Integrating timetabling and vehicle scheduling in public bus transportation

A.P.R. van den Heuvel

J.M. van den Akker

M.E. van Kooten Niekerk

Technical Report UU-CS-2008-003

February 2008

Department of Information and Computing Sciences

Utrecht University, Utrecht, The Netherlands

www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Integrating timetabling and vehicle scheduling in public bus transportation

A.P.R. van den Heuvel[*],
Piozum
Korte Linschoten WZ 23, 3461 EB Linschoten, The Netherlands
Email address: Jan-Pieter@piozum.com

J.M. van den Akker[‡]
Department for Information and Computing Sciences
Utrecht University
P.O. Box 80089, 3508 TB Utrecht, The Netherlands
Email address: marjan@cs.uu.nl

M.E. van Kooten Niekerk,
Connexxion,
P.O. Box 224, 1200 AE Hilversum, The Netherlands
Email address: m.v.kootenniekerk@connexxion.nl

[*]The research for this paper was performed while this author was at Utrecht University
[‡]Corresponding author; Supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society)

February 26, 2008

Abstract

The planning process for public bus transportation roughly consists of four steps: network route design, timetable generation, the generation of vehicle schedules and the generation of crew schedules. A lot of research has been done on optimizing the individual steps, but the integration of multiple steps has received less attention. However, this may lead to a more efficient planning. In this paper we present multiple integer linear programming models to integrate the timetable generation process and the vehicle scheduling process. The models are based on a time-space network with preprocessing steps. For the generation of arrival and departure times, these models are combined with local search.

The models were implemented and tested on real life data sets from the Dutch bus company Connexxion. The results indicate that a significant reduction of the operational costs can be achieved by optimization of the type and number of vehicles performing a service trip and by splitting service trips. Using local search to change the departure and arrival times in the current timetable can save even more.

1 Introduction

In recent years many public transportation companies have been privatized and public transport in an area is then tendered for by the public transportation companies. Because of this competition, more time and effort is spent on reducing operational costs.

A lot of research has been done on several aspects of planning in public bus transportation companies. The operational planning process can roughly be divided into four steps, route network design, timetable generation, vehicle scheduling and crew planning. Before we can generate a timetable the connections or lines have to be determined. This is also known as the network route design. In the timetabling problem the frequency of the lines and the departure and arrival times of the trips are determined. When multiple vehicle types are available, the vehicle type will often also be determined in the timetabling process. A part of the total timetable could be: "Line 128 will depart 2 times an hour; at -.12 and -.42". In the vehicle scheduling process each trip will be assigned to a vehicle. The outcome of this process is a vehicle schedule containing which vehicle will service which trips. Finally, the crew is assigned to the vehicles and trips. In this paper we will investigate the integration of two steps in the planning process: the generation of timetables and vehicle schedules. Examples of questions concerning this topic are: 'In a rural area would it be efficient to use in the morning peak 4 small buses per hour instead of two large buses, especially if you are also using small buses during the day?'. Or 'Can you save a number of buses by shifting the departure times in the timetable?' According to the Dutch situation, we assume that the timetable is clock-face. In the Netherlands a timetable that is not clock-face is only allowed for lines with a high frequency e.g within large cities.

In many bus companies including Connexxion timetables are generated manually and software is only used to support the process by means of e.g. graphical presentation. However, a considerable amount of research has been performed on automatic timetable generation. For example, Ceder (Ceder, 2001) proposes to change the timetable with a procedure to evenly spread the amount of passengers amongst the buses. This will, however, not produce a clock-face timetable and can therefore not be used by a Dutch bus company such as Connexxion. The Dutch railway company (NS) has done a lot of research on the automatic generation of periodic timetables for trains. (Peeters and Kroon, 2001) have shown an interesting way to automatically generate a cyclic timetable given the lines and frequencies. An hourly pattern is constructed using an Integer Linear Program which is adjusted for rush hours and periods with low demands on the capacity by manually adding or removing service trips. They use a transformed formulation of the Cyclic Railway Timetabling Problem (CRTP) and the – more general – Periodic Event Scheduling Problem (PESP) which was introduced by (Nachtigall,

1998).

Moreover a lot of research has been on multi-depot vehicle scheduling. An example of a survey is (Bussieck et al., 1997). A column generation technique was used by Ribeiro et al. (Ribeiro and Soumis, 1994) to solve the problem to optimality for sizes up to 300 trips and 6 depots. (Löbel, 1999) solves an ILP formulation derived from a multi-commodity flow formulation by column generation and is able to handle large real-life instances. A recent approach from (Kliewer et al., 2006) extended in (Gintner et al., 2005) shows good results. The problem is represented as a time-space network with covering constraints. The use of time-space networks was first suggested by (Hane et al., 1995) for solving routing problems in airline scheduling. Kliewer et al. have adapted them for application in bus scheduling by greatly reducing the number of deadhead edges. The reason this works well is because the values of the variables in the solution of the LP relaxation are mostly integer since the formulation resembles the LP formulation for a minimum cost flow problem.

The integration of vehicle scheduling and crew scheduling has also been investigated. (Huisman et al., 2005) propose two algorithms for integrated vehicle and crew scheduling in the multiple depot case. Both algorithms are based on a combination of column generation and Lagrangian relaxation. In (Huisman and Wagelmans, 2006)a solution approach for the dynamic version vehicle and crew scheduling is proposed. Vehicle and crew scheduling is also investigated by (Rodrigues et al., 2006), whose techniques are based on integer linear programming coupled with heuristics.

In (Kliewer and Bunte, 2007) vehicle scheduling is combined with the possibility to make small changes to the timetable, i.e. by shifting the departure times of trips that have been identified as critical within a limited time window. This is achieved by extending the time-space network. Their computational results are encouraging.

The contribution of this paper is the development of models for vehicle scheduling which include the optimization of the type and/or number of vehicles for each trip and a local search algorithm for combining the computation of timetables times with vehicle scheduling. We developed integer linear programming models for vehicle scheduling where multiple vehicle types are allowed for each service trip, i.e., trip defined in the timetable. These models use a so-called Time-Space Network and extend the models presented in (Kliewer et al., 2006). One of our models has full flexibility in the assignment of bus types to a service trip as long as there is enough capacity to transport the expected number of passengers, e.g. if the expected number of passengers is 60 one standard bus with capacity 50 and one small bus with capacity 10 can be used. Another model chooses one vehicle type per service trip and then assigns the required number of buses to it. If there are more buses of the selected type required for the trip, the trip is split into multiple trips and these trip are divided evenly over time. For

example, if initially there are two trips per hour and one trip is executed by two buses these buses will run with an interval of 15 minutes. To include the optimization of the timetable we introduce an hierarchical approach. Recall that we assume that a timetable has to be clock-face. We apply a local search algorithm where in each iteration the timetable is changed and for a fixed timetable and fixed vehicle types the cost are computed by solving a minimum cost flow problem. To the best of our knowledge this the first algorithm for integration of timetabling and vehicle scheduling allowing all possible departure and arrival times in the timetable under the condition that the timetable remains clock face. We implemented the algorithms and performed experiments with real-life data from bus company Connexxion.

The remainder of the paper is organized as follows. In Section 2, we present the integer linear programming models for vehicle scheduling and the local search algorithms. After that, in Section 3 we present our computational results. Finally, in Section 4 we conclude by discussing the applicability of the results and further research.

2 Solution models

The models that we have developed are based on a 'Time-Space Network', which was also applied in e.g. ((Hane et al., 1995), (Kliewer et al., 2006), and (Kliewer and Bunte, 2007)). In the next subsection, we will explain the network in more detail. After that we will explain models with a fixed vehicle type per trip in Section 2.2 and models with different possible vehicle types per trip in Section 2.3. Finally in Section 2.4 we will explain the local search algorithm.

2.1 The time-space network

In the time-space network, every departure and arrival of a bus at a specific time and a specific station or the depot is a vertex. In most cases this corresponds to the beginning or end of a service trip. The edges represent actions that can be performed by a vehicle. The network has four types of edges:

- *Service trip edges* representing driving from a starting point of a trip as defined in the timetable to the end point of that trip with passengers
- *Waiting edges* representing waiting at a station or in a depot
- *Pull-in/-out edges* representing driving from a depot to a station and from a station to a depot

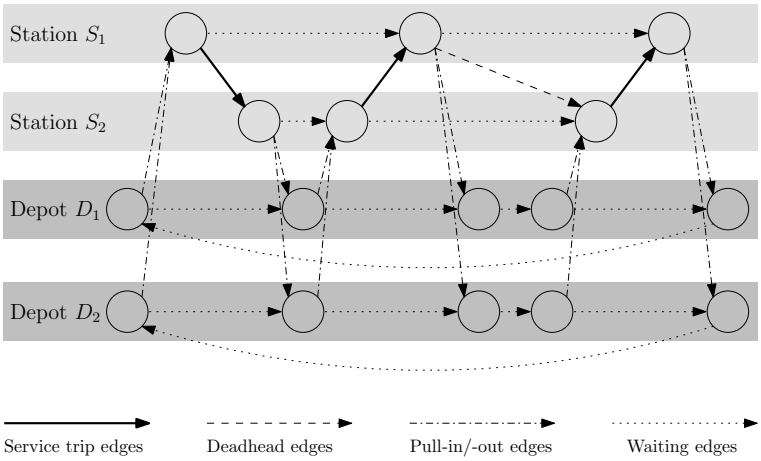

Figure 1: Example of a time-space network for the MDVSP

- *Deadhead edges* representing driving without passengers after operating a service trip from one station to another station to operate another service trip.

Before and after each service trip pull-in and -out edges are added from and to the depot respectively. The network can be represented in such a way that the vertices are horizontally ordered by time and vertically ordered by space. An example is depicted in Figure 1.

A vehicle schedule corresponds to a flow through the network, where every unit of flow corresponds to a vehicle. To each edge the corresponding costs are assigned which are calculated per minute and include fuel and driver costs. An additional circulation edge is added to each depot from the last vertex (in time) to the first vertex. This enables us to minimize the number of vehicles by assigning fixed vehicle cost to the circulation edges. The computation of the optimal vehicle schedule now boils down to calculating a minimum cost circulation through this network.

In principle, deadhead edges are added whenever the travel time between two stations is smaller than the time between the end of the first service trip and the beginning of the second service trip and hence there is no maximum length of a deadhead trip. To reduce the number of deadhead edges we will use the following preprocessing steps suggested in (Kliewer et al., 2006). We only add so called *First matches*. If from a service trip arrival at station A a deadhead trip can be performed to two subsequent service trip departures at station B, only the deadhead edge to the first departure is added together with a waiting edge to the second departure to ensure that the deadhead trip to this departure is also allowed in the network. From the first matches, even more edges can be removed keeping only the *Latest first matches*: when there are multiple deadhead edges from station A to the same departure at station B, only the last one is preserved, since it is still reachable through waiting edges on station A.

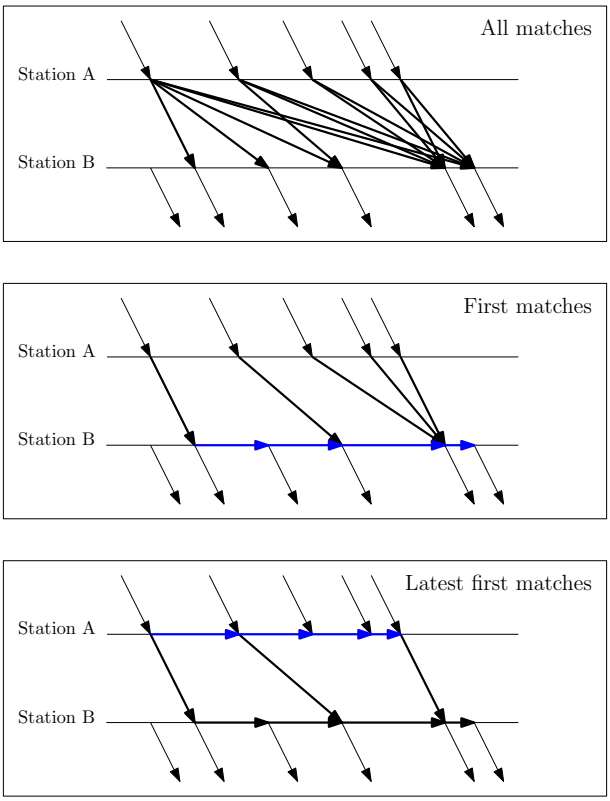

Figure 2: Reducing deadhead edges

An example can be seen in Figure 2. These steps can reduce the number of deadhead edges to 1% of the number of edges in the original graph representation (All matches).

Finally, we apply an additional preprocessing step to reduce the size of the network. If two subsequent vertices at a depot only have pull-in edges, i.e., outgoing edges, these can be merged into the first vertex by including the waiting at the depot (modeled by the waiting edge between the two vertices) into the outgoing deadhead edges. If the second vertex would have pull-out, i.e. ingoing edges, this is impossible because the ingoing edges cannot be moved to an earlier point in time. In the same way, if two of these vertices only have pull-out (ingoing) edges these can be merged into the last vertex.

2.2 Fixed vehicle type per service trip

To introduce the models, we consider the case with one type of vehicle. There can be multiple depots and we assume that vehicles do not need to end in the depot from which they have started, but for each depot the number of vehicles at the beginning of the day must be the same as the number of vehicles at the end of the day. This corresponds to the current situation at Connexxion.

A vehicle schedule corresponds to a minimum cost circulation through the time-space network, which is essentially a minimum cost flow without sources and sinks, i.e., for each vertex the inbound flow must equal the outbound flow. A minimum cost circulation can be found with special purpose algorithms such as the minimum-mean cycle cancelling algorithm from (Tardos, 1985), a successive shortest path algorithm by (Orlin, 1988) that uses the Edmonds-Karp scaling technique. However, since we want to extend the model to other cases e.g. with multiple vehicle types, we use a linear programming formulation.

Let V be the set of vertices in the time-space network, E the set of edges, and $S \subset E$ the set of service trip arcs. For each vertex $v \in V$ the sets In_v and Out_v denote the set of incoming and outgoing edges, respectively. By c_e, we denote the cost of the trip corresponding the edge e. Recall that these cost include fuel and driver cost and that the fixed cost of a bus are included on the circulation edges. We define decision variables x_e representing the amount of flow through edge e. The linear programming problem formulation is given below.

$$\text{Minimize} \quad \sum_{e \in E} c_e \cdot x_e \tag{1}$$

$$\text{subject to} \quad \sum_{e \in In_v} x_e - \sum_{e \in Out_v} x_e = 0 \qquad \forall v \in V \tag{2}$$

$$x_e = 1 \qquad \forall e \in S \tag{3}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{4}$$

The objective function (1) describes the total cost of the generated vehicle schedule, the constraint (2) ensures the flow properties in the network and constraint (3) guarantees that all service trips are performed exactly once. We want x_e to be integral, but since the constraint matrix for this problem is known to be totally unimodular, the variables will always have integral values.

When a minimum cost circulation has been found, the flow needs to be decomposed into individual bus schedules. Several methods exist, such as First In First Out (FIFO) and Last In First Out (LIFO). It is also possible to design a customized decomposition method which for example minimizes the number of times a bus will change lines or routes.

A large bus company like Connexxion owns multiple vehicle types. Examples are buses with a low floor which are designed for easy access for disabled people and articulated buses which have a higher capacity. Even bi-articulated buses are deployed in the city of Utrecht by the GVU, a full daughter of Connexxion. On the other hand, in rural areas sometimes small taxi buses are used. As also explained in (Kliewer et al., 2006), the situation with multiple types of vehicles but a predetermined vehicle type for each trip, this can be included

in the above model by creating a separate network (layer) for each type of vehicle including a separate depot. Since the service trips and depots of the different vehicle types do not overlap and there are no edges between these network layers, the minimum cost circulation can be solved separately for each vehicle type. A similar model is described in (Kliewer et al., 2006), but there also a separate layer was necessary for each depot to ensure that a bus returns to its starting depot.

2.3 Multiple vehicle types on a service trip

It may be expected that allowing multiple vehicle types on a trip improves the vehicle scheduling. For example, in a rural area one might use large buses during the rush hour and small buses during the rest of the day. In such a case it might be interesting to use two small buses on a rush hour trip, instead of one large bus.

Again, for each vehicle type a connected component (or layer) is added to the graph. Note that a service trip can now be in multiple layers, and hence these layers can not be solved independently. Consequently, the problem becomes NP-hard and is no longer solvable with special purpose minimum cost circulation algorithms. However, the linear programming formulation of a minimum cost circulation can be applied with some additional constraints. In this section, we present three ways of including multiple possible vehicle types per trip in the models.

A first approach which is also presented in (Kliewer et al., 2006), is that each trip is performed by a single vehicle, but vehicles of different types are allowed. Let T be the set of service trips and for each $t \in T$, let S_t be the set of service trips arcs associated with trip t, i.e., corresponding to the vehicle types which are allowed for t. The (Integer) Linear Program now becomes:

$$\text{Minimize} \quad \sum_{e \in E} c_e \cdot x_e \tag{5}$$

$$\text{subject to} \quad \sum_{e \in In_v} x_e - \sum_{e \in Out_v} x_e = 0 \qquad \forall v \in V \tag{6}$$

$$\sum_{e \in S_t} x_e = 1 \qquad \forall t \in T \tag{7}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{8}$$

$$x_e \text{ integral} \qquad \forall e \in E \tag{9}$$

Constraint (3) is replaced by the so-called multiple choice constraint (7) modeling the choice for one bus type. Moreover, since the constraint matrix is not totally unimodular, the integrality constraint cannot be omitted anymore.

The previous model is limited in the sense that it does not allow more than one vehicle per trip, and hence excludes for example using multiple small vehicles instead of one large vehicle to service a trip. To include this in the model we are going to replace the multiple choice constraint (7) by a constraint which enforces that on each trip there is enough capacity for the expected number of passengers. Each service trip edge $e \in S$ corresponds to a bus type in which we denote by $b(e)$. We define $p_{b(e)}$ as the passenger capacity of the type of bus corresponding to edge e. Furthermore let P_t be the expected number of passenger on service trip t. We now replace constraint (7) by:

$$\sum_{e \in S_t} p_{b(e)} \cdot x_e \quad \geq \quad P_t \quad \forall t \in T \tag{10}$$

The previous model is very intuitive and allows full flexibility in the selection buses for a trip, but – as will be shown in the computational results – it turns out to be computationally quite hard to solve. For this reason, we also developed a model to only allow one vehicle type per service trip. So a service trip can for example be serviced by two normal sized buses or three small sized buses but not by one normal sized bus and one small sized bus. This might be a reasonable assumption in many practical situations. To this end a new binary variable $y_{b,t}$ is introduced to decide if service trip t will be serviced by vehicles of type b. We define $n_{b,t}$ as the number of buses of type b that is needed to transport the expected number of passengers in trip t. When multiple buses are needed to accommodate the passengers, the service trip is split into single bus trips that are spread evenly in time. If for example there is a trip every hour and we will need two vehicles of a certain vehicle type, a vehicle would leave every half hour. This is achieved by including additional service trips in the time-space network accordingly. In the model, this means that constraint (10) has to be replaced by the constraints:

$$\Big(\sum_{e \in S_{b,t}} \frac{1}{n_{b,t}} \cdot x_e \Big) - y_{b,t} \quad = \quad 0 \qquad \forall b \in B, t \in T \tag{11}$$

$$\sum_{b \in B} y_{b,t} \quad = \quad 1 \qquad \forall t \in T \tag{12}$$

$$y_{b,t} \quad \in \quad \{0,1\} \qquad \forall b \in B, t \in T \tag{13}$$

Here $S_{b,t}$ denotes the set of service trip edges of bus type b associated with service trip t.

2.4 Local search

To actually change the timetable times we use local search in combination with a network flow model. In each iteration of the local search we apply a small change to the timetable

and optimize the vehicle schedule by the network flow model. Local search algorithms usually require a lot of iterations to obtain a good solution and hence the speed of an iteration is very important. For this reason we use the model (1)-(4) with different vehicle types. This means that we assume the type of vehicle of each trip is fixed. The minimum cost circulation that is required to solve this model can be calculated in polynomial time by a special purpose algorithm or by using the LP formulation and an LP solver.

The local search strategy we have used is simulated annealing. In each iteration a small change is made to the timetable and we will determine if we will accept or reject the new timetable. Improvements will always be accepted and degradations will be accepted with a chance that decreases while the algorithm runs. In each iterations, the timetable is changed by randomly selecting a line and then shift this line 1, 2, 3, 4, 5 or 10 minutes forward or backward in time. In this way, the current clock face timetable is preserved. Observe that with this neighborhood structure the vehicle type of each service trip remains fixed. The next step is to allow changes of vehicle types. Therefore, we have to determine for each service trip the set of possible vehicle configurations, i.e. the number of buses and their types. Then after a fixed number of iterations, we can randomly select a service trip and change its vehicle configuration. This is not yet included in our implementation.

Several experiments have been performed to determine the best number of iterations and and the speed of decreasing the probability of accepting a worse timetable. The progress of the algorithm is represented by a temperature variable. The temperature starts at 5000 and is multiplied by 0.99 in each iteration. The algorithm stops when the temperature drops below 1. The chance of accepting a worse solution is $\exp\left(\frac{(l-c)\cdot 0.1}{t}\right)$ where l is the cost of the last accepted solution, c is the cost of the current solution and t is the current temperature.

3 Computational results

To evaluate the performance of the models and algorithms, we have implemented them and tested them with real-life data provided by Connexxion. All integer linear programming models are implemented in the Java programming language. After the preprocessing steps described in Section 2.2, the minimum cost circulations are computed by means of the presented ILP formulations which are solved by ILOG CPLEX (ILOG,) version 9.1. The local search algorithm is implemented in the C++ programming language. For this algorithm, the graphs are built using the Library of Efficient Data types and Algorithms (Mehlhorn and Näher, 1995) which solves the minimum cost circulations very fast. The ILP problems were solved on one of the available processors on a machine with 8 Intel Xeon 1.86 GHz processors. The local search was run on a PC with an with an Intel Core2 Duo 3,0 Ghz processor and

1024MB RAM.

The data sets provided by Connexxion are from the area "Noord Holland Noord". The area contains both local and regional bus lines. The spring 2007 timetable is provided for weekdays (WE), for Saturday (SA) and for Sunday (SU). Detailed statistics about these timetables can be found in Table 1.

day	lines	service trips	passengers
Weekday (WE)	49	1862	30095
Saturday (SA)	29	1198	16138
Sunday (SU)	23	822	8865

Table 1: Current timetable statistics

The passenger counts for all service trips are available. Traveling times between all pairs of stations (including deadhead trips) were partly obtained by measurements with special devices in the bus and partly derived from these measurement results. The area contains three depots: in Alkmaar (AMR), Den Helder (DHE) and Hoorn (HRN). In this area Connexxion uses several types of buses. The bus type L12 is the most commonly used vehicle type. This is a twelve meter long bus with a low floor. An 18 meter articulated version of this type is also used (L18). There is a special Service bus (SER) that is disabled-friendly. The H10 bus type is a short 10 meter version of a standard bus, which is only used on a specific line. Finally, small taxi vans (TX08) are used.

Name	Vehicles per service trip	Model
SV-FT	Single vehicle, fixed type	(1)-(4) with multiple vehicle types
SV-MT	Single vehicle, multiple types	(5)-(9)
MV-MT	Multiple vehicles, multiple types	(5), (6), (8), (9), (10)
MV-ST	Multiple vehicles of the same type	(5), (6), (8), (9), (11)-(13)
MV-ST-H10	Multiple vehicles of the same type	MV-ST with H10 allowed on all lines

Table 2: Scenarios

We first report results on the ILP models from the previous section. The different scenarios are given in Table 2. We indicate the options for assigning vehicle to a service trip and the corresponding ILP formulation. We will use SV-FT as a reference scenario. In SV-MT for each service trip the type of bus currently used by Connexxion or a larger type is allowed. In MV-ST, if the are multiple vehicles on a service trip, the resulting trips are evenly spread in time between the other service trips.

In Table 3 we present the results for the different ILP models. For each timetable, we report on the cost reduction relative to the reference scenario SV-FT for that day. Note that the absolute cost on a weekday are much larger than the cost on a Saturday, which on its turn are much larger that the cost on a Sunday. We also present the number of used buses of different types, and the computation time (in seconds). The default settings of CPLEX appeared to have some difficulties, especially with the MV-MT scenarios. To improve the performance of CPLEX, we used more 'aggressive' CPLEX settings, such as more extensive cut generation and strong branching (which applies some look-ahead when choosing the next node). In case the optimal solution was not found after 10 hours, the computation was stopped. This happened for the Saturday instance, which appeared to be the most difficult one. For these cases we give the integrality gap, i.e., the difference between the value of the LP-relaxation and the value of the best integral solution. For the MV-ST cases we also give the number split trips.

The results indicate that allowing a larger bus type on a trip (SV-MT) gives a slight cost reduction and significantly reduces the number of buses used. The models for MV-MT, where multiple vehicles are allowed on one trip and with only the constraint that the passenger demand is met, are relatively hard from a computational point of view. Our results indicate that allowing multiple vehicles on a service trip on a service trip leads to a significant cost reduction. For the Saturday, still after 10 hours the integrality gap is reasonably small.

In MV-ST a trip is split into trips that are evenly spread in time between the current service trips, in case of multiple vehicles on a service trip. Hence, this model is not strictly comparable to the previous models. The computational results show that a lot of money can be saved. However, it seems that the cost reduction of MV-ST is very close to the cost reduction of MV-MT (for Saturday, if the gap is added). This would imply that in case of multiple vehicles on a service trip, the condition that these are all of the same type is not a strong restriction.

In our original experiment buses of type H10 could only service trips from one particular line. After dropping this restriction – allowing H10 vehicles to service trips in the whole area – the results show something interesting. A lot of buses of type H10 were used. These buses have a slightly smaller capacity than the standard L12 buses and the costs of the H10 buses is slightly less. These results indicate that it would be cheaper (2 to 3 % additional savings) to have more buses with a smaller capacity.

On Saturday and Sunday, the number of vehicles increases when we 'split' service trips over multiple vehicles, while the cost decrease. This is caused by the use of taxi vans. An interesting observation is that the number of buses needed to operate the timetable on a Weekday is smaller when trips are split than when trips are not split. One would expect

that servicing trips with more than one bus would result in a vehicle schedule with more buses. This may be explained by the following. Some trips were already serviced by multiple vehicles and those trips are serviced by one larger vehicle in the new vehicle schedule and thus reducing the number of buses needed. Another effect that might occur (although this is not the case with Connexxion) is that smaller buses are used during the periods that have a low demand on the capacity (midday and evening) and that those buses were in the depot during the rush hour periods. In the new vehicle schedule, these smaller buses could also be used during the rush hour periods and thus reducing the total amount of buses needed.

In Table 4 we present the results of the simulated annealing algorithm. The vehicle type for each service trip is the one that is currently being used by Connexxion to operate the service trip and is not changed. The results are compared to the basic scenario SV-FT; they show the improvement that can be obtained by moving the timetable times. For each timetable we give the cost reduction with respect to the basic scenario and the computation time in seconds. The results indicate that a significant cost reduction can be obtained by changing the timetable times.

4 Discussion and further research

In this paper we have proposed several models that provide an integration of timetabling and vehicle scheduling, including more flexibility of assigning vehicle types to service trips. Models from Kliewer et al. based on a so-called time space network have been extended. We developed a model to perform service trips with multiple vehicles possibly of different types and a model where a service trip can be performed by multiple vehicles that have to be of the same type. In the latter case the service trip is split into a set of trips evenly divided over time. A local search algorithm is proposed to integrate the vehicle scheduling and the optimization of the departure and arrival times given in the timetable under the condition that the timetable remains clock-face.

Computational experiments on real life data sets provided by Connexxion indicate that a significant (up to about 8 %) reduction of the operational costs can be achieved when there is more flexibility in the type and number of vehicles performing the service trips and some trips are split and evenly spread in time. The experiments with our local search algorithm suggest that changing the timetable times may lead to 1 or 2 % additional cost reduction. Our current implementation considers a fixed vehicle type for a service trip and does not allow splitting of trips results in up to about 2 % reduction of the operational costs. Including multiple and different types of vehicles per service trip and splitting of service trips into the local search algorithms results, enables us to analyze this effect more precisely. Finally, possible

connections between different bus lines, which limit the freedom of changing the timetable, are not included in our algorithm. These issues are a topic for further research.

Given the large amounts of money that are involved in the operation of a large bus company like Connexxion, our models suggest that integration of vehicle scheduling and timetabling could save millions of Euros. Some remarks have to be made regarding the applicability of the model. The models are based on passenger counts of two weeks, so there is some inaccuracy in the numbers. In the future with the introduction of the 'OV-Chipcard (electronic payment by all passengers) more elaborate passengers counts will be available. Another possibility is to include some kind of buffer in the passenger number, i.e., to reserve some spare capacity on trips. In our models we did not include any turn-around time, i.e. when a bus arrives at a station it can immediately depart for a new trip. In principle these times could be included by adding them to the driving time of the trip. Since the turn-around time equals zero in all models, including the basic scenario, the models are comparable. An estimation of the cost of the crew is included in the cost numbers. However, since crew scheduling is not included the effect on the cost of the crew can be analyzed in more detail by extending the model. This requires a significant amount of further research. Note that previous results Huisman et al (Huisman et al., 2005) have shown that integration of crew and vehicle scheduling can be very beneficial. Finally, it might be interesting to investigate if the integration of vehicle scheduling and time-tabling can be fully performed by Integer Linear Programming.

Acknowledgment

The authors would like to thank Guido Diepen for his 'tips and tricks' on how to complete the computational experiments.

References

Bussieck, M., Winter, T., and Zimmerman, U. (1997). Discrte optimization in public rail transport. In *Mathematical Programming: A Publication of the Mathematical Programming Society*, pages 415–444, Amsterdam. Elsevier Science B.V.

Ceder, A. (2001). Bus timetables with even passenger loads as opposed to even headways. *Transportation research record*, 1760:3–9.

Gintner, V., Kliewer, N., and Suhl, L. (2005). Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27:507–523.

Hane, C., Barnhart, C., Johnson, E., Marsten, R., Nemhauser, G., and Sigismondi, G. (1995).

The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1):211–232.

Huisman, D., Freling, R., and Wagelmans, A. (2005). Multiple-Depot Integrated Vehicle and Crew Scheduling. *Transportation Science*, 39(4):491.

Huisman, D. and Wagelmans, A. (2006). A Solution Approach for Dynamic Vehicle and Crew Scheduling. *European Journal of Operational Research*, 172(2):453–471.

ILOG. Ilog cplex website. http://www.ilog.com/products/cplex/.

Kliewer, N. and Bunte, S. (2007). Resource optimization in public transport with variable departure times.

Kliewer, N., Mellouli, T., and Suhl, L. (2006). A time-space network based exact optimization model for multi-depot bus scheduling. *European journal of operational research*, 175(3):1616–1627.

Löbel, A. (1999). Solving large-scale multi-depot vehicle scheduling problems. *Lecture Notes in Economics and Mathematical Systems*.

Mehlhorn, K. and Näher, S. (1995). *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press.

Nachtigall, K. (1998). Periodic network optimization and fixed interval timetables. *Deutsches Zentrum für Luft-und Raumfahrt, Institut für Flugführung, Braunschweig*.

Orlin, J. (1988). A faster strongly polynomial minimum cost flow algorithm. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 377–387, New York, NY, USA. ACM Press.

Peeters, L. and Kroon, L. (2001). A cycle based optimization model for the cyclic railway timetabling problem. *Lecture Notes in Economics and Mathematical Systems*, pages 275–296.

Ribeiro, C. and Soumis, F. (1994). A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. *Operations Research*, 42(1):41–52.

Rodrigues, M. M., Souza, C. C. d., and Moura, A. V. (2006). Vehicle and Crew Scheduling for Urban Bus Lines. *European Journal of Operational Research*, 170(3):844–862.

Tardos, É. (1985). A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255.

day	scenario	cost red.	#buses (L12;SER;L18;H10;TX08)	time	gap	#trips spl.
WE	SV-FT		121 (98;7;11;1;4)	0.383		
	SV-MT	1.18 %	112 (87;9;11;1;4)	27.601		
	MV-MT	5.17 %	107 (73;9;9;1;15)	17918.688		
	MV-ST	5.24 %	107 (72;10;8;1;16)	294.982		24
	MV-ST-H10	7.97 %	106 (29;7;7;51;12)	3242.803		15
SA	SV-FT		69 (53;7;5;1;3)	0.217		
	SV-MT	0.45 %	67 (49;10;5;1;2)	15.107		
	MV-MT	3.61 %	71 (47;10;0;1;13)	$>$ 10 hrs	0.11 %	
	MV-ST	3.83 %	71 (46;10;0;1;14)	$>$ 10 hrs	0.04 %	37
	MV-ST-H10	7.07 %	71 (8;8;0;43;12)	$>$ 10 hrs	0.05 %	25
SU	SV-FT		44 (39;0;1;0;4)	0.157		
	SV-MT	0.06 %	43 (39;0;1;0;3)	6.111		
	MV-MT	4.75 %	48 (32;0;0;0;16)	9934.973		
	MV-ST	4.86 %	47 (31;0;0;0;16)	388.849		34
	MV-ST-H10	8.66 %	47 (5;0;0;27;15)	460.484		26

Table 3: Computational results ILP with 'aggressive' CPLEX settings

day	cost red.	time
WE	2.23 %	1266.69
SA	0.70 %	565.89
SU	1.83 %	316.60

Table 4: Computational results local search