The Simultaneous Vehicle Scheduling and Passenger Service Problem

Hanne L. Petersen, Allan Larsen, Oli B.G. Madsen
Bjørn Petersen, Stefan Ropke

DTU Transport
Technical University of Denmark
Bygningstorvet 1, DK-2800 Kgs. Lyngby
{bhp,ala,ogm,bpe,sv}@transport.dtu.dk

September 4, 2009

Abstract

Passengers using public transport systems often experience waiting times when transferring between two scheduled services. In this paper we propose a planning approach which seeks to obtain a favourable trade-off between the two contrasting objectives passenger service and operating cost by modifying the timetable. The planning approach is referred to as the Simultaneous Vehicle Scheduling and Passenger Service Problem (SVSPSP). The SVSPSP is modelled as an integer programming problem, and solved using a large neighborhood search (LNS) metaheuristic. The proposed framework is tested on data inspired by the express-bus network in the Greater Copenhagen Area. The results are encouraging and indicate a potential decrease of passenger waiting times in the network of 10–20%, with the vehicle scheduling costs remaining unaffected.

1 Introduction

In every larger public transport system massive amounts of time are wasted due to waiting time when transferring between different parts of the journey. For the Greater Copenhagen area it has been estimated that the time lost on an average weekday by passengers waiting for connecting buses or trains approaches 65,000 hours (based on 400,000 daily transfers with an average of 10 minutes transfer waiting time\footnote{http://www.dtu.dk/centre/modelcenter/TU/Standard\%20tabeller/}). Hence, generating timetables which optimise for temporal correspondences has an enormous socio-economic potential. Clearly, this could be achieved through an increase in the frequency of the trips offered in
the timetable, however this would require an unacceptable increase in operating costs.

The traditional sequential framework for planning of public transport has been excellently described by Desaulniers and Hickman [2007] and is sketched in Figure 1. Given the route network, the frequencies are determined to ensure demand coverage and to comply with politically determined service levels, under practical constraints such as fleet size. The timetabling process then determines the exact timings for all trips while respecting the previously determined frequencies/headways. Both of these first phases are concerned with maximising some measure of passenger service, and are carried out by the public transport service provider, who typically works by appointment by the local authorities. The timetabling phase may take schedule synchronisation and transfer times into account.

![Diagram of planning phases](image)

**Figure 1: Traditional sequential planning approach**

Once the timetable has been established, the resource scheduling starts. During this phase the first problem to be solved is the scheduling of the physical resources necessary to carry out the trips in the timetable, i.e. the vehicles. The purpose of the vehicle scheduling is to be able to execute the timetable at the lowest possible cost. The costs considered in this phase include empty mileage performed by the vehicles, both in connection to the depot, and in the form of deadheading, i.e. transport between the end point of one trip and the starting point of another. Once the vehicle schedules have been established, the crew pairing and rostering phases are carried out. The last three phases are all carried out by the public transport operator, who is appointed by the service provider to operate a set of trips, and they all have the purpose of operating the requested timetable at the lowest possible cost.

Today, efficient systems for generating near-optimal vehicle schedules exist within all modes of transport. However, these systems treat the timetable as fixed input, meaning that potential savings in operating costs from moving a set of trips
in the timetable are lost. Only very limited research has been done on models that address the problem of minimising the operating costs by modifying the timetable. Furthermore, research is scarce on models that focus on minimisation of the waiting time during transfer.

In this paper we introduce the Simultaneous Vehicle Scheduling and Passenger Service Problem (SVSPSP) which addresses the multiple objective planning problem of improving timetables such that they remain economically satisfactory for the operator, and at the same time offer high-quality service to the passengers by reducing the unproductive time spent on waiting during transfers. Please note that whenever we refer to waiting time throughout this paper we are solely referring to the waiting time associated with transfers, and not the waiting time of passengers entering the system. The SVSPSP framework is sketched in Figure 2, and integrates the planning processes of timetabling and vehicle scheduling.

![Diagram of SVSPSP framework]

Figure 2: The role of the SVSPSP shown in relation to the traditional sequential planning approach.

Its main input is the original timetable and estimates of passenger demand in the network. The natural problem owner of the SVSPSP is the public transport service provider, as this is the authority which on the one hand is committed to provide a high-quality timetable to the customers (in terms of e.g. minimum waiting times) and on the other hand holds the responsibility of ensuring that the offered timetable is feasible from an operating costs perspective. By integration of the vehicle scheduling phase, which previously belonged to the operator, the service provider can obtain a better negotiating position towards the operator, since the operating costs have already been considered during the optimisation of the timetables.

The contributions of this paper are fourfold: 1) we formally introduce a new interesting problem, motivated by a real-life case, 2) we make a realistic data set available, that can be used for future studies, 3) we propose a heuristic
solution method that is able to handle data sets of realistic size, 4) we show that substantial reductions in passenger waiting time are possible using the proposed methodology. The paper is organised as follows: Section 2 reviews the literature on the multiple depot vehicle scheduling problem as well as work on minimising passenger transfer times. In section 3 we formulate the SVSPSP as an integer programming model. Section 4 discusses how the proposed problem can be solved by the large neighborhood search metaheuristic. Section 5 introduces the data set used in this study which is based on the bus network of the Greater Copenhagen area, and in Section 6 we discuss the results obtained. Finally, we provide our concluding remarks and suggest directions for further research in Section 7.

2 Literature review

Our approach for the integrated vehicle scheduling and timetabling problem is based on the multiple depot vehicle scheduling problem (MDVSP). Desrochers et al. [1995] provide an excellent introduction to the problem and survey the literature prior to 1995. A more recent, but short literature survey is presented by Pepin et al. [2009] who also presents an interesting comparison of heuristic approaches for the problem. Section 4.1 in Desaulniers and Hickman [2007] also contains a recent survey. Some of the currently best exact methods for the MDVSP are proposed by Hadjar et al. [2006] and Löbel [1999]. We are aware of two papers that extend vehicle scheduling problems to handle parts of the timetabling process. The paper by van den Heuvel et al. [2008] studies the integration of timetabling and multi depot vehicle scheduling with the aim of reducing costs (reducing the number of vehicles) while ignoring passenger waiting times. On the timetabling level the approach allows the trip starting times for each line to be shifted in time to allow greater flexibility in the vehicle scheduling part. The paper presents integer programming models as well as a local search algorithm that solves a network flow problem in each local search iteration. Guhaire and Hao [2008] also integrate vehicle scheduling and timetable synchronisation in their optimisation problem. They consider several terms in their objective: number of vehicles required, number and quality of transfer possibilities and the so-called headway evenness. The second term aims at minimising passenger inconvenience. The last term attempts to make arrivals of vehicles, serving a particular line, occur with a regular frequency. The three terms are weighted together. In terms of the vehicle scheduling problem, the paper considers a single depot setup while our approach handles the multiple depot case. The problem studied in this paper is probably the one that resembles our problem the most.

Several papers focus on optimising timetables in order to minimise passenger waiting times, without explicitly considering the impact such changes have on the physical resource requirements (e.g. more buses may be needed to carry out
the modified plan). Examples of such approaches are Jansen and Pedersen [2002] who formulate the problem as a mathematical model and propose simulated annealing and tabu search algorithms to solve the problem (see also Pedersen [2003]); Ceder et al. [2001] who synchronise bus timetables by maximising the number of times two buses arrive at the same time at any node in the network; Kleint and Stemme [1988], Bookbinder and Désilets [1992] and Daduna and Voß [1993] who synchronise timetables by solving a quadratic semi-assignment problem. Worth mentioning is also the paper by Chakroborty et al. [2001], which studies timetable synchronisation and “optimal fleet size” using a genetic algorithm heuristic. They do not study the vehicle scheduling aspect of the problem, instead the term “optimal fleet size” refers to the fact that the number of departures on a specific line is a variable, decided by the proposed model.

As explained in the introduction, SVSPSP integrates the timetabling and vehicle scheduling phases. The integrated problem has not been widely studied in the literature but some papers on the topic do exist. One approach for handling the integrated problem has been the so-called periodic event scheduling problem (PESP). The PESP is mainly used for timetabling but has been extended to handle some aspects of vehicle scheduling as well. The PESP model was proposed by Serafini and Ukovich [1989]. It is a general framework for modelling optimisation problems with a periodic nature. Liebchen and Möhring [2007] show how the PESP and extensions can be used to handle many aspects of railway timetabling. One of these is to minimise the changeover time for passengers and another is the minimisation of the number of vehicles needed to perform the timetable. The complexity of the vehicle minimisation depends on whether trains are allowed to switch line when they reach their endpoint. Contrary to our approach the paper does not model the situation where vehicles can perform deadheading in order to switch terminal (this does not seem practical when the vehicles are trains running on tracks, but can be useful for buses). The material in Liebchen and Möhring [2007] builds on the work of Liebchen and Peeters [2002] which focuses on vehicle minimisation, but arrives at a model with a quadratic objective function. Other recent works on the PESP and railway timetabling include Liebchen and Möhring [2002], Peeters [2003], and Kroon et al. [2007].

Wong et al. [2008] studies the Mass Transit Railway in Hong Kong that contains 6 train lines. They minimise the overall passenger waiting time in a non-periodic fashion. The number of vehicles needed to carry out the plan is determined in advance and is kept constant. In this way it is ensured that the proposed timetable does not become too expensive to carry out, while optimising customer satisfaction. The authors present a MIP model and solve it using a heuristic that incorporates a standard MIP solver as an important component. Fleurent et al. [2007] describe an optimisation system and an interactive tool for minimising passenger waiting time while keeping vehicle costs under control. The suggested approach is tested on a case from the city of Montreal, Canada, and the results indicate that the passenger waiting time can be improved while keeping the vehicle count constant. The paper provides little detail about the optimisation
algorithm used to obtain these results.

We can conclude that the work on integrating time tabling and vehicle scheduling is rather limited and that Guihaire and Hao [2008] is the paper that presents a problem that is most similar to the SVSPSP. The SVSPSP model is, regarding some aspects, more ambitious than the model studied by Guihaire and Hao [2008] as it considers a multi-depot setting which is not the case in the aforementioned paper.

3 The SVSPSP: modelling

In a classical multi-depot vehicle scheduling problem (MDVSP) one has to cover a set of trips with a set of vehicles (based at several depots) while minimising costs. A trip has a start and end location, as well as a departure and arrival time. In a bus scheduling setting a trip corresponds to the movement from the start to the end of a bus line. A line is a collection of trips that have the same start and end locations but different departure and arrival times. A line also contains trips going in the opposite direction. The MDVSP can be modelled as follows (see Desrosiers et al. [1995]): let \( N = \{1, \ldots, n\} \) denote the set of trips and \( K \) the set of depots. With each depot \( k \in K \) we associate a graph \( G^k = (V^k, A^k) \) where the set of nodes is defined as \( V^k = N \cup \{n+k\} \) with \( n+k \) being the node representing the \( k^{th} \) depot. The set of arcs \( A^k \) is a subset of the set \( V^k \times V^k \), with all infeasible arcs removed. An arc is infeasible if it forms an impossible connection between two trips; typically this is caused by timing constraints. For each depot \( k \in K \) and each arc \((i,j) \in A^k \) we define an arc cost \( c_{ij}^k \) and we are given an upper bound \( v^k \) on the number of vehicles located at \( k \). Using a binary variable \( x_{ij}^k \) for all \( k \in K, (i,j) \in A^k \), having value 1 if and only if a vehicle from depot \( k \) travels from node \( i \) to \( j \) we can write an integer multi-commodity flow model as follows:

\[
\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k 
\]

subject to

\[
\sum_{k \in K} \sum_{j \in V^k} x_{ij}^k = 1 \quad i \in N 
\]

\[
\sum_{k \in K} \sum_{j \in V^k} x_{n+k,j}^k \leq v^k \quad k \in K 
\]

\[
\sum_{i \in V^k} x_{ij}^k - \sum_{i \in V^k} x_{ji}^k = 0 \quad k \in K, j \in V^k 
\]

\[
x_{ij}^k \in \{0,1\} \quad k \in K, (i,j) \in A^k 
\]

The objective (1) minimises the total cost. The arc costs \( c_{ij}^k \) can be set such that the total cost reflects a fixed cost per vehicle and deadheading costs. Constraints
(2) ensure that all trips are served, constraints (3) ensure that we do not use more than the available number of vehicles and, constraints (4) are flow conservation constraints.

The SVSPSP generalises the MDVSP as follows: in the SVSPSP we group trips into so called metatrips. The set of metatrips, \( \Omega \), forms a partitioning of the set \( N \), that is, \( \cup_{M \in \Omega} M = N \) and \( \forall M_1, M_2 \in \Omega, M_1 \neq M_2 : M_1 \cap M_2 = \emptyset \). Furthermore, we relax the condition that every trip must be covered. Instead we require that exactly one trip from each metatrip must be covered. In the context of this paper, we assume that each metatrip corresponds to a trip from the original timetable, and the (sub)trips belonging to the metatrip represent copies of the original trip, with alternative departure times. Thus, the requirement that each metatrip is covered corresponds to the MDVSP-requirement that each trip is covered (2). The idea behind this, in relation to our goal of increasing passenger service, is that selecting alternative departure times may reduce waiting times and thereby improve the passenger service level.

We will now introduce some useful concepts that will be used in our treatment of the SVSPSP. Trips in the SVSPSP model can be *incompatible* for various reasons, as we shall see later. This is captured by a set \( \Phi \subseteq 2^N \) containing sets of mutually incompatible trips. Thus, if \( \phi \in \Phi \) then any pair \( i, j \in \phi \) is incompatible and cannot be used together in a feasible solution. For the SVSPSP we maintain the definition of a *line* that is known from the MDVSP; a line \( L \) is a sequence of stops to be visited in a given order. A line can be travelled in both directions, and we use the term d-line (directed line) for a line in a particular direction. Each metatrip, and the trips contained in it, belongs to exactly one d-line. Therefore we can view a d-line \( L \) as a subset of the set of metatrips: \( L \subseteq \Omega \). For every bus line a number of stops are defined. The stops are the locations where the bus stops to pick up and unload passengers. Several bus lines may share one stop and a stop can provide connection to other modes of timetabled transportation like trains or ferries. Any transfer of passengers takes place at a stop. We are only interested in stops where transfers can take place, hence, when mentioning stops in the rest of this paper we assume a stop with at least one transfer opportunity.

Figure 3 shows an example of trips and metatrips. The nodes \( \{ 1, \ldots, 12 \} \) represent trips, and two metatrips \( \{ 2, \ldots, 6 \} \) and \( \{ 7, \ldots, 11 \} \) are shown. The time of day is shown along the top of the figure. Trips 4 and 9, marked with grey, are the two original trips, from which the metatrips are constructed. The remaining trips in each metatrip are constructed by creating duplicates of the original trip, spread evenly in the available time interval. The nodes 1 and 12 belong to other metatrips, not illustrated in the figure. All trips shown in the figure belong to the same d-line.

The usage of incompatible trips to impose passenger service is apparent: trips belonging to the same d-line and departing within a short time interval should be incompatible, for example trip 6 and 7 on Figure 3 could be incompatible because they depart within 4 minutes. Similarly, two consecutive departures on
a d-line should not be too far apart. Therefore it would make sense to make trip 2 incompatible with trip 11. If departures at regular intervals are required on a bus line for a specific period of the day or the entire day this could also be modelled using incompatible trips. If we desire departures every 20 minutes in the example on Figure 3 we must make trip 2 incompatible with trips 8, 9, 10, and 11 (by adding the set \{2, 8, 9, 10, 11\} to \(\Phi\)), trip 3 should be incompatible with trips 7, 9, 10, and 11, and so on.

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c}
9.20 & 9.30 & 9.40 & 9.50 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
\end{array}
\]

\text{metatrip 1} \hspace{1cm} \text{metatrip 2}

Figure 3: Example of trips and metatrips.

Using the notation from the MDVSP we can now present a mathematical model for a simple version of the SVSPSP, denoted SVSPSP\(^0\).

\[
\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} x_{ij}^k
\]

subject to

\[
\sum_{i \in M} \sum_{k \in K} \sum_{j \in V^k} x_{ij}^k = 1 \quad M \in \Omega \tag{7}
\]

\[
\sum_{i \in \phi} \sum_{k \in K} \sum_{j \in V^k} x_{ij}^k \leq 1 \quad \phi \in \Phi \tag{8}
\]

\[
\sum_{j \in N} x_{n+k,j}^k \leq v^k \quad k \in K \tag{9}
\]

\[
\sum_{i \in V^k} x_{ij}^k - \sum_{i \in V^k} x_{ji}^k = 0 \quad k \in K, j \in V^k \tag{10}
\]

\[
x_{ij}^k \in \{0, 1\} \quad k \in K, (i, j) \in A^k \tag{11}
\]

Constraints (9) and (10) are identical to (3) and (4) in the original MDVSP formulation. Constraints (7) ensure that exactly one trip from each metatrip is selected and constraints (8) ensure that no incompatible trips are selected at the same time.

In order to discuss how passenger service can be taken into account in the SVSPSP\(^0\) we need to define exactly how we measure passenger service. The area we focus on in relation to passenger service is waiting time during transfers. We first introduce the central concept \textit{transfer opportunity}. A transfer opportunity is a triple \((s, M, L)\). Here \(s\) is the stop where the transfer takes place, \(M\) is a
metatrip that stops at $s$, and $L$ is a connecting line that exchanges passengers with $M$ at $s$. For each transfer opportunity we assume that an estimate $D^s_{ML}$ of the number of passengers disembarking metatrip $M$ and transferring to line $L$ at stop $s$, as well as an estimate $E^s_{ML}$ of the number of passengers embarking metatrip $M$ transferring from line $L$ at stop $s$ are available. It is assumed that all passengers disembarking a metatrip to transfer to line $L$ take the earliest possible departure on line $L$ and all passengers embarking a metatrip $M$ come from the latest possible arrival on line $L$. For the SVSPSP$^0$, $L$ is a line external to the model, but we will later generalise it to include those lines that are rescheduled by the model.

To improve passenger service we desire to minimise the total number of passenger minutes wasted by waiting for a connection, at the same time as we want to minimise the cost of serving all trips. This results in two goals that are weighted together in the cost coefficients of the objective function. The SVSPSP$^0$ model can accommodate a part of the waiting times that we desire to include in the model, namely a penalty for waiting times related to lines that are external to the model, such as already timetabled train departures: for each trip $i$ in $N$ we find the transfer opportunities $(s, M, L)$ of the metatrip $M$ that $i$ belongs to. As stated above, $L$ is an external line with fixed departures and arrivals, therefore we can a priori find the arrival and departure on line $L$ that are used by passengers embarking and disembarking trip $i$ at stop $s$ and we can calculate the associated waiting times. The two waiting times are multiplied by the passenger estimates $E^s_{SM}$ and $D^s_{SM}$ and summed to give the total number of minutes waited for the particular trip and transfer opportunity. By summing over all the transfer opportunities that the trip is involved in we obtain the total number of waiting minutes incurred by the trip. This number, weighted in a suitable way, is added to the cost of all arcs leaving the node corresponding to the trip.

The SVSPSP$^0$ model cannot take the transfer of passengers from bus to bus into account if both buses are rescheduled by the model. We therefore introduce the model SVSPSP, that generalises SVSPSP$^0$ to accommodate this. The overall idea is to introduce two new sets of binary variables $y^s_{ij}$ and $z^s_{ij}$ that indicate if transfers between trip $i$ and $j$ are taking place at stop $s$. For each transfer opportunity $(s, M, L)$ involving a d-line $L$ which is timetabled by the model we create a number of variables $y^s_{ij}$ where $i \in M$, $j \in \cup_{M \in L} M'$. Each variable indicates if the transfer opportunity of passengers disembarking metatrip $M$ to transfer to d-line $L$ is realised by transferring from trip $i$ to $j$. Similarly, for the same transfer opportunity, we create a number of variables $z^s_{ij}$ where $j \in M$, $i \in \cup_{M \in L} M'$. These variables indicate if the transfer opportunity of passengers embarking $M$, coming from $L$ is realised by transferring from trip $i$ to $j$. We assign a cost $\tilde{c}^s_{ij} > 0$ for each $y^s_{ij}$ variable and a cost $\tilde{c}^s_{ij} > 0$ for each $z^s_{ij}$ variable. The cost is based on the time between arrival and departure on the two trips and the number of passengers expected to take advantage of the transfer opportunity.

Consider the following example: the bus lines 200 and 300 both visit Lynghby.
Station. Assume that a trip for line 200 northbound (200-N) has been chosen by the model such that the bus arrives at Lyngby station at 9:29. A number of the passengers on board the bus wish to disembark the bus to transfer to line 300 heading north (300-N). Their waiting time depends on the departure time of the next 300-N, which is also decided by the model. Figure 4 shows this situation. The chosen trip for bus 200-N (trip a) is shown at the top of the figure along with alternative 200-N arrivals and nine trips belonging to line 300-N are shown on the bottom. Passengers from trip a cannot transfer to bus 300-N on the departure times marked with grey circles: departure 4 is impossible because it departs before bus 200-N arrives, while departure 5 departs one minute later than trip a arrives and there is not enough time for the transfer (passengers have to walk). The other departures are all feasible transfers. Note that trips 7 to 11 constitute a metatrip, so exactly one of these trips must be selected. This means that no passenger from trip a heading for line 300-N would transfer to trip 12 because an earlier, feasible departure will exist in the plan. On the other hand, if trip 12 is selected by the model and trip a is the latest selected bus from 200-N that allows a transfer to trip 12 then embarking passengers on trip 12 arriving from 200-N would perform the transfer. Since both embarking and disembarking passengers are considered, both y and z variables are necessary. The y variables handle passengers disembarking a specific trip to the first possible trip on the specified d-line. The z variables handle passengers embarking a specific trip from the last possible trip on the specified d-line.

Let S be the set of all stops that are visited by more than one bus line. We introduce a graph $G^s = (V^s, A^s)$ for each stop $s \in S$. The set of vertices $V^s$ is the set of all trips that visit stop $s$ and the set of arcs is defined as

$$A^s = \{ (i, j) : i, j \in V^s, \text{ passengers can transfer from trip } i \text{ to trip } j \text{ at stop } s \}.$$

For example, if $s$ is Lyngby station as shown in Figure 4 we would have that

$$\{ (a, 6), (a, 7), (a, 8), (a, 9), (a, 10), (a, 11), (a, 12) \} \subset A^s$$

but $\{ (b, 1), (b, 2) \} \cap A^s = \emptyset$. The variables $y^s_{ij}$ and $z^s_{ij}$ are defined for every $s \in S$ and every arc $(i, j) \in A^s$. We can use Figure 4 to show the meaning of the y variables. If, for example, trips b and 7 are chosen and none of the trips $\{ 3, 4, 5, 6 \}$ are chosen then $y^s_{b,7} = 1$ and $y^s_{b,j} = 0$ for $j \in \{ 3, 4, 5, 6, 8, 9, 10 \}$. If
both trip 3 and 7 were chosen then we would have \( y_{b3}^i = 1 \) and \( y_{b7}^i = 0 \) because all passengers disembarking \( b_i \) bound for 300-N, would transfer to trip 3.

For a trip \( i \in N \) and a stop \( s \) on its line we define \( t(i, s) \) to be the departure time of trip \( i \) at stop \( s \). For a trip \( i \) we define \( dl(i) \) to be the d-line that the trip belongs to. For a stop \( s \) and an arc \((i, j)\) \( \in \hat{A}^s \) we define

\[
\pi(i, j, s) = \{ j' \in \cup_{M' \in dl(i)} M' : (i, j') \in \hat{A}^s, t(j') < t(j) \},
\]

that is, \( \pi(i, j, s) \) is the set of trips \( j' \) from the same d-line as \( j \) that are earlier than \( j \) but that still are feasible transfer destinations from trip \( i \). Similarly we define

\[
\sigma(i, j, s) = \{ i' \in \cup_{M' \in dl(i)} M' : (i', j) \in \hat{A}^s, t(i) < t(i') \},
\]

which is the set of trips \( i' \) from the same d-line as \( i \) that are later than \( i \) but where a transfer to trip \( j \) still is feasible. We can now present an extended model that also handles the bus-to-bus transfers:

\[
\begin{aligned}
\min & \sum_{k \in K} \sum_{(i,j) \in \hat{A}^k} c_{ij}^k x_{ij}^k + \sum_{s \in S} \sum_{(i,j) \in \hat{A}^s} c_{ij}^s y_{ij}^s + \sum_{s \in S} \sum_{(i,j) \in \hat{A}^s} c_{ij}^s z_{ij}^s \\
\text{subject to} & \sum_{i \in M} \sum_{k \in K} \sum_{j \in V} x_{ij}^k = M \in \Omega \\
& \sum_{i \in \phi} \sum_{k \in K} \sum_{j \in V} x_{ij}^k \leq 1 \quad \phi \in \Phi \\
& \sum_{j \in N} x_{n+k,j}^k \leq v^k \quad k \in K \\
& \sum_{i \in V} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \quad k \in K, j \in V^k \\
& \sum_{k \in K} \sum_{i \in V} x_{il}^k + \sum_{k \in K} \sum_{i \in V} x_{ji}^k - 1 \\
& \quad - \sum_{j' \in \pi(i, j, s)} \sum_{k \in K} \sum_{i \in V} x_{j'i}^k \leq y_{ij}^s \quad s \in S, (i, j) \in \hat{A}^s \\
& \sum_{k \in K} \sum_{i \in V} x_{il}^k + \sum_{k \in K} \sum_{i \in V} x_{ji}^k - 1 \\
& \quad - \sum_{i' \in \sigma(i, j, s)} \sum_{k \in K} \sum_{i \in V} x_{il}^k \leq z_{ij}^s \quad s \in S, (i, j) \in \hat{A}^s \\
& x_{ij}^k \in \{0,1\} \quad k \in K, (i, j) \in A^k \\
& y_{ij}^s \in \{0,1\} \quad s \in S, (i, j) \in \hat{A}^s \\
& z_{ij}^s \in \{0,1\} \quad s \in S, (i, j) \in \hat{A}^s
\end{aligned}
\]

Two changes have been performed compared to model SVSPSP\textsuperscript{0}: a) two terms have been added to the objective function (12) to model the cost of passengers
waiting during transfers between two buses that are both re-timetabled by the model, and b) inequalities (17) and (18) have been added to ensure that the $y_{ij}^x$ and $z_{ij}^x$ variables are set correctly. For example, $y_{ij}^x$ is set to 1 by (17) when both trip $i$ and trip $j$ are used (the first two sums on the left hand side) and when none of the feasible transfer destinations earlier than $j$ are in use (the last sum on the left hand side). The constraints only enforce a lower bound on $y_{ij}^x$, but the minimisation in the objective and assumption that $c_{ij}^x$ is positive ensure that the $y$ variables take the lowest possible value. Constraints (18) are similar to (17), but work on $z$ rather than $y$ variables.

The mathematical model presented in (6)–(11) has been implemented in CPLEX, but CPLEX was not able to solve instances with the dimensions considered in this paper. No attempts have been made to solve the model presented in (12)–(21) with a general purpose solver, since the number of variables and constraints used in the advanced model is even larger than in the model presented in (6)–(11). However, by presenting the models here, they have served as an instrument to give a precise definition of the problem to be studied. Using techniques like reformulation or cut or column generation it might be possible to solve realistically sized instances using the mathematical models — in particular, model (6)–(11) lends itself to a column based solution approach. However, we have worked in a different direction, and in the following section we shall present a metaheuristic for solving the problem.

4 Solution method

The solution method we propose for solving the SVSPSP is based on the large neighborhood search (LNS) metaheuristic. The LNS was proposed by Shaw [1998]. As many other metaheuristics, the LNS is based on the idea of finding improving solutions in the neighborhood of an existing solution. What sets the LNS apart from other metaheuristics is that the neighborhood searched (or sampled) in the LNS is huge.

The term LNS is often confused with the term very large scale neighborhood search (VLSN) defined in Ahuja et al. [2002]. While the LNS is a heuristic framework, VLSN is the family of heuristics that searches neighborhoods whose sizes grow exponentially as a function of the problem size, or neighborhoods that simply are too large to be searched explicitly in practice, according to Ahuja et al. [2002]. The LNS is one example of a VLSN heuristic.

We are aware of one application of LNS to the MDVSP, this approach is described in Pepín et al. [2009]. That LNS implementation is more complex than ours as it uses column generation and branch and bound to solve restricted instances of the MDVSP. The computational results reported in Pepín et al. [2009] show that the LNS is competitive against 4 other heuristics. LNS has also been successful in solving the related vehicle routing problem with time windows. See
for example Bent and Van Hentenryck [2004] and Pisinger and Ropke [2007].

4.1 Large neighborhood search

A LNS heuristic moves from the current solution to a new, hopefully better, solution by first destroying the current solution and then repairing the destroyed solution. To illustrate this, consider the traveling salesman problem (TSP). In the TSP we are given \( n \) cities and a cost matrix that specifies the cost of traveling between each pair of cities. The goal of the TSP is to construct a minimum cost cycle that visits all cities exactly once (see e.g. Applegate et al. [2006]). A destroy method for the TSP could be to remove 10% of the cities in the current tour at random (shortcutting the tour where cities are removed). The repair method could insert the removed cities again using a cheapest insertion principle (see e.g. Jünger et al. [1995]).

The LNS heuristic is outlined on Algorithm 1. In the pseudo-code we use the symbols \( x \) for the current solution, \( x^* \) for the best solution observed during the search and \( x' \) for a temporary solution. The operator \( d( \cdot ) \) is the destroy method. When applied to a solution \( x \) it returns a partially destroyed solution. The operator \( r( \cdot ) \) is the repair method. It can be applied to a partially destroyed solution and returns a normal solution. The expression \( r(d(x)) \) therefore returns a solution created by first destroying \( x \) and then rebuilding it.

The LNS heuristic takes an initial solution as input and makes it the current and best known solutions in lines 1 and 2. Lines 4 to 10 form the main body of the heuristic. In line 4 the current solution is first destroyed and then repaired, resulting in a new solution \( x' \). In line 5 the new solution is evaluated to see if it should replace the current solution, this is done using the function \( \text{accept} \) which is described in Section 4.2.3 below. In lines 8 to 10 the best known solution is updated if necessary. Line 11 checks the stopping criterion which in our implementation simply amounts to checking if \( t_{\text{max}} \) seconds have elapsed.
4.2 Large neighborhood search applied to the SVSPSP

This section describes how the LNS heuristic has been tailored to solve the SVSPSP. In particular, we describe the implemented destroy and repair methods and the acceptance criterion.

4.2.1 Destroy methods

Destroy methods for the SVSPSP remove trips from the current solution. Every time a destroy method is invoked the number of trips to remove is selected randomly in the interval [5, 30]. Two simple destroy methods for the SVSPSP have been implemented. The first method simply remove trips at random, which is a good method for diversifying the search.

The second method is based on the relatedness principle proposed by Shaw [1998]. Here we assign a relatedness measure \( R(i, j) \) to each pair of trips \((i, j)\). A high relatedness measure indicates that the two trips are highly related. The relatedness of two trips \(i\) and \(j\) are defined as

\[
R(i, j) = 30 \times 1_{s(i) = s(j)} + 30 \times 1_{e(i) = e(j)} + 20 \times 1_{s(i) = s(j)} + 20 \times 1_{e(i) = e(j)} - |t(i) - t(j)|
\]

where \(s(i)\) and \(e(i)\) are the start and end locations of trip \(i\) respectively, \(t(i)\) is the start time of trip \(i\) (start time in the current solution). The notation \(1_{expr}\) is used to represent the indicator function which evaluates to one if \(expr\) evaluates to true and zero otherwise. The measure defines two trips to be related if they start around the same time and if the share start and/or end locations. The measure is used to remove trips as follows. An initial seed trip is selected at random and added to a set of removed trips \(S\). For each trip \(i\) still in the solution we calculate the relatedness

\[
v(i, S) = \max_{j \in S} \{ R(i, j) \}
\]

The trips still in the solution are sorted according a non-increasing \(v(i, S)\) in a sequence \(T\), a random number \(p\) in the interval \([0, 1)\) is drawn and the trip at position \(\lfloor T[p] \rfloor\) in \(T\) is selected. This selection rule favours trips with high \(v(i, S)\) value. The selected trip is added to the set of removed trips, and \(v(i, S)\) is recalculated after adding a trip to \(S\). We continue to add trips to \(S\), until we have reached the target number of removed trips.

The two destroy methods are mixed in the LNS heuristic. Before removing a trip from the solution it is decided which destroy method that should be used to select the trip. With probability 0.15 the first method (random) is used and with probability 0.85 the second method (relatedness) is used.

The trips that have been removed from the solution are still active in the sense that they will be used in the trip incompatibility check defined by constraints (8) and (14). That is when adding a trip to a solution in the repair step below,
we check if it is compatible with the trips in the solution and the trips removed in the previous destroy operation. A trip $i$ is made inactive when another trip, belonging to the same metatrip as $i$, is inserted into the solution.

4.2.2 Repair methods

The repair method for the SVSPSP reinserts the trips that were removed from the solution by the destroy method. The repair method uses a randomised greedy heuristic. For each unassigned metatrip $S$ the heuristic calculates an insertion cost $f(S)$ given the current solution. When inserting a metatrip $S$ we have a choice of which trip $i \in S$ that should be inserted. With probability $\rho$ we insert the same trip that was used in the solution before destruction and with probability $1 - \rho$ we insert a random trip from $S$. The chosen trip should be compatible with all active trips. Such a trip exists because we are sure that the trip from the pre-destruction solution is compatible with all trips. The requirement ensures that we never get to a situation where one or more metatrips cannot be inserted because of the compatibility constraints (8) and (14).

Given the choice of trip $i$, we define the cost $f(S)$ as the cost of inserting trip $i$ at the best possible position in the current solution multiplied by a random factor that is meant to diversify the insertion procedure. More precisely the cost is defined as:

$$f(S) = \begin{cases} \min_{r \in R} \{c(i, r)\} \cdot (1 + \text{rand}(\delta, \delta)) & \text{if } \min_{r \in R} c(i, r) \neq \infty \\ c(i, \emptyset) & \text{otherwise} \end{cases}$$

where $c(i, r)$ is the cost of inserting trip $i$ in route $r$ at the best possible position, $R$ is the set of routes in the current solution, $c(i, \emptyset)$ is the cost of serving the trip using a new vehicle from the best possible depot, $\delta$ is a parameter and rand($-\delta, \delta$) is a function that returns a random number in the interval $[-\delta, \delta]$. The parameter $\delta$ controls the amount of randomisation applied by the insertion procedure. The heuristic chooses to insert the metatrip $S$ with lowest cost. It does this by inserting the trip $i$ that was used as a representative for $S$ and inserts this at its best possible position. This continues until all metatrips have been inserted. With to the assumption that $v^k = |\Omega|$ it is always possible to insert a metatrip — we will always be able to serve it using a new vehicle.

4.2.3 Acceptance criterion

The acceptance criterion used in our implementation of the LNS heuristic is the one used in simulated annealing metaheuristics: The function $\text{accept}(x', x)$ used in line 6 of Algorithm 1 accepts the new solution $x'$ if it is at least as good as the current solution $x$, that is, $f(x') \leq f(x)$. If $f(x') > f(x)$ then the solution is accepted with probability

$$\frac{e^{\frac{f(x) - f(x')}{\tau_k'}}}{e^{\frac{f(x) - f(x')}{\tau_k'}}}.$$
The parameter $T$ is called the temperature and controls the acceptance probability: a high temperature makes it more likely that worse solutions are accepted. Normally the temperature is reduced in every iteration using the formula $T^{\text{new}} = \alpha T^{\text{old}}$ where $0 < \alpha < 1$ is a parameter that is set relative to desired start and end temperatures and desired number of iterations. Because we use elapsed time as stopping criterion we calculate the current temperature by the formula

$$T(t) = T_s \cdot \left( \frac{T_e}{T_s} \right)^{\frac{t}{t_{\text{max}}}}$$

here $t$ is the elapsed time since the start of the heuristic, $T_s$ is the starting temperature and $T_e$ is the end temperature. Because of the acceptance criterion the LNS heuristic can be seen as a simulated annealing heuristic with a complex neighbourhood definition.

### 4.2.4 Starting solution

A starting solution is necessary because the LNS heuristic improves an existing solution. It is constructed using the greedy heuristic outlined in Algorithm 2. The generation heuristic does not consider time shifting, instead it only considers insertion of the original trip from each metatrip. Therefore, when writing earliest metatrip in Algorithm 2 we refer to the metatrip whose original trip is the earliest. The heuristic constructs vehicle routes one at a time and attempts to create routes where little time is wasted in between trips. Lines 2–12 deal with the construction of a single route for a vehicle. Lines 2–4 select the first trip on the route and the depot which should provide the vehicle for the route. Lines 5–12 add trips to the partial route. The selection of which trip to add is based on the terminal where the partial route is ending at the moment. The algorithm adds the first trip that leaves that terminal or closes the route if the route cannot be extended with a trip starting in the current terminal.

## 5 Data

The data set that has been developed for the SVSPSP during the preparation of this paper has been described in further detail in a technical report by Petersen et al. [2008], and in this section we will give a brief description of the background and the resulting data set. The data set can be obtained from http://www.transport.dtu.dk/SVSPSP/.

The local train network in the Greater Copenhagen area roughly has the form of a fan or the fingers of a hand, as shown in Figure 5. A network of express bus lines complements the train lines across and in parallel, as can be seen in Figure 6. The data set that has been developed for the SVSPSP is based on this structure, where the radial train lines are operated on a fixed timetable, and the
Algorithm 2 Heuristic for generating an initial solution

1: while there are non-served metatrips left do
2: Select a random station \( s \) with unserved metatrips;
3: Select earliest non-served metatrip \( S \) starting from \( s \);
4: Start a new route \( r \) serving \( S \). Use a vehicle from the depot nearest to \( s \);
5: repeat
6: Let \( s' \) be the station where route \( r \) is ending;
7: if \( r \) can be extended with a non-served metatrip starting in \( s' \) then
8: Select earliest non-served metatrip \( S' \) starting in \( s' \) that can extend \( r \). Add \( S' \) to \( r \);
9: else
10: End route \( r \) by returning to the depot;
11: end if
12: until \( r \) has returned to the depot;
13: end while;

timetables for the bus lines (of which most are circular) are adjusted according to this.

A data set for the SVSPSP consists of several parts: 1) a distance matrix, containing all distances between depots and line end-points, 2) fixed time tables of all fixed-schedule train connections, 3) number of transferring passengers for each transfer opportunity, 4) an initial schedule used to determine the available set of trips, 5) costs of different activities, and other parameters such as turnaround times, passenger transfer times, etc.

Among these elements the distances and fixed time tables are generally relatively easy to obtain. Furthermore the initial schedule, in the form of the current bus schedule, is required to provide information regarding frequencies and service level, which will be maintained by the new solution. Given a suitable generation strategy, the set of potential trips can be generated based on these time tables. The current schedule can also be used to generate an initial feasible (VSP) solution for the heuristics.

The problem objectives of operating cost and passenger waiting time have been combined by expressing both in monetary units. The various costs required for calculating the total cost of a solution have been estimated for the data set, in particular the cost of passenger waiting time has been estimated based on the recommended value of travel time by the Danish Ministry of Transport.

What then remains to be estimated is the number of passengers and their transfer patterns. This transfer information will allow us to calculate the number of (dis)embarking passengers using each available transfer opportunity, for any arrival or departure of a bus at a station.

For this project these data have been obtained by a two-stage process: First we estimated the number of (dis)embarking passengers, as a function of the station,
Figure 5: The local train network of Copenhagen
Figure 6: The S-bus network; trains are shown as thin lines
bus line and time of day, and then we estimated the percentage of (dis)embarking passengers that could perform each possible transfer.

The number of (dis)embarking passengers at each station is calculated as \( f_{i} \cdot f_{s} \cdot n \) where \( f_{i} \) is a time factor, \( f_{s} \) is a line factor, and \( n \) is a random number evenly distributed in the interval \([32, 48]\). The values of \( n \) is chosen to roughly reflect the capacity of a vehicle, and the introduction of randomness increases the variation of data, to make them more realistic.

The distribution of transferring passengers between available connections has been estimated based on knowledge of the network, and considering the direction of trains (towards the town centre or away from it). A random element has been added to provide a better spread of the obtained values. Connections have been specified either for a particular train line or as e.g. "the first departure going into town". For modelling purposes this could be obtained by adding artificial train lines.

Metatrips are created from trips in the original timetable. Let \( T_{i} \) be the departure time of a trip in the original timetable, belonging to a particular d-line \( L \). We create an interval \([T_{i}^{-}, T_{i}^{+}]\) around \( T_{i} \) and distribute \( \kappa \) trips in this interval to form a metatrip. Assume that \( \kappa \) is an uneven number. We express the start and end of the interval as follows \( T_{i}^{-} = T_{i} - \tau_{i}^{-} \) and \( T_{i}^{+} = T_{i} + \tau_{i}^{+} \). The symbols \( \tau_{i}^{-} \) and \( \tau_{i}^{+} \) are expressed in terms of the departure times \( T_{i-1} \) and \( T_{i+1} \) of the previous and next, respectively, trip on \( L \) as follows: \( \tau_{i}^{-} = \left[ \frac{T_{i-1} - T_{i+1}}{2} \right] \), \( \tau_{i}^{+} = \left[ \frac{T_{i+1} - T_{i}}{2} \right] \). This construction ensures that the intervals around the trips on each d-line are disjoint. The set of departure times constructed are

\[
\left\{ T_{i} - \frac{2j}{\kappa} \tau_{i}^{-} : j = 1 \ldots \left[ \frac{\kappa}{2} \right] \right\} \cup \{ T_{i} \} \cup \left\{ T_{i} + \frac{2j}{\kappa} \tau_{i}^{+} : j = 1 \ldots \left[ \frac{\kappa}{2} \right] \right\}
\]

with the time expressions rounded to the nearest integer to ensure that departures occur at integer valued times. If the trips in the original timetable are close then we may end up with fewer than \( \kappa \) departure times because some departures get mapped to the same integer due to the round-off. In that case we only create as many trips as we have departure times for. In our test we used \( \kappa = 5 \). Figure 7 shows an example of how the trips of a metatrip are distributed.

![Figure 7: Example of the distribution of trips in metatrips](image)

The only incompatibilities used in this project are found by multiplying the current interval between two trips by a factor to determine lower and upper
bounds allowed for the same interval. This factor has been set to 0.5 for the lower bound and 1.5 for the upper bound.

Instances of three different sizes have been considered for this project. These instances have been constructed by considering a meaningful subset of the actual operated bus routes, i.e. a subset that in itself constitutes a realistic problem. This means that the routes selected for the smaller subset have characteristics that may differ from the routes added in the larger subsets. Thus the smaller problem consists of the most central lines, and the lines that are added in the larger sets are more rural, and/or have fewer intersections with the train network.

The properties of the three different instances will be summarised below:

3 lines. 538 trips. All lines are circular lines with 5–6 intersections with the train network, but only few interconnections between the buses. Many passengers. Subset of

5 lines. 792 trips. All lines are circular lines with 4–6 intersections with the train network, and only few interconnections between the buses. Some lines are passenger intensive. Subset of

8 lines. 1400 trips. Combination of circular and radial lines. The radial lines only have 2–3 connections to trains, but more connections to other buses. Most lines are passenger intensive.

6 Computational experiments

To evaluate the quality and usefulness of the algorithm, we have performed a series of tests to examine its behaviour with different instance sizes and settings, which will be presented in this section. The tests have been performed on an Intel Pentium 4, 2.8 GHz, with 2GB RAM, running Windows XP.

The current vehicle schedules used for the data set were not available, so these had to be constructed initially. This has been done by using the implemented LNS as a regular VSP solver, i.e. by not allowing any time shifts. The generated solutions have been used as initial solutions when solving the SVSPSP, and also as reference solutions representing current practice, when evaluating the quality of the obtained final solutions. As we know that the actual current schedules are not created with dedicated software, this should produce reference solutions that are not worse than the currently used solution. For each instance a running time of 24 hours was allowed for the construction of the reference solution.

Table 1 shows the results from running the implemented LNS algorithm on instances of different sizes with different running times. For each run we report the cost reduction compared to the initial solution, the number of vehicles used, the reduction of empty mileage costs (i.e. a negative value indicates that the
<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>avg.</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost</td>
<td>veh.</td>
<td>empty</td>
<td>time</td>
<td>shifts</td>
<td></td>
<td>shift</td>
<td>reg.</td>
<td></td>
</tr>
<tr>
<td>1h</td>
<td>2.9%</td>
<td>0.0%</td>
<td>−14.2%</td>
<td>16.5%</td>
<td>74.2%</td>
<td>2.19</td>
<td>39.7%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6h</td>
<td>3.1%</td>
<td>0.0%</td>
<td>−13.0%</td>
<td>17.4%</td>
<td>73.4%</td>
<td>2.22</td>
<td>43.2%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24h</td>
<td>3.3%</td>
<td>0.0%</td>
<td>−8.9%</td>
<td>18.1%</td>
<td>73.8%</td>
<td>2.11</td>
<td>48.1%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>avg.</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost</td>
<td>veh.</td>
<td>empty</td>
<td>time</td>
<td>shifts</td>
<td></td>
<td>shift</td>
<td>reg.</td>
<td></td>
</tr>
<tr>
<td>1h</td>
<td>2.8%</td>
<td>0.0%</td>
<td>−10.1%</td>
<td>19.8%</td>
<td>77.0%</td>
<td>2.58</td>
<td>39.8%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6h</td>
<td>3.1%</td>
<td>0.0%</td>
<td>−9.2%</td>
<td>21.8%</td>
<td>79.3%</td>
<td>2.68</td>
<td>43.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24h</td>
<td>3.2%</td>
<td>0.0%</td>
<td>−7.8%</td>
<td>22.5%</td>
<td>78.2%</td>
<td>2.61</td>
<td>40.5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>avg.</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost</td>
<td>veh.</td>
<td>empty</td>
<td>time</td>
<td>shifts</td>
<td></td>
<td>shift</td>
<td>reg.</td>
<td></td>
</tr>
<tr>
<td>1h</td>
<td>1.1%</td>
<td>0.0%</td>
<td>−7.8%</td>
<td>9.5%</td>
<td>64.2%</td>
<td>1.88</td>
<td>30.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6h</td>
<td>1.6%</td>
<td>0.0%</td>
<td>−6.4%</td>
<td>13.3%</td>
<td>76.6%</td>
<td>2.38</td>
<td>31.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24h</td>
<td>2.0%</td>
<td>0.0%</td>
<td>−7.1%</td>
<td>16.4%</td>
<td>76.4%</td>
<td>2.39</td>
<td>36.0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Solution improvements for different problem sizes
empty cost has increased), the reduction of total passenger waiting time, the percentage of trips that have been time shifted, the average amount of time that each trip is shifted, and the percentage of trips that are regular. A regular/memorable trip is a trip for which the gap to the preceding trip on the same line is a multiple of 5. This makes the schedule easier to remember, and is thus an advantage to the passengers. For the current schedule the percentage of regular trips is around 72% for the largest instance, and 83–84% for the others. However, memorability has not been an objective of the implemented algorithm.

The table shows that good results can be obtained, and that a considerable reduction of passenger waiting time is possible. The reduced waiting times lead to an increase in the amount of empty travel, however the total operating cost still shows improvement of around 3% for the smaller instances, and 1–2% for the 8 line instances.

**Alternative small instances**

As stated previously the different tested instances differ not only in size, but also in some characteristics regarding the type of lines that are used. Thus the variation in cost and time reduction obtained for the different instances may well depend just as much on the change in these characteristics as on the actual size of the problems. The tests of Table 1 have been repeated on two additional small instances that have been created with a mix of lines more similar to those of the largest instance. These instances represent subproblems that would most likely not be considered in real-life, but can hopefully demonstrate the behaviour on smaller instances without being affected by the different characteristics of the problem. Each instance consists of two circular lines (of which one is passenger intensive) and one radial line. The results for these two instances can be found in Table 2, and indicate that it is difficult to compare the properties of instance just by looking at simple properties of the included lines. The results also indicate that the achievable cost improvement does indeed depend on the choice of lines to include in the problem.

<table>
<thead>
<tr>
<th>total cost</th>
<th>veh.</th>
<th>empty</th>
<th>time</th>
<th>shifts</th>
<th>avg. shift</th>
<th>reg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1h</td>
<td>1.3%</td>
<td>0.0%</td>
<td>-7.2%</td>
<td>12.2%</td>
<td>73.2%</td>
<td>2.0</td>
</tr>
<tr>
<td>6h</td>
<td>1.6%</td>
<td>0.0%</td>
<td>-7.7%</td>
<td>14.7%</td>
<td>76.4%</td>
<td>2.1</td>
</tr>
<tr>
<td>1h</td>
<td>2.9%</td>
<td>0.0%</td>
<td>-8.6%</td>
<td>20.4%</td>
<td>79.4%</td>
<td>2.8</td>
</tr>
<tr>
<td>6h</td>
<td>3.1%</td>
<td>0.0%</td>
<td>-5.6%</td>
<td>21.3%</td>
<td>76.5%</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Table 2: Solution improvements for more “realistic” small instances
Random variation of the instances

The network structure and the existing time tables are fixed, so in order to produce a series of different data sets/problem instances that still reflect the real world, the only adjustable parameter has been the random element of the spread of the passengers over different available connections. This has been done for the medium-sized instances (5 lines), using running times of 1 and 6 hours, and the results can be found in Table 3.

<table>
<thead>
<tr>
<th>total</th>
<th>cost</th>
<th>veh.</th>
<th>empty</th>
<th>time</th>
<th>shifts</th>
<th>avg.</th>
<th>shift</th>
<th>reg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1h</td>
<td>2.8%</td>
<td>0.0%</td>
<td>-10.5%</td>
<td>19.7%</td>
<td>78.8%</td>
<td>2.7</td>
<td>37.3%</td>
<td></td>
</tr>
<tr>
<td>2.2%</td>
<td>0.0%</td>
<td>-6.4%</td>
<td>15.4%</td>
<td>75.5%</td>
<td>2.5</td>
<td>30.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.8%</td>
<td>0.0%</td>
<td>-11.8%</td>
<td>20.1%</td>
<td>77.8%</td>
<td>2.7</td>
<td>34.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.7%</td>
<td>0.0%</td>
<td>-11.6%</td>
<td>19.7%</td>
<td>76.8%</td>
<td>2.7</td>
<td>39.6%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6h</td>
<td>3.2%</td>
<td>0.0%</td>
<td>-6.2%</td>
<td>21.8%</td>
<td>76.4%</td>
<td>2.6</td>
<td>39.9%</td>
<td></td>
</tr>
<tr>
<td>2.6%</td>
<td>0.0%</td>
<td>-4.8%</td>
<td>17.8%</td>
<td>77.1%</td>
<td>2.7</td>
<td>43.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1%</td>
<td>0.0%</td>
<td>-9.0%</td>
<td>21.8%</td>
<td>78.3%</td>
<td>2.6</td>
<td>43.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2%</td>
<td>0.0%</td>
<td>-5.4%</td>
<td>21.8%</td>
<td>76.4%</td>
<td>2.5</td>
<td>39.5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Solution results with modified transfer distributions

These results show that the actual distribution of the passengers to some extent influences the size of the reductions that can be obtained, but also that the improvements are consistently around 2.6% for the shorter running times, and around 3% for the 6 hour running times.

7 Conclusion

We have introduced a new problem that integrates the timetabling and vehicle scheduling phases in public transportation planning. It does so by simultaneously considering resource costs and passenger waiting time at transfers. The problem has been defined formally and a metaheuristic based on the LNS principle has been designed and tested. The metaheuristic has been tested on a data set based on a subset of the buses serving the Greater Copenhagen area. The results obtained are encouraging; for the full data set we have observed that a 16% reduction of passenger transfer waiting times are possible. This reduction was possible without using more buses to provide the service, but an increase in the amount of deadheading was necessary. We consider the increase in deadheading negligible compared to the total cost involved in operating a public transport system and when considering the increased passenger service obtained.

A topic for future research is how to make the timetables produced by the heuristic easier for the passengers to memorise. This could be achieved either
by adding a term penalising solutions with low memorability to the objective function or ensuring that blocks of subsequent departures have fixed headway.

Acknowledgment

This project has been supported by a grant from The Danish Social Science Research Council.

References


