conditional model is significant. The approximate methods become accurate with tight and loose transfer margins and the difference is greatest with $-1 < M < 2$.

6.2.1. Experiments

In the following three experiments we study the differences between the results obtained by the conditional, history independent and unconditional models.

Experiment 1

Let us first study the differences between the three models defined by Equations (3), (14) and (16). At first we compare the models as a function of transfer margin. We study the transfer

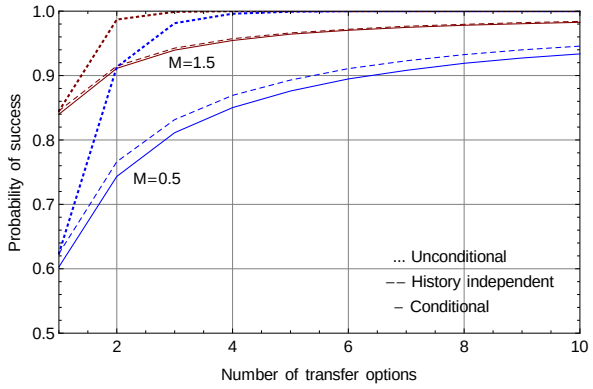


Figure 10: Experiment 2: Comparison of three probability models as a function of the number of transfer options K. The solid, dashed and dotted lines represent the probability of reaching the destination in the conditional model, history independent model and the unconditional model, respectively, with $N = 2$ and $M = 0.5, 1.5$. The unconditional model becomes less accurate when the number of transfer options is increased. With $M = 1.5$, the difference between the history independent model and the conditional model is relatively small and with $M = 0.5$, the difference is greater but does not increase significantly with K.

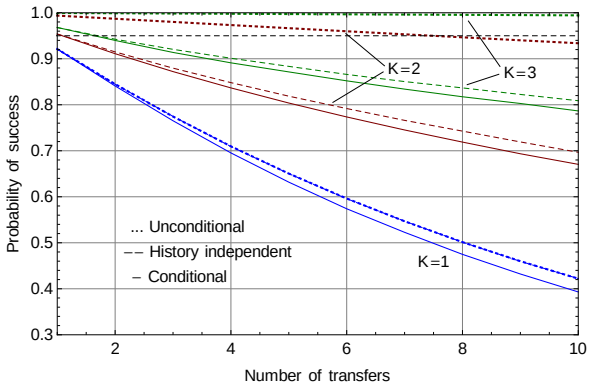


Figure 11: Experiment 3: Comparison of three probability models as a function of the number of transfers N. The solid, dashed and dotted lines represent the probability of reaching the destination in the conditional model, history independent model and the unconditional model, respectively, with $M = 1.5$ and $K = 1, 2, 3$. The difference between the unconditional model and the conditional model increases with N. The history independent model remains relatively accurate even if the number of transfers is increased.

margin as proportional to standard deviation: at each level i, the transfer margin M_i is defined by $M_i = \sigma_i M = \sqrt{E_i}\beta M$, where E_i is the expeced arrival time at level i and M is the ratio of transfer margin M_i to standard deviation σ_i.

The solid, dashed and dotted lines in Figure 9 represent the probability of a successful journey in the conditional model, history independent model and the unconditional model, respectively, with $N = 2$ and $K = 1, 3$.

As mentioned in the end of Section 5.4, we see that the history independent and unconditional models coincide when $K = 1$. With $K = 3$, the difference between the unconditional model and the conditional model is significant, whereas the history independent model is relatively accurate in this case. Note that the approximate methods become accurate with tight and loose transfer margins and that the difference is greatest with $-1 < M < 2$.

Experiment 2

Let us then compare the three models as a function of the number of transfer options K. The solid, dashed and dotted lines in Figure 10 represent the probability of a successful journey in the conditional model, history independent model and the unconditional model, respectively, with $N = 2$ and $M = 0.5, 1.5$.

By looking at the figure it is clear that the unconditional model becomes less accurate when the number of transfer options is increased. With $M = 1.5$, the difference between the history independent model and the conditional model is relatively small and with $M = 0.5$, the difference is greater but does not increase significantly with K.

Experiment 3

Finally, we examine the difference between the three models as a function of the number of transfers N. The solid, dashed and dotted lines in Figure 11 represent the probability of a successful journey in the conditional model, history independent model and the unconditional model, respectively, with $M = 1.5$ and $K = 1, 2, 3$.

Clearly, the difference between the unconditional model and the conditional model increases with N. The history independent model remains relatively accurate even if the number of transfers is increased.

7. Conclusions

We consider a dynamic journey planning problem in scheduled transportation networks. Due to uncertainty in travel times, the arrival times of transport services at stops are defined as random variables. The problem is modeled as a Markov decision process in which the travel history is included in the definition of a state. We present an algorithm for producing an optimal policy for traveling from a given origin to a given destination, assuming that the remaining path to the destination can be modified at any time during the journey. We also show that the importance of being able to reconsider the remaining path to the destination is emphasized when the number of transfers is increased.

The history dependent solution is further approximated by assuming history independence and unconditionality of successful transfers between transport services. The different models are evaluated by means of analysis and numerical experiments. The results suggest that the proposed approximations may be useful for practical purposes due to their computational simplicity.

References

Androutsopoulos, K. N., Zografos, K. G., 2009. Solving the multi-criteria time-dependent routing and scheduling problem in a multimodal fixed scheduled network. European Journal of Operational Research 192, 18–28.

Azaron, A., Kianfar, F., 2003. Dynamic shortest path in stochastic dynamic networks: Ship routing problem. European Journal of Operational Research 144, 138–156.

Bander, J., White, C. C., 1991. A new route optimization algorithm for rapid decision support. Proc. Vehicle Navigat. Inf. Syst. Conf. Part 2 P-253, 709–728.

Bander, J. L., White III, C. C., 2002. A heuristic search approach for a nonstationary shortest path problem with terminal costs. Transportation Science 36, 218–230.

Bard, J. F., Bennett, J. E., 1991. Arc reduction and path preference in stochastic acyclic networks. Management Science 37 (2), 198–215.

Bellman, R., 1957. A markovian decision process. Journal of Mathematics and Mechanics (6).

Boyan, J., Mitzenmacher, M., 2001. Improved results for route planning in stochastic transportation networks. In: Proceedings of the Twelfth Annual Symposium on Discrete Algorithms. pp. 895–902.

Brownstone, D., Small, K. A., 2005. Valuing time and reliability: assessing the evidence from road pricing demonstrations. Transportation Research Part A 39, 279–293.

Brub, J., Potvin, J., Vaucher, J., 2006. Time-dependent shortest paths through fixed sequence of nodes: Application to a travel planning problem. Comput. Oper. Res. 33 (6), 1838–1856.

Cai, X., Kloks, T., Wong, C. K., 1997. Time-varying shortest path problems with constraints. Networks 29 (3), 141–149.

Chabini, I., 1998. Discrete dynamic shortest path problems in transportation applications. Transp. Res. Rec. 1645, 170–175.

Cheung, R. K., 1998. Iterative methods for dynamic stochastic shortest path problems. Naval Research Logistics 45, 769–789.

Cheung, R. K., Muralidharan, B., 2000. Dynamic routing for priority shipments in ltl service networks. Transportation Science 34, 86–98.

Chiang, Y. S., , Roberts, P. O., 1980. A note on transit time and reliability for regular-route trucking. Transport Research Part B 14s (1-2), 59–65.

Cooke, K. L., Halsey, E., 1966. The shortest route through a network with time-dependent intermodal transit times. J. Math. Anal. Appl. 14, 493–498.

Datar, M., Ranade, A., 2000. Commuting with delay prone buses. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 22–29.

Davies, C., Lingras, P., 2003. Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks. European Journal of Operational Research 144, 27–38.

Fan, Y., Kalaba, R., Moore, J., 2005a. Arriving on time. Journal of Optimization Theory and Applications 127 (3), 497–513.

Fan, Y., Kalaba, R., Moore, J., 2005b. Shortest paths in stochastic networks with correlated link costs. Computers and Mathematics with Applications 49, 1549–1564.

Ferris, M. C., Ruszczynski, A., 2000. Robust path choice in networks with failures. Networks 35, 181–194.

Fosgerau, M., Karlström, A., 2010. The value of reliability. Transportation Research Part B 44, 38–49.

Fu, L., Rillet, L., 1998. Expected shortest paths in dynamic and stochastic traffic networks. Transportation Research 32 (7), 499–516.

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

Hamacher, H., Ruzika, S., Tjandra, S., 2006. Algorithms for time-dependent bicriteria shortest path problems. Discrete Optim. 3 (3), 238–254.

Helsinki City Transport, Nov. 2011. Helsinki city transport annual reports.
URL `http://www.hel.fi/hki/hkl/en/About+HKL/Annual+Reports`

Helsinki Region Traffic, Oct. 2010a. Jouko neighbourhood routes and service routes.
URL `http://www.hsl.fi/EN/passengersguide/ServiceroutesandJoukoroutes/`

Helsinki Region Traffic, Oct 2010b. Kalkati.net XML representation of helsinki regional transport authority (HSL) timetable data.
URL `http://developer.reittiopas.fi/`

Horn, M., 2003. An extended model and procedural framework for planning multi-modal passenger journeys. Transp. Res. Part B 37 (7), 641–660.

Huang, R., Peng, Z. R., 2001. An integration of network data model and routing algorithms for online transit trip planning. In: Proc. 80th Annu. Meeting Transp. Res. Board. Washington, DC.

Huang, R., Peng, Z. R., 2002. Schedule-based path finding algorithms for online transit trip planning. In: Proc. 81st Annu. Meeting Transp. Res. Board. Washington, DC.

Jaillet, P., 1992. Shortest path problems with node failures. Networks 22, 589–605.

Jula, H., Dessouky, M., Ioannou, P. A., 2006. Truck route planning in nonstationary stochastic networks with time windows at customer locations. IEEE Transactions on Intelligent Transportation Systems 7 (1), 51–62.

Kamburowski, J., 1985. A note on the stochastic shortest route problem. Operations Research 33 (6), 696–698.

Kao, E. P. C., 1978. A preference order dynamic program for a stochastic traveling salesman problem. Operations Research 26 (6), 1033–1045.

Kenyon, A. S., Morton, D. P., 2003. Stochastic vehicle routing with random travel times. Transportation Science 37 (1), 69–82.

Kim, S., Lewis, M., White III, C., 2005a. Optimal vehicle routing with real-time traffic information. IEEE Transactions on Intelligent Transportation Systems 6 (2), 178–188.

Kim, S., Lewis, M., White III, C. C., 2005b. State space reduction for non-stationary stochastic shortest path problems with real-time traffic congestion information. IEEE Transactions on Intelligent Transportation Systems 6 (3), 273–284.

Kostreva, M., Wiecek, M., 1993. Time dependency in multiple objective dynamic programming. J. Math. Anal. Appl. 173 (1), 289–307.

Lam, T. C., Small, K. A., 2001. The value of time and reliability: measurement from a value pricing experiment. Transportation Research Part E: Logistics and Transportation Review 37 (2-3), 231–251.

Loui, R. P., 1983. Optimal paths in graphs with stochastic or multidimensional weights. Communications of the ACM

26, 670–676.
Meyer, C. D., 2000. Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia.
Miller-Hooks, E. D., Mahmassani, H. S., 2000. Least expected time paths in stochastic, time-varying transportation networks. Transportation Science 34, 198–215.
Mirchandani, P. B., Soroush, H., 1985. Optimal paths in probabilistic networks: A case with temporary preferences. Computers & Operations Research 12, 365–383.
Modesti, P., Siomachen, A., 1998. A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. Eur. J. Oper. Res. 111 (3), 495–508.
Mulley, C., Nelson, J. D., 2009. Research in Transportation Economics 25 (1), 39 – 45.
Murthy, I., Sarkar, S., 1997. Exact algorithms for the stochastic shortest path problem with a decreasing deadline utility function. European Journal of Operational Research 103, 209–229.
Nie, Y. M., Wu, X., 2009. Shortest path problem considering on-time arrival probability. Transportation Research Part B 43, 597–613.
Nikolova, E., Brand, M., Karger, D., 2006a. Optimal route planning under uncertainty. In: Proceedings of International Conference on Planning and Scheduling, Lake District, England.
Nikolova, E., Kelner, J., Brand, M., Mitzenmacher, M., 2006b. Stochastic shortest paths via quasi-convex maximization. In: Proceedings of European Symposium of Algorithms, Zurich, Switzerland. pp. 552–563.
Peer, S., Sharma, D. K., 2007. Finding the shortest path in stochastic networks. Computers and Mathematics with Applications 53, 729–740.
Peng, Z. R., Huang, R., 2000. Design and development of interactive trip planning for web-based transit information systems. Transp. Res. Part C 8 (1), 409–425.
Polychronopoulos, G. H., Tsitsiklis, J. N., 1996. Stochastic shortest path problems with recourse. Networks 27, 133–143.
Psaraftis, H. E., Tsitsiklis, J. N., 1993. Dynamic shortest paths in acyclic networks with markovian arc costs. Operations Research 41 (1), 91–101.
Ritzerveld, N. G. H., 2007. Doctoral Thesis, Leiden University.
Russell, R. A., Urban, T. L., 2007. Vehicle routing with soft time windows and erlang travel times. Journal of the Operational Research Society 59, 1220–1228.
Sigal, C. E., Pritsker, A. A. B., Solberg, J. J., 1980. The stochastic shortest route problem. Operations Research 28 (5), 1122–1129.
Small, K., Winston, C., Yan, J., 2005. Uncovering the distribution of motorists' preferences for travel time and reliability. Econometrica 73 (4), 1367–1382.
Tan, M. C., Tong, C. O., Wong, S. C., Xu, J., 2007. An algorithm for finding reasonable paths in transit networks. J. Adv. Transp. 41 (3), 285–305.
Thomas, B., White, C., 2007. The dynamic shortest path problem with anticipation. European Journal of Operational Research 176 (2), 836–854.
Tong, C. O., Richardson, A. J., 1984. A computer model for finding the timedependent minimum path in transit systems with fixed schedules. J. Adv. Transp. 18 (2), 145–161.
Tong, C. O., Wong, S. C., 1999. A stochastic transit assignment model using a dynamic schedule-based network. Transp. Res. Part B 33 (2), 107–121.
Waller, S., Ziliaskopoulos, A., 2002. On the online shortest path problem with limited arc cost dependencies. Networks 40 (4), 216–227.
Wong, S. C., Tong, C. O., 1998. Estimation of time-dependent origin-destination matrices for transit networks. Transp. Res. Part B 32 (1), 35–48.
Ziliaskopoulos, A., Mahmassani, H., 1993. Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. Transp. Res. Rec. 1408, 94–100.
Ziliaskopoulos, A., Wardell, W., 2000. An intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays. Eur. J. Oper. Res. 125 (3), 486–502.
Zografos, K. G., Androutsopoulos, K. N., 2008. Algorithms for itinerary planning in multimodal transportation networks. IEEE Transactions on Intelligent Transportation Systems 9 (1).

Publication VII

Lauri Häme, Harri Hakula, Saara Hyvönen. Dynamic journeying in scheduled networks. Submitted to *Under review for publication in IEEE Transactions on Intelligent Transportation Systems*, 16.1.2012 .

Routing by Ranking: A Link Analysis Method for the Constrained Dial-A-Ride Problem

Lauri Häme*,[1]

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Harri Hakula

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Abstract

Using a modified version of hyperlink-induced topic search (HITS), we characterize hubs as nodes in a directed graph with many out-links to other hubs and calculate a hub score for each node. Ranking the nodes by hub score gives guidance to a dynamic programming algorithm for efficiently finding feasible solutions to the dial-a-ride problem.

Key words: Dial-a-Ride Problem, Link Analysis, HITS, Topological Ordering

Introduction

Several web information retrieval (IR) methods have been developed for finding the most appropriate web pages corresponding to queries given to search engines. The most sophisticated methods, such as HITS [1], PageRank [2] and SALSA [3] in use today make use of the hyperlinked structure of the web, since the goodness of a web page and the position of the page with respect to other web pages seem to have a certain connection. For example, a web page may be considered good if there are many other web pages linking to that page. In other words, web pages are ranked by search engines not only by means of the content of the page, but also by exploiting information regarding the hyperlink-induced relationships between pages.

The bringing of hyperlinks to bear on the ordering of web pages has given rise to a mathematical analysis related to hyperlink-induced web IR methods, such as in [4–7], in which the behaviour of several IR methods is studied from the computational point of view.

In this work, we focus on the HITS (Hyperlink-Induced Topic Search) algorithm, which defines *authorities* (web pages with several inlinks) and *hubs* (several outlinks). The HITS thesis is that *good hubs point to good authorities and good authorities are pointed to by good hubs*. Based on this thesis, HITS assigns both a hub score and authority score to each web page [5]. In this paper, we use the term ”hub” similarly as in the above context and not in the context of travel dispatch centers.

We present an application of HITS on the dial-a-ride problem (DARP) [8–20], in which the goal is to construct a set of vehicle routes, serving a set of customers, satisfying the given time, capacity and precedence constraints. We examine the DARP as a *constraint satisfaction problem*, in which the goal is to find a set of m feasible vehicle routes that serve all customers, where m is the number of vehicles, or to prove that such a set of routes does not exist, as in [20]. In this reference, it is noted that an algorithm for checking the feasibility of a DARP instance has two main applications: 1) Determining the feasibility can be the first phase in an optimization algorithm in a *static* setting, where all trip requests are known, for example, one day in advance. 2) In *dynamic* services, a constraint satisfaction algorithm can be used for deciding whether to accept or reject incoming user requests. In contrast to most works related to vehicle routing and dial-a-ride problems, no specific cost function is considered. For approaches to vehicle routing and dial-a-ride problems with cost functions, we refer to [21–23] and [13, 19].

In the context of the dial-a-ride problem, we define links between nodes as feasible transitions with respect to the constraints of the problem: If node j can be visited after i, a link from i to j is formed. Thus, a good hub score of a pick-up or drop-off node i means that many nodes can be reached in time from i. Thus, in order to efficiently find feasible solutions to the dial-a-ride problem, we suggest that nodes with large hub scores should be visited first since there are many nodes that can be visited after such nodes.

This work is partially motivated by a dynamic demand-responsive transport (DRT) service currently being planned to operate in Helsinki. Helsinki Region Transport board has approved a plan under which the trial period of the service takes place from 2012 to 2014. Similarly as the current service routes [24], the new DRT service is designed to operate on a demand-responsive basis, that is, each trip is booked in advance and the vehicle routes are modified according to the trips. The main difference to existing services is that no pre-order times for trips are required and the trips can be booked ”on the fly” by means of an interactive user interface.

1. The Ranking Method

In the following sections we introduce a ranking method based on the HITS algorithm (Sections 1.1 and 1.2) and show

*Corresponding author

Email addresses: Lauri.Hame@tkk.fi (Lauri Häme), Harri.Hakula@tkk.fi (Harri Hakula)

[1]Tel.: +358 40 576 3585, Fax: +358 9 470 23016

how it can be used to solve the dial-a-ride problem (Section 1.3).

1.1. The HITS algorithm [1]

Given a web graph $G = (V, A)$ consisting of pages V and links A between pages, the authority and hub scores a_i and h_i are computed for each page i as follows. Letting (i, j) represent a link from page i to page j, given that each page has been assigned an initial authority score $a_i(0)$ and hub score $h_i(0)$, HITS successively refines these scores by computing

$$a_i(k) = \sum_{(j,i)\in A} h_j(k-1)$$
$$h_i(k) = \sum_{(i,j)\in A} a_j(k-1)$$

for $k \in \{1, 2, \ldots\}$. By using matrix notation, these equations can be written the form $a(k) = L^T h(k-1)$ and $h(k) = La(k-1)$, where $a(k)$ is the authority vector containing the authority scores of each of the pages at step k, $h(k)$ is the corresponding hub vector and L is the adjacency matrix of the graph with elements $L_{ij} = 1$ if $(i, j) \in A$ and $L = 0$ otherwise [5].

It has been shown in [4] that the authority and hub vectors describing the authority and hub scores of nodes of a given graph are given by the dominant eigenvectors of the matrices $L^T L$ and LL^T (or equivalently, dominant singular vectors of L), where L is the adjacency matrix of the graph.

1.2. Modified HITS

For constrained routing problems, we present a modified version of the HITS algorithm, in which only the hub scores are considered. More precisely, our thesis is that *good hubs point to good hubs*. This formulation is motivated by Theorem 1, which states that for a specific class of graphs, the hub score of a node i corresponds to the number of self-avoiding paths from i to a given destination node. When attempting to construct a path that visits all nodes, or to maximize the number of visited nodes, the modified HITS idea induces the following intuitive policy: Good hubs are visited first, since many nodes can be reached from good hubs.

The hub scores are calculated as follows. Let L denote the adjacency matrix of a directed graph G. Similarly as in the HITS algorithm, the hub vector containing the *hub scores* of nodes is first initialized, $h(0) = (1, 1, \ldots, 1)$ and the hub vector is successively updated by means of the power method

$$h'(k) = Lh(k-1). \tag{1}$$

Similarly as in the original HITS algorithm, the hub vector converges to a dominant eigenvector of L.

Theorem 1 characterizes the hub scores produced by the modified HITS algorithm for *sink graphs* defined as follows.

Definition. *Let $G = (V, A)$ be a directed acyclic graph and let $s \in V$ be a node such that $(s, i) \notin A$ for all $i \in V$. The graph $G_s = (V, A \cup (s, s))$ is called a* sink graph.

In other words, a sink graph is a directed acyclic graph (V, A) with the exception that one node $s \in V$ with zero outdegree is associated with a loop (s, s).

Theorem 1. *Let L denote the adjacency matrix of a sink graph $G_s = (V, A)$, where $V = \{1, \ldots, |V|\}$, let h_i denote the number of self-avoiding paths from i to s for $i \in V \setminus \{s\}$ and let $h_s = 1$. Then, $h = (h_1, \ldots, h_{|V|})^T$ is a unique dominant eigenvector of L.*

Proof. Since the adjacency matrix of a directed acyclic graph is an upper triangular matrix with zeros on the diagonal, the adjacency matrix of a sink graph is an upper triangular matrix with diagonal elements $L_{ii} = 0$ except for the sink node s, for which we have $L_{ss} = 1$. The eigenvalues of an upper triangular matrix are equal to the diagonal elements [25] and thus there exists a unique dominant eigenvalue 1. Let us show that $h = (h_1, \ldots, h_{|V|})^T$ is the corresponding eigenvector of L.

Since all paths in a directed acyclic graph are self-avoiding and by definition we have $h_s = 1$, the number of self-avoiding paths h_i from node i to the sink node s in the sink graph G_s satisfies $h_i = \sum_{j\in V} L_{ij} h_j$ for $i \in V \setminus \{s\}$ and h_s satisfies $h_s = 1 = L_{ss} h_s = \sum_{j\in V} L_{sj} h_j$. In matrix form, we have $h = Lh$ and thus h is the unique eigenvector corresponding to eigenvalue 1. □

Note that since $h_i \leq h_j$ for all $i, j \in V$ for which $L_{ij} = 1$, the vector h defines a *topological ordering* [26] of the nodes for which $h_i > 0$. Although Theorem 1 considers a special class of graphs[2], the result gives us an idea of the behaviour of the modified HITS method: There are many paths beginning from nodes with high hub scores.

In the following section, we show how the hub scores are used to give indicative guidance to a dynamic programming algorithm for the dial-a-ride problem.

1.3. The dial-a-ride problem

The dial-a-ride problem is defined as follows [20]. Let $G = (V, A)$ be a complete and directed graph with node set $V = \{0\} \cup P$, where node 0 represents the depot, and P represents the set of pick-up and drop-off nodes, where $(|P| = 2n)$. The set P is partitioned into sets P^+ (pick-up nodes) and P^- (drop-off nodes). Each arc $(i, j) \in A$ has a non-negative travel time T_{ij}. The travel times satisfy the triangle equation, that is, $T_{ij} + T_{jk} \geq T_{ik}$ for all $i, j, k \in R$. With each node $i \in V$ are associated a time window $[E_i, L_i]$, a service duration D_i and a load q_i, where $D_0 = 0$ and $q_0 = 0$. Let $H = \{1, \ldots, n\}$ be the set of customers and let $T^{\max}$ be the maximum ride time for any customer. With each customer i is associated a pickup node $i^+ \in P^+$, a delivery node $i^- \in P^-$ and a load $q_{i^+} = q_{i^-}$. The above parameters are illustrated in Figure 1.

Let $K = \{1, \ldots, m\}$ be the set of available vehicles, each with capacity Q. A *route* is a circuit over a set of nodes in P, starting and finishing at the depot 0. The goal is to construct m vehicle routes such that: (i) for every customer i, the pick-up node and the drop-off node are visited by the same route and the pick-up node is visited before the drop-off node; (ii) the load of the vehicles does not exceed the capacity Q at any time; (iii) the ride time of each customer is at most $T^{\max}$; (iv) the service at node i begins within the interval $[E_i, L_i]$.

[2] The calculation of self-avoiding paths in graphs with cycles is discussed in [27].

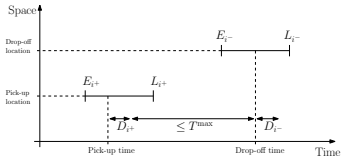


Figure 1: Time pick-up and drop-off time windows. The pick-up point of customer i is denoted by i^+ and the drop-off point is denoted by i^-. The customer should be picked up at i^+ within the time window $[E_{i^+}, L_{i^+}]$ and the customer should be dropped off at i^- within the time window $[E_{i^-}, L_{i^-}]$. The service times needed for the customer to get on the vehicle and get off the vehicle are denoted by D_{i^+} and D_{i^-}. The time between the drop-off and the pick-up (excluding D_{i^+}) should not exceed the maximum ride time $T^{\max}$.

1.3.1. Single-vehicle solution

In this section, we propose an exact constraint programming method for the single-vehicle DARP ($m = 1$) that produces a feasible solution whenever one exists or proves that the problem is infeasible. In the latter case, the algorithm returns a route that maximizes the number of served customers. The main challenge is to find a sequence of pick-up and drop-off nodes for which the time and precedence constraints are satisfied. While capacity constraints are also considered, the focus is on problems, in which time limits are more restrictive. The solution is extended to the multi-vehicle case in Section 1.4.

Letting node $0 \in V$ be the location of the vehicle at $t = 0$, we aim to find a feasible path $(0, p_1, \ldots, p_{2n}, 0)$ consisting of the pick-up and drop-off nodes of n customers. The number of permutations is $(2n)!$ and the number of feasible permutations with respect to precedence constraints is $(2n)!/2^n$ [17].

Briefly, our approach to the problem is a *depth-first* search, in which the remaining nodes are ranked by means of hub scores at each step and the order of the depth-first search is determined by the ranking. The search begins from node 0 by ranking the pick-up and drop-off nodes P of all customers. Then, the node p^* with the highest ranking is added to the sequence and the ranking procedure is repeated for the remaining nodes $P \setminus \{p^*\}$.

Formally, the search is executed by means of the recursion $\mathrm{REC}(S, R_S)$ shown in Algorithm 1, where S denotes a sequence of nodes (initially $S = (0)$), R_S denotes the set of remaining nodes at S (initially $R_S = \{1^+, 1^-, \ldots, n^+, n^-\}$) and $S_{\max}$ denotes the sequence that maximizes the number of served customers (initially $S_{\max} = (0)$). $C(S)$ denotes the number of served customers in sequence S, which is equal to the number of drop-off points in S.

The main idea of the recursion is that at each step, we have the current sequence S and the set of remaining nodes R_S (nodes that can possibly be added to the sequence). At first, we check the time, capacity and precedence constraints of S. If S is feasible and all customers have been served, that is, if $C(S) = n$, a feasible solution is found and the recursion is terminated. Otherwise, the remaining nodes R_S are ranked by hub scores and the infeasible nodes are removed from R_S (see Section 1.3.2). Then, the recursive function $\mathrm{REC}((S, i), R \setminus \{i\})$ is called for all sequences (S, i) in ranked order. In the following, we describe the ranking of remaining nodes in more detail.

```
Check time, capacity and precedence constraints;
if S is infeasible then
    Return;
end if
if C(S) > C(S_max) then
    S_max ← S;                          (Store the current sequence S.)
    if C(S) = n
     Terminate recursion;                (Feasible route.)
    end if
end if
Rank the remaining nodes i ∈ R_S and remove infeasible nodes from R_S;
(See Section 1.3.2.)
for all i ∈ R_S (in ranked order) do
    REC((S, i), R_S \ {i});
end for
```

Algorithm 1: A recursive solution $\mathrm{REC}(S, R_S)$ to the single-vehicle DARP. S denotes a sequence of nodes (initially $S = (0)$), R_S denotes the set of remaining nodes (initially $R_S = \{1^+, 1^-, \ldots, n^+, n^-\}$) and $S_{\max}$ denotes the sequence that maximizes the number of served customers (initially $S_{\max} = (0)$). $C(S)$ denotes the number of served customers in sequence S, which is equal to the number of drop-off points in S.

1.3.2. Ranking and pruning

The ranking of the remaining nodes R_S is determined in two phases: First, the adjacency matrix of the remaining nodes is determined by studying feasible transitions between the nodes. Then, the hub vector is calculated by means of Equation (1).

Adjacency matrix

Given that the vehicle is at the last node s of sequence S, we study which sequences (S, i, j), where $i, j \in R_S$, are feasible with respect to the time and precedence constraints. Let t_s denote the departure time and Q_s denote the load of the vehicle at s, and $t_i = t_s + T_{si}$ denote the arrival time at $i \in R_S$. For clarity, service times are assumed equal to zero. However, the modification of the following equations for non-zero service times is straightforward. In addition, we consider fixed time windows $[E_i, L_i]$ for each node. However, maximum ride time constraints [12] are taken into account by updating the upper bounds L_i of the time windows during run-time as described in [28].

First, all nodes that are seen to be non-reachable from S are removed from the set R_S of remaining nodes.

Definition. *A node $i \in R_S$ is said to be infeasible if there does not exist a feasible sequence $(S, \ldots, i)$ ending at i.*

Clearly, if the time of arrival t_i staisfies $t_i > L_i$, the node i is infeasible. The nodes $i \in I$ for which $t_i > L_i$ are removed from R_S, that is, $R_S \leftarrow R_S \setminus I$. This elimination criteria is similar to the one presented in [29].

For nodes that are not seen to be infeasible, we define feasible transitions as follows.

Definition. *The transition $i \rightarrow j$ from node $i \in R_S$ to node $j \in R_S$ is said to be* feasible*, if* $\max(t_i, E_i) + T_{ij} \leq L_j$.

The above definition is similar to the one studied in [30], in which the set of *admissible arcs* between the pick-up and delivery nodes is constructed as a preprocessing step in the pickup and delivery problem. The set of admissible arcs is made up of arcs which a priori satisfy the precedence, capacity and time constraints of the problem. The difference in our formulation is that we define the feasibility of transitions as a function of the state of the vehicle (see Figure 2).

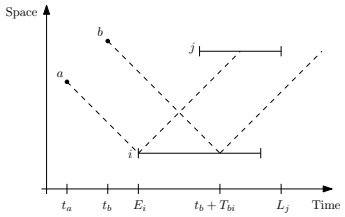


Figure 2: Feasible transitions. The figure shows the locations and time windows of two nodes i and j. If the vehicle left from a at the instant t_a, visited node i and moved directly to node j, the time constraint L_j would be satisfied. However, assuming that at the instant t_b the vehicle is located at b, the transition $i \rightarrow j$ is seen to be infeasible.

The elements of the adjacency matrix are defined for all $i, j \in R_S$ by $L_{ij} = 1$, if $i \neq j$ and $i \rightarrow j$ is feasible, $L_{ij} = 0$ otherwise.

Link analysis

After the adjacency matrix L has been determined, the hub vector $h = (h_1, \ldots, h_{|R_S|})$ of L is determined by Equation (1) (here the remaining nodes $i \in R_S$ are numbered from 1 to $|R_S|$ for clarity). In our formulation, a large hub score of node i means that many nodes can be visited after i. The hub ranking method is defined as follows.

Definition (Hub ranking)**.** *The remaining nodes $i \in R_S$ are sorted in descending order of the hub scores h_i: The branches are evaluated in the order $i_1, \ldots, i_{|R_S|}$, where $h_{i_1} \geq h_{i_2} \geq \ldots \geq h_{i_{|R_S|}}$.*

The hub scores are used to give guidance to the algorithm regarding the order in which the depth-first search visits the nodes. Since the goal is to maximize the number of served customers, the sequences that can not improve the maximum number of served customers, can be discarded by studying the neighborhoods of the remaining nodes.

Definition. *The neighborhood of node $i \in R_S$ is defined by $N(i) = \{i\} \cup \{j \in R_S \mid L_{ij} = 1\}$.*

Suboptimal sequences are detected as follows.

Theorem 2. *Let $S_{\max}$ denote the best found solution. The sequence S is suboptimal if*

$$C(S) + \max_{i \in R_S} C(N(i)) \leq C(S_{\max}),$$

where $C(S), C(N(i))$ and $C(S_{\max})$ denote the number of drop-off points in $S, N(i)$ and $S_{\max}$, respectively.

Proof. Suppose there exists a feasible sequence $(S, i_1, \ldots, i_h)$ for which $C((i_1, \ldots, i_h)) > \max_{i \in R_S} C(N(i))$. Then, since $\{i_1, \ldots, i_h\} \subset N(i_1)$, we have $C(N(i_1)) \geq C((i_1, \ldots, i_h)) > \max_{i \in R_S} C(N(i))$, which is a contradiction. □

1.3.3. A heuristic extension

Algorithm 1 describes an exact procedure for maximizing the number of served customers in a single route. In order to find the exact solution, the algorithm goes through all branches until no further nodes can be added to the current sequence or the sequence is seen to be suboptimal.

The effort of the algorithm can be controlled by limiting the number of branches that are evaluated by means of a positive parameter J. For example, if $J = 1$, the algorithm constructs a single sequence and stops when the set of remaining nodes in empty. By increasing J the search space is expanded and if $J = (2n)!/2^n$ (the number of permutations that satisfy precedence constraints), the heuristic coincides with the exact algorithm.

1.4. Multi-vehicle solution

In this section, we propose an exact approach to the multi-vehicle DARP as a constraint satisfaction problem. By using Algorithm 1 as a subroutine, the method produces a feasible solution to any instance or proves that the instance is infeasible. The main idea is that the vehicle routes are constructed one by one, each maximizing the number of served customers in the set of remaining customers. Customers are denoted by numbers $1, \ldots, n$ and vehicles are denoted by numbers $1, \ldots, m$.

First, a route is constructed for vehicle 1, serving as many customers as possible. Then, the process is repeated with vehicle 2 for the set of customers that were not served by vehicle 1 and so forth (see Figure 3). Letting X denote a set of customers, we denote by $\mathrm{SV}(X)$ the single-vehicle algorithm (Algorithm 1) that returns a route that maximizes number of served customers in set X and the corresponding set of served customers. This set will be referred to as a maximum cluster of X, which is formally defined as follows.

Definition. *Let X be a set of customers and $M \subset X$ be a subset of X, for which there exists a feasible route serving all customers in M. If $|M| \geq |Y|$ for all sets $Y \subset X$ for which there exists a feasible route serving all customers in Y, then M is a* maximum cluster *of X.*

Note that if there exists a feasible route serving all customers in X, we have $M(X) = X$. The solution to the multi-vehicle case is outlined in Algorithm 2.

for all $k \in \{1, \ldots, m\}$ **do**
  $[S_k, X'_k] \leftarrow \mathrm{SV}(X_k)$; ($S_k$ = route, X'_k = set of served customers)
  $X_{k+1} \leftarrow X_{k+1} \cup (X_k \setminus X'_k)$, where $X_{m+1} = X_1$;
end for

Algorithm 2: Outline of the multi-vehicle algorithm. X_k denotes the set of customers assigned to vehicle k and $\mathrm{SV}(X_k)$ denotes the single-vehicle subroutine presented in Algorithm 1 that returns a maximum cluster X'_k of X_k and the corresponding route S_k. Initially, all customers are assigned to the first vehicle, that is, $X_1 = \{1, \ldots, n\}$ and $X_k = \emptyset$ for all $k \in \{2, \ldots, m\}$.

If a feasible solution is not found directly by using the procedure described in Algorithm 2, the process is repeated. The approach is to find a set of customer-vehicle assignments such that for each vehicle there exists a feasible route serving all customers assigned to the vehicle or to prove infeasibility by going through all possible sets of customer-vehicle assignments [28].

We note that since the number of possible partitions of n customers into m sets is equal to the Stirling number of the second kind, that is, $|\mathcal{P}| = S_2(n, m) = \frac{1}{m!} \sum_{i=0}^{m} (-1)^i \binom{m}{i} (m - i)^n$, going through all possible partitions is computationally taxing. However, some instances are rendered infeasible by studying the feasibility of routes involving two customers: If there exist no feasible routes involving customers $\{i, j\}$, an arc between i and j is formed (customers i and j can not be assigned to the

same vehicle). Then, we find the *maximum clique* C within the set of customers, that is, the largest set of customers for which there is an arc between all pairs of nodes $i, j \in C$, see [31]. If the size $|C|$ of the maximum clique is greater than the number m of vehicles, the instance is infeasible.

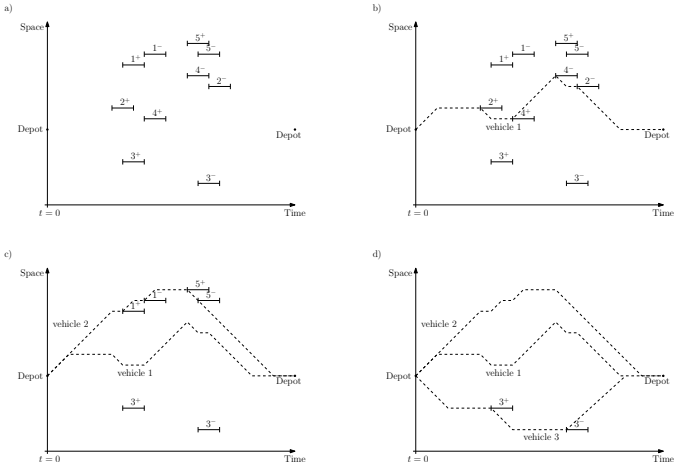


Figure 3: A one-dimensional example of the approach to the multi-vehicle problem involving five customers and three vehicles. The first route is constructed by maximizing the number of customers that can be served by a single vehicle (Figure b). Then, the customers 2 and 4 served by the first vehicle are removed from the set of remaining customers and the process is repeated for the second vehicle (Figure c). Finally, a route is construced for the third vehicle, serving the last remaining customer 3 (Figure d).

2. Structure and complexity

The structure of the single vehicle algorithm is as follows. Briefly described, each recursion step involves checking the the feasibility of sequences (S, i, j), where $i, j \in R_S$. Clearly, this process has a complexity of order $O(|R_S|^2)$, where $|R_S|$ is the number of remaining nodes. After the adjacency matrix L of the remaining nodes has been determined, the complexity of calculating k steps of the power method (Equation (1)) is of order $O(k|R_S|^2)$.

The overall complexity of finding a feasible solution in the single vehicle case is strongly dependent on the number of recursion steps needed to find the solution. Finding a feasible solution to a problem with n customers involves at least $2n$ recursion steps: At first, the number of remaining nodes is $2n$ and each time a feasible node is added to the end of the service sequence, the number of remaining nodes is decreased by one (see Figure 3). In the best case, no infeasible states are encountered and thus there are $2n$ recursion steps (one for each node added to the sequence). At step $k \in \{1, \ldots, 2n\}$, the number of remaining nodes is $2n+1-k$ and thus the complexity is of order $O(\sum_{k=1}^{2n}(2n+1-k)^2) = O(n^3)$. Roughly, the computational work is linearly increased with the number of dead ends M encountered during the recursion. The overall complexity of the single vehicle algorithm is thus $O(n^3 + M)$.

The multi-vehicle algorithm involves solving the single vehicle case for each vehicle in r iterations. Thus the complexity is bounded above by $O(rmn^3)$. Note that the complexity is in general lower since the size of the set of customers is n only the first vehicle. For the other vehicles, the number of customers is smaller since the customers that are already included in another vehicle route are not a part of the subproblem. For example, if the customers were divided equally among vehicles a priori, the complexity would be of order $O(rm(n/m)^3) = O(r(n^3/m^2))$.

3. Numerical experiments

In the following, we present extracts of the results of computational tests reported in [28]. An algorithm using the hub ranking idea is compared with two existing solution methods, namely, a tabu search algorithm [13] and a constraint programming (CP) algorithm [20].

The hub algorithm was implemented in Matlab and the tests were performed on a 2.2 GHz Dual Core Intel PC. The tabu and CP algorithms were tested on a 2.5 GHz Dual Core AMD Opteron computer [18].

In the studied instances the pick-up and drop-off points are located in a 20×20 square and the ride times between points (in minutes) are equal to Euclidean distances. The time windows have 15 minutes of length. In the first set of instances marked with a, the capacity of a vehicle equals $Q = 3$ and the load of each customer is 1. In the second set marked with b, we have $Q = 6$ and the load associated to each customer is chosen randomly according to a uniform distribution on the set $\{1, \ldots, Q\}$. In both sets, the maximum ride time is equal to $R = 22$ and the service time is proportional to the number of passengers, namely $d_i = d_{n+i} = q_i$. The instances are described in more detail in [16, 20, 32].

The results of the tests are shown in Table 3. The first column shows the instance labels of the form *am-n* or *bm-n*, where m indicates the number of vehicles and n corresponds to the number of customers. The other columns show the average time (in seconds) needed to solve the instances and the corresponding modifications by using the different solution methods, calculated over ten runs. The results obtained by the first two algorithms, tabu and CP, have been reported previously in [18, 20]. and the results for the hub algorithm have been reported in [28]. A number in parentheses indicates that the instance was proven to be infeasible, a dash indicates that a solution was not found in three minutes computing time and a star indicates that results for the instance have not been reported. The Matlab-implementations of the hub algorithm are available at `http://math.tkk.fi/~lehame/exact/`. The heuristic version of the algorithm with $J = 1$ was used (see Section 1.3.3).

By looking at the results obtained by the hub algorithm we see that most instances were solved within a fraction of a second. The CPU times obtained by the hub algorithm are typically of order ten times smaller compared to the results of the tabu and CP algorithms. Although the algorithms were tested on different platforms, the results seem to justify the efficiency of the hub algorithm on the test instances. We acknowledge that there are instances in wich tabu and CP produced a feasible solution or proved infeasibility faster than the hub algorithm. However, the results suggest that the multi-vehicle hub algorithm may have practical importance since it is capable of handling large problems in short computation times.

Instance	Tabu	CP	Hub	Instance	Tabu	CP	Hub
a4-40	0.5	0.3	0.05	b4-40	0.4	0.3	0.05
a4-48	1.3	0.4	0.05	b4-48	-	(0.1)	(0.32)
a5-40	0.3	0.3	0.02	b5-40	-	(0.1)	(0.27)
a5-50	0.6	0.6	0.03	b5-50	1.1	0.9	0.18
a5-60	1.5	0.9	0.05	b5-60	1.6	1.2	0.04
a6-48	0.4	0.7	0.03	b6-48	*	*	0.02
a6-60	-	(1.1)	(0.63)	b6-60	1.0	1.4	0.04
a6-72	-	(1.7)	(0.77)	b6-72	2.4	2.4	0.05
a7-56	-	(0.6)	(0.40)	b7-56	0.5	1.5	0.04
a7-70	1.4	7.6	0.06	b7-70	1.3	2.7	0.05
a7-84	3.0	4.1	0.08	b7-84	-	(0.1)	(0.64)
a8-64	-	(1.9)	(0.84)	b8-64	0.8	1.9	0.04
a8-80	1.8	6.0	0.07	b8-80	-	(0.5)	(1.28)
a8-96	-	(3.7)	(1.72)	b8-96	3.9	56.1	0.15

Table 1: Comparison between a tabu search algorithm [13], a constraint programming algorithm [18] and the hub algorithm. The results are given in [18, 20, 28]. The upper table shows the average time (in seconds) needed to solve the instance of the dial-a-ride problem in the first column by using the different solution methods, calculated over ten runs. The times without parentheses indicate that a feasible solution was found and the times in parentheses indicate that the instance was proven to be infeasible. A star (*) indicates that the computing time has not been reported.

4. Conclusions

In this work, an efficient dynamic programming method for the time constrained dial-a-ride problem is suggested. The purpose of the method is to produce routes that satisfy the constraints of the problem, since finding such routes is seen to be a major challenge in highly constrained problems. The main result is an exact constraint satisfaction algorithm for the multi-vehicle case.

Our approach is motivated by HITS, a link-analysis algorithm originally developed for web information retrieval. We define links between the pick-up and drop-off nodes as feasible transitions with respect to the constraints of the problem, *hubs* as nodes with several out-links. Ranking the nodes by hub score gives guidance to a depth-first search used to maximize the number of served customers in a single vehicle route. The method is extended to solve the multi-vehicle case. Numerical experiments suggest that the proposed ranking method is capable of producing feasible solutions to large problems in relatively small computation times.

This work is an example on how recommendation-type link analysis can be used to solve combinatorial problems. Although we have focused on the Dial-A-Ride Problem (DARP), we note that the DARP is a generalization of the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) and thus the proposed hub ranking method could be useful in solving other constrained routing problems as well.

References

[1] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: Proceedings of the Ninth Annual ACM-SIAM Symposium of Discrete Algorithms, ACM Press, New York, 1998, pp. 668–677.

[2] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the Seventh International Conference on World Wide Web, Vol. 7, 1998, pp. 107–117.

[3] R. Lempel, S. Moran, The stochastic approach for link-structure analysis (salsa) and the tkc effect, in: The 9th Intl. WWW Conference, 2000.

[4] A. Farahat, T. LoFaro, J. C. Miller, G. Rae, L. Ward, Authority rankings from hits, pagerank, and salsa: existence, uniqueness, and effect of initialization, SIAM Journal on Scientific Computing 27 (2006) 1181–1201.

[5] A. Langville, C. Meyer, A survey of eigenvector methods of web information retrieval, SIAM Review 47 (2005) 135–161.

[6] A. Y. Ng, A. X. Zheng, M. I. Jordan, Stable algorithms for link analysis, SIGIR01, New Orleans, Louisiana, USA.

[7] M. Agosti, L. Pretto, A theoretical study of a generalized version of kleinberg's hits algorithm, Information Retrieval 8 (2005) 219–243.

[8] H. Psaraftis, An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows, Transportation Science 17 (1983) 351–357.

[9] J. Jaw, A. Odoni, H. Psaraftis, N. Wilson, A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows, Transportation Research Part B 20 (1986) 243–257.

[10] O. Madsen, H. Ravn, J. Rygaard, A heuristic algorithm for the a dial-a-ride problem with time windows, multiple capacities, and multiple objectives, Annals of Operations Research 60 (1995) 193–208.

[11] P. Toth, D. Vigo, Heuristic algorithms for the handicapped persons transportation problem, Transportation Science 31 (1997) 60–71.

[12] B. Hunsaker, M. W. P. Savelsbergh, Efficient feasibility testing for dial-a-ride problems, Operations Research Letters 30 (2002) 169–173.

[13] J.-F. Cordeau, G. Laporte, A tabu search heuristic for the static multi-vehicle dial-a-ride problem, Transportation Research B 37 (2003a) 579–594.

[14] M. Diana, M. Dessouky, A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows, Transportation Research Part B 38 (2004) 539–557.

[15] K. Wong, M. Bell, Solution of the dial-a-ride problem with multi-dimensional capacity constraints, International Transactions in Operational Research 13 (2006) 195–208.

[16] J.-F. Cordeau, A branch-and-cut algorithm for the dial-a-ride problem, Operations Research 54 (2006) 573–586.

[17] L. Häme, An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows, European Journal of Operational Research. 209 (1) (2011) 11 – 22.

[18] G. Berbeglia, Complexity Analyses and Algorithms for Pickup and Delivery Problems, Ph.D. Thesis, HEC Montreal, 2009.

[19] G. Berbeglia, J.-F. Cordeau, G. Laporte, A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem, forthcoming (2011).

[20] G. Berbeglia, G. Pesant, L.-M. Rousseau, Checking feasibility of the dial-a-ride problem using constraint programming, forthcoming (2011).

[21] O. Bräysy, M. Gendreau, Vehicle routing problem with time windows, part i: Route construction and local search algorithms, Transportation Science 39 (1) (2005) 104118.

[22] O. Bräysy, M. Gendreau, Vehicle routing problem with time windows, part ii: Metaheuristics, Transportation Science 39 (1) (2005) 119139.

[23] C. Groër, B. Golden, E. Wasil, A library of local search heuristics for the vehicle routing problem, Mathematical Programming Computation 2 (2).

[24] Helsinki Region Traffic, Jouko neighbourhood routes and service routes (Oct. 2010).
URL `http://www.hsl.fi/EN/passengersguide/`

[25] S. Axler, Linear Algebra Done Right, Springer-Verlag, ISBN 0-387-98258-2, 1996.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms (2nd ed.), MIT Press and McGraw-Hill, 2001.

[27] J. Ponstein, Self-avoiding paths and the adjacency matrix of a graph 14 (3) (1966) 600–609.

[28] L. Häme, H. Hakula, A maximum cluster algorithm for checking the feasibility of dial-a-ride instances, manuscript submitted to Transportation Science.

[29] J. Desrosiers, Y. Dumas, F. Soumis, A dynamic programming solution of the large-scale singlevehicle dial-a-ride problem with time windows, American Journal of Mathematical and Management Sciences 6 (1986) 301–325.

[30] Y. Dumas, J. Desrosiers, F. Soumis, The pickup and delivery problem with time windows, European Journal of Operational Research 54 (1991) 7–22.

[31] D. R. Wood, An algorithm for finding a maximum clique in a graph, Operations Research Letters 21 (5) (1997) 211–217.

[32] S. Ropke, J.-F. Cordeau, G. Laporte, Models and branch-and-cut algorithm for pickup and delivery problems with time windows, Networks 49 (2007) 258–272.

Publication VIII

Lauri Häme, Harri Hakula. A Maximum Cluster Algorithm for Checking the Feasibility of Dial-A-Ride Instances. Submitted to *Under review for publication in Transportation Science*, 16.1.2012 .

TRANSPORTATION SCIENCE
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0041-1655 | EISSN 1526-5447 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

A Maximum Cluster Algorithm for Checking the Feasibility of Dial-A-Ride Instances

Lauri Häme
Aalto University School of Science, lauri.hame@tkk.fi,

Harri Hakula
Aalto University School of Science, harri.hakula@tkk.fi,

The Dial-A-Ride Problem (DARP) involves the dispatching of a fleet of vehicles in order to transport customers requesting service and is one of the most challenging tasks of combinatorial optimization. Berbeglia et al. (2011b) models the DARP as a constraint satisfaction problem, where the goal is to find a feasible solution with respect to the time, capacity and precedence constraints, or to prove infeasibility. The main contribution of our work is a new exact method for this problem formulation. The algorithm is based on a dynamic subroutine which finds for any set of customers a maximal set of customers that can be served by a single vehicle. The performance of the algorithm is analyzed and evaluated by means of computational experiments, justifying the efficiency of the solution method. In addition to the dial-a-ride problem, the algorithm has an important application in food delivery services, where each meal needs to be delivered within guaranteed time limits.

Key words: Dial-A-Ride Problem, Algorithms, Dynamic Programming

Introduction

The dial-a-ride problem (DARP) is a generalization of the vehicle routing problem (VRP) arising in contexts where passengers are transported, either in groups or individually, between specified origins and destinations, as defined by Cordeau et al. (2007a).

In most studies related to the dial-a-ride problem, the customers' trips are restricted by *time windows* for pick-up and drop-off, see for example Psaraftis (1983), Jaw et al. (1986), Madsen et al. (1995), Toth and Vigo (1997), Hunsaker and Savelsbergh (2002), Cordeau and Laporte (2003a), Diana and Dessouky (2004), Wong and Bell (2006), Cordeau (2006), Häme (2011), Berbeglia (2009), Berbeglia et al. (2011a,b). In these studies it is noted that in dynamic settings, time windows eliminate the possibility of indefinite deferment of customers and strict time limits help provide reliable service. In addition to time windows, the vehicle routes are restricted by *maximum ride time constraints* that limit the time spent by passengers on the vehicle between their pick-up and their drop-off, *capacity* and *precedence* constraints. For an exhaustive summary on the models and algorithms for the DARP, the reader is referred to (Cordeau and Laporte 2007c).

We examine the DARP as a *constraint satisfaction problem*, in which the goal is to find a set of m feasible vehicle routes that serve all customers, where m is the number of vehicles, or to prove that such a set of routes does not exist, as in (Berbeglia et al. 2011b). In this reference, it is noted that an algorithm for checking the feasibility of a DARP instance has two main applications: 1) Determining the feasibility can be the first phase in an optimization algorithm in a *static* setting, where all trip requests are known, for example, one day in advance. 2) In *dynamic* services, a constraint satisfaction algorithm can be used for deciding whether to accept or reject incoming user requests.

This work is partly motivated by the latter application, namely, a dynamic demand-responsive transport (DRT) service currently being planned to operate in Helsinki. The service, run by Helsinki Region Transport, The Finnish Transport Agency and Aalto University, will be deployed by the end of 2012. Similarly as the current service routes (Helsinki Region Traffic 2010), the new DRT service is designed to operate on a demand-responsive basis, that is, each trip is booked in advance and the vehicle routes are modified according to the trips. The main difference to existing services is that no pre-order times for trips are required and

the trips can be booked "on the fly" by means of an interactive user interface. A main goal is to provide high-quality service that could compete with private car traffic.

In addition to passenger transportation, we note that a time-constrained problem similar to the DARP arises in food delivery services. Instead of passengers with pick-up and drop-off time windows, the goal is to transport *meals* from restaurants to specified delivery points within guaranteed time limits. Similarly as in the dynamic DARP, the food delivery service provider needs to decide quickly whether to take an incoming order or not, if there is a guaranteed maximum delivery time, for example, one hour.

In both contexts (passenger transportation and food service), a good level of service means that the problem is highly constrained (Jokinen et al. 2011). In such problems, the number of feasible solutions becomes so limited that often any feasible solution will be relatively close to the optimal solution with respect to the objective function (Psaraftis 1983). Thus, we suggest that in highly constrained cases, solving the consraint satisfaction problem provides, if not the optimal solution, at least a good initial solution.

The main result of this work is an exact algorithm for the constraint satisfaction dial-a-ride problem introduced by Berbeglia et al. (2011b). In contrast to most works related to vehicle routing and dial-a-ride problems, no specific cost function is considered[1]. The algorithm stops as soon as a feasible solution is found regardeless of the quality of the solution. If a feasible solution is not found through exhaustive search, the algorithm proves that the problem instance is infeasible. We also show how some infeasible instances can be detected in advance by studying the feasiblility of routes involving two customers. The computational results obtained by the proposed solution method are compared to the results presented in (Berbeglia et al. 2011b). In addition, we solve three instances, for which results have not been previously reported. The experiments suggest that algorithm is capable of solving large instances efficiently.

This document is organized as follows. In Section 1, we formalize the multi-vehicle DARP with time windows. In Section 2.1, we present a dynamic programming algorithm which produces for any set of customers the maximal set of customers that can be served by a single vehicle. The extension of the algorithm to the multi-vehicle case is discussed in Section 2.2. This is followed by an analysis of the structure and complexity of the solution method in Section 3 and computational experiments in Section 4.

1. Problem formulation

The dial-a-ride problem is defined as follows (Berbeglia et al. 2011b). Let $G = (V, A)$ be a complete and directed graph with node set $V = \{0\} \cup P$, where node 0 represents the depot, and P represents the set of pick-up and drop-off nodes, where ($|P| = 2n$). The set P is partitioned into sets P^+ (pick-up nodes) and P^- (drop-off nodes). Each arc $(i, j) \in A$ has a non-negative travel time T_{ij}. The travel times satisfy the triangle equation, that is, $T_{ij} + T_{jk} \geq T_{ik}$ for all $i, j, k \in V$. With each node $i \in V$ are associated a time window $[E_i, L_i]$, a service duration D_i and a load q_i, where $D_0 = 0$ and $q_0 = 0$. Let $H = \{1, \ldots, n\}$ be the set of customers and let $T^{\max}$ be the maximum ride time for any customer. With each customer i is associated a pickup node $i^+ \in P^+$, a delivery node $i^- \in P^-$ and a load $q_{i^+} = q_{i^-}$. The above parameters are illustrated in Figure 1.

Let $K = \{1, \ldots, m\}$ be the set of available vehicles, each with capacity Q. A *route* is a circuit over a set of nodes in P, starting and finishing at the depot 0. The DARP consists of constructing m vehicle routes (possibly empty) such that: (i) for every customer i the pick-up node and the drop-off node are visited by the same route and the pick-up node is visited before the drop-off node; (ii) the load of the vehicles does not exceed the capacity Q at any time; (iii) the ride time of each customer is at most $T^{\max}$; (iv) the service at node i begins within the interval $[E_i, L_i]$.

2. Problem solution

Our solution to the dial-a-ride problem defined in the previous section is based on a single-vehicle algorithm which produces for any set $X \subset H$ of customers a single route that serves as many customers in X as possible (Section 2.1). This algorithm is extended to solve the multi-vehicle dial-a-ride problem in Section 2.2.

[1] For approaches to vehicle routing and dial-a-ride problems with cost functions, we refer to Bräysy and Gendreau (2005a,b), Groër et al. (2010) and Cordeau and Laporte (2003a), Berbeglia et al. (2011a).

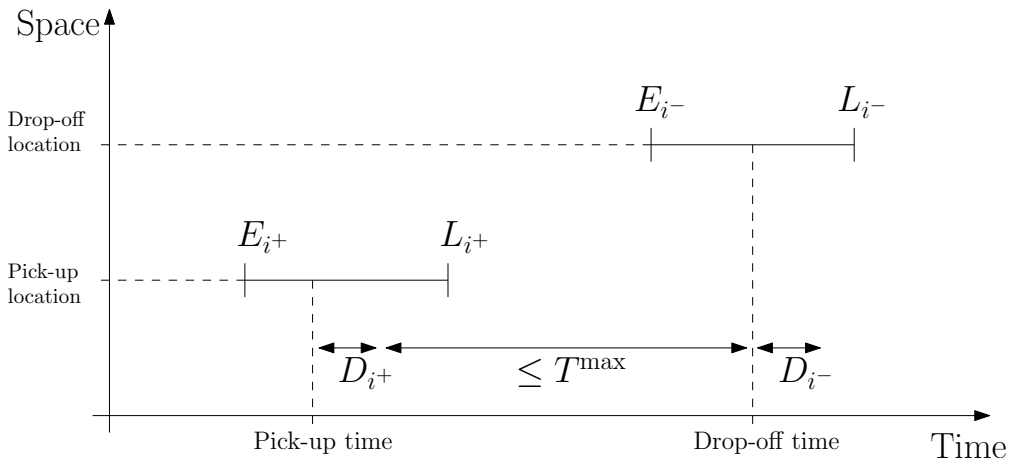


Figure 1 **Time pick-up and drop-off time windows. The pick-up point of customer i is denoted by i^+ and the drop-off point is denoted by i^-. The customer should be picked up at i^+ within the time window $[E_{i^+}, L_{i^+}]$ and the customer should be dropped off at i^- within the time window $[E_{i^-}, L_{i^-}]$. The service times needed for the customer to get on the vehicle and get off the vehicle are denoted by D_{i^+} and D_{i^-}. The time between the drop-off and the pick-up (excluding D_{i^+}) should not exceed the maximum ride time $T^{\max}$.**

The single-vehicle dial-a-ride problem has been previously studied by Psaraftis (1980), Bianco et al. (1994), Hernández-Pérez and Salazar-González (2009), Psaraftis (1983), Desrosiers et al. (1986), Sexton (1979), Sexton and Bodin (1985a,b), Häme (2011). The difference between this work and existing approaches is that we study the problem as a constraint satisfaction problem and the objective is to maximize the number of served customers.

2.1. Maximum cluster algorithm

In this section, we propose a dynamic programming method for the single-vehicle DARP ($m = 1$) with customer set X that produces a feasible solution whenever one exists or proves that the problem is infeasible. In the latter case, the algorithm returns a route that maximizes the number of served customers in X. The main challenge is to find a sequence of pick-up and drop-off nodes for which the time and precedence constraints are satisfied. While capacity constraints are also considered, the focus is on problems, in which time limits are more restrictive.

For clarity, let us denote the customers in X by numbers $\{1,\dots,N\}$ and the sets of pick-up and drop-off nodes by $P^+ = \{1^+,\dots,N^+\}$ and $P^- = \{1^-,\dots,N^-\}$. The goal is to find a feasible service sequence $(0, p_1,\dots,p_{2N}, 0)$ consisting of the pick-up and drop-off nodes of N customers. The number of permutations is $(2N)!$ and the number of feasible permutations with respect to precedence constraints is $(2N)!/2^N$ (Häme 2011). As an example, the 24 possible sequences of the pick-up and drop-off nodes of two customers are represented as a *tree structure* in Figure 2.1. The six sequences that are feasible with respect to the precedence constraints are marked with 'p' and the sequences that are feasible with respect to time and capacity constraints as well are marked with dotted lines. In this example, there is one feasible sequence serving all customers, namely, $(0, 2^+, 1^+, 2^-, 1^-, 0)$. The primary goal is to find this feasible sequence.

Briefly, our approach to the problem is a *depth-first* search, in which the infeasible branches are detected a priori and then discarded. The search begins from node 0 by determining the infeasible branches beginning from 0. In the example case presented in Figure 2.1, the branches beginning from the drop-off points 1^- and 2^- could be discarded since they would not satisfy the precedence constraint. Thus, we would only have to consider the branches beginning from 1^+ and 2^+. Similarly, at state $(0, 2^+)$, the branch beginning with 1^- could be discarded due to the precedence constraint and the search would proceed to the branches beginning from 1^+ and 2^-. In addition to precedence constraints, branches are discarded due to time and capacity constraints (see Section 2.1.3). Since only infeasible branches are discarded, this method finds a feasible solution whenever such a solution exists. If a feasible sequence serving all customers is found, the search can be terminated.

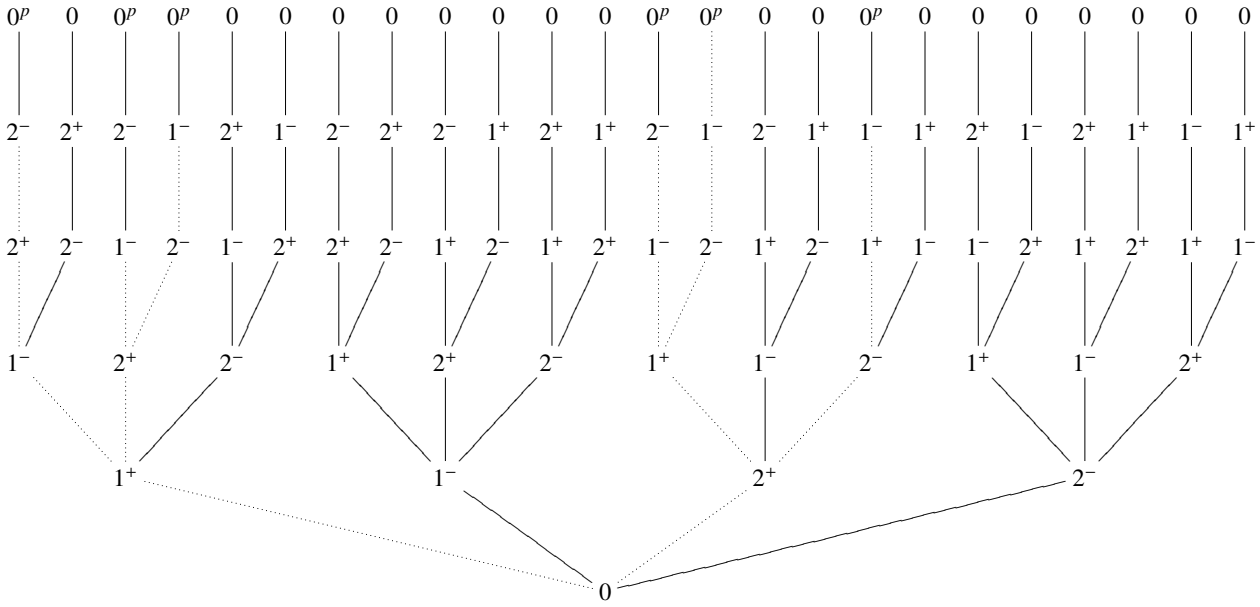


Figure 2 **The single-vehicle DARP as a tree structure. There are** 24 **permutations of the nodes** $1^+, 1^-, 2^+, 2^-$ **and six permutations (marked with 'p') that are feasible with respect to precedence constraints. The goal is to find a sequence that is feasible with respect to time and capacity constraints as well, without going through all permutations.**

2.1.1. Ranking feasible branches In order to find a feasible solution whenever one exists, we have to consider at each state S all branches that are not rendered infeasible (branches $(0, 2^+, 1^+)$ and $(0, 2^+, 2^-)$ in the previous example). However, the order in which these branches are evaluated has a significant effect on the effort needed to find a feasible solution. For example, if the search proceeded by evaluating the branch $(0, 2^+, 1^+)$, producing a feasible solution, the search could be terminated and there would be no need to evaluate the branch $(0, 2^+, 2^-)$ (which would prove to be a dead end).

Given that the search is at state S, our approach is to sort the feasible successor branches (S, i) by means of a two-step procedure as follows:

1. The feasibility of branches (S, i) is evaluated for all successor branches of S.
2. The number of feasible sequences (S, i, j) is calculated for all feasible branches (S, i).

The search proceeds by first evaluating the branches (S, i), for which the number of feasible sequences (S, i, j), where $j \in X$, is greatest.

In the example case (Figure 2.1) with $S = (0, 2^+)$, the search would proceed to branch $(0, 2^+, 1^+)$, since there are two feasible sequences $(S, 1^+, j)$, namely $(0, 2^+, 1^+, 1^-)$ and $(0, 2^+, 1^+, 2^-)$ and only one feasible sequence $(S, 2^-, j)$, that is, $(0, 2^+, 2^-, 1^+)$. A similar example is shown in Figure 3.

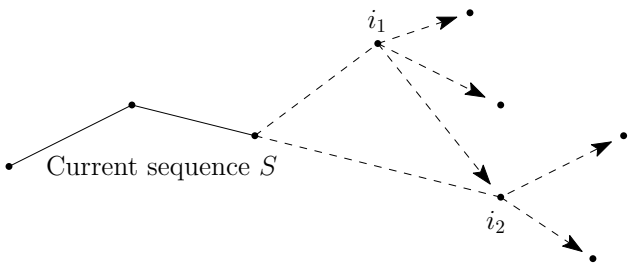


Figure 3 **Ranking feasible branches. The current sequence** S **consisting of three nodes is marked with a solid line. The branches** (S, i_1) **and** (S, i_2) **are ranked by means of the number of feasible sequences** (S, i_1, j) **and** (S, i_2, j) **marked with dashed lines. In this case, the search proceeds to state** (S, i_1) **since there are three feasible sequences** (S, i_1, j) **and two feasible sequences** (S, i_2, j)**.**

2.1.2. Recursive solution Formally, the search is executed by means of the recursion REC(S, R_S) shown in Algorithm 1, where S denotes a sequence of nodes (initially $S = (0)$), R_S denotes the set of remaining nodes at S (initially $R_S = \{1^+, 1^-, \ldots, N^+, N^-, 0\}$) and $S_{\max}$ denotes the sequence that maximizes the number of served customers (initially $S_{\max} = (0)$). $C(S)$ denotes the number of served customers in sequence S, which is equal to the number of drop-off points in S.

Algorithm 1 A recursive solution REC(S, R_S) to the single-vehicle DARP. S denotes a sequence of nodes (initially $S = (0)$), R_S denotes the set of remaining nodes (initially $R_S = \{1^+, 1^-, \ldots, N^+, N^-, 0\}$) and $S_{\max}$ denotes the sequence that maximizes the number of served customers (initially $S_{\max} = (0)$). $C(S)$ denotes the number of served customers in sequence S.

```
if C(S) > C(S_max) then
  S_max ← S;                                   (Store the current sequence S.)
  if C(S) = N
    Terminate recursion;                       (All customers have been served.)
  end if
end if
if R_S = ∅ then
  Return;                                      (Dead end)
end if
Determine the set I of nodes for which (S, i) is infeasible;   (See Section 2.1.3, Algorithm 2.)
Determine the set of remaining nodes R_(S,i) for all i ∈ R_S \ I;
Sort the remaining nodes i ∈ R_S \ I by |R_(S,i)|;
for all i ∈ R_S \ I (in sorted order) do
  REC((S, i),R_(S,i));
end for
```

The main idea of the recursion is that at each step, we have the current service sequence S and the set of remaining nodes R_S (nodes that can possibly be added to the sequence). At first, we check if the current sequence serves more customers than the previously visited sequences. If the maximum number of served customers is improved, the current sequence is stored, $S_{\max} \leftarrow S$. If all customers have been served, that is, if $C(S) = N$, a feasible solution is found and the recursion is terminated. Otherwise, we check if the set of remaining nodes is empty. If $R_S = \emptyset$, no further nodes can be added to the sequence and the search proceeds to the next branch. Otherwise, the feasibility of sequence (S, i) is checked for all $i \in R_S$ and the set of remaining nodes $R_{(S,i)}$ is calculated for all nodes $i \in R_S$ for which (S, i) is feasible (see Section 2.1.3). Then, the set of remaining nodes R_S (excluding the nodes for which (S, i) is infeasible) is sorted by $|R_{(S,i)}|$. Finally, the recursive function REC$((S, i), R_{(S,i)})$ is called for all feasible sequences (S, i) in sorted order.

In the following, we describe the feasibility screening and the calculation of the set of remaining nodes in more detail.

2.1.3. Feasibility considerations The feasibility screening is executed in two steps: In the *first step*, we check if the precedence, time and capacity constraints are satisfied at (S, i) for all remaining nodes $i \in R_S$. In the *second step*, we determine the set of remaining nodes $R_{(S,i)}$ at (S, i) by considering the possibilities of adding nodes $j \in R_S \setminus \{i\}$ to the sequence *after* i.

The first step is similar as in existing dynamic programming algorithms for the single-vehicle DARP with time windows (Psaraftis 1983, Desrosiers et al. 1986). By considering a second step, a *ranking* of the branches (S, i) is obtained by considering the number $|R_{(S,i)}|$ of remaining nodes after the first step. A summary of the feasibility conditions is presented in Table 1.

Table 1 **A summary of feasibility conditions. Conditions** (1), (2) **and** (3) **are related to the feasibility of a successor sequence** (S,i)**, whereas condition** (4) **renders the sequence** (S,i,j) **infeasible.** t_i**,** E_i **and** L_i **represent the arrival time and lower and upper bound of the time window at node** i**.** Q_i **denotes the load on the vehicle after visiting node** i**.**

Step	Condition	Impact	Equation number
1	$t_i > L_i$	(S,i) is infeasible, $R_{(S,j)} \leftarrow R_{(S,j)} \setminus \{i\}$ for all $j \in R_S \setminus \{i\}$	(1)
1	$i = h^- \in P^-$ and $h^+ \notin S$	(S,i) is infeasible	(2)
1	$Q_i > C$	(S,i) is infeasible	(3)
2	$\max(t_i, E_i) + T_{ij} > L_j$	$R_{(S,i)} \leftarrow R_{(S,i)} \setminus \{j\}$	(4)

First step In the following, s denotes the last node in S, t_s denotes the departure time at s and Q_s denotes the load on the vehicle after visiting s. For clarity, service times are assumed equal to zero. However, the modification of the following equations for non-zero service times is straightforward.

First, the set of remaining nodes $R_{(S,i)}$ at (S,i) is initialized, that is, $R_{(S,i)} \leftarrow R_S \setminus \{i\}$ for all $i \in R_S$. The time of arrival t_i at i is given by $t_i = t_s + T_{si}$ and the load Q_i after visiting i is given by $Q_i = Q_s + q_i$.

We note that the sequence (S,i) is infeasible, if

$$t_i > L_i. \tag{1}$$

In this case, there exist no sequences from S to i such that the time constraint at i is satisfied. Thus, the sequence (S,i) is marked infeasible and node i is removed from the set of remaining nodes $R_{(S,j)}$ at (S,j) for all $j \in R_S$. This state elimination criteria is similar to the one presented in (Desrosiers et al. 1986). Maximum ride time constraints are checked similarly with the exception that L_i is modified by means of the procedure described in Section 2.1.4.

In addition, the sequence (S,i) is infeasible, if at least one of the following conditions is true:

$$i = h^- \in P^- \text{ and the node } h^+ \text{ is not included in } S, \tag{2}$$
$$Q_i > Q. \tag{3}$$

In this case, the sequence (S,i) is marked infeasible but i is not removed from the sets of remaining nodes $R_{(S,j)}$, since there may exist feasible sequences (S,j) that can be extended to include node i.

Second step Given that the vehicle is at the last node of sequence S, let us consider the possibilities of adding a node $j \in R_{(S,i)}$ to the sequence (S,i). The time of arrival t_j at j is given by $\max(t_i, E_i) + T_{ij}$.

Similarly as in condition (1), if there exists a node $j \in R_{(S,i)}$ such that

$$\max(t_i, E_i) + T_{ij} > L_j, \tag{4}$$

the sequence (S,i,j) is infeasible and node j is removed from the set of remaining nodes $R_{(S,i)}$ at (S,i). For the checking of maximum ride time constraints, see Section 2.1.4.

This feasibility condition remotely similar to the one studied in (Dumas et al. 1991), in which the set of *admissible arcs* between the pick-up and delivery nodes is constructed as a preprocessing step in the pickup and delivery problem. The set of admissible arcs is made up of arcs which a priori satisfy the precedence, capacity and time constraints of the problem. The difference in our formulation is that we study the feasibility of sequences (S,i,j) during the search of feasible routes (see Figure 4).

Note that even if the precedence or capacity constraint is violated by adding node j to the sequence (S,i), there may exist a feasible sequence $(S,i,k_1,\ldots,k_h,j)$ in which i precedes j. That is, if the sequence (S,i) is not rendered infeasible by condition (4), there *may exist* a feasible sequence in which j is visited after i (not necessarily immediately after i) and thus node j is not removed from $R_{(S,i)}$.

The above two-step feasibility screening procedure is presented in Algorithm 2.

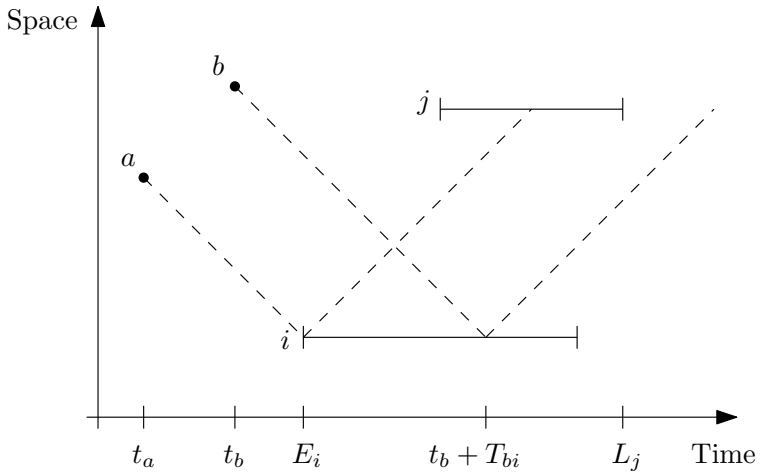


Figure 4 **Admissible arcs. The figure shows the locations and time windows of two nodes i and j. If the vehicle left from a at the instant t_a, visited node i and moved directly to node j, the time constraint L_j would be satisfied. However, assuming that at the instant t_b the vehicle is located at b, the arc (i, j) is seen to be inadmissible.**

Algorithm 2 Two-step feasibility screening. The algorithm returns the set of nodes I for which the sequence (S, i) is infeasible and the set $R_{(S,i)}$ of remaining nodes for each feasible branch (S, i).

$I = \emptyset$, $R_{(S,i)} = R_S \setminus \{i\}$ for all $i \in R_S$; (I = the set of nodes for which (S, i) is infeasible)
for all $i \in R_S$ **do**
 if $t_s + T_{si} > l_i$ (Condition (1))
  $I = I \cup \{i\}$; ((S, i) is infeasible)
  $R_{(S,j)} = R_{(S,j)} \setminus \{i\}$ for all $j \in R_S \setminus \{i\}$;
  continue; (continue to next i)
 end if
 if $(i = h^- \in P^-$ and $h^+ \notin S)$ or $(Q_i > Q)$ (Conditions (2) and (3))
  $I = I \cup \{i\}$; ((S, i) is infeasible)
  continue; (continue to next i)
 end if
 for all $j \in R_{(S,i)}$ **do**
  if $\max(t_i, e_i) + T_{ij} > l_j$ (Condition (4))
   $R_{(S,i)} = R_{(S,i)} \setminus \{j\}$; ($(S, i, j)$ is infeasible)
  end if
 end for
end for
return $(I, \{R_{(S,i)}\}_{i \in R_S})$;

2.1.4. Maximum ride time constraints The feasibility of maximum ride time constraints is checked as follows. Let $T_{\max}$ denote the maximum ride time. When a pick-up node j^+ is added to the sequence, the earliest arrival time t_{j^+} at j^+ is calculated, and the updated latest drop-off time of customer j is calculated by means of the formula

$$L'_{j^-} = \min(L_{j^-}, \max(t_{j^+}, E_{j^+}) + T_{\max}), \tag{5}$$

where $\max(t_{j^+}, E_{j^+})$ denotes the earliest pick-up time of customer j.

The updated latest drop off time L'_{j^-} describes the latest drop-off time given that customer j is picked up at the earliest pick-up time. However, it may sometimes be profitable to delay the beginning of service at

node j^+ so as to reduce the unnecessary in-vehicle waiting time at any node visited between j^+ and j^- and thus, the ride time associated with customer j (Cordeau and Laporte 2003a).

In order to take into account the possibility of delaying the pick-up of customer j, we propose the following procedure. First, at the pick-up node j^+, we define the *time slack* s_j of customer j by $s_j = L_{j^-} - \max(t_{j^+}, E_{j^+})$, where $\max(t_{j^+}, E_{j^+})$ is the earliest pick-up time of customer j. The time slack describes the amount of time the pick-up of customer j can be delayed at most. Since the pick-up of customer can be delayed by at most s_j, we define the *delayed latest drop-off time* L''_j of customer j by

$$L''_{j^-} = \min(L_{j^-}, L'_{j^-} + s_j). \tag{6}$$

In addition, we initialize the *in-vehicle waiting time* w_j of customer j by $w_j = 0$.

Then, at each node h visited between j^+ and j^-, the time slack s_j, in-vehicle waiting time w_j and the delayed latest drop off time L''_{j^-} of customer j are updated:

$$s_j \leftarrow \min(s_j, w_j + L_h - t_h), \tag{7}$$
$$L''_{j^-} \leftarrow \min(L_j, L'_{j^-} + s_j), \tag{8}$$
$$w_j \leftarrow w_j + \max(E_h - t_h, 0), \tag{9}$$

where t_h denotes the earliest time of arrival at node h. Note that if h is a drop-off point, L_h in Equation (7) is replaced by L''_h. Finally, at the drop-off point j^-, the maximum ride time constraint is violated if

$$t_{j^-} > L''_{j^-}, \tag{10}$$

where t_{j^-} is the earliest arrival time at node j^- (see Figure 5).

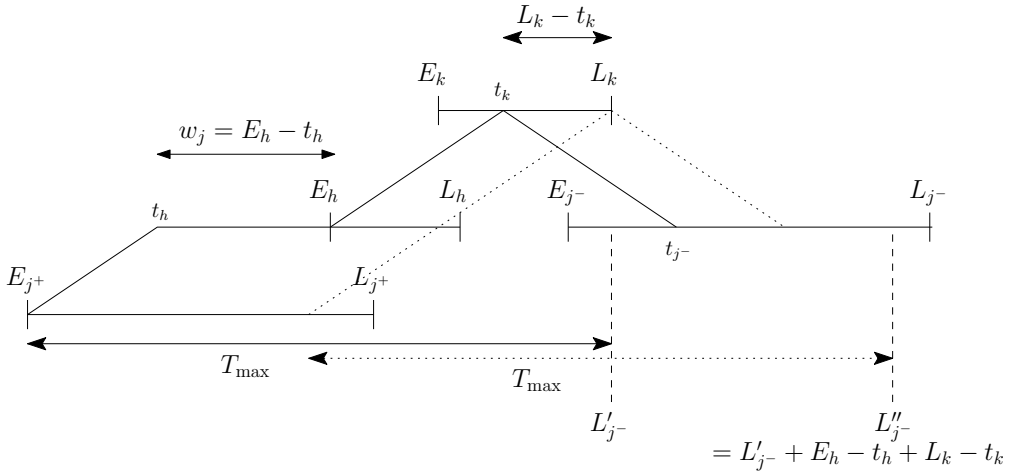


Figure 5 **Maximum ride time constraints. The figure shows the time windows of nodes j^+, h, k, j^-, a non-delayed route (solid line) and a delayed route (dotted line). At the pick-up point j^+, the updated latest drop-off time L'_{j^-} of customer j is calculated by means of Equation (5). At the nodes h and k between j^+ and j^-, the time slack s_j, delayed latest drop-off time L''_{j^-} and in-vehicle waiting time w_j of customer j are updated by Equations (7), (8) and (9). Finally, at j^-, the maximum ride time constraint is checked by Equation (10).**

2.1.5. Discarding suboptimal sequences Since the goal is to maximize the number of served customers, the sequences that can not improve the maximum number of served customers, can be discarded (in addition to infeasible sequences). That is, given a sequence S, set of remaining nodes R_S and the best found solution $S_{\max}$, the sequence S is discarded if

$$C(S) + C(R_S) \leq C(S_{\max}),$$

where $C(S), C(R_S)$ and $C(S_{\max})$ denote the number of drop-off points in S, R_S and $S_{\max}$, respectively.

2.1.6. A heuristic extension Algorithm 1 describes an exact procedure for maximizing the number of served customers in a single route. In order to find the exact solution, the algorithm goes through all branches until no further nodes can be added to the current sequence or the sequence is seen to be suboptimal.

The effort of the algorithm can be controlled by limiting the number of branches that are evaluated by means of a positive parameter L. For example, if $L = 1$, the algorithm constructs a single sequence and stops when the set of remaining nodes in empty. By increasing L the search space is expanded and if $L = (2N)!/2^N$ (the number of permutations that satisfy precedence constraints), the heuristic coincides with the exact algorithm.

2.2. Multi-vehicle solution

In this section, we propose an exact approach to the multi-vehicle DARP as a constraint satisfaction problem. By using Algorithm 1 described in Section 2.1 as a subroutine, the method produces a feasible solution to any instance or proves that the instance is infeasible. The main idea is that the vehicle routes are constructed one by one, each maximizing the number of served customers in the set of remaining customers. The customers are denoted by numbers $1, \dots, n$ and the vehicles are denoted by numbers $1, \dots, m$.

First, a route is constructed for vehicle 1, serving as many customers as possible. Then, the process is repeated with vehicle 2 for the set of customers that were not served by vehicle 1 and so forth (see Figure 6). Letting X denote a set of customers, we denote by $\mathrm{MC}(X)$ the single-vehicle maximum cluster algorithm (Algorithm 1) that returns a route that maximizes number of served customers in set X and the corresponding set of served customers. This set will be referred to as a maximum cluster of X, which is formally defined as follows.

Definition 1. Let X be a set of customers and $M \subset X$ be a subset of X, for which there exists a feasible route serving all customers in M. If $|M| \geq |Y|$ for all sets $Y \subset X$ for which there exists a feasible route serving all customers in Y, then M is a *maximum cluster* of X.

Note that if there exists a feasible route serving all customers in X, we have $M(X) = X$. The solution to the multi-vehicle case is outlined in Algorithm 3.

Algorithm 3 Outline of the multi-vehicle algorithm. X_k denotes the set of customers assigned to vehicle k and $\mathrm{MC}(X_k)$ denotes the single-vehicle subroutine presented in Algorithm 1 that returns a maximum cluster X'_k of X_k and the corresponding route S_k. Initially, all customers are assigned to the first vehicle, that is, $X_1 = \{1, \dots, n\}$ and $X_k = \emptyset$ for all $k \in \{2, \dots, m\}$.

for all $k \in \{1, \dots, m\}$ **do**
  $[S_k, X'_k] \leftarrow \mathrm{MC}(X_k)$; ($S_k$ = route, X'_k = set of served customers)
  $X_{k+1} \leftarrow X_{k+1} \cup (X_k \setminus X'_k)$, where $X_{m+1} = X_1$;
end for

2.2.1. Iteration If a feasible solution is not found directly by using the procedure described in Algorithm 3, the process is repeated. The goal is to find a set of customer-vehicle assignments such that for each vehicle there exists a feasible route serving all customers assigned to the vehicle or to prove infeasibility by going through all possible sets of customer-vehicle assignments.

Formally, we define a *partition* of customers as a disjoint cover of $\{1, \dots, n\}$ consisting of m sets $(X_1, \dots, X_m)$, where X_k denotes the set of customers assigned to vehicle k. The set of all partitions is denoted by $\mathcal{P}$. Since each customer can be assigned to any of the m vehicles, the total number of possible partitions is equal to the Stirling number of the second kind, that is, $|\mathcal{P}| = S_2(n, m) = \frac{1}{m!} \sum_{i=0}^{m} (-1)^i \binom{m}{i} (m - i)^n$. A partition is said to be *feasible*, if $M(X_k) = X_k$ for all $k \in \{1, \dots, m\}$.

Note that if the multi-vehicle instance has a solution, there exists a partition $(X_1^*, X_2^*, \dots, X_m^*)$ for which $M(X_k^*) = X_k^*$ for all $k \in \{1, \dots, m\}$. Thus, infeasibility can be proved by iteratively checking all partitions for feasibility.

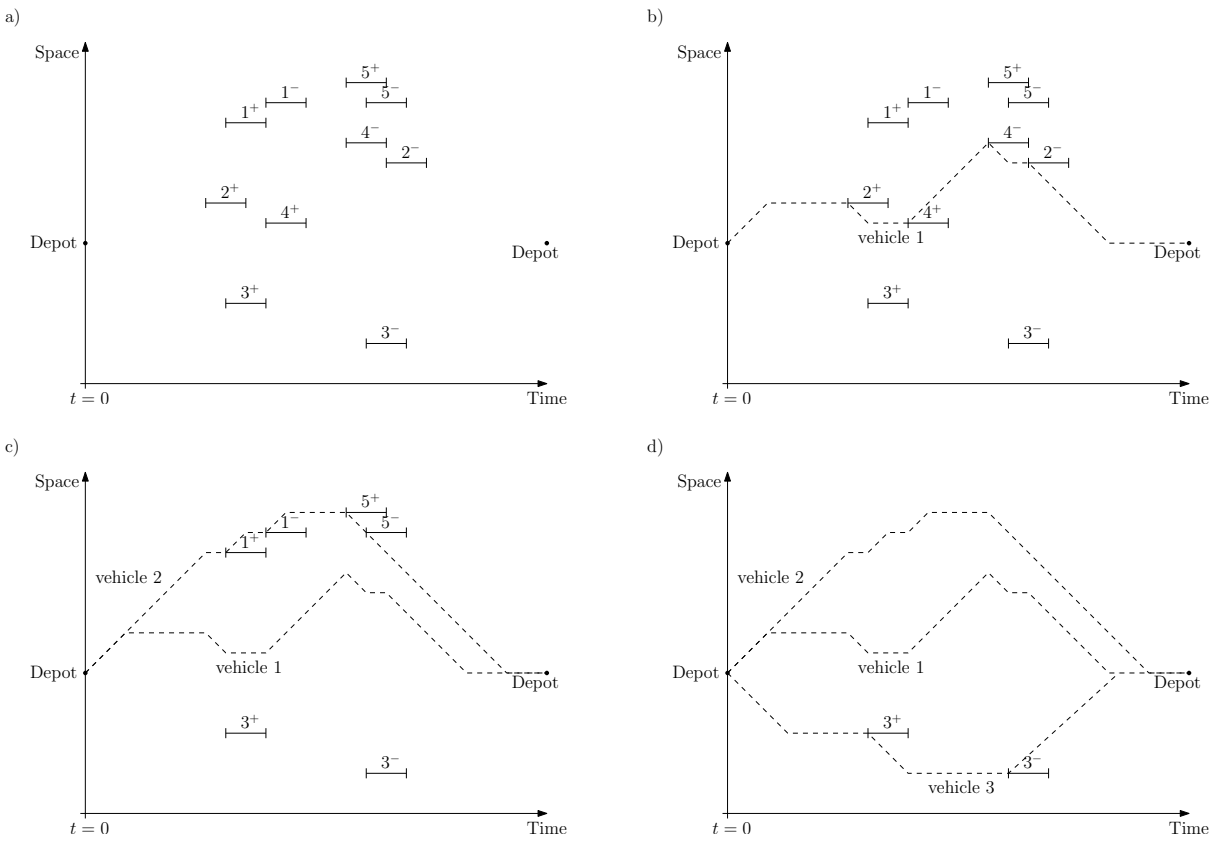


Figure 6 **A one-dimensional example of the approach to the multi-vehicle problem involving five customers and three vehicles. The first route is constructed by maximizing the number of customers that can be served by a single vehicle (Figure b). Then, the customers 2 and 4 served by the first vehicle are removed from the set of remaining customers and the process is repeated for the second vehicle (Figure c). Finally, a route is construced for the third vehicle, serving the last remaining customer 3 (Figure d).**

Each iteration starts with the *candidate* partition $(X_1, \ldots, X_m)$ and results in another partition $(X'_1, \ldots, X'_m)$. If the partition $(X'_1, \ldots, X'_m)$ is not feasible, the candidate partition $(X_1, \ldots, X_m)$ is marked as 'checked'. Otherwise, a feasible solution has been found and the algorithm is terminated.

If $(X'_1, \ldots, X'_m)$ is marked as 'checked', the algorithm proceeds to the next candidate partition that is not marked as 'checked' (see Section 2.2.2). Otherwise, $(X'_1, \ldots, X'_m)$ becomes the candidate partition, $(X_1, \ldots, X_m) \leftarrow (X'_1, \ldots, X'_m)$.

The main idea is that the first iteration (Algorithm 3) produces an *initial candidate partition* of customers, which is improved during later iterations. This procedure is motivated by the fact that the initial vehicle routes are constructed by attempting to serve as many customers as possible from the initial set $\{1, \ldots, n\}$ of all customers. After the initial routes have been determined, we know that at least the customers in X'_k can be served by vehicle k. Thus, for any set X, the number of served customers from the set $X \cup X'_k$ is *at least* equal to the number of customers in the set X'_k. A customer x that has relatively few admissible arcs to all nodes may be in a good position with respect to the customers assigned to a single vehicle k. That is, even if customer x was left out of the initial vehicle routes due to a small number of out admissible arcs to other nodes, x can be added to one of the routes during a later iteration.

2.2.2. A priori screening In order to detect infeasible instances without going through all partitions and to reduce the number of partitions that are considered, we propose the following procedure.

First, all routes involving a single customer are checked for feasibility. If any of these routes is infeasible, there exist no feasible solutions to the problem.

Second, the routes involving two customers are checked for feasibility. If there exist no feasible routes involving customers $\{i, j\}$, an arc between i and j is formed. This means that customers i and j can not be assigned to the same vehicle. Then, we find the *maximum clique* C within the set of customers, that is, the largest set of customers for which there is an arc between all pairs of nodes $i, j \in C$, see (Wood 1997). If the size of the maximum clique is greater than the number m of vehicles, the instance is infeasible since there are $|C| > m$ customers which should all be served by different vehicles.

Otherwise, since the customers in the maximum clique $C = \{c_1, \ldots, c_{|C|}\}$ all have to be assigned to different vehicles, with no loss of generality we may initially assign customer c_j to vehicle j for all $j \in \{1, \ldots, |C|\}$. Then, we determine the set of *feasible vehicles* V_i for each customer i as follows. 1) For $c_j \in \{c_1, \ldots, c_{|C|}\}$, we have $V_{c_j} = \{j\}$. For other customers $i \in \{1, \ldots, n\} \setminus C$, we initially define $V_i = \{1, \ldots, m\}$. 2) If $|V_j| = 1$ and there is an arc between j and i, customer i may not be assigned to the same vehicle as j, that is, $V_i \leftarrow V_i \setminus V_j$. This step is repeated for all $i, j \in \{1, \ldots, n\}$ until convergence.

After the set of feasible vehicles V_i has been determined for each customer $i \in \{1, \ldots, n\}$, the set of *a priori feasible partitions* is defined as follows.

Definition 2. A partition $(X_1, \ldots, X_m)$ of customers is feasible a priori, if $i \in X_k \Rightarrow k \in V_i$ for all $k \in \{1, \ldots, m\}$. The set of a priori feasible partitions is denoted by $\mathcal{P}'$.

Note that assigning each customer i a vehicle number $v_i \in V_i$ defines a unique partition. Since the vehicle number of customer i can be chosen from the set V_i of feasible vehicles, the number of a priori feasible partitions is given by $|\mathcal{P}'| = \prod_{i=1}^{n} |V_i|$.

Theorem 1. *The number of a priori feasible partitions satisfies* $|\mathcal{P}| \leq m^{n-|C|}$.

Proof. Since $|V_i| = 1$ for all $i \in C$, we get $|\mathcal{P}| = \prod_{i=1}^{n} |V_i| = \prod_{i \in \{1,\ldots,n\} \setminus C} \overbrace{|V_i|}^{\leq m} \prod_{i \in C} \overbrace{|V_i|}^{=1} \leq m^{n-|C|}$. □

The set of a priori feasible partitions is represented by the set $V_1 \times \ldots \times V_n$ and each a priori feasible partition is represented by a vector $(v_1, \ldots, v_n) \in V_1 \times \ldots \times V_n$. Information on checked partitions is stored in a sparse array 'CHECKED'. The partitions not marked as checked are included in the set of remaining partitions I (initially $I = V_1 \times \ldots \times V_n$). During run-time, each candidate partition is marked as 'checked' by setting $\mathrm{CHECKED}(v_1, \ldots, v_n) = 1$ and removed from the set of remaining partitions, that is, $I \leftarrow I \setminus \{(v_1, \ldots, v_n)\}$. If the iteration results in a partition that is marked as 'checked', the next candidate partition is chosen randomly from the set of remaining partitions I. This way, the iteration described in Section 2.2.1 goes through all feasible partitions.

2.2.3. Filtering The algorithm can be accelerated by using information on previously calculated clusters as follows.

Theorem 2. *Let X denote a set of customers and $M(X)$ a maximum cluster of X. If X' is a subset of X satisfying $M(X) \subset X' \subset X$, the set $M(X)$ is a maximum cluster of X'.*

Proof. If $M(X)$ is not a maximum cluster of X', there exists a set $Y \subset X'$ for which $|Y| > |M(X)|$ and thus $M(X)$ is not a maximum cluster of X. □

In other words, for any set X' satisfying $M(X) \subset X' \subset X$, we do not need to run the single-vehicle maximum cluster algorithm to find a maximum cluster of X'.

Theorem 3. *Let $H = \{1, \ldots, n\}$ denote the set of all customers and $M(H)$ a maximum cluster of H. If Y is a set of customers served in a single route in $X \subset H$ satisfying $|Y| = |M(H)|$, the set Y is a maximum cluster of X.*

Proof. If Y is not a maximum cluster of X, there exists a route that serves customers $Y' \subset X$ for which $|Y'| > |Y| = |M(H)|$ and thus $M(H)$ is not a maximum cluster of H. □

Theorem 3 means that if the maximum cluster algorithm finds a route that serves as many customers as in the maximum cluster $M(H)$ of all customers, the algorithm can be terminated since there exist no larger clusters than $M(H)$.

3. Structure and complexity

The structure of the single-vehicle maximum cluster algorithm (Algorithm 1) is as follows. Briefly described, each recursion step involves checking the the feasibility of sequences (S, i, j), where $i, j \in R_S$.

Clearly, this process has a complexity of order $O(|R|^2)$, where $|R_S|$ is the number of remaining nodes.

The overall complexity of finding a feasible solution in the single-vehicle case is strongly dependent on the number of recursion steps needed to find the solution. Finding a feasible route involving N customers involves at least $2N$ recursion steps: At first, the number of remaining nodes is $2N$ and each time a feasible node is added to the end of the service sequence, the number of remaining nodes is decreased by one (see Figure 6). In the best case, no infeasible states are encountered and thus there are $2N$ recursion steps (one for each node added to the sequence). At step $s \in \{1, \dots, 2N\}$, the number of remaining nodes is $2N + 1 - s$ and thus the complexity is of order $O(\sum_{i=1}^{2N}(2N + 1 - s)^2) = O(N^3)$.

The efficiency of the detection of infeasible branches has a significant effect on the performance of the single-vehicle algorithm. The computational work is linearly increased with the number of dead ends $A(N)$ encountered during the recursion. The overall complexity of the single-vehicle algorithm is thus $O(N^3 + A(N))$. The number of dead ends encountered satisfies $A(N) \leq (2N)!/2^N$. If the heuristic version of the algorithm is used (see Section 2.1.6), we have $A(N) \leq L$.

The multi-vehicle algorithm involves solving the single-vehicle case for each vehicle in r iterations. Thus the complexity is bounded above by $O\left(rm(n^3 + A(n))\right)$. Note that the complexity is in general lower since the size of the set of customers is n only for the first vehicle in the first iteration. For the other vehicles, the number of customers is smaller since the customers that are already included in another vehicle route are not a part of the subproblem. For example, if the customers were divided equally among vehicles a priori, the complexity would be of order $O\left(rm((n/m)^3 + A(n/m))\right) = O\left(r(n^3/m^2) + rA(n/m)\right)$.

Since the total number of possible partitions is equal to the Stirling number of the second kind, the number of iterations r needed to find a feasible solution or to prove infeasibility satisfies $r \leq S_2(n, m) = \frac{1}{m!}\sum_{i=0}^{m}(-1)^i\binom{m}{i}(m-i)^n$. Given the maximum clique C (see section 2.2.2), we have $r \leq |\mathcal{P}'| = \prod_{i=1}^{n}|V_i| \leq m^{n-|C|}$.

In summary, the worst case complexity is high especially when the set of a priori feasible partitions $\mathcal{P}'$ is large. However, as will be seen in the next section, solutions to problems with loose constraints are usually found with little effort. On the other hand, tight constraints reduce the set of a priori feasible partitions $\mathcal{P}'$ and thus also the complexity of the algorithm.

4. Numerical experiments

In the following, we compare the performance of the multi-vehicle maximum cluster (MC) algorithm presented in Section 2.2 with two existing solution methods, namely, a tabu search algorithm (Cordeau and Laporte 2003a) and a constraint programming (CP) algorithm (Berbeglia 2009).

Berbeglia (2009) presents computational tests to compare the tabu and CP algorithms. We repeated these tests for the maximum cluster algorithm and compared our results with the ones presented in (Berbeglia 2009). The maximum cluster algorithm was implemented in Matlab and the tests were performed on a 2.2 GHz Dual Core Intel PC. The tabu and CP algorithms were tested on a 2.5 GHz Dual Core AMD Opteron computer (Berbeglia 2009).

In the studied instances the pick-up and drop-off points are located in a 20×20 square and the ride times between points (in minutes) are equal to Euclidean distances. The time windows have 15 minutes of length.

In the first set of instances marked with a, the capacity of a vehicle equals $Q = 3$, the load of each customer is 1 and the maximum ride time is $RT = 30$ minutes. In the second set marked with b, we have $Q = 6, RT = 45$ and the load associated to each customer is chosen randomly according to a uniform distribution on the set $\{1, \dots, Q\}$. In both sets, the service time is proportional to the number of passengers, namely $d_i = d_{n+i} = q_i$. The instances are described in more detail in (Cordeau 2006, Ropke et al. 2007).

In addition to the original sets a and b, the tests were performed on the modified sets presented in (Berbeglia 2009). In the first and second modifications, the maximum ride time equals $RT = 30$ and $RT = 22$,

respectively. The third modification is obtained by reducing the number of vehicles to 75% of the original number, rounded down to the nearest integer.

The results of the tests are shown in Table 4. The first column shows the instance labels of the form *am-n* or *bm-n*, where *m* indicates the number of vehicles and *n* corresponds to the number of customers. The other columns show the average time (in seconds) needed to solve the instances and the corresponding modifications by using the different solution methods, calculated over ten runs. The results obtained by the first two algorithms, tabu and CP, have been reported previously in (Berbeglia 2009, Berbeglia et al. 2011b)[2]. A number in parentheses indicates that the instance was proven to be infeasible, a dash indicates that a solution was not found in three minutes computing time and a star indicates that results for the instance have not been reported. For comparison, we have added our results obtained by the maximum cluster algorithm to the table. The Matlab-implementations of the maximum cluster algorithm are available at `http://math.tkk.fi/~lehame/exact/`. We first executed the heuristic version of the algorithm (see Section 2.1.6) with $L = 1$. If a solution was not found and the problem was not rendered infeasible by the a priori screening procedure described in Section 2.2.2, the exact version of the algorithm was executed.

By looking at the results obtained by the maximum cluster algorithm we see that most instances were solved within a fraction of a second. Except for a single modified instance (b7-84, 75% of vehicles), the maximum cluster algorithm produced a feasible solution or proved ineasibility in all instances. Note that the maximum cluster algorithm produced a feasible solution or proved infeasibility in the modified instances with six vehicles and 48 customers (b6-48), for which results have not been previously reported.

The CPU times obtained by the maximum cluster algorithm are typically of order ten times smaller compared to the results of the tabu and CP algorithms. The best improvement factor is $109.5/0.04 \approx 2700$ compared to CP (instance b5-60) and $78.5/0.06 \approx 1300$ compared to tabu (instance a4-48, 75% of vehicles). Although the algorithms were tested on different platforms, the results seem to justify the efficiency of the maximum cluster algorithm on the test instances. We acknowledge that there are instances in wich tabu and CP produced a feasible solution or proved infeasibility faster than the maximum cluster algorithm. However, the results suggest that the multi-vehicle maximum cluster algorithm may have practical importance since it is capable of handling large problems in short computation times.

The lower part of Table 4 shows the computational profile of the maximum cluster algorithm for the instance b8-96. The profile shows that most of the run time of the algorithm is consumed in the feasibility screening phase. Thus, in order to further optimize the run times of the maximum cluster algorithm, one might attempt to reduce the computational effort of the feasibility screening phase.

5. Conclusions

In this work, an exact constraint satisfaction algorithm for the multi-vehicle dial-a-ride problem is suggested. The purpose of the method is 1) to produce a solution that satisfies the constraints of the problem or 2) to prove infeasibility. In dynamic demand-responsive transport services, the algorithm can be used for deciding whether to accept or reject incoming user requests in real-time. The algorithm can also be used as a preprocessing step in a static problem setting.

The multi-vehicle solution is based on a recursive single-vehicle algorithm that maximizes the number of served customers in a given set. The single-vehicle algorithm is based on looking two steps ahead at each recursion step: The next decision is determined by the possibilities of making feasible decisions *after* the next one. This single-vehicle subroutine is successively executed for all available vehicles until each customer is assigned to a vehicle route or the problem is seen to be infeasible.

The multi-vehicle algorithm is compared to existing solution approaches (Cordeau and Laporte 2003a, Berbeglia 2009) by means of numerical tests, justifying the efficiency of the solution approach.

Acknowledgments

This work was partly supported by the Finnish Funding Agency for Technology and Innovation, Finnish Ministry of Transport and Communications and Helsinki Region Transport.

[2] The computing time for the instance 'b6-48' has been reported for the tabu algorithm in (Ropke et al. 2007).

Instance	Original			RT=30			RT=22			75 % of vehicles		
	Tabu	CP	MC	Tabu	CP	MC	Tabu	CP	MC	Tabu	CP	MC
a4-40	0.8	0.5	0.02	0.8	0.5	0.02	0.5	0.3	0.05	1.7	0.3	0.64
a4-48	1.0	0.5	0.05	1.0	0.5	0.05	1.3	0.4	0.05	78.5	0.6	0.08
a5-40	0.3	0.3	0.02	0.3	0.3	0.02	0.3	0.3	0.02	1.3	0.3	0.02
a5-50	0.7	0.5	0.04	0.7	0.5	0.04	0.6	0.6	0.03	3.9	1.3	2.0
a5-60	1.4	1.0	0.05	1.4	1.0	0.05	1.5	0.9	0.05	-	24.5	64.7
a6-48	0.4	0.6	0.03	0.4	0.6	0.03	0.4	0.7	0.03	1.1	0.5	0.07
a6-60	1.0	5.6	0.04	1.0	5.6	0.04	-	(1.1)	(0.63)	11.6	6.2	10.7
a6-72	1.9	5.0	0.06	1.9	5.0	0.06	-	(1.7)	(0.77)	5.7	2.0	0.90
a7-56	0.5	1.8	0.04	0.5	1.8	0.04	-	(0.6)	(0.40)	0.9	1.5	0.06
a7-70	1.7	41.5	0.06	1.7	41.5	0.06	1.4	7.6	0.06	3.2	5.1	0.06
a7-84	2.7	3.4	0.08	2.7	3.4	0.08	3.0	4.1	0.08	7.5	3.5	0.07
a8-64	0.8	6.2	0.05	0.8	6.2	0.05	-	(1.9)	(0.84)	1.1	2.5	0.04
a8-80	1.5	8.5	0.07	1.5	8.5	0.07	1.8	6.0	0.07	3.0	3.6	0.07
a8-96	3.5	37.7	0.11	3.5	38.7	0.11	-	(3.7)	(1.72)	8.1	5.3	0.15
b4-40	0.6	0.4	0.02	0.4	0.3	0.05	0.4	0.3	0.05	-	(0.1)	(0.40)
b4-48	1.3	0.4	0.03	1.6	0.4	0.08	-	(0.1)	(0.32)	-	(0.1)	(0.59)
b5-40	0.4	0.3	0.03	0.4	0.4	0.03	-	(0.1)	(0.27)	-	(0.1)	(0.37)
b5-50	1.2	6.8	0.06	0.9	0.8	0.05	1.1	0.9	0.18	-	(0.1)	(0.58)
b5-60	1.5	109.5	0.04	1.8	3.1	0.04	1.6	1.2	0.04	-	(0.1)	(0.78)
b6-48	0.3	*	0.02	*	*	0.02	*	*	0.02	*	*	(0.54)
b6-60	0.9	5.1	0.04	1.5	1.5	0.04	1.0	1.4	0.04	5.3	3.8	3.2
b6-72	2.1	27.3	0.05	2.3	2.4	0.06	2.4	2.4	0.05	9.7	25.6	5.1
b7-56	0.5	13.3	0.04	0.6	1.5	0.03	0.5	1.5	0.04	2.1	3.9	0.19
b7-70	1.4	5.6	0.08	1.6	18.4	0.08	1.3	2.7	0.05	12.6	78.7	4.3
b7-84	3.0	25.8	1.60	2.9	6.3	0.08	-	(0.1)	(0.64)	-	-	-
b8-64	0.7	12.7	0.09	0.8	1.9	0.09	0.8	1.9	0.04	1.8	4.8	0.98
b8-80	1.9	23.9	0.07	1.8	19.2	0.07	-	(0.5)	(1.28)	4.0	13.9	0.27
b8-96	4.1	149.1	0.12	3.9	34.8	0.12	3.9	56.1	0.15	8.2	-	0.74

Computational profile of the maximum cluster algorithm for a5-60, 75% of vehicles	
Single-vehicle subroutine (89.8%)	Other (10.2%)
- Feasibility screening (> 89.1 %)	- Initialization - List operations

Table 2 **Comparison between a tabu search algorithm (Cordeau and Laporte 2003a), a constraint programming algorithm (Berbeglia 2009) and the maximum cluster algorithm. The results obtained by the first two algorithms are given in (Berbeglia 2009, Berbeglia et al. 2011b). The upper table shows the average time (in seconds) needed to solve the instance of the dial-a-ride problem in the first column by using the different solution methods, calculated over ten runs. The times without parentheses indicate that a feasible solution was found and the times in parentheses indicate that the instance was proven to be infeasible. A star (*) indicates that the computing time has not been reported. The lower table shows the computational profile of the maximum cluster algorithm for the instance b8-96. The profile shows that most of the run time of the algorithm is consumed in the feasibility screening phase.**

References

Berbeglia, G. 2009. *Complexity Analyses and Algorithms for Pickup and Delivery Problems*. Ph.D. Thesis, HEC Montreal.

Berbeglia, G., J.-F. Cordeau, G. Laporte. 2011a. A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. Forthcoming.

Berbeglia, G., G. Pesant, L.-M. Rousseau. 2011b. Checking feasibility of the dial-a-ride problem using constraint programming. Forthcoming.

Bianco, L., A. Mingozzi, S. Ricciardelli, M. Spadoni. 1994. Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. *INFOR* **32** 19–31.

Bräysy, O., M. Gendreau. 2005a. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science* **39**(1) 104118.

Bräysy, O., M. Gendreau. 2005b. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science* **39**(1) 119139.

Cordeau, J.-F. 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* **54** 573–586.

Cordeau, J.-F., G. Laporte. 2003a. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research B* **37** 579–594.

Cordeau, J.-F., G. Laporte. 2007c. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research* **153** 29–46.

Cordeau, J.-F., G. Laporte, J.-Y. Potvin, M.W.P. Savelsbergh. 2007a. Transportation on demand. *Transportation*. Amsterdam: North-Holland, 429–466.

Desrosiers, J., Y. Dumas, F. Soumis. 1986. A dynamic programming solution of the large-scale singlevehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences* **6** 301–325.

Diana, M., M.M. Dessouky. 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B* **38** 539–557.

Dumas, Y., J. Desrosiers, F. Soumis. 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research* **54** 7–22.

Groër, C., B. Golden, E. Wasil. 2010. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation* **2**(2).

Häme, L.E. 2011. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows. *European Journal of Operational Research.* **209**(1) 11 – 22.

Helsinki Region Traffic. 2010. Jouko neighbourhood routes and service routes. URL `http://www.hsl.fi/EN/passengersguide/ServiceroutesandJoukoroutes/Pages/default.aspx`.

Hernández-Pérez, H., J.-J. Salazar-González. 2009. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research* **196** 987–995.

Hunsaker, B., M. W. P. Savelsbergh. 2002. Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters* **30** 169–173.

Jaw, J.J., A.R. Odoni, H.N. Psaraftis, N.H.M. Wilson. 1986. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. *Transportation Research Part B* **20** 243–257.

Jokinen, Jani-Pekka, Teemu Sihvola, Esa Hyytiä, Reijo Sulonen. 2011. Why urban mass demand responsive transport? *IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS)*. Vienna, Austria.

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

Psaraftis, H.N. 1980. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science* **14** 130–154.

Psaraftis, H.N. 1983. An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science* **17** 351–357.

Ropke, S., J.-F. Cordeau, G. Laporte. 2007. Models and branch-and-cut algorithm for pickup and delivery problems with time windows. *Networks* **49** 258–272.

Sexton, T. 1979. *The single vehicle many-to-many routing and scheduling problem*. Ph.D. dissertation, SUNY at Stony Brook.

Sexton, T., L. D. Bodin. 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science* **19** 378–410.

Sexton, T., L. D. Bodin. 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: Ii. routing. *Transportation Science* **19** 411–435.

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

Wong, K.I., M.G.H. Bell. 2006. Solution of the dial-a-ride problem with multi-dimensional capacity constraints. *International Transactions in Operational Research* **13** 195–208.

Wood, D. R. 1997. An algorithm for finding a maximum clique in a graph. *Operations Research Letters* **21**(5) 211–217.

Publication IX

Lauri Häme, Harri Hakula. Routing by Ranking: A Link Analysis Method for the Constrained Dial-A-Ride Problem. Submitted to *Under review for publication in Operations Research Letters*, 16.1.2012 .

Routing by Ranking: A Link Analysis Method for the Constrained Dial-A-Ride Problem

Lauri Häme*,[1]

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Harri Hakula

Aalto University School of Science, PL 14100, 00076 Aalto, Finland

Abstract

Using a modified version of hyperlink-induced topic search (HITS), we characterize hubs as nodes in a directed graph with many out-links to other hubs and calculate a hub score for each node. Ranking the nodes by hub score gives guidance to a dynamic programming algorithm for efficiently finding feasible solutions to the dial-a-ride problem.

Key words: Dial-a-Ride Problem, Link Analysis, HITS, Topological Ordering

Introduction

Several web information retrieval (IR) methods have been developed for finding the most appropriate web pages corresponding to queries given to search engines. The most sophisticated methods, such as HITS [1], PageRank [2] and SALSA [3] in use today make use of the hyperlinked structure of the web, since the goodness of a web page and the position of the page with respect to other web pages seem to have a certain connection. For example, a web page may be considered good if there are many other web pages linking to that page. In other words, web pages are ranked by search engines not only by means of the content of the page, but also by exploiting information regarding the hyperlink-induced relationships between pages.

The bringing of hyperlinks to bear on the ordering of web pages has given rise to a mathematical analysis related to hyperlink-induced web IR methods, such as in [4–7], in which the behaviour of several IR methods is studied from the computational point of view.

In this work, we focus on the HITS (Hyperlink-Induced Topic Search) algorithm, which defines *authorities* (web pages with several inlinks) and *hubs* (several outlinks). The HITS thesis is that *good hubs point to good authorities and good authorities are pointed to by good hubs*. Based on this thesis, HITS assigns both a hub score and authority score to each web page [5]. In this paper, we use the term ”hub” similarly as in the above context and not in the context of travel dispatch centers.

We present an application of HITS on the dial-a-ride problem (DARP) [8–20], in which the goal is to construct a set of vehicle routes, serving a set of customers, satisfying the given time, capacity and precedence constraints. We examine the DARP as a *constraint satisfaction problem*, in which the goal is to find a set of m feasible vehicle routes that serve all customers, where m is the number of vehicles, or to prove that such a set of routes does not exist, as in [20]. In this reference, it is noted that an algorithm for checking the feasibility of a DARP instance has two main applications: 1) Determining the feasibility can be the first phase in an optimization algorithm in a *static* setting, where all trip requests are known, for example, one day in advance. 2) In *dynamic* services, a constraint satisfaction algorithm can be used for deciding whether to accept or reject incoming user requests. In contrast to most works related to vehicle routing and dial-a-ride problems, no specific cost function is considered. For approaches to vehicle routing and dial-a-ride problems with cost functions, we refer to [21–23] and [13, 19].

In the context of the dial-a-ride problem, we define links between nodes as feasible transitions with respect to the constraints of the problem: If node j can be visited after i, a link from i to j is formed. Thus, a good hub score of a pick-up or drop-off node i means that many nodes can be reached in time from i. Thus, in order to efficiently find feasible solutions to the dial-a-ride problem, we suggest that nodes with large hub scores should be visited first since there are many nodes that can be visited after such nodes.

This work is partially motivated by a dynamic demand-responsive transport (DRT) service currently being planned to operate in Helsinki. Helsinki Region Transport board has approved a plan under which the trial period of the service takes place from 2012 to 2014. Similarly as the current service routes [24], the new DRT service is designed to operate on a demand-responsive basis, that is, each trip is booked in advance and the vehicle routes are modified according to the trips. The main difference to existing services is that no pre-order times for trips are required and the trips can be booked ”on the fly” by means of an interactive user interface.

1. The Ranking Method

In the following sections we introduce a ranking method based on the HITS algorithm (Sections 1.1 and 1.2) and show

*Corresponding author

Email addresses: Lauri.Hame@tkk.fi (Lauri Häme), Harri.Hakula@tkk.fi (Harri Hakula)

[1]Tel.: +358 40 576 3585, Fax: +358 9 470 23016

how it can be used to solve the dial-a-ride problem (Section 1.3).

1.1. The HITS algorithm [1]

Given a web graph $G = (V, A)$ consisting of pages V and links A between pages, the authority and hub scores a_i and h_i are computed for each page i as follows. Letting (i, j) represent a link from page i to page j, given that each page has been assigned an initial authority score $a_i(0)$ and hub score $h_i(0)$, HITS successively refines these scores by computing

$$a_i(k) = \sum_{(j,i)\in A} h_j(k-1)$$
$$h_i(k) = \sum_{(i,j)\in A} a_j(k-1)$$

for $k \in \{1, 2, \ldots\}$. By using matrix notation, these equations can be written the form $a(k) = L^T h(k-1)$ and $h(k) = La(k-1)$, where $a(k)$ is the authority vector containing the authority scores of each of the pages at step k, $h(k)$ is the corresponding hub vector and L is the adjacency matrix of the graph with elements $L_{ij} = 1$ if $(i, j) \in A$ and $L = 0$ otherwise [5].

It has been shown in [4] that the authority and hub vectors describing the authority and hub scores of nodes of a given graph are given by the dominant eigenvectors of the matrices $L^T L$ and LL^T (or equivalently, dominant singular vectors of L), where L is the adjacency matrix of the graph.

1.2. Modified HITS

For constrained routing problems, we present a modified version of the HITS algorithm, in which only the hub scores are considered. More precisely, our thesis is that *good hubs point to good hubs*. This formulation is motivated by Theorem 1, which states that for a specific class of graphs, the hub score of a node i corresponds to the number of self-avoiding paths from i to a given destination node. When attempting to construct a path that visits all nodes, or to maximize the number of visited nodes, the modified HITS idea induces the following intuitive policy: Good hubs are visited first, since many nodes can be reached from good hubs.

The hub scores are calculated as follows. Let L denote the adjacency matrix of a directed graph G. Similarly as in the HITS algorithm, the hub vector containing the *hub scores* of nodes is first initialized, $h(0) = (1, 1, \ldots, 1)$ and the hub vector is successively updated by means of the power method

$$h'(k) = Lh(k-1). \tag{1}$$

Similarly as in the original HITS algorithm, the hub vector converges to a dominant eigenvector of L.

Theorem 1 characterizes the hub scores produced by the modified HITS algorithm for *sink graphs* defined as follows.

Definition. *Let $G = (V, A)$ be a directed acyclic graph and let $s \in V$ be a node such that $(s, i) \notin A$ for all $i \in V$. The graph $G_s = (V, A \cup (s, s))$ is called a* sink graph.

In other words, a sink graph is a directed acyclic graph (V, A) with the exception that one node $s \in V$ with zero outdegree is associated with a loop (s, s).

Theorem 1. *Let L denote the adjacency matrix of a sink graph $G_s = (V, A)$, where $V = \{1, \ldots, |V|\}$, let h_i denote the number of self-avoiding paths from i to s for $i \in V \setminus \{s\}$ and let $h_s = 1$. Then, $h = (h_1, \ldots, h_{|V|})^T$ is a unique dominant eigenvector of L.*

Proof. Since the adjacency matrix of a directed acyclic graph is an upper triangular matrix with zeros on the diagonal, the adjacency matrix of a sink graph is an upper triangular matrix with diagonal elements $L_{ii} = 0$ except for the sink node s, for which we have $L_{ss} = 1$. The eigenvalues of an upper triangular matrix are equal to the diagonal elements [25] and thus there exists a unique dominant eigenvalue 1. Let us show that $h = (h_1, \ldots, h_{|V|})^T$ is the corresponding eigenvector of L.

Since all paths in a directed acyclic graph are self-avoiding and by definition we have $h_s = 1$, the number of self-avoiding paths h_i from node i to the sink node s in the sink graph G_s satisfies $h_i = \sum_{j\in V} L_{ij} h_j$ for $i \in V \setminus \{s\}$ and h_s satisfies $h_s = 1 = L_{ss} h_s = \sum_{j\in V} L_{sj} h_j$. In matrix form, we have $h = Lh$ and thus h is the unique eigenvector corresponding to eigenvalue 1. □

Note that since $h_i \leq h_j$ for all $i, j \in V$ for which $L_{ij} = 1$, the vector h defines a *topological ordering* [26] of the nodes for which $h_i > 0$. Although Theorem 1 considers a special class of graphs[2], the result gives us an idea of the behaviour of the modified HITS method: There are many paths beginning from nodes with high hub scores.

In the following section, we show how the hub scores are used to give indicative guidance to a dynamic programming algorithm for the dial-a-ride problem.

1.3. The dial-a-ride problem

The dial-a-ride problem is defined as follows [20]. Let $G = (V, A)$ be a complete and directed graph with node set $V = \{0\} \cup P$, where node 0 represents the depot, and P represents the set of pick-up and drop-off nodes, where $(|P| = 2n)$. The set P is partitioned into sets P^+ (pick-up nodes) and P^- (drop-off nodes). Each arc $(i, j) \in A$ has a non-negative travel time T_{ij}. The travel times satisfy the triangle equation, that is, $T_{ij} + T_{jk} \geq T_{ik}$ for all $i, j, k \in R$. With each node $i \in V$ are associated a time window $[E_i, L_i]$, a service duration D_i and a load q_i, where $D_0 = 0$ and $q_0 = 0$. Let $H = \{1, \ldots, n\}$ be the set of customers and let $T^{\max}$ be the maximum ride time for any customer. With each customer i is associated a pickup node $i^+ \in P^+$, a delivery node $i^- \in P^-$ and a load $q_{i^+} = q_{i^-}$. The above parameters are illustrated in Figure 1.

Let $K = \{1, \ldots, m\}$ be the set of available vehicles, each with capacity Q. A *route* is a circuit over a set of nodes in P, starting and finishing at the depot 0. The goal is to construct m vehicle routes such that: (i) for every customer i, the pick-up node and the drop-off node are visited by the same route and the pick-up node is visited before the drop-off node; (ii) the load of the vehicles does not exceed the capacity Q at any time; (iii) the ride time of each customer is at most $T^{\max}$; (iv) the service at node i begins within the interval $[E_i, L_i]$.

[2] The calculation of self-avoiding paths in graphs with cycles is discussed in [27].

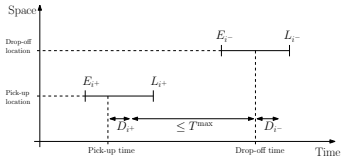


Figure 1: Time pick-up and drop-off time windows. The pick-up point of customer i is denoted by i^+ and the drop-off point is denoted by i^-. The customer should be picked up at i^+ within the time window $[E_{i^+}, L_{i^+}]$ and the customer should be dropped off at i^- within the time window $[E_{i^-}, L_{i^-}]$. The service times needed for the customer to get on the vehicle and get off the vehicle are denoted by D_{i^+} and D_{i^-}. The time between the drop-off and the pick-up (excluding D_{i^+}) should not exceed the maximum ride time $T^{\max}$.

1.3.1. Single-vehicle solution

In this section, we propose an exact constraint programming method for the single-vehicle DARP ($m = 1$) that produces a feasible solution whenever one exists or proves that the problem is infeasible. In the latter case, the algorithm returns a route that maximizes the number of served customers. The main challenge is to find a sequence of pick-up and drop-off nodes for which the time and precedence constraints are satisfied. While capacity constraints are also considered, the focus is on problems, in which time limits are more restrictive. The solution is extended to the multi-vehicle case in Section 1.4.

Letting node $0 \in V$ be the location of the vehicle at $t = 0$, we aim to find a feasible path $(0, p_1, \ldots, p_{2n}, 0)$ consisting of the pick-up and drop-off nodes of n customers. The number of permutations is $(2n)!$ and the number of feasible permutations with respect to precedence constraints is $(2n)!/2^n$ [17].

Briefly, our approach to the problem is a *depth-first* search, in which the remaining nodes are ranked by means of hub scores at each step and the order of the depth-first search is determined by the ranking. The search begins from node 0 by ranking the pick-up and drop-off nodes P of all customers. Then, the node p^* with the highest ranking is added to the sequence and the ranking procedure is repeated for the remaining nodes $P \setminus \{p^*\}$.

Formally, the search is executed by means of the recursion $\mathrm{REC}(S, R_S)$ shown in Algorithm 1, where S denotes a sequence of nodes (initially $S = (0)$), R_S denotes the set of remaining nodes at S (initially $R_S = \{1^+, 1^-, \ldots, n^+, n^-\}$) and $S_{\max}$ denotes the sequence that maximizes the number of served customers (initially $S_{\max} = (0)$). $C(S)$ denotes the number of served customers in sequence S, which is equal to the number of drop-off points in S.

The main idea of the recursion is that at each step, we have the current sequence S and the set of remaining nodes R_S (nodes that can possibly be added to the sequence). At first, we check the time, capacity and precedence constraints of S. If S is feasible and all customers have been served, that is, if $C(S) = n$, a feasible solution is found and the recursion is terminated. Otherwise, the remaining nodes R_S are ranked by hub scores and the infeasible nodes are removed from R_S (see Section 1.3.2). Then, the recursive function $\mathrm{REC}((S, i), R \setminus \{i\})$ is called for all sequences (S, i) in ranked order. In the following, we describe the ranking of remaining nodes in more detail.

```
Check time, capacity and precedence constraints;
if S is infeasible then
    Return;
end if
if C(S) > C(S_max) then
    S_max ← S;                          (Store the current sequence S.)
    if C(S) = n
     Terminate recursion;                (Feasible route.)
    end if
end if
Rank the remaining nodes i ∈ R_S and remove infeasible nodes from R_S;
(See Section 1.3.2.)
for all i ∈ R_S (in ranked order) do
    REC((S, i), R_S \ {i});
end for
```

Algorithm 1: A recursive solution $\mathrm{REC}(S, R_S)$ to the single-vehicle DARP. S denotes a sequence of nodes (initially $S = (0)$), R_S denotes the set of remaining nodes (initially $R_S = \{1^+, 1^-, \ldots, n^+, n^-\}$) and $S_{\max}$ denotes the sequence that maximizes the number of served customers (initially $S_{\max} = (0)$). $C(S)$ denotes the number of served customers in sequence S, which is equal to the number of drop-off points in S.

1.3.2. Ranking and pruning

The ranking of the remaining nodes R_S is determined in two phases: First, the adjacency matrix of the remaining nodes is determined by studying feasible transitions between the nodes. Then, the hub vector is calculated by means of Equation (1).

Adjacency matrix

Given that the vehicle is at the last node s of sequence S, we study which sequences (S, i, j), where $i, j \in R_S$, are feasible with respect to the time and precedence constraints. Let t_s denote the departure time and Q_s denote the load of the vehicle at s, and $t_i = t_s + T_{si}$ denote the arrival time at $i \in R_S$. For clarity, service times are assumed equal to zero. However, the modification of the following equations for non-zero service times is straightforward. In addition, we consider fixed time windows $[E_i, L_i]$ for each node. However, maximum ride time constraints [12] are taken into account by updating the upper bounds L_i of the time windows during run-time as described in [28].

First, all nodes that are seen to be non-reachable from S are removed from the set R_S of remaining nodes.

Definition. *A node $i \in R_S$ is said to be infeasible if there does not exist a feasible sequence $(S, \ldots, i)$ ending at i.*

Clearly, if the time of arrival t_i staisfies $t_i > L_i$, the node i is infeasible. The nodes $i \in I$ for which $t_i > L_i$ are removed from R_S, that is, $R_S \leftarrow R_S \setminus I$. This elimination criteria is similar to the one presented in [29].

For nodes that are not seen to be infeasible, we define feasible transitions as follows.

Definition. *The transition $i \rightarrow j$ from node $i \in R_S$ to node $j \in R_S$ is said to be* feasible*, if* $\max(t_i, E_i) + T_{ij} \leq L_j$.

The above definition is similar to the one studied in [30], in which the set of *admissible arcs* between the pick-up and delivery nodes is constructed as a preprocessing step in the pickup and delivery problem. The set of admissible arcs is made up of arcs which a priori satisfy the precedence, capacity and time constraints of the problem. The difference in our formulation is that we define the feasibility of transitions as a function of the state of the vehicle (see Figure 2).

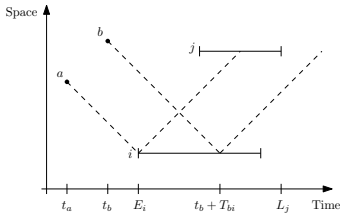


Figure 2: Feasible transitions. The figure shows the locations and time windows of two nodes i and j. If the vehicle left from a at the instant t_a, visited node i and moved directly to node j, the time constraint L_j would be satisfied. However, assuming that at the instant t_b the vehicle is located at b, the transition $i \rightarrow j$ is seen to be infeasible.

The elements of the adjacency matrix are defined for all $i, j \in R_S$ by $L_{ij} = 1$, if $i \neq j$ and $i \rightarrow j$ is feasible, $L_{ij} = 0$ otherwise.

Link analysis

After the adjacency matrix L has been determined, the hub vector $h = (h_1, \ldots, h_{|R_S|})$ of L is determined by Equation (1) (here the remaining nodes $i \in R_S$ are numbered from 1 to $|R_S|$ for clarity). In our formulation, a large hub score of node i means that many nodes can be visited after i. The hub ranking method is defined as follows.

Definition (Hub ranking)**.** *The remaining nodes $i \in R_S$ are sorted in descending order of the hub scores h_i: The branches are evaluated in the order $i_1, \ldots, i_{|R_S|}$, where $h_{i_1} \geq h_{i_2} \geq \ldots \geq h_{i_{|R_S|}}$.*

The hub scores are used to give guidance to the algorithm regarding the order in which the depth-first search visits the nodes. Since the goal is to maximize the number of served customers, the sequences that can not improve the maximum number of served customers, can be discarded by studying the neighborhoods of the remaining nodes.

Definition. *The neighborhood of node $i \in R_S$ is defined by $N(i) = \{i\} \cup \{j \in R_S \mid L_{ij} = 1\}$.*

Suboptimal sequences are detected as follows.

Theorem 2. *Let $S_{\max}$ denote the best found solution. The sequence S is suboptimal if*

$$C(S) + \max_{i \in R_S} C(N(i)) \leq C(S_{\max}),$$

where $C(S), C(N(i))$ and $C(S_{\max})$ denote the number of drop-off points in $S, N(i)$ and $S_{\max}$, respectively.

Proof. Suppose there exists a feasible sequence $(S, i_1, \ldots, i_h)$ for which $C((i_1, \ldots, i_h)) > \max_{i \in R_S} C(N(i))$. Then, since $\{i_1, \ldots, i_h\} \subset N(i_1)$, we have $C(N(i_1)) \geq C((i_1, \ldots, i_h)) > \max_{i \in R_S} C(N(i))$, which is a contradiction. □

1.3.3. A heuristic extension

Algorithm 1 describes an exact procedure for maximizing the number of served customers in a single route. In order to find the exact solution, the algorithm goes through all branches until no further nodes can be added to the current sequence or the sequence is seen to be suboptimal.

The effort of the algorithm can be controlled by limiting the number of branches that are evaluated by means of a positive parameter J. For example, if $J = 1$, the algorithm constructs a single sequence and stops when the set of remaining nodes in empty. By increasing J the search space is expanded and if $J = (2n)!/2^n$ (the number of permutations that satisfy precedence constraints), the heuristic coincides with the exact algorithm.

1.4. Multi-vehicle solution

In this section, we propose an exact approach to the multi-vehicle DARP as a constraint satisfaction problem. By using Algorithm 1 as a subroutine, the method produces a feasible solution to any instance or proves that the instance is infeasible. The main idea is that the vehicle routes are constructed one by one, each maximizing the number of served customers in the set of remaining customers. Customers are denoted by numbers $1, \ldots, n$ and vehicles are denoted by numbers $1, \ldots, m$.

First, a route is constructed for vehicle 1, serving as many customers as possible. Then, the process is repeated with vehicle 2 for the set of customers that were not served by vehicle 1 and so forth (see Figure 3). Letting X denote a set of customers, we denote by $\mathrm{SV}(X)$ the single-vehicle algorithm (Algorithm 1) that returns a route that maximizes number of served customers in set X and the corresponding set of served customers. This set will be referred to as a maximum cluster of X, which is formally defined as follows.

Definition. *Let X be a set of customers and $M \subset X$ be a subset of X, for which there exists a feasible route serving all customers in M. If $|M| \geq |Y|$ for all sets $Y \subset X$ for which there exists a feasible route serving all customers in Y, then M is a* maximum cluster *of X.*

Note that if there exists a feasible route serving all customers in X, we have $M(X) = X$. The solution to the multi-vehicle case is outlined in Algorithm 2.

for all $k \in \{1, \ldots, m\}$ **do**
  $[S_k, X'_k] \leftarrow \mathrm{SV}(X_k)$; ($S_k$ = route, X'_k = set of served customers)
  $X_{k+1} \leftarrow X_{k+1} \cup (X_k \setminus X'_k)$, where $X_{m+1} = X_1$;
end for

Algorithm 2: Outline of the multi-vehicle algorithm. X_k denotes the set of customers assigned to vehicle k and $\mathrm{SV}(X_k)$ denotes the single-vehicle subroutine presented in Algorithm 1 that returns a maximum cluster X'_k of X_k and the corresponding route S_k. Initially, all customers are assigned to the first vehicle, that is, $X_1 = \{1, \ldots, n\}$ and $X_k = \emptyset$ for all $k \in \{2, \ldots, m\}$.

If a feasible solution is not found directly by using the procedure described in Algorithm 2, the process is repeated. The approach is to find a set of customer-vehicle assignments such that for each vehicle there exists a feasible route serving all customers assigned to the vehicle or to prove infeasibility by going through all possible sets of customer-vehicle assignments [28].

We note that since the number of possible partitions of n customers into m sets is equal to the Stirling number of the second kind, that is, $|\mathcal{P}| = S_2(n, m) = \frac{1}{m!} \sum_{i=0}^{m} (-1)^i \binom{m}{i} (m - i)^n$, going through all possible partitions is computationally taxing. However, some instances are rendered infeasible by studying the feasibility of routes involving two customers: If there exist no feasible routes involving customers $\{i, j\}$, an arc between i and j is formed (customers i and j can not be assigned to the

same vehicle). Then, we find the *maximum clique* C within the set of customers, that is, the largest set of customers for which there is an arc between all pairs of nodes $i, j \in C$, see [31]. If the size $|C|$ of the maximum clique is greater than the number m of vehicles, the instance is infeasible.

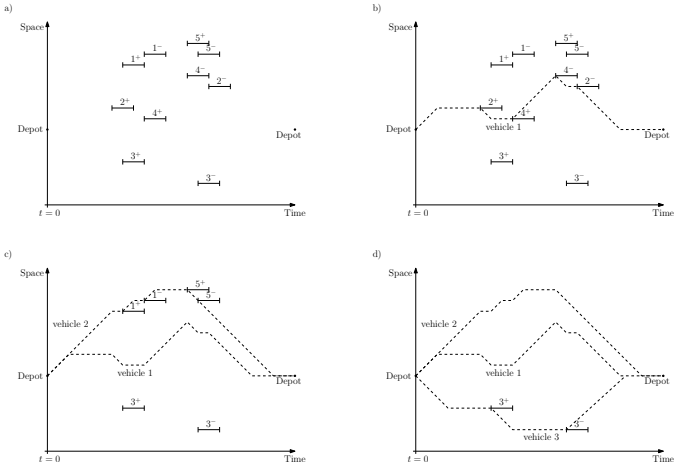


Figure 3: A one-dimensional example of the approach to the multi-vehicle problem involving five customers and three vehicles. The first route is constructed by maximizing the number of customers that can be served by a single vehicle (Figure b). Then, the customers 2 and 4 served by the first vehicle are removed from the set of remaining customers and the process is repeated for the second vehicle (Figure c). Finally, a route is construced for the third vehicle, serving the last remaining customer 3 (Figure d).

2. Structure and complexity

The structure of the single vehicle algorithm is as follows. Briefly described, each recursion step involves checking the the feasibility of sequences (S, i, j), where $i, j \in R_S$. Clearly, this process has a complexity of order $O(|R_S|^2)$, where $|R_S|$ is the number of remaining nodes. After the adjacency matrix L of the remaining nodes has been determined, the complexity of calculating k steps of the power method (Equation (1)) is of order $O(k|R_S|^2)$.

The overall complexity of finding a feasible solution in the single vehicle case is strongly dependent on the number of recursion steps needed to find the solution. Finding a feasible solution to a problem with n customers involves at least $2n$ recursion steps: At first, the number of remaining nodes is $2n$ and each time a feasible node is added to the end of the service sequence, the number of remaining nodes is decreased by one (see Figure 3). In the best case, no infeasible states are encountered and thus there are $2n$ recursion steps (one for each node added to the sequence). At step $k \in \{1, \ldots, 2n\}$, the number of remaining nodes is $2n+1-k$ and thus the complexity is of order $O(\sum_{k=1}^{2n}(2n+1-k)^2) = O(n^3)$. Roughly, the computational work is linearly increased with the number of dead ends M encountered during the recursion. The overall complexity of the single vehicle algorithm is thus $O(n^3 + M)$.

The multi-vehicle algorithm involves solving the single vehicle case for each vehicle in r iterations. Thus the complexity is bounded above by $O(rmn^3)$. Note that the complexity is in general lower since the size of the set of customers is n only the first vehicle. For the other vehicles, the number of customers is smaller since the customers that are already included in another vehicle route are not a part of the subproblem. For example, if the customers were divided equally among vehicles a priori, the complexity would be of order $O(rm(n/m)^3) = O(r(n^3/m^2))$.

3. Numerical experiments

In the following, we present extracts of the results of computational tests reported in [28]. An algorithm using the hub ranking idea is compared with two existing solution methods, namely, a tabu search algorithm [13] and a constraint programming (CP) algorithm [20].

The hub algorithm was implemented in Matlab and the tests were performed on a 2.2 GHz Dual Core Intel PC. The tabu and CP algorithms were tested on a 2.5 GHz Dual Core AMD Opteron computer [18].

In the studied instances the pick-up and drop-off points are located in a 20×20 square and the ride times between points (in minutes) are equal to Euclidean distances. The time windows have 15 minutes of length. In the first set of instances marked with a, the capacity of a vehicle equals $Q = 3$ and the load of each customer is 1. In the second set marked with b, we have $Q = 6$ and the load associated to each customer is chosen randomly according to a uniform distribution on the set $\{1, \ldots, Q\}$. In both sets, the maximum ride time is equal to $R = 22$ and the service time is proportional to the number of passengers, namely $d_i = d_{n+i} = q_i$. The instances are described in more detail in [16, 20, 32].

The results of the tests are shown in Table 3. The first column shows the instance labels of the form *am-n* or *bm-n*, where m indicates the number of vehicles and n corresponds to the number of customers. The other columns show the average time (in seconds) needed to solve the instances and the corresponding modifications by using the different solution methods, calculated over ten runs. The results obtained by the first two algorithms, tabu and CP, have been reported previously in [18, 20]. and the results for the hub algorithm have been reported in [28]. A number in parentheses indicates that the instance was proven to be infeasible, a dash indicates that a solution was not found in three minutes computing time and a star indicates that results for the instance have not been reported. The Matlab-implementations of the hub algorithm are available at `http://math.tkk.fi/~lehame/exact/`. The heuristic version of the algorithm with $J = 1$ was used (see Section 1.3.3).

By looking at the results obtained by the hub algorithm we see that most instances were solved within a fraction of a second. The CPU times obtained by the hub algorithm are typically of order ten times smaller compared to the results of the tabu and CP algorithms. Although the algorithms were tested on different platforms, the results seem to justify the efficiency of the hub algorithm on the test instances. We acknowledge that there are instances in wich tabu and CP produced a feasible solution or proved infeasibility faster than the hub algorithm. However, the results suggest that the multi-vehicle hub algorithm may have practical importance since it is capable of handling large problems in short computation times.

Instance	Tabu	CP	Hub	Instance	Tabu	CP	Hub
a4-40	0.5	0.3	0.05	b4-40	0.4	0.3	0.05
a4-48	1.3	0.4	0.05	b4-48	-	(0.1)	(0.32)
a5-40	0.3	0.3	0.02	b5-40	-	(0.1)	(0.27)
a5-50	0.6	0.6	0.03	b5-50	1.1	0.9	0.18
a5-60	1.5	0.9	0.05	b5-60	1.6	1.2	0.04
a6-48	0.4	0.7	0.03	b6-48	*	*	0.02
a6-60	-	(1.1)	(0.63)	b6-60	1.0	1.4	0.04
a6-72	-	(1.7)	(0.77)	b6-72	2.4	2.4	0.05
a7-56	-	(0.6)	(0.40)	b7-56	0.5	1.5	0.04
a7-70	1.4	7.6	0.06	b7-70	1.3	2.7	0.05
a7-84	3.0	4.1	0.08	b7-84	-	(0.1)	(0.64)
a8-64	-	(1.9)	(0.84)	b8-64	0.8	1.9	0.04
a8-80	1.8	6.0	0.07	b8-80	-	(0.5)	(1.28)
a8-96	-	(3.7)	(1.72)	b8-96	3.9	56.1	0.15

Table 1: Comparison between a tabu search algorithm [13], a constraint programming algorithm [18] and the hub algorithm. The results are given in [18, 20, 28]. The upper table shows the average time (in seconds) needed to solve the instance of the dial-a-ride problem in the first column by using the different solution methods, calculated over ten runs. The times without parentheses indicate that a feasible solution was found and the times in parentheses indicate that the instance was proven to be infeasible. A star (*) indicates that the computing time has not been reported.

4. Conclusions

In this work, an efficient dynamic programming method for the time constrained dial-a-ride problem is suggested. The purpose of the method is to produce routes that satisfy the constraints of the problem, since finding such routes is seen to be a major challenge in highly constrained problems. The main result is an exact constraint satisfaction algorithm for the multi-vehicle case.

Our approach is motivated by HITS, a link-analysis algorithm originally developed for web information retrieval. We define links between the pick-up and drop-off nodes as feasible transitions with respect to the constraints of the problem, *hubs* as nodes with several out-links. Ranking the nodes by hub score gives guidance to a depth-first search used to maximize the number of served customers in a single vehicle route. The method is extended to solve the multi-vehicle case. Numerical experiments suggest that the proposed ranking method is capable of producing feasible solutions to large problems in relatively small computation times.

This work is an example on how recommendation-type link analysis can be used to solve combinatorial problems. Although we have focused on the Dial-A-Ride Problem (DARP), we note that the DARP is a generalization of the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) and thus the proposed hub ranking method could be useful in solving other constrained routing problems as well.

References

[1] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: Proceedings of the Ninth Annual ACM-SIAM Symposium of Discrete Algorithms, ACM Press, New York, 1998, pp. 668–677.

[2] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the Seventh International Conference on World Wide Web, Vol. 7, 1998, pp. 107–117.

[3] R. Lempel, S. Moran, The stochastic approach for link-structure analysis (salsa) and the tkc effect, in: The 9th Intl. WWW Conference, 2000.

[4] A. Farahat, T. LoFaro, J. C. Miller, G. Rae, L. Ward, Authority rankings from hits, pagerank, and salsa: existence, uniqueness, and effect of initialization, SIAM Journal on Scientific Computing 27 (2006) 1181–1201.

[5] A. Langville, C. Meyer, A survey of eigenvector methods of web information retrieval, SIAM Review 47 (2005) 135–161.

[6] A. Y. Ng, A. X. Zheng, M. I. Jordan, Stable algorithms for link analysis, SIGIR01, New Orleans, Louisiana, USA.

[7] M. Agosti, L. Pretto, A theoretical study of a generalized version of kleinberg's hits algorithm, Information Retrieval 8 (2005) 219–243.

[8] H. Psaraftis, An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows, Transportation Science 17 (1983) 351–357.

[9] J. Jaw, A. Odoni, H. Psaraftis, N. Wilson, A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows, Transportation Research Part B 20 (1986) 243–257.

[10] O. Madsen, H. Ravn, J. Rygaard, A heuristic algorithm for the a dial-a-ride problem with time windows, multiple capacities, and multiple objectives, Annals of Operations Research 60 (1995) 193–208.

[11] P. Toth, D. Vigo, Heuristic algorithms for the handicapped persons transportation problem, Transportation Science 31 (1997) 60–71.

[12] B. Hunsaker, M. W. P. Savelsbergh, Efficient feasibility testing for dial-a-ride problems, Operations Research Letters 30 (2002) 169–173.

[13] J.-F. Cordeau, G. Laporte, A tabu search heuristic for the static multi-vehicle dial-a-ride problem, Transportation Research B 37 (2003a) 579–594.

[14] M. Diana, M. Dessouky, A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows, Transportation Research Part B 38 (2004) 539–557.

[15] K. Wong, M. Bell, Solution of the dial-a-ride problem with multi-dimensional capacity constraints, International Transactions in Operational Research 13 (2006) 195–208.

[16] J.-F. Cordeau, A branch-and-cut algorithm for the dial-a-ride problem, Operations Research 54 (2006) 573–586.

[17] L. Häme, An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows, European Journal of Operational Research. 209 (1) (2011) 11 – 22.

[18] G. Berbeglia, Complexity Analyses and Algorithms for Pickup and Delivery Problems, Ph.D. Thesis, HEC Montreal, 2009.

[19] G. Berbeglia, J.-F. Cordeau, G. Laporte, A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem, forthcoming (2011).

[20] G. Berbeglia, G. Pesant, L.-M. Rousseau, Checking feasibility of the dial-a-ride problem using constraint programming, forthcoming (2011).

[21] O. Bräysy, M. Gendreau, Vehicle routing problem with time windows, part i: Route construction and local search algorithms, Transportation Science 39 (1) (2005) 104118.

[22] O. Bräysy, M. Gendreau, Vehicle routing problem with time windows, part ii: Metaheuristics, Transportation Science 39 (1) (2005) 119139.

[23] C. Groër, B. Golden, E. Wasil, A library of local search heuristics for the vehicle routing problem, Mathematical Programming Computation 2 (2).

[24] Helsinki Region Traffic, Jouko neighbourhood routes and service routes (Oct. 2010).
URL `http://www.hsl.fi/EN/passengersguide/`

[25] S. Axler, Linear Algebra Done Right, Springer-Verlag, ISBN 0-387-98258-2, 1996.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms (2nd ed.), MIT Press and McGraw-Hill, 2001.

[27] J. Ponstein, Self-avoiding paths and the adjacency matrix of a graph 14 (3) (1966) 600–609.

[28] L. Häme, H. Hakula, A maximum cluster algorithm for checking the feasibility of dial-a-ride instances, manuscript submitted to Transportation Science.

[29] J. Desrosiers, Y. Dumas, F. Soumis, A dynamic programming solution of the large-scale singlevehicle dial-a-ride problem with time windows, American Journal of Mathematical and Management Sciences 6 (1986) 301–325.

[30] Y. Dumas, J. Desrosiers, F. Soumis, The pickup and delivery problem with time windows, European Journal of Operational Research 54 (1991) 7–22.

[31] D. R. Wood, An algorithm for finding a maximum clique in a graph, Operations Research Letters 21 (5) (1997) 211–217.

[32] S. Ropke, J.-F. Cordeau, G. Laporte, Models and branch-and-cut algorithm for pickup and delivery problems with time windows, Networks 49 (2007) 258–272.