Dedicated Heuristic for a Back-and-Forth Single-Line Bus Trip Timetabling Problem

Sylvain FOURNIER

[1]WPLEX Software
Rod SC 401, 8600 Corporate Park bloco 5 sala 101
88050-000 Santo Antônio de Lisboa, Florianópolis SC

sylvain@wplex.com.br

ABSTRACT

The problem tackled in this paper is a Trip Timetabling Problem where only one line is considered and this line has both a forward and backward way. The forward and backward trips are generated together in order to create a better timetable for vehicle scheduling, which is the next step in the bus transit scheduling process. For this, we first create forward trips time periods from the initial time periods and demands in backward and forward trips. For each of these created time periods, the demand in number of trips is computed using a integer-programming model and the initial demands. Then, trips are generated from these demands using a simple greedy algorithm, such that in each time period, the duration between each generated trip departure time is constant. This procedure, followed by vehicle scheduling, shows very interesting results.

KEYWORDS. Trip Timetabling, Heuristic, Linear Programming, Logistics and Transportation

RESUMO

Neste artigo apresentamos um Problema de Programação Horária de Viagens de uma linha de transporte urbano, com viagens de ida e volta. Estas viagens são geradas conjuntamente para criar um quadro horário, visando otimizar o Escalonamento de Veículos. Para isso, são dadas faixas horárias iniciais e as demandas de viagens na ida e volta. Esses dados são utilizados para criar novas faixas horárias para determinação das viagens de ida. Para cada uma dessas faixas criadas, a demanda em número de viagens é calculada usando um modelo de programação inteira, que leva em consideração as demandas iniciais. Em seguida, viagens são geradas a partir dessas demandas usando um algoritmo guloso simples tal que, em cada faixa horária, a duração entre o tempo de partida de cada viagem gerada é constante. Este processo, seguido pelo Escalonamento de Veículos, apresentou bons resultados.

PALAVRAS-CHAVE. Programação Horária de Viagens, Heurística, Programação Linear, Logística e Transporte

1. Introduction

Bus transit scheduling includes various steps, including route design, trip timetabling, vehicle scheduling, crew scheduling and crew rostering. Due to the complexity of the overall scheduling, each of these steps are usually tackled separately, in spite of the strong dependences between each other. However some research effort is made to treat the problems at the same time. In this paper, we will focus on the trip timetabling aspect, with a view to the vehicle scheduling. Indeed, when generating the bus timetables for a given day, the allocation of these created trips to vehicles must be taken into account in order to minimize the number of vehicles needed and hence the costs for the bus company.

WPLEX Software is a company that provides tools for planning and controlling bus transit operations as well as passengers information. The software WplexON, which also proposes solutions for vehicle scheduling and crew scheduling (see Fournier (2009)), has a fast engine for solving the trip timetabling problem which is presented in this paper.

In section 2, we describe the Trip Timetabling Problem that we face. An overview for the general solution method is given in section 3. The first step of creating time periods from the initial forward and backward trips time periods, and determining the demand for each time period is detailed in section 4. Then, section 5 describes the greedy way of generating the timetable from this demand. Some results are presented and analysed in section 6, comparing the algorithm with a method where the backward trip generation is separated from the forward trip generation. Section 7 is dedicated to some conclusions and perspectives.

2. A Back-and-Forth Single-Line Trip Timetabling Problem

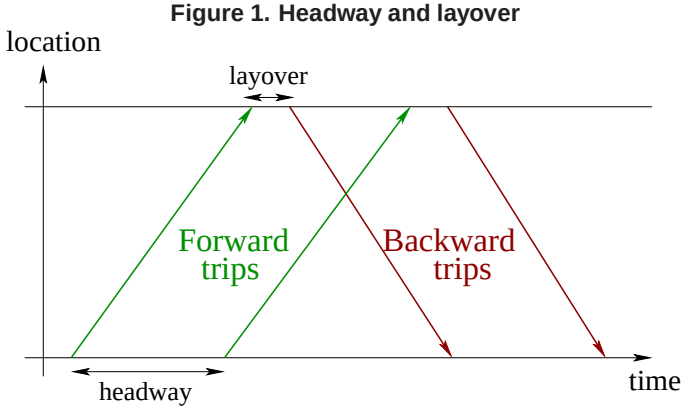
In the Trip Timetabling Problem (also known as Transit Network Timetabling Problem), the aim is to provide a schedule of trips in order to meet the passenger demand, and minimizing the costs and the passenger waiting time. There are plenty of variations of this problem, either with respect to the objective or to the input. In our case, the trip schedule has to satisfy the passenger demand for a number of time periods during the day. The passenger demand can be used to compute the number of needed trips for each time period and with some additional data such as the vehicle capacity and the wanted occupancy rate.

The multi-line case has been widely studied and in this case, the objective is often to maximize the synchronization between the trips of each couple of lines, in order to minimize the waiting time for any transferring passenger.

Fleurent et al. (2004) use a dedicated heuristic to maximize the number of meeting times between the buses, using time windows and a piecewise-linear cost function. Voss (1992) proposes a quadratic semi-assignment model for the synchronization of timetables and Schröder and Solchenbach (2006) use and customize this model to solve a problem with mixed train and bus lines. These last three papers start from an existing timetable and allow only small changes in the schedule. Ceder et al. (2001) show that a mixed-integer formulation for this problem is large and hard to solve, and propose a polynomial heuristic to maximize the number of simultaneous arrivals.

Some researchers also solve both the timetabling and the vehicle scheduling prob-

Figure 1. Headway and layover



lems in an integrated approach. For example, van den Heuvel et al. (2006) tackle this problem using integer programming models combined with local search in a time-space network.

We consider the case of a back-and-forth single-line, where each direction has its own time periods and demands in trips. For example, the forward direction of a line from the suburbs to the center will have a higher demand in the morning and its backward direction in the evening. The case of a line with only one way (for example a circular line) will not be treated here as the solution can be obviously deduced from the algorithm presented in this paper. The reason why we focus on this special case is that various transit companies prefer to split their overall schedule into parts based on each line of the bus network, for both an easier handling of the data and a better view of the schedule, and the possible necessity to prevent buses or drivers from changing lines.

Figure 1 is an illustration for the important definitions of **layover** (time between a forward trip and its associated backward trip) and **headway** (time between two consecutive forward trips).

3. Simultaneous back and forth generation of trips

A simple way to generate trips for a back-and-forth single line is to provide the timetable for each direction separately. Each of these trip generations is very simple, but this method has two main drawbacks, considering the vehicle scheduling which is the following step of the overall process:

- there can be a lot of forward trips without associated backward trip (or vice-versa), which would result either in the creation of a lot of deadheads (without any passengers in the bus), or in the use of a lot of addicional blocks (or buses) performing only few trips,
- the generated forward trips are not synchronized with the backward trips, which may create more blocks with a significant total idle time in the vehicle scheduling step.

These drawbacks can be resolved by a post-processing, creating deadheads and time shifting trips for a better synchronization. This procedure may be heavy and time-consuming.

In our approach, we chose to generate simultaneously the forward and backward trips, such that:

- there are as many forward trips as backward trips,
- the forward trip is always defined before its associated backward trip, with a minimum layover between both trips.

This way, we are sure that in the vehicle scheduling step, each vehicle will be able to perform a backward trip just after a forward trip and will be made available at the starting point of the line, which is generally the main bus terminal, in order to perform another forward trip.

4. Computing the demand in forward trips for each time period

Let a set of time periods $T_0, T_1, \ldots, T_n$ be defined by the limit times $(t_1, t_2, \ldots, t_n)$ (with $t_1 < t_2 < \ldots < t_n$), meaning that the $n+1$ time periods are the following:

- $T_0 = [t_0, t_1[$ where t_0 is the day beginning time,
- $T_1 = [t_1, t_2[$,
- $\ldots$
- $T_{n-1} = [t_{n-1}, t_n[$,
- $T_n = [t_n, t_N[$ where t_N is the day ending time.

Each direction (forward and backward) has one such set of time periods (not necessarily the same) and is characterized by demands in integral number of trips for each of these time windows, for instance $(d(T_0), d(T_1), \ldots, d(T_n))$ ($d(T_i)$ being the demand in time period T_i, $0 \leq i \leq n$).

The very first step of our procedure is to create time periods for the generation of forward trips as detailed in Figure 2. Every backward trip will be automatically generated from the associated forward trip. Hence, in order to be able to generate the forward trips in order to meet the backward trips demand, we need to change the backward trips time periods into time periods on forward trips.

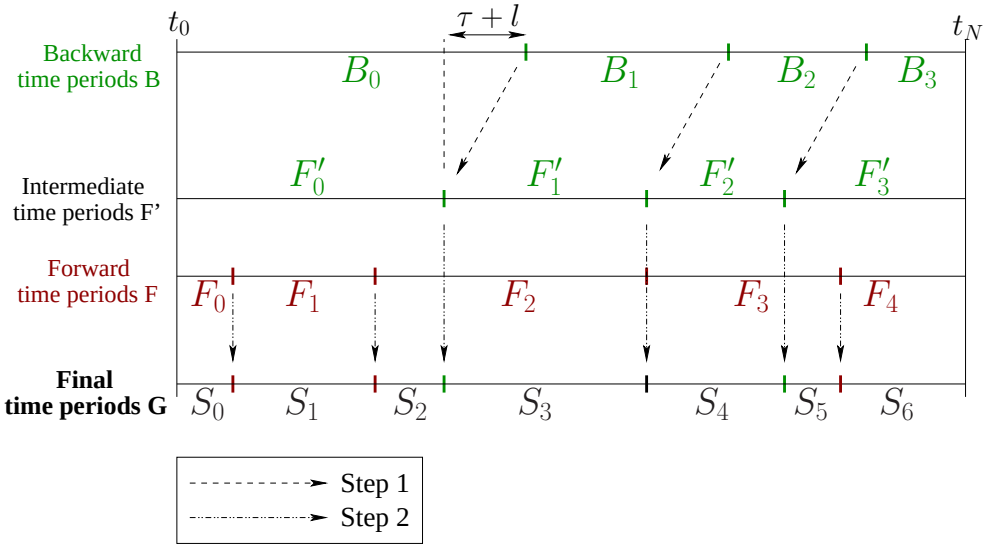
Let's suppose, for example, that the forward trip duration is a constant τ. Let's keep the same layover l for every couple of trips (forward / backward). This value can be chosen as the minimum layover between forward and backward trips in order to keep a good quality of service. For example, if $l = 0$ there is a high probability of delays running along the day due to the trip duration uncertainty.

Using these notations, if a forward trip starts at time t, the associated backward trip will start at time $t + \tau + l$. Inversely, if a backward trip is needed to start at time t, its associated forward trip will have to start **at most** at time $t - (\tau + l)$.

Let the backward trips time periods $B_0, B_1, \ldots, B_m$ be defined by the limit times $(b_1, b_2, \ldots, b_m)$. Then, we define the forward trips time periods to meet the backward demand with the following times: $F' = (b_1 - (\tau + l), b_2 - (\tau + l), \ldots, b_m - (\tau + l))$. This is represented in Figure 2 by the arrows between the backward time periods and the intermediate time periods (Step 1).

Note that here, we supposed that $b_1 - (\tau + l) > t_0$. If this was not the case, we can artificially set a new t_0 that satisfies the inequality, and add a new time period (between the new t_0 and the former) with null demand, for each given set of time periods.

Figure 2. Steps for the final time periods creation



At this point, the forward trips have two sets of time periods:

- the time periods $F = (f_1, f_2, \ldots, f_n)$ to satisfy the forward trips demand $D = (d(F_0), d(F_1), d(F_2), \ldots, d(F_n))$,
- the time periods $F' = (f'_1, f'_2, \ldots, f'_m)$ to satisfy the backward demand $D' = (d(F'_0), d(F'_1), d(F'_2), \ldots, d(F'_m)) = (d(B_0), d(B_1), \ldots, d(B_m))$.

The forward trips have to be timetabled in such a way that they meet the two sets of demands. Let's consider the intersection G of the two sets of time periods. More formally, $G = \{S_0, S_1, \ldots, S_p\}$ is defined by $(s_1, s_2, \ldots, s_p)$ such that:

- $s_1 < s_2 < \ldots < s_p$,
- $\forall k \in \{1, \ldots, p\}, \exists j \in \{1, \ldots, n\}, s_k = f_j$ or $\exists j \in \{1, \ldots, m\}, s_k = f'_j$,
- $\forall j \in \{1, \ldots, n\}, \exists k \in \{1, \ldots, p\}, f_j = s_k$,
- $\forall j \in \{1, \ldots, m\}, \exists k' \in \{1, \ldots, p\}, f'_j = s_{k'}$.

The arrows in the lower part of Figure 2 (Step 2) show an illustration of this intersection for a simple example. As a result:

- each time period F_j in F is the union of some time periods in G: $S_{k(j)}, S_{k(j)+1}, \ldots, S_{q(j)}$ $(0 \leq k(j) \leq q(j) \leq p)$,
- each time period F'_j in F' is the union of some time periods in G: $S_{k'(j)}, S_{k'(j)+1}, \ldots, S_{q'(j)}$ $(0 \leq k'(j) \leq q'(j) \leq p)$.

Hence, recalling that $\forall j \in \{0, \ldots, m\}, d(F'_j) = d(B_j)$, the demands $d(S_i)$ have to satisfy the following inequalities to be valid:

$$\forall j \in \{0, \ldots, n\} \quad , \quad d(F_j) \leq \sum_{i=k(j)}^{q(j)} d(S_i) \tag{1}$$

$$\forall j \in \{0, \ldots, m\} \quad , \quad d(B_j) \leq \sum_{i=k'(j)}^{q'(j)} d(S_i) \tag{2}$$

In the example of Figure 2, these constraints are:

$$\begin{aligned}
d(F_0) &\leq d(S_0) \\
d(F_1) &\leq d(S_1) \\
d(F_2) &\leq d(S_2) + d(S_3) \\
d(F_3) &\leq d(S_4) + d(S_5) \\
d(F_4) &\leq d(S_6) \\
d(B_0) &\leq d(S_0) + d(S_1) + d(S_2) \\
d(B_1) &\leq d(S_3) \\
d(B_2) &\leq d(S_4) \\
d(B_3) &\leq d(S_5) + d(S_6)
\end{aligned}$$

We have to determine the demand in number of trips $d(S_i)$ for each i, in such a way to create as few trips as possible and let the trips as periodical as possible (try to minimize the variations of headway). For this purpose, we introduce an integer program, where the (integer) variables are $d(S_i), 0 \leq i \leq p$ and the constraints are all the inequalities of type (1) and (2). We chose to use the following linear objective:

$$\min \sum_{i=0}^{p} \frac{d(S_i)}{\pi(S_i)} \tag{3}$$

where $\pi(S_i)$ is the duration of time period S_i. The advantages of this cost function are the following:

- it is linear and easy to solve,
- it will minimize the total number of trips,
- if there are several possible time periods for some trips, it chooses the longest time period.

This small size model (usually less than 30 variables and constraints) is then solved by an integer-programming solver to obtain the demands in number of trips, for each time period S_i created from the forward trips and the backward trips time periods.

In the following, we will omit the “forward” and refer simply to “trips”, as the backward trips will be created simply from each forward trip, as described earlier.

Note that in the special case of a line with a single direction (without backward trip), the resulting time periods are the initial forward trip time periods, with the original demands.

5. Generating trip timetables in each time period

Inside each of the created time periods, we now have to generate the trip timetable in order to let the headway between two forward trips as constant as possible. For this purpose, we chose a greedy approach with the following basic principles:

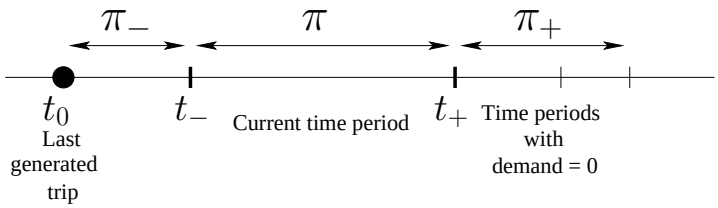
- the headway between two consecutive trips is constant in each of the (created) time periods,

- the time of the first trip in the current time period is calculated from the time of the last trip of the previous time period and the forecasted headway of the current time period, in order to let headways as uniform as possible,
- the headway for the current time period is raised as much as possible if it is followed by time periods with null demand.

Keeping a constant headway within a time period reduces the waiting time and makes the schedule easier to remember for passengers.

We consider each time period one after another. We have to determine, for the current time period, the headway and the time of departure of the first trip in the time period. Let t_0 be the departure time of the last generated trip, and the current time period defined by the times $[t_-, t_+]$. Let $\pi_- = t_- - t_0$ be the time between the last generated trip and the beginning time of the current time period, $\pi = t_+ - t_-$ the duration of the current time period, and π_+ the total duration of the next time periods without demand. Figure 3 illustrates these definitions.

Figure 3. Definitions for the determination of the headway



Let d be the demand in number of trips in the current time period. We assume that $d \geq 1$, as the case $d = 0$ is obvious (no trip created).

1. If possible, the headway is set to $h = \frac{\pi_- + \pi + \pi_+}{d+1}$, and the first trip departure time to $t_1 = t_0 + h$.
2. Using these values, if the last trip departure time is out of the time period (that is $t_d = t_1 + (d-1)h = t_0 + d \cdot h > t_+$), meaning that the additional time after the time period (π_+) is rather high, then we try to schedule the last trip at the end of the time period, using the following values:

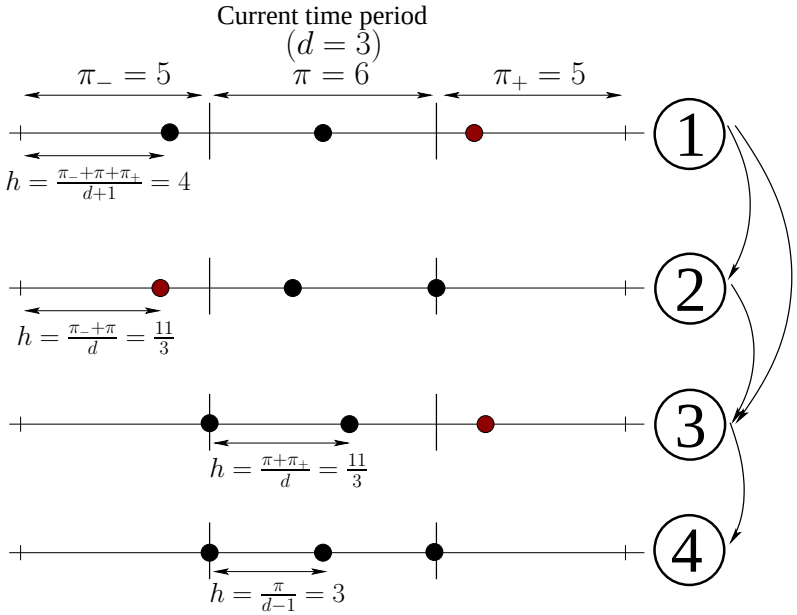
$$\begin{aligned} h &= \frac{\pi_- + \pi}{d} \\ t_1 &= t_0 + h \end{aligned}$$

3. Considering the values defined previously (step 1 or 2), if the first trip departure time is out of the time period ($t_1 < t_-$), we try to schedule it at the beginning of the time period:

$$\begin{aligned} t_1 &= t_- \\ h &= \frac{\pi + \pi_+}{d} \end{aligned}$$

4. With the values of step 3, once again it is possible that the last trip departure time goes out of the time period ($t_d = t_- + (d-1)h > t_+$), in which case it is scheduled at the end of the time period by simply reducing the headway: $h = \frac{\pi}{d-1}$. Of course, this can be done only if $d \geq 2$, as in the case $d = 1$, the (only) trip was already scheduled at the beginning of the time period at step 3.

Figure 4. Choice of the headway for the current time period



All these steps are illustrated with a 3-trip example in Figure 4.

More formally, h and t_1 can be seen as optimal solution values for the following linear program:

$$\max h$$

Subject to:

$$\begin{aligned} t_1 &\geq t_- && (4)\\ t_1 + (d-1)h &\leq t_+ && (5)\\ t_1 - t_- + \pi_- &\geq h && (6)\\ t_1 + dh &\leq t_+ + \pi_+ && (7) \end{aligned}$$

Constraints (4) and (5) ensure that the departure times for the first and the last trip (and hence all the currently generated trips) are included in the time period. Constraints (6) and (7) guarantee that the idle time before the first trip and after the last trip is not lower than the headway (so as to make the timetable more regular). Note that simplificated constraint (7) can be expressed more explicitly in the following way: $t_+ - (t_1 + (d-1)h) + \pi_+ \geq h$.

It can be shown easily (for example using a two-dimensional graphical approach) that the values given previously for h and t_1 are optimal with respect to this linear model.

6. Computational results

In order to get an evaluation of our algorithm, we decided to test the simultaneous forward and backward trip generation algorithm, and compare its results with the approach where the forward and backward trips are generated separately (often referred to in the following as “separated approach”), using only the part of the algorithm described in section 5 (as in this case, the initial time periods and demands can be used). As the application to

Table 1. Total demand in number of trips for forward trips and for backward trips, for each line

Line	Forward dem.	Backward dem.
L1	90	82
L2-1	30	49
L2-2	26	38
L3	47	67
L4	30	50
L5	29	51
L6	11	16
L7	20	24
L8-1	31	32
L8-2	35	38
All	328	371

real life data is the main point here, the tests are performed on datasets from a WplexON customer. The total numbers of trips required for the backward and the forward directions are summed up for each line in Table 1. In most of the line datasets, the daytime is divided into 18 time periods.

L2-1 and L2-2 are two different datasets for the same line (L2), as well as L8-1 and L8-2 for line L8. At the bottom of the table, “All” stands for all the lines together, including two circular lines (of total demand 40) which demands were added to the total demand in forward trips. In the “All” case, the alternatives L2-1 (for line L2) and L8-1 (for line L8) were chosen.

The trips are generated for every line separately and the results are given in Table 2. For each test, the blocks were generated after the trip timetabling, using the WplexON vehicle scheduling module and allowing deadheads between each pair of bus terminals.

Table 2 indicates in the first columns the number of trips and of blocks produced by the algorithm on the one hand and considering the forward and backward directions separately on the other hand. For example, line L1 produces 15 blocks using the algorithm and 18 in the separated case. Note that in the separated approach, the number of generated trips is exactly the sum of the demand in forward and backward trips, whereas for the algorithm, the number of generated forward trips equals the number of generated backward trips and is greater than the maximal value between the forward and the backward demand. The last columns contains the ratios between the improductive time of the algorithm and the improductive time of the separated approach, and the ratios of total duration of all the blocks of the schedule, generated from the trips of the timetable. The improductive time is the time during which a vehicle that performs a block is improductive (sum of idle time and deadhead time).

The first thing that can be pointed out is that the number of trips generated by the algorithm is always greater than for the separated approach. This is not surprising, as the separated approach always generate the exact number of trips from the demand, whereas the algorithm can generate more trips due to the restriction to generate a backward trip for each forward trip. In spite of this first observation, the number of blocks generated in the vehicle scheduling step is very often lower for the algorithm than in the separated

Table 2. Comparison between the algorithm and the solution obtained after separating the forward and backward ways

	# Trips		# Blocks		Time ratio	
Line	Alg.	Sep.	Alg.	Sep.	Improd.	Total
L1	212	172	**15**	18	1,0963	1,1953
L2-1	102	79	6	6	0,8074	1,0837
L2-2	84	64	5	5	0,6191	1,0238
L3	136	114	**6**	7	0,4330	0,8596
L4	102	80	6	6	0,9065	1,1499
L5	112	80	**10**	11	0,4541	0,9914
L6	42	27	**2**	4	0,5377	0,8195
L7	48	44	**4**	6	0,5043	0,8501
L8-1	80	63	8	8	0,5345	0,9582
L8-2	92	73	**10**	9	0,6188	0,9955
All	874	699	**48**	53	0,8949	1,1085

approach. For line L6 for example, only two blocks were necessary to perform the 42 trips generated by the algorithm, whereas the 27 trips generated when separating the forward and backward directions needed 4 blocks.

The total duration of the blocks (last column) is also often lower for the algorithm than for the separated approach, despite the higher quantity of trips. This means that the algorithm manages to leave a good set of trips for vehicle scheduling. We can also stress the fact that the improductive time ratio is lower than 1. The algorithm doesn't lead to improductive time, in spite of the fact that it generates more trips, because these trips are easily gathered in blocks, which is also the case in the multiline case (last line of Table 2).

All of these tests were performed within a very low computing time (two seconds for the case with all the lines together).

7. Conclusion and perspectives

In this paper, we presented a single line trip timetabling problem with the necessity to generate a good vehicle schedule from the created trips. Our algorithm, based on a redefinition of time periods that are the intersection of the forward trips time periods and the ones from the backward trips, generates more trips than necessary but outperforms a simple generation (considering the forward and backward directions separately) when creating schedules for vehicles. This is a good solution for the bus transit company, as using fewer buses lowers the total cost, but also for the passenger, who has a regular schedule of buses and a low waiting time. The fact that the algorithm creates slightly more trips than necessary is also positive for the customer satisfaction.

The repartition of the trips between the time periods can be improved, because with the linear cost (3) we used, the algorithm chooses a time period independently of the number of trips to generate. For example, if two trips can be generated in any of two time periods, the integer program will enforce the generation of both trips in the same time period (the one with the greatest duration), although it would be better in some cases to

generate one trip in each time period, especially if the difference of durations is small. To overcome this issue, one could use a quadratic objective or a dedicated heuristic. Another improvement would be to adapt the headway in each time period taking vehicle scheduling into account. This could be made possible minimizing the idle time between a generated backward trip and the next forward trip or "pushing" a forward trip in order to allow a same vehicle to perform both an already generated backward trip and this forward trip.

References

Ceder, A., Golany, B. and Tal, O. (2001). Creating bus timetables with maximal synchronization. *Transportation Research Part A: Policy and Practice* **35**, 913–928.

Fleurent, C., Lessard, R. and Séguin, L. (2004). Transit timetable synchronization: Evaluation and optimization. In *Proceedings of the 9th International Conference on Computer-Aided Scheduling of Public Transport*. San Diego CA (U.S.A.).

Fournier, S. (2009). Branch-and-price algorithm for a real-life bus crew scheduling problem. In L. Buriol, M. Ritt and A. Benavides, eds., *ERPOSul 2009 Anais*. Porto Alegre (RS, Brazil).

Schröder, M. and Solchenbach, I. (2006). Optimization of transfer quality in regional public transit. Technical Report 84, Fraunhofer Institut für Techno- und Wirtschaftsmathematik, Germany.

van den Heuvel, A. P. R., van den Akker, J. M. and van Kooten Niekerk, M. E. (2006). Integrating timetabling and vehicle scheduling in public bus transportation. Technical Report UU-CS-2008-003, Utrecht University, The Netherlands.

Voss, S. (1992). Network design formulation in schedule synchronization. In Desrochers and Rousseau, eds., *Computer-Aided Transit Scheduling*. Springer-Verlag, Berlin, pages 137–152.