Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 170 (2006) 844–862
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

Vehicle and crew scheduling for urban bus lines

Maikol M. Rodrigues [a,b], Cid C. de Souza [b,1], Arnaldo V. Moura [b,*,2]

[a] *Departamento de Informática, Universidade de Fortaleza, Fundação Edson Queiroz, Avenida Washington Soares 1321, 60.811-905 Fortaleza, Brazil*
[b] *Instituto de Computação, Universidade Estadual de Campinas, P.O. Box 6176, 13084-971—Campinas, Brazil*

Received 29 August 2002; accepted 18 June 2004
Available online 30 November 2004

Abstract

A solution to the urban transportation problem is given by vehicle and crew schedules. These schedules must meet the passenger demand and satisfy technical and contractual restrictions stemming from the daily operation of the lines, while optimizing some measure of operational cost. This work describes a computational tool developed to solve the urban transportation problem in the large metropolitan area of São Paulo, Brazil. The techniques used are based on integer programming models coupled with heuristics. The former produces good feasible solutions, and the latter improves the quality of the final solutions. While the operational and labor restrictions are specific to the city of São Paulo, the same ideas can inspire similar approaches for solving the urban transportation problem arising in other metropolitan areas.

Keywords: Scheduling; Crew; Vehicle; Transportation; Timetabling

1. Introduction

The urban transportation problem (UTP) has been the focus of several studies [34,8,32,17]. Most of its variations give rise to NP-hard problems [16,32,19]. Generally, the difficulties stem from a large set of complex and conflicting restrictions that must be satisfied by any solution. Most of these restrictions are reflected in a sizable number of operational conditions that involve specific characteristics of each bus line, among which are

* Corresponding author. Tel.: +55 19 3788 5859; fax: +55 19 3788 5847.
E-mail addresses: maikol@unifor.br (M.M. Rodrigues), cid@ic.unicamp.br (C.C. de Souza), arnaldo@ic.unicamp.br (A.V. Moura).
URL: http://goa.pos.ic.unicamp.br/otimo
[1] Supported by FINEP (ProNEx 107/97), and CNPq (300883/94-3).
[2] Supported by FAPESP, grant 99/05999-4.

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.06.035

the number of vehicles, the number of passengers that must be transported, the varying duration of each trip along the day and the vehicle capacities. Also, a number of local labor and safety regulations, such as the maximum number of work hours in a day and mandatory rest periods, further restrict the construction of viable journeys. The solutions must also optimize some objective function, which includes precisely measured items, such as operational costs, but may also involve other items whose measures may not be completely clear, such as maintaining trip departure times well spaced along the day.

In one possible strategy for solving the UTP, short term vehicle and crew schedules can be obtained first. Next, the long term planning of crew rostering is addressed. Of course, this particular sequence of steps admits variations [2,5,35,36]. In this work, we will concentrate on the vehicle and crew scheduling subproblems for which, broadly speaking, there are two approaches.

First, we have the sequential approach where these subproblems are treated separately [13,14]. In vehicle scheduling, the problem is to construct blocks of consecutive trips. Each block must start and end at a depot, while satisfying appropriate operational restrictions. The set of blocks so constructed must also meet the passenger demand and minimize some objective criteria, such as the distance traveled or fuel consumed [8,18,19,22–24]. A number of computational tools have been developed to construct vehicle blocks [7,31,19,28]. For the crew scheduling problem, a set of work schedules is sought, a work schedule being a sequence of consecutive trips, maybe interspersed with rest time [14,13]. Besides covering all trips occurring in the vehicle blocks, work schedules must also satisfy a number of labor and operational restrictions. In general, the objective function tend to be more complex in crew scheduling, it being a combination of fixed cost items, such as wages, and variable cost items, such as extra duty time. As a result, crew scheduling is usually harder to solve than vehicle scheduling, and simple heuristics for this problem often run into poor local optima [32]. Earlier codings for crew scheduling have been based on heuristics [30,20]. More recent results are based on mathematical programming techniques [3,34,12], or have used a hybrid approach [35,36]. After both subproblems have been solved, the solutions are unified by mapping work schedules, possibly separated by relief periods, into vehicle blocks, while still preserving the quality of the solutions [17,21,35,4,34,26]. The sequential approach has been applied in a number of practical cases [33,11], and such experiences revealed that while it may work well in certain situations, it may not yield adequate results when exercised over slightly different scenarios. Also, often the solutions obtained must be manually adjusted before they can be enforced in practice.

When there are interdependent restrictions that involve crew and vehicles, solving both problems independently may not lead to operational solutions [14]. A second alternative for solving the UTP resorts to more integrated approaches, which tend to yield more adequate final solutions [13]. Since the set of restrictions that apply to crews is usually more stringent than those that apply to vehicles, some of the restrictions inherent to the former may be shifted to the latter, leaving the final crew scheduling for a second step [9,27]. Earlier attempts towards an integrated approach used heuristics [1] and graph pairing techniques [29,25,10]. More recently, linear programming partition models have been proposed [15], combined with column generation techniques. The results indicated that problems with up to 25 trips could be solved, while problems with 30 or more trips remained intractable. A combination of Lagrangian relaxation and column generation techniques have also been implemented to simultaneously solve vehicle and crew scheduling problems with multiple depots for real instances of the UTP, in Italy [6]. Since full integration between vehicle and crew scheduling usually results in a much more complex problem, this approach has received less attention [13,14]. In this work, a loosely integrated approach is developed into a computational tool that can be used to automatically solve the UTP for large urban areas in and around the city of São Paulo, in Brazil. Since, for a typical company operating in that area, about 50% of the operational costs comes from wages and related items, even small improvements in the schedules can produce sizable savings.

The algorithms proposed here are based on a combination of integer programming models combined with some end game heuristics that improve upon intermediate results. The combination was able to deliver quite adequate final results, when compared to solutions produced manually by experts, for a number of real instances. Due to peculiarities in the problem restrictions, algorithms that produce good solutions for some urban area may not yield appropriate solutions when applied to other areas operating under slightly different conditions. Even so, the general ideas developed here might still be applicable to other regions, once they are adjusted to take into account discrepancies in the objective function or in the set of restrictions.

The article is organized as follows. The specific UTP investigated here is described in Section 2. A model for the UTP is discussed in Section 3. The next section presents results obtained by executing the algorithms over real data sets. The final section offers some concluding remarks.

2. The urban transportation problem

It will be assumed that there is a single depot. A control point (CP) is a previously defined location, along the line, where drivers can make a stop and rest. A trip is the act of driving the bus between two CPs. Each trip has both a duration and a direction, that is, an origin CP and a destination CP.

In the area under consideration, two kinds of urban lines are responsible for the bulk of the passenger traffic. Of the first kind are lines that go from a central region of the city to a point in the periphery, and back to the central region. In this case, two CPs are defined, usually located at a central terminal in the city and at the farthermost neighborhood point reached by the line. The other important kind are the lines that travel in a closed circuit through a certain neighborhood. In this case, only one CP is defined, at some stop conveniently located along the line.

The input data to the UTP consists of the following set of items: (i) a table, describing the number of passengers to be transported, in each hour of the day and in each direction; (ii) the number of vehicles available, all vehicles being identical; (iii) a parameter, referred to as the vehicle capacity, indicating the number of passengers that can be transported by one vehicle; (iv) the maximum number of vehicles that can be stationed at a CP, assumed uniform among all CPs; (v) the average trip duration between CPs, per hour and in each direction; and (vi) the average trip duration between CPs and the depot, per hour and in each direction. Fig. 1 illustrates the typical passenger demand distribution along a day.

The vehicle restrictions are: (i) the number of trips, in each hour and direction, must be enough to cover the passenger demand; (ii) the number of vehicles waiting at CPs must not exceed the allowed limit; (iii) the number of vehicles can not

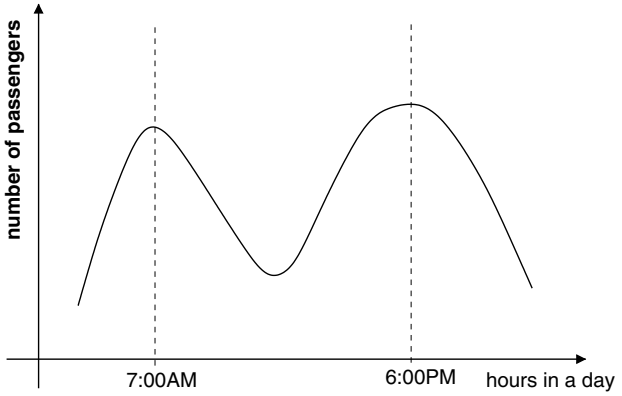

Fig. 1. Passengers to be transported along a typical day.

exceed the stated maximum; (iv) the departure times of the first and last daily trips, at each CP, must obey the respective times stipulated by the city authorities, these times being input as problem parameters; (v) a contractual requirement dictates that at the morning and the afternoon peaks, 7:00 AM and 6:00 PM, respectively, see Fig. 1, all vehicles must be out of the depot; and (vi) consecutive trip departure times, at each CP and in each hour, must be as uniformly spaced as possible, in which case the schedule is said to be well spaced, with a typical tolerance of two minutes from the ideal uniform distribution.

Crew restrictions are: (i) the maximum daily working time, excluding extra duty time, is 440 minutes; (ii) there is a limit on the number of extra duty minutes a crew can work on a day, this number being input as a problem parameter; (iii) every crew is entitled to 30 minutes of rest time; (iv) rest times must start no earlier than the second and no later than the sixth working hour; (v) the company might opt for a crew schedule with no rest time, provided that the maximum daily working time is reduced to 410 minutes; (vi) crews can start their work shifts at the depot or at a CP; (vii) a crew is designated to a single vehicle for the entire duration of its daily work schedule; and (viii) a constant relief time should be added to the duration time of the last trip assigned to each crew, unless this trip ends at the depot.

The objective is to construct vehicle and crew schedules satisfying the problem restrictions, while optimizing some operational criteria such as the number of vehicles, crews, or extra duty minutes.

3. An algorithm for solving the UTP

In this section, a hierarchical algorithm for solving the UTP is presented. The algorithm has four phases: (i) a preliminary schedule generator; (ii) a vehicle block generator; (iii) a final schedule generator; and (iv) a heuristics that adjusts trip departure times. Using an informal style, we now offer brief comments about each phase, since it is convenient to convey the overall idea before presenting specific algorithms for each phase. We assume that there are two CPs, designated CP_1 and CP_2—this being, by far, the most common case in practice.

In the first phase, a bipartite graph is constructed. Each node partition is associated with a distinct CP, and contains 1440 nodes, one for each minute of a day. Nodes are numbered from 1 to 1440, and we use node names in arithmetic expressions with the obvious meaning. For example, if s is a node then $s + 30$ stands for the time 30 minutes ahead of the minute denoted by node s. Using information about the duration of a trip between CPs, in each hour and direction, all viable edges are inserted in the graph. Let t and s be nodes in this graph associated with distinct CPs, say CP_i and CP_j, respectively. Edge (t,s) is viable if $D \leqslant (s - t) \leqslant D + W$, where D is the duration of a trip from CP_i to CP_j, at the hour corresponding to minute t, and W is the maximum allowed wait time for vehicles stationed at CP_j, at hour corresponding to minute $t + D$. Recall that the constants D and W are input data available to the algorithms. Now, the problem is to select vertices of this graph, representing trip departure times, in such a way as to satisfy the demand in each hour and at each CP. At the same time, the number of edges that are covered by the selected vertices should be maximized, where an edge is covered if both of its end nodes have been selected. Note that, for such an edge, the two trips associated with its end nodes can be done by the same vehicle without requiring too much waiting time on a CP, therefore reducing stationing. An integer linear programming model is constructed to solve this problem. The departure times so obtained are called primary start times. We stress that the set of primary start times will not necessarily coincide with the trip departure times in the final schedule. In the next phase, the primary start times are used to generate vehicle blocks.

A vehicle block comprises a sequence of consecutive trips designated to one vehicle, from the time it leaves the depot till it returns to the depot, where all vehicle blocks originate and end. When generating vehicle blocks, if a vehicle arrives in a CP at time t, possible continuations for this block are all primary start times from the same CP and lying the interval $[t, t + \delta]$. The choice of δ depends on the maximum wait times in the corresponding

CPs, and on the hour corresponding to minute t. Small values for δ tend to prevent the accumulation of vehicles stationed at CPs. Also, during this phase, a small constant is added to the duration of each trip. The stretching of trip durations will become important in the last phase, when an adjustment on the trip departure times is attempted.

In the third phase, the vehicle blocks are used in a classical packing or covering model in order to construct vehicle schedules satisfying the passenger demand in each CP and each hour. In the objective function, costs are computed by adding a bonus for each trip that departs from a primary start time and subtracting a penalty for each trip that does not depart from a primary start time.

Often, the schedule obtained at the end of phase three comprises a number of trip departures that do not coincide with primary start times and, also, are not well spaced. In these cases, the set of all present departure times is used to start the cycle again, at phase one. This is done by assuming that the demand, given in terms of number of trips at each CP and at each hour, is the maximum between the corresponding values computed at the initialization step and the number of trips in the solution at hand. Tests have demonstrated that, in these cases, a much better schedule results at the end of the second cycle. However, repeating the cycle more times produces an excessive number of primary start times, at phase one, and computation becomes inefficient.

After once or twice through the cycle involving phases one, two and three, the fourth and final phase starts. Here, a simple greedy heuristic adjusts the trip departure times that are still not adequately well spaced by the end of the cycle.

In what follows, each of the four phases is described in more detail.

3.1. Obtaining a set of primary start times

Suppose that there are two CPs. In the sequel, variables i and j will index one of the CPs; variables g and h, $0 \leqslant g$, $h \leqslant 24$ will denote hour values along a day; and variables t and s, $0 \leqslant t$, $s \leqslant 59$, will denote minutes. Before the first phase starts, the algorithm adds a constant Δ to the duration of each trip that starts at a CP. The effect is that, at later phases, each trip can have its duration time diminished by Δ minutes without violating the condition on the duration of each trip. The typical value for Δ, determined by experimenting, was five minutes. The new trip duration values so obtained will be indicated by D_{ih}, for CP_i and hour h. As mentioned earlier, parameters W_{ih} will represent the maximum stationing time for a vehicle at CP_i at hour h, and these values are given as input. When stationing is highly restricted, the parameter is usually set to one or two minutes, otherwise it may take a value from 1 up to 15 minutes. The number of departures from CP_i that are needed in order to meet the passenger demand in hour h will be indicated by d_{ih}. These values can be readily calculated from the input data.

The bipartite graph is $G = (V_1 \cup V_2, E)$, where (i) V_i contains all nodes n_{ihs} representing a possible departure time in CP_i at hour h and minute s and (ii) E contains all arcs (n_{ihs}, n_{jgt}) for which $D_{ih} \leqslant (60g + t) - (60h + s) \leqslant D_{ih} + W_{ih}$, $i \neq j$, see Fig. 2. In what follows, variables u and v will also be used to denote nodes in $V_1 \cup V_2$, when the internal structure of the nodes, namely, the corresponding CP, hour and minute, is not important.

Given G, the problem is to select subsets $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ such that the number of vertices in U_i, corresponding to hour h, is equal to d_{ih}. Moreover, the number of arcs of E that connect vertices between U_1 and U_2 is to be maximized. Before presenting an IP model for this problem, some extra notation is needed. Usually, the UTP is solved for a number of consecutive hours during the day, starting at early morning and ending at late evening. These hours will be denoted by $1, \ldots, H$. Tolerances are defined to control the minimum and maximum spacing between consecutive trip departures. For each CP_i and hour h, $h \in \{1, \ldots, H\}$, the upper and lower tolerances are given by T_{ih} and t_{ih}, respectively, where $T_{ih} = \lfloor 60/(d_{ih} + 1) \rfloor + \beta + 1$ and $t_{ih} = \lfloor 60/(d_{ih} + 1) \rfloor - \beta$. The values chosen for the constant β decreases as the value of d_{ih} increases, as shown in Table 1.

Returning to the IP model, for each node $u \in V_1 \cup V_2$, a binary variable x_u is defined to be 1 when u is in $U_1 \cup U_2$. For all nodes u and v the binary variable y_{uv} is defined to be 1 when $u \in U_i$

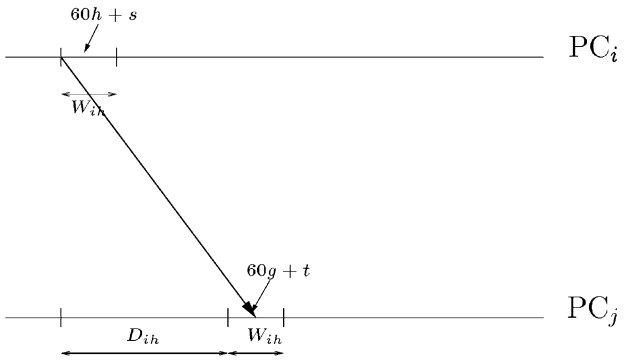

Fig. 2. Compatible start times.

Table 1
Constants for tolerances

d_{ih}	>15	⩾9, ⩽15	⩾4, ⩽8	⩾1, ⩽3
β	0	1	2	4

and $v \in U_j$ with $i \neq j$. The objective function is written as $\max \sum_{(u,v)\in E} y_{uv}$. There are three sets of restrictions. First, x_u and y_{uv} must be compatible, that is, $y_{uv} \geqslant x_u + x_v - 1$, $y_{uv} \leqslant x_u$ and $y_{uv} \leqslant x_v$, for all $(u,v) \in E$. Next, departure times must be properly spaced:

$$\sum_{s=t}^{t+t_{ih}-1} x_{n_{ihs}} \leqslant 1, \quad h = 1,\ldots,H; \quad t = 0,\ldots,60 - t_{ih}; \quad i = 1,2;$$

$$\sum_{s=t}^{t+t_{ih}-1} x_{n_{ihs}} \geqslant 1, \quad h = 1,\ldots,H; \quad t = 0,\ldots,60 - t_{ih}; \quad i = 1,2.$$

If a trip is required from CP_i at hour h and minute s, then the restriction $x_{n_{ihs}} = 1$ is added to the model. Third, the number of departures must meet the demand, for each CP and in each hour:

$$\sum_{s=0}^{59} x_{n_{ihs}} = d_{ih}, \quad h = 1,\ldots,H; \quad i = 1,2.$$

The output of this phase is the set of primary start times whose corresponding x variables have value of 1 in the optimal solution for the IP model.

3.2. Constructing a set of vehicle blocks

The output of this phase is a set of vehicle blocks computed from the set of primary start times generated in the previous phase.

The algorithm creates three types of blocks, similar to those produced by experts who solve the UTP manually. Blocks that span only the morning and afternoon peaks, see Fig. 1, are of type I_A and I_P, respectively. The third type of block comprises a longer journey, spanning both demand peaks. We call these blocks of type I_L. Any sequence of trips that form a shorter block is guaranteed to satisfy all restrictions that apply to crew shifts. The same holds for the sequences of trips separated by relief points in the longer blocks. As a consequence, once the vehicle blocks of a solution have been selected, a complete solution to the crew assignment problem is at hand. Vehicles assigned to shorter blocks will be operated by a single crew and, of course, vehicles assigned to longer blocks will be driven by two distinct crews, with a relief period in between. Also, blocks of type I_A and I_P are initially treated as independent entities. After all blocks have been chosen, the algorithm attempts to pair shorter blocks to form longer blocks, of type I_L. In the next paragraphs, we describe the three steps of vehicle block construction: initiation, iteration and termination.

Block initiation: For each CP_i consider the interval $[t', 420 - \gamma]$, where 420 is the time, in minutes, of the morning peak, t' corresponds to one

minutes before the earliest primary start time computed for CP_i, and γ is an internal constant whose typical value is 30 minutes. For each primary start time $t \in [t', 420 - \gamma]$, blocks of both types I_A and I_L are initiated by a single trip starting at the depot and ending at CP_i, at time $t - 1$. Departure times for these initial trips can be computed from the input data and the hour corresponding to time $t - 1$. In a similar way, for each CP and each primary start time in the interval $[1080 - \gamma', 1080 - \gamma]$, a block of type I_P is initiated. Here, 1080 corresponds, in minutes, to the afternoon peak, γ is the same constant as before, and γ' is another internal constant, with a typical value of 150 minutes. Values for γ and γ' were established by experimentation.

Block iteration: To see how the next trip is selected, suppose that the last trip ended in CP_i, at time t. Consider the interval $[t, t + \delta]$, where δ represents a fraction of the maximum allowed wait time, W_{ih}, for vehicles stationed at CP_i and at hour h corresponding to minute t. For all primary start times $t' \in [t, t + \delta]$, a possible continuation for the current block is attempted by adjoining to it the trip that starts at time t' and goes from CP_i to the other CP, each such possible continuation giving rise to a different vehicle block. If no primary start time exists in this interval, a trip departing at time $t + 1$ is added to the block, thereby preventing the vehicle from being stationed at the CP. A constant δ was selected for each hour, in such a way that a smaller value was used at those hours that required more departures. Typical values for δ ranged from 5 to 10 minutes. Some variations of this method were also implemented, but they all proved to be inferior, in the sense that the set of blocks produced lead to final solutions that were not as adequate as those obtained with the algorithm presented. In particular, using a constant value for δ, or choosing only one of the possible primary start times within the corresponding time interval, did not lead to better final solutions to the UTP.

Block termination: Consider a block with elapsed time t_ℓ and whose last trip ended at CP_i at time t. Let t' be the primary start time selected for the next attempt at adding a trip to this block, and let t_d be the duration of such a trip. From the input data, the algorithm retrieves t_p, the duration of a trip from CP_i to the depot at the hour corresponding to minute $(t + t_d + 1)$. If this trip were added to the block, its new elapsed time would be given by $f = (t_\ell + (t' - t) + t_d + 1 + t_p)$. If the block is a short one, i.e. of type I_A or I_P, the algorithm checks if f exceeds the total working time, t_w, permitted for a crew. If so, the block is terminated by adding to it a trip from CP_i to the depot, departing at time $(t + 1)$. Otherwise, the selected trip is added to the block and its construction proceeds. Here, the value of t_w is the sum of two terms: the regular daily work time, given as input, and the maximum amount of extra time that the company is willing to compensate for. This last term, set by the user, is a parameter that allows for a certain degree of control over the costs incurred by extra time. Typical values for this parameter were from one to two hours. Now, consider a longer block, of type I_L. When these blocks are initiated they are marked as having one crew assigned to them. If this is the case when the next trip is selected, the algorithm checks if $f \leqslant t_w$. If so, this trip is added to the block. Otherwise, the minimum relief time is added to the block, and it is now marked as having two crews assigned to it. In any case, the block construction continues. Finally, if by the time the next trip is selected the block is already marked as a two-crew block and $f \leqslant 2t_w$, then the trip is added to the block and construction continues. Otherwise, the construction is terminated by adding a trip to the depot, departing at time $(t + 1)$ from CP_i. The minimum relief time is a parameter also given as input data and is usually set to 10 minutes.

Manually generated solutions may also include a fourth kind of block, which encompasses the duties of three different crews, separated by two intermediate relief points. But the latter is used much more rarely and, therefore, is not considered by the algorithm.

Tests have shown that the behavior of the algorithm is sensitive to the values of the minimum relief time and the maximum wait time between two consecutive trips, and most critically to the latter. If the wait time is too long, the number of blocks generated could be excessive, and the procedure

used to select promising blocks, discussed in the next subsection, could stall.

3.3. Constructing the schedule

The input data is the set of vehicle blocks enumerated in the preceding phase. The output will be a solution to the problem, namely, a subset of blocks that constitutes an adequate schedule. The schedule is constructed by solving an IP model, which is described next.

First, we describe the variables of the model. Assume that there are two control points, CP_1 and CP_2. Let B be the total number of vehicle blocks obtained in the second phase. Let NP_ℓ and NT_ℓ be the number of primary start times and trips, respectively, that occur in block ℓ, for $\ell = 1, \ldots, B$. The total number of primary start times in the solution is $NP = \sum_{\ell=1}^{B} NP_\ell$ and the total number of trips is $NT = \sum_{\ell=1}^{B} NT_\ell$. Define the binary matrix A by letting $A_{k\ell}$ be 1 if and only if block ℓ has a trip departing at primary start time k, where $k \in [1, NP]$ and $\ell \in [1, B]$. Define the binary variable z_ℓ to be 1 if and only if the ℓth block is in the solution, for $\ell \in [1, B]$.

The first set of restrictions avoids multiple departures at the same primary start time by requiring $\sum_{\ell=1}^{B} A_{k\ell} z_\ell \leqslant 1$, for $k \in [1, NT]$. If trip k is a required trip, this constraint is written as an equality. Let V be the number of vehicles available to operate the line. Recall from Section 3.2 that there are three kinds of blocks. Let VA be the set of short blocks of type $\mathrm{I_A}$ that span the morning peak, let VP be the set of short blocks of type $\mathrm{I_P}$ that span the afternoon peak, and let VL be the set of long blocks of type $\mathrm{I_L}$, spanning both peaks. It is clear that only blocks from VA and VP can be designated to the same vehicle. The restriction on the number of vehicles can be written as $(\sum_{\ell \in VA} z_\ell) + (\sum_{\ell \in VL} z_\ell) \leqslant V$ and $(\sum_{\ell \in VP} z_\ell) + (\sum_{\ell \in VL} z_\ell) \leqslant V$.

It remains to describe the objective function. Having fixed, in the previous phases, the constants which directly affect the cost of the solution, such as maximum number of vehicles, the number of crews per block and the limit for extra duty time, we seek to maximize the occurrences of primary start times in the final solution. Ideally, all departure times would coincide with primary start times. The objective function is written as $\max \sum_{\ell=1}^{\ell=B} (C \times NP_\ell - P_\ell) z_\ell$, where P_ℓ is a penalty associated with block ℓ, for $\ell \in [1, B]$, and C is an internal constant. We note that NP_ℓ can be computed by the end of the previous phase. As will be seen, the same is true for the P_ℓ values, which will measure how distant the actual departures are from the primary start times, in block ℓ. By varying P_ℓ, a large number of possibilities can be tried. The internal constant C was chosen to satisfy $C > \sum_{\ell=1}^{\ell=B} P_\ell$. Therefore, between two solutions, the algorithm will always prefer the one with more occurrences of primary start times. In addition, the objective function should also lead the algorithm toward solutions where departure times are as uniformly spaced as possible. Two different strategies to compute P_ℓ were implemented, as discussed below.

Maxmin: Here, we let $P_\ell = NT_\ell - NP_\ell$, charging a constant penalty of one for each departure time which is not a primary start time. Fig. 3(a) illustrates this cost function. Clearly, P_ℓ can be computed by the end of the previous phase of the algorithm.

Quadratic: In this case, each departure time that is not a primary start time imposes a penalty that decreases quadratically with the distance to the nearest primary start time in the same block. The idea is depicted in Fig. 3(b). More precisely, we let $P_\ell = \sum_{e \in DB_\ell} p_e$, where DB_ℓ represents the set of departures in block ℓ, for $\ell \in [1, B]$, and p_e is a penalty associated with departure e, whose value is computed as follows. Suppose that t and t', $t < t'$, are the two consecutive primary start times such that $e \in [t, t']$. Let $a_1 x^2 + a_2 x + a_3$ be the unique parabola passing through the points (t, C'), (t', C') and $((t + t')/2, 0)$, with C' being a constant that, in our experiments, was set to 10. Then, $p_e = a_1 e^2 + a_2 e + a_3$.

Table 2 gives some results when three real instances of the UTP were put through the first pass of the cycle. In this table, column labeled "Deficit" registers how many additional trips were needed in order to meet the demand. Column under "Trips" shows the total number of trips in the solution, and column labeled "% Opt" shows the deviation from the optimal solution of the corresponding IP

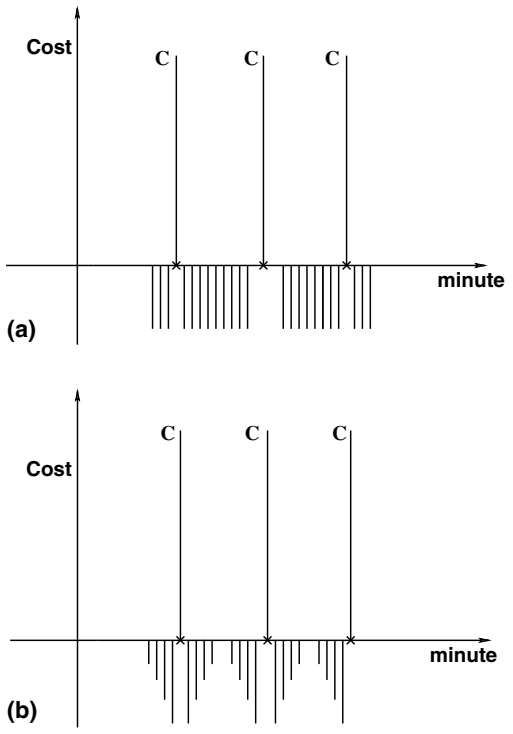

Fig. 3. Two cost strategies: (a) Maxmin strategy and (b) Quadratic strategy.

Table 2
Comparing strategies

Instance	Maxmin			Quadratic		
	Deficit	Trips	% Opt	Deficit	Trips	% Opt
OSO1	0	220	0	3	190	0
OSO2	2	199	0	0	193	0
OSO3	0	257	0	3	190	0

packing problem. Note that the number of trips is significantly smaller when the quadratic strategy is used. This same tendency was present when other real instances were tested. In some cases, the number of trips generated was not enough to satisfy the passenger demand, although this deficit was often reduced after the second pass through the cycle. Also, the solutions obtained using the quadratic strategy were substantially better with respect to the spacing between trips. Using the maxmin strategy, from 15 to 30 trips were not well spaced. With the quadratic strategy, this number was always under 10. Here, "well spaced" reflects a subjective indication of "uniform spacing", as expressed by trained personnel who build schedules by hand on a daily basis.

3.4. A heuristic for spacing consecutive departures

Usually, after the second pass through the cycle, the algorithm yields a fairly adequate solution. However, it may still happen that the spacing between consecutive trips, at some hour intervals, is not yet satisfactory. In order to obtain a better spacing, a greedy strategy was implemented as the fourth and final phase. The heuristics moves along the day, adjusting departure times within each hour interval in turn. The whole process is iterated a number of times in order to obtain a cumulative effect. In this section, the heuristic is described in more detail.

The input is the schedule obtained in the preceding phase. The output is a similar schedule, were departure times may appear shifted by small amounts.

Let NT be the total number of trips. Recall from Section 3.1 that, before the first phase begins, Δ minutes are added to the duration of each trip. The effect is that, at the present stage, each trip can have its duration time diminished by Δ minutes without violating the restrictions on trip duration. The heuristics moves along the minutes of the day maintaining a set of values l_a and r_a representing, respectively, the maximum left and right shifts that can be added to the departure time of trip a. Initially, both l_a and r_a are set to Δ, for $a \in [1, NT]$, unless a is a required trip in which case, $l_a = 0$ and $r_a = 0$.

Suppose that all trips have been ordered in a sequence and let b, $c \in [1, NT]$ also represent trips. Suppose that the algorithm is now considering trip b, occurring in hour h. Let a and c be the trips that come before and after b, respectively, and departing from the same CP. Let t_w be the departure time of trip w, for $w \in \{a, b, c\}$, with the proviso that t_a is the first minute of h if b is the first trip in that hour, and t_c is the last minute of h when b is the last trip of h. Note that the heuristics has just repositioned trip a. Let $\sigma = \lfloor (t - t_a)/(n + 1) \rfloor$, where t is the last minute in h and n is the number of trips

Table 3
Urban lines characteristics

Instance	CP_1	CP_2	Region	Comment
OSO1	182	181	SP	Large number of trips at peaks
OSO2	183	170	SBC	Initial and final trips at CP_1
OSO3	78	42	SBC	A micro-bus line
OSO4	100	75	SP	Limited number of stationed vehicles at CP_2
OSO5	85	60	SP	–
OSO6	100	–	SP	Only one CP
OSO7	95	97	SP	Trips must start at CP_1

that were not yet repositioned in h. Clearly, $t' = t_a + \sigma$ is the ideal departure minute for trip b. If $t' < t_b$, the new departure time for trip b is set to $t'_b = \max(l_b, t')$, otherwise it is set to $t'_b = \min(r_b, t')$. It is also necessary to prepare for the next iteration. When $t'_b < t_b$, the new values of r_a and l_c are set to $r'_a = r_a - (t_b - t'_b)$ and $l'_c = l_c + (t_b - t'_b)$. Similarly, when $t'_b \geqslant t_b$ the new values of r_a and l_c are $r'_a = r_a + (t'_b - t_b)$ and $l'_c = l_c - (t'_b - t_b)$. Notice that these updates do not take place when a or c are required trips.

There are several ways to sequence the trips: order them by departure times; list the trips from one CP followed by the trips from the other CP; interleave the trips from both CPs based on their departure times, to name just a few. Experimentation showed that the best results with respect to the final spacing between consecutive departure times were obtained by listing first the departure times from CP_1, followed by the departure times from CP_2.

After a few passes of this left to right scanning and repositioning cycle, the departure times tend to reach a stable position. Experience showed that three iterations through the cycle were enough to reach a much better solution. Tests over real data have shown that the heuristics is very effective in improving the distribution of departure times in the final schedule, as discussed in the next section.

4. Results on real data

This section reports on some results obtained by running the algorithm over seven real instances of the UTP. All data was collected from three companies that operate in the large metropolitan regions of São Paulo and São Bernardo do Campo, in Brazil.

Table 3 presents the test instances. Columns labeled "CP_1" and "CP_2" give the minimum number of trips necessary to meet the passenger demand at each CP; column marked "Region" gives the region were the lines operate, SP standing for São Paulo and SBC for São Bernardo do Campo; and column labeled "Comments" display peculiarities of the corresponding line. Instances OSO3 and OSO7 are typical and will be the most used when presenting results. More details about these instances appear in Table 4. The maximum on duty time is smaller for line OSO7, as a result of different local union agreements in the corresponding regions. Table 5 shows the trip input data for these two instances, where the "Hours" column indicates the hour, in a 24 hour clock. Columns labeled "T(1-2)" and "T(2-1)" indicate the duration of trips, in minutes, from CP_1 to CP_2 and from CP_2 to CP_1, respectively. Columns marked "P(1-2)" and "P(2-1)" indicate the number of passengers to be transported, at each hour interval, between the two CPs. It is clear that trip

Table 4
Input data for test instances OSO3 and OSO7

Item	OSO3	OSO7
Number of vehicles	15	26
Vehicle capacity	27	80
First trip at time	5:30 AM	4:30 AM
Mandatory trip at time	23:00 PM	24:00 PM
Rest time	30 minutes	30 minutes
Time from depot to CP_1	25 minutes	25 minutes
Time from depot to CP_2	–	50 minutes
Maximum on duty time	465 minutes	430 minutes
Vehicle schedules must start at	CP_1	CP_2

Table 5
Input data for test instances OSO3 and OSO7

Hour	OSO3				OSO7			
	T(1-2)	P(1-2)	T(2-1)	P(2-1)	T(1-2)	P(1-2)	T(2-1)	P(2-1)
04–05	–	–	–	–	60	23	0	50
05–06	50	60	50	00	65	144	50	50
06–07	50	128	50	35	70	358	55	218
07–08	55	189	55	62	90	596	70	304
08–09	55	135	55	51	90	320	80	380
09–10	55	99	55	48	90	312	80	270
10–11	55	79	50	35	90	340	80	260
11–12	55	101	50	30	90	264	80	250
12–13	50	80	50	38	90	304	75	270
13–14	50	98	50	29	95	445	75	300
14–15	50	102	60	55	100	348	80	354
15–16	50	138	60	72	105	373	85	355
16–17	50	182	60	86	105	265	85	360
17–18	60	155	50	101	110	320	85	560
18–19	60	160	50	145	100	160	85	400
19–20	60	122	50	140	85	109	70	160
20–21	50	115	50	78	80	63	70	85
21–22	50	80	50	69	65	58	65	96
22–23	50	50	50	8	65	34	65	15
23–24	–	–	–	–	65	11	55	11

durations are more uniform in line OSO3, due to less intense traffic in that region.

The general UTP is a complex optimization problem. In practice, experts relax the problem and consider only a small number of candidate blocks when constructing manual solutions. Blocks are constructed by using the full length of a crew working day, including extra duty hours the company is willing to pay for. While the problem restrictions have to be observed, its complexity and size often force experts to tolerate violations, to a certain extent.

4.1. Vehicle schedules

Fig. 4 shows the details of a manual and an automatic solution for the vehicle schedule, using data from OSO7. The vertical axis displays the blocks, one per line. The horizontal axis displays the time, using a 24 hour clock. For each block, a horizontal scan along the corresponding line shows its trips. Each blue rectangle represents one trip and the diagonal lines inside the rectangles show the alternation of trips that depart from both CPs. The gray rectangles stand for rest times and the green rectangles at the extremities indicate trips from and to the depot. In Fig. 4(a), blocks from lines 1 to 6 are short blocks of type I_A, while from lines 7 to 12 we have I_P type blocks. Clearly, these blocks can be paired to form longer vehicle blocks. Thus, the total number of vehicles used both in the manual and automatic solutions was 26. This was the case for all tested instances. In fact, the algorithm was able to construct a solution that used only 25 vehicles for the OSO7 instance. Similarly, for the OSO3 instance a solution with one less vehicle than the manual solution was also found. In both cases, however, departure times were not as evenly distributed when compared to the solutions that used the same number of vehicles. We stress again that the automatic solutions, of course, obeyed all of the problem restrictions, while in the manual solutions a certain degree of violation was always present. Also, as can be seen, the algorithmic solution shows a preference for pairing shorter blocks. This facilitates the scheduling of preventive vehicle servicing, since short blocks bring the cars back to the depot for a number of hours around midday.

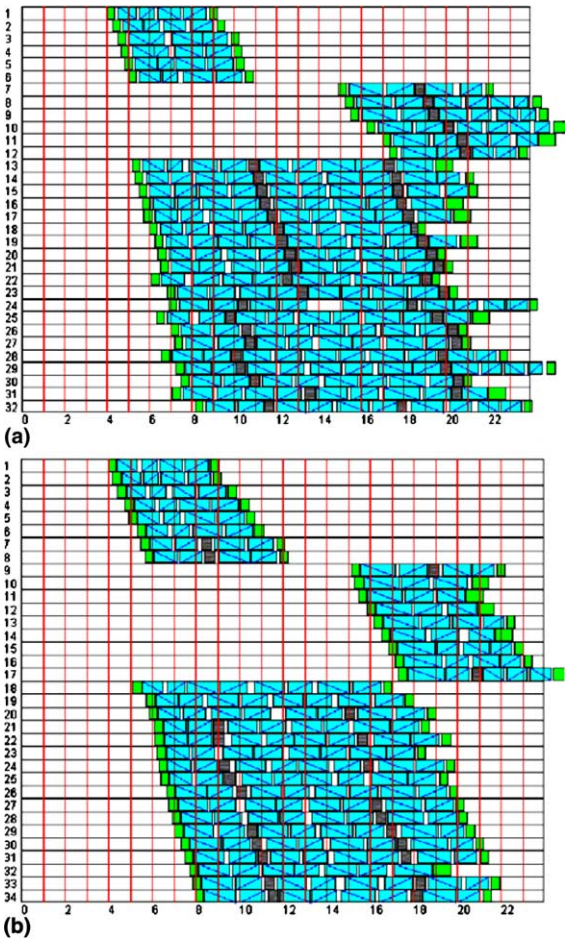

Fig. 4. Comparing vehicle schedules for OSO7. (a) Vehicle schedule, manual solution for OSO7. (b) Vehicle schedule, algorithmic solution for OSO7. (For interpretation of the references in color in this figure, the reader is referred to the web version of this article.)

It is also important to verify the number of vehicles stationed at each CP and if all vehicles are out of the depot by the morning and afternoon peaks. As discussed in Section 3, the packing formulation for the vehicle block generator does not explicitly enforce the rule for the maximum number of vehicles stationed at CPs. What the models do require is that the maximum allowed wait time at CPs is not exceeded. When the maximum wait time is fixed to a small constant, the algorithm is able to exercise some control over the number of vehicles stationed at CPs. Fig. 5 shows the total number of vehicles stationed at CP_1, for instance OSO3. The vertical axis depicts the number of vehicles, and the horizontal axis shows the time in a 24 hour clock. It is clear that in the algorithmic solution the number of vehicles stationed in both CPs was smaller, when compared with the same number derived from the manual solutions. The same effect was observed with most of the other test instances. In some cases, however, the same number of maximum-stationed vehicles was attained by both solutions. Such was the case with instance OSO7 at CP_1, in which both solutions showed a maximum of five stationed vehicles, at some moments.

4.2. Crew schedules

Fig. 6 shows both the manual and the algorithmic solutions. Legends are the same as for Fig. 4 with the difference that the vertical axis displays the crew and the red bars indicate extra duty hours. Clearly, the automatic solution uses one less

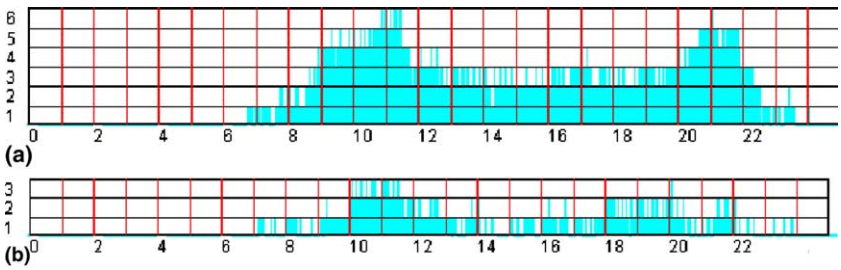

Fig. 5. Total number of vehicles stationed at CP_1 for OSO3. (a) Manual solution, vehicles stationed at CP_1 for OSO3. (b) Algorithmic solution, vehicles stationed at CP_1 for OSO3. (For interpretation of the references in color in this figure, the reader is referred to the web version of this article.)

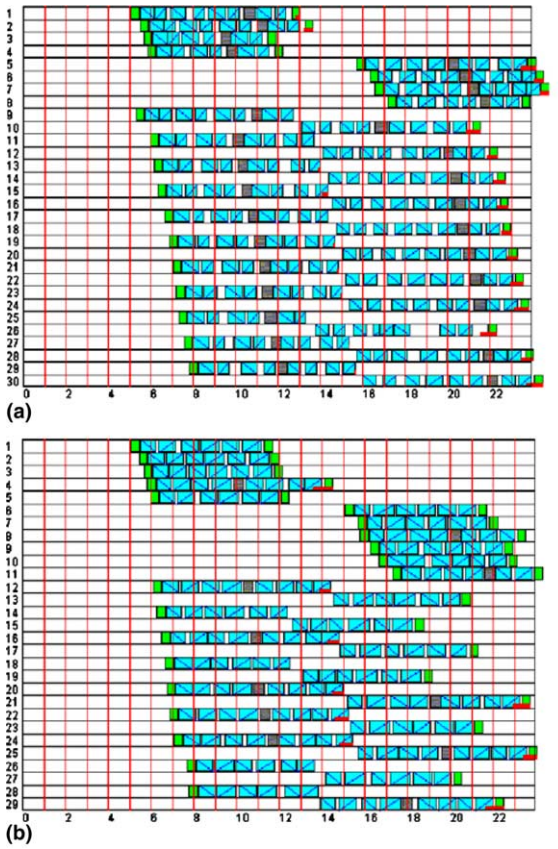

Fig. 6. Comparing crew schedules for OSO3: (a) Manual solution and (b) Algorithmic solution. (For interpretation of the references in color in this figure, the reader is referred to the web version of this article.)

crew to operate OSO3, when compared with the solution produced manually. This behavior was typical when testing several other instances.

The automatic solution showed a sizable gain in the number of extra duty hours. This was one of the most important advantages of the automatic solution over manually constructed ones, since labor costs are responsible for more than 50% of the total operational cost.

Fig. 7 compares the total working hours in the manual and the algorithmic solutions, using instance OSO3. The vertical axis lists crews and the horizontal axis shows the workload, measured in hours. The blue color stands for hours spent in driving, the yellow color shows inactive time that accumulated along the day, and the red color measures extra duty hours. As can be seen, the automatic solution showed a larger number of inactive hours, but reduced the extra duty hours.

4.3. Number of passengers transported

Here, the instance used in these tests was OSO7. Fig. 8 shows the result of the manual and automatic schedules. As usual, the horizontal axis shows the time, in a 24 hour clock. The vertical axis shows the number of passengers transported, per hour. In each hour interval, the first bar refers to CP_1 and the second to CP_2. A red bar indicates a demand that was not met, while the blue bar indicates transported passengers. As can be seen, the automatic solution did not meet the demand in a few hour intervals close to the beginning and the end of the day, although the deficit was small. On the other hand, the algorithm always respected the maximum work time for all crew and the maximum number of passengers transported in each trip, never exceeding the vehicle capacities. In the manual solutions, it was not rare to encounter trips were the number of passengers transported was above the vehicle capacity or the crew was working beyond its maximum daily working hours, including extra duty hours. By extending a little the number of extra working hours of only two crew, the algorithm was able to construct a schedule where all passengers were indeed transported. This strategy always produced adequate results, in all tested cases.

Table 6 shows the result of running the algorithm over all test instances. Columns labeled "HI" and "HT" show the number of hour intervals were the demand was not met, and the total number of hour intervals considered, respectively, summing up for both CPs. Columns labeled "TMax." and "TTot." indicate, respectively, the maximum number of trips that were not scheduled in an hour interval and the total number of trips that were not scheduled, over all hour intervals. The last column shows the number of passengers that were not transported. Clearly, in all cases, with the exception of OSO2, the maximum number of trips not scheduled was two.

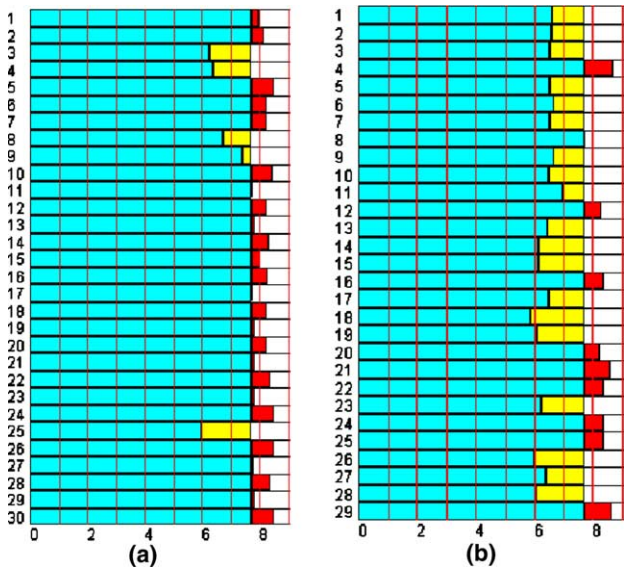

Fig. 7. Comparing work hours for OSO3: (a) Manual solution and (b) Algorithmic solution. (For interpretation of the references in color in this figure, the reader is referred to the web version of this article.)

4.4. Interval between consecutive trips

Trips lumped together at small intervals make for a bad schedule and can cause discomfort to passengers that may have to wait too long for the next trip. Tables 7 and 8 show the manual and algorithmic departure times, respectively, at CP_1 and when taking OSO3 as the input instance. The top horizontal line marks the hours, in a 24 hour clock. The vertical list below each hour gives the trip departure times, in minutes within that hour. The manual solution produced a somewhat more uniform spacing between consecutive trips, although the automatic solution was also quite adequate except, possibly, when the hour barriers were crossed. This reflects the fact that the heuristics works by taking each hour interval at a time, independently, and does not consider discrepancies that might occur when there is a change to the next hour interval. For CP_2, essentially the same characteristics were present in the solutions, although the spacing between trips in the algorithmic solution at CP_1 was always somewhat more uniform when compared with the automatic solution at CP_2. This can be attributed to the scanning strategy used by the heuristics, which sweeps the departure times at CP_1, and only after all of these have been considered it scans the times at CP_2.

When examining data from other instances, it was noticed that the interval between trips is more evenly distributed when there is a larger number of trips in an hour interval. One possible reason for such behavior is that the small displacements introduced by the heuristics can cause more impact in the overall distribution of the spacings between trips in an hour interval with more departures.

4.5. Processing time

Fig. 9 shows the processing times over four of the tested instances, namely OSO2, OSO3, OSO5 and OSO7. The vertical axis shows the time, in hours. For each instance, the lighter bar shows the time required when running on a powerful DEC-DS20 server, and the darker one indicates the running times when using a PC platform. The DEC server was a 675 MHz machine, with 4 GB of main memory and equipped with an Alpha 2264 processor with 64 bit capability. The PC platform was a PENTIUM III 450 MHz processor with 384MB of main memory. The LP solver used was cplex-7.0 on the DS20 server and

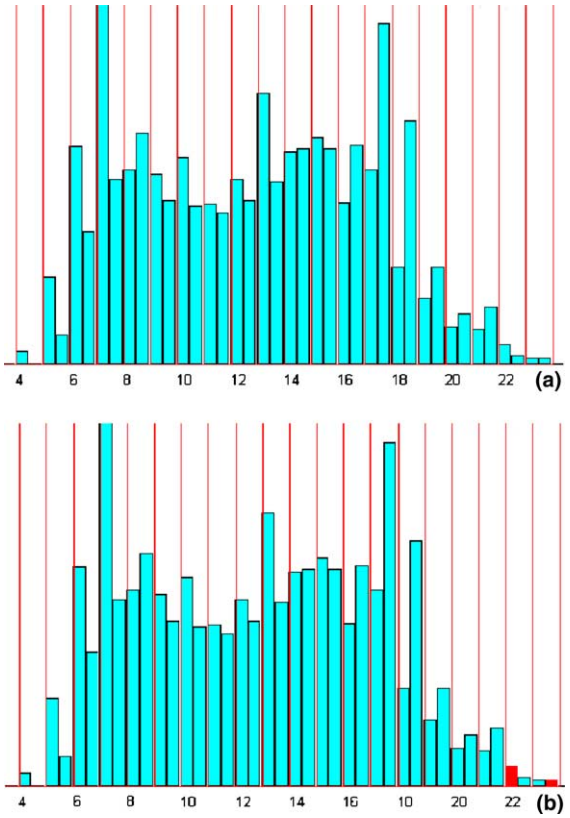

Fig. 8. Passengers transported using OSO7: (a) Manual solution and (b) Algorithmic solution. (For interpretation of the references in color in this figure, the reader is referred to the web version of this article.)

cplex-6.5 on the PC. Such processing times, although in the range of a few hours, are still quite an improvement over the corresponding times, in the range of a few days, that the company experts take to construct the manual solutions. Moreover, the processing times in the PENTIUM PC, a modest machine by today standards, shows that the algorithm can be run on a typical and affordable office machine.

Table 6
Passengers not transported, automatic solutions

Inst.	HI	HT	TMax.	TTot.	TPass.
OSO1	0	40	0	0	0
OSO2	4	40	2	6	150
OSO3	1	38	1	2	26
OSO4	0	40	0	0	0
OSO5	0	40	0	0	0
OSO6	1	39	2	2	130
OSO7	2	39	1	2	50

4.6. Summary

This section offers a summary of the strong and weak points of the automatic solutions. Tables 9 and 10 group some important data. The second and third columns list the number of inactive hours and the number of extra duty hours, respectively, in the corresponding solution. A time interval is inactive if the crew is on duty but not working at that interval. Columns labeled "Trips" and "DTrips", respectively, indicate the number of trips and the deficit of trips in the solution. The deficit indicates the number of trips that must be added to the schedule in order for the passenger demand to be met. Column labeled "NVS" indicates if the restrictions about vehicles simultane-

Table 7
Departure times at CP_1, manual solution for OSO3

5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
30	05	03	06	07	05	05	07	07	07	03	00	00	05	05	05	08	00
45	15	10	14	15	15	14	17	17	15	13	07	08	14	13	17	18	10
55	25	17	22	23	25	24	27	28	25	23	15	15	24	21	30	30	22
	35	23	33	30	35	31	37	38	35	32	23	22	32	30	40	40	35
	45	28	43	39	45	38	47	47	45	42	32	29	44	38	50	48	50
	55	33	51	46	55	47	57	57	55	52	45	36	55	48	58		
		38	59	55		56					53	43		56			
		45										50					
		52										58					
		59															

Table 8
Departure times at CP_1, algorithmic solution for OSO3

5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
30	05	06	06	12	08	12	05	06	10	10	06	05	08	05	10	37	08
53	14	09	12	20	16	26	15	15	20	20	12	12	16	13	20	55	18
	24	17	18	28	27	41	26	24	31	30	19	20	24	21	30		
	33	24	25	37	36	54	37	33	42	40	26	28	33	30	43		
	42	30	32	46	44		48	44	51	50	33	36	42	38	54		
	54	38	39	54	52			54			42	44	51	48			
		46	48								48	54		56			
		53	54								54						

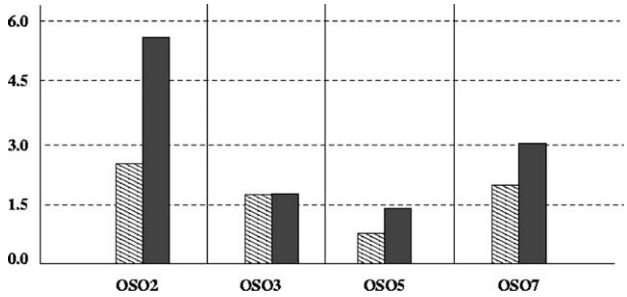

Fig. 9. Processing times.

Table 9
Characteristics of the manual solutions

Inst.	NIH	EDH	Trips	DTtrips	NVS	TNV	TNC
OSO1	12:54	12:44	412	00	Yes	40	80
OSO2	03:55	52:50	437	00	Yes	23	46
OSO3	03:45	18:25	232	00	Yes	15	30
OSO4	–	–	–	–	–	28	–
OSO5	–	–	–	–	–	25	–
OSO6	–	26:48	177	00	Yes	21	42
OSO7	16:23	25:28	235	00	Yes	26	52

Table 10
Characteristics of the algorithmic solutions

Inst.	NIH	EDH	Trips	DTtrips	NVS	TNV	TNC
OSO1	15:20	12:05	395	00	Yes	40	80
OSO2	10:07	24:23	351	06	Yes	23	46
OSO3	25:53	06:25	193	02	Yes	15	29
OSO4	44:54	02:09	291	00	Yes	28	56
OSO5	32:45	04:29	274	00	Yes	25	50
OSO6	27:03	07:31	133	02	Yes	20	39
OSO7	42:01	02:30	200	02	Yes	26	51

ously stationed at CPs were observed. Columns labeled "TNV" and "TNC" show the total number of vehicles and crew, respectively, in the corresponding solutions. A dash indicates unavailable data.

Positive points in favor of the automatic solutions are: (i) The number of extra duty hours is an important component in the operational costs for the companies. The algorithmic solutions showed a quite large reduction in the number of extra duty hours needed to operate the corresponding solutions. The decrease was by at least 20% and, in some cases, it reached 90%. (ii) In some cases, the algorithmic solutions used less crew. In one case, the algorithmic schedule even showed one less vehicle, a very difficult goal to reach. (iii) In general, the workloads in the algorithmic solutions are much better balanced among all the crew. (iv) With respect to the time needed to reach a solution, of course, the algorithmic method was orders of magnitude faster than the manual process. (v) The total number of trips, usually, is smaller in the algorithmic solutions. This is important when the public transportation authority pays a bonus for schedules that cover the demand with less trips, thereby improving the traffic and pollution marks. (vi) The maximum number of stationed vehicles was not violated in any of the algorithmic solutions constructed. In fact, in many cases, this number was lower than the corresponding number showed by the manual solutions. (vii) The duration of each trip was always faithfully respected. This was not the case in the manual solutions, were some trips had their durations artificially shortened, or had their departure times not respected. Such practice may lead to the companies being fined by the public transportation authority.

Some negative aspects of the algorithmic solutions are: (i) Sometimes, not all the necessary trips were scheduled, and some passengers would not be transported, if the maximum number of passengers in each trip was to be strictly observed. This may typically happen at the beginning or at the end of a day. This effect may be corrected by small modifications in the input data, attributing some extra duty minutes of working time to a small number of selected crew. Or, alternatively, by slightly increasing the number of passengers permitted in a trip. (ii) In some cases, the spacing between consecutive trips is not as uniform in the algorithmic solution as it is in the corresponding manual solutions. This effect is more acute in the borderline between consecutive hour intervals. (iii) The number of inactive hours was smaller in the manual solutions.

5. Conclusions

The article presents a suite of integrated algorithms and heuristics that can be used to construct adequate automatic solutions to the urban transportation problem. The algorithms were tested over real data, collected from urban transportation companies that operate in large metropolitan areas.

The problem in its entirety is difficult to be treated mathematically, since it involves a sizable set of conflicting restrictions that stem from operational conditions and labor agreements. In order to cope with such restrictions, mathematical programming models and heuristics were combined. This hybrid strategy was able to produce quite adequate solutions, in a fraction of the time that experts take to construct manual solutions. Also, the operational cost of the automatic solution showed considerable gains over the manual ones. The algorithm may fail to schedule a few trips at the beginning and at the end of a day and, in some extreme cases, the spacing between consecutive trips in the final solution may not be the best possible. These inconveniences could be corrected by small adjustments in the input parameters, although, in a few cases, at the expense of some additional minutes of extra working time, over the minimum required by the automatic solutions.

Even not being the focus of this work, it should be mentioned that a friendly user interface has also been implemented. It allows a non-expert user to input the data, activate the algorithms and examine the final solution. The interface also endows the user with the capacity of making local adjustments to the algorithmic solutions, in order to perfect the final schedule. Most of the figures shown in Section 4, such as Figs. 6–8, were generated

by the interface. The intention now is to submit both the algorithm and the interface to intensive field tests at selected companies.

References

[1] M.O. Ball, L.D. Bodin, R. Dial, A matching based heuristic for scheduling mass transit crews and vehicles, Transportation Science 17 (1983) 4–31.

[2] L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli, M. Spandoni, A heuristic procedure for the crew rostering problem, European Journal of Operational Research 58 (2) (1992) 272–283.

[3] J.Y. Blais, J.M. Rosseau, Overview of HASTUS current and future versions, in: J.R. Daduna, A. Wren (Eds.), Computer-Aided Transit Scheduling, Springer-Verlag, 1988, pp. 175–187.

[4] A. Caprara, M. Fischetti, P. Toth, D. Vigo, P.L. Guida, Algorithms for railway crew management, Mathematical Programming 79 (1–3) (1997) 125–141.

[5] A. Caprara, M. Fischetti, P. Toth, D. Vigo. Modeling and solving the crew rostering problem, Technical Report OR-95-6, DEIS, University of Bologna, Italy, 1995.

[6] P. Carraresi, L. Girardi, M. Nonato, Network models, Lagrangean relaxation and subgradients bundle approach in crew scheduling problems, in: I. Branco, J.R. Daduna, J.M.P. Paixao (Eds.), Proceedings of the Sixth International Workshop on Computer-Aided Transit Scheduling, 1995, pp. 188–212.

[7] M. Chamberlain, A. Wren, Developments and recent experience with the BUSMAN and BUSMAN II system, in: M. Desrochers, J.-M. Rousseau (Eds.), Computer-Aided Transit Scheduling, Springer-Verlag, 1992, pp. 1–16.

[8] J.R. Daduna, J.M.P. Paixão, Vehicle scheduling for public mass transit—an overview, Lecture Notes in Economics and Mathematical Systems 430 (July) (1993) 76–90.

[9] K. Darby-Dowman, J.K. Jachnik, R.L. Lewis, G. Mitra, Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience, in: A. Wren, J.R. Daduna (Eds.), Proceedings of the Fourth International Workshop on Computer-Aided Transit Scheduling, Springer-Verlag, Berlin, 1988, pp. 226–239.

[10] J.C. Falkner, D.M. Ryan, Express: Set partitioning for bus crew scheduling in Christchurch, in: M. Desrochers, J.-M. Rousseau (Eds.), Proceedings of the Fifth International Workshop on Computer-Aided Transit Scheduling, Springer-Verlag, Berlin, 1992, pp. 359–378.

[11] M.A.N.A. Filho, R.S.K. Kwan, A. Wren, Scheduling and their drivers in Brasil: Some pratical experience, in: Anais do VII Congresso de Pesquisa e Ensino em Transportes, 1994, pp. 231–242 (in Portuguese).

[12] S. Fores, L. Proll. Driver scheduling by integer linear programming—the tracs ii approach, Technical Report 98.01, University of Leeds, January 1998.

[13] R. Freling, D. Huisman, A.P.M. Wagelmans, Models and algorithms for integration of vehicle and crew scheduling, in: 8th International Conference of Computer-Aided Scheduling of Public Transport, 2000, pp. 441–460.

[14] R. Freling, A.P.M. Wagelmans, J.M.P. Paixão, An overview of models and techniques for integrating vehicle and crew scheduling, in: N.H.M. Wilson (Ed.), Computer-Aided Transit Scheduling, 1999, pp. 441–460.

[15] C. Friberg, K. Haase. An exact algorithm for the vehicle and crew scheduling problem, Technical Report 416, Universität Kiel, 1996, pp. 36–52.

[16] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, CA, 1979.

[17] K.L. Hoffman, M. Padberg, Solving airline crew-scheduling problems by branch-and-cut, Management Science 39 (6) (1993) 657–682.

[18] R.S.K. Kwan, M.A. Rahin, Bus scheduling with trip coordination and complex constrains, Lecture Notes in Economics and Mathematical Systems 430 (1995) 91–101.

[19] R.S.K. Kwan, M.A. Rahin, Object oriented bus vehicle scheduling—the boost system, in: Proceedings of the 7th International Workshop on Computer-Aided Scheduling of Public Transport, August 1997, pp. 36–52.

[20] B. Manington, A. Wren, A general computer method for bus crew scheduling, Preprints of the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, 1975.

[21] R.E. Marsten, F. Shepardson, Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications, Networks 11 (1981) 165–177.

[22] J.M.P. Paixão, I.M. Branco, A quasi-assignment algorithm for bus scheduling, Networks 17 (1987) 249–269.

[23] J.M.P. Paixão, I.M. Branco, Bus scheduling with a fixed number of vehicles, in: Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop, vol. 308, 1988, pp. 28–40.

[24] J.M.P. Paixão, A.P.M. Wagelmans, Models and algorithms for vehicle scheduling, Transportation Science 18 (1996) 1138–1162.

[25] I. Patrikalakis, D. Xerocostas, A new decomposition scheme of the urban public transport scheduling problem, in: M. Desrochers, J.-M. Rousseau (Eds.), Proceedings of the Fifth International Workshop: Computer-Aided Transit Scheduling, Springer-Verlag, Berlin, 1992, pp. 407–425.

[26] L. Sarah, L. Proll, A. Wren, A column generation approach to bus driver scheduling, in: Proceedings of the 4th Meeting of the EURO Working Group on Transportation, September 1996, pp. 36–52.

[27] D. Scott, A large linear programming approach to the public transport scheduling and cost model, in: J.M. Rousseau (Ed.), Computer Scheduling of Public Transport, Amsterdam, North Holland, vol. 2, 1985, pp. 473–491.

[28] B.M. Smith, A. Wren, VAMPIRES and TASC: Two successfully applied bus scheduling programs, Computer Scheduling of Public Transport (1981) 97–129.
[29] E. Tosini, C. Vercellis, An interactive system for extra-urban vehicle and crew scheduling problems, in: J.R. Daduna, A. Wren (Eds.), Proceedings of the Fourth International Workshop on Computer-Aided Transit Scheduling, 1988, pp. 41–53.
[30] E.B. Wilhelm, Overview of the RUCUS package driver run cutting program (RUNS), Preprints of the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, 1975.
[31] A. Wren, The development of Micro-BUSMAN: Scheduling on micro-computers, in: J.R. Daduna, A. Wren (Eds.), Computer-Aided Transit Scheduling, Springer-Verlag, Berlin, 1988, pp. 160–174.
[32] A. Wren, Heuristics ancient and modern: Transport scheduling through the ages, Journal of Heuristics 4 (1998) 87–100.
[33] A. Wren, N.D.F. Gualda, Integrated scheduling of buses and drivers, in: Seventh International Workshop on Computer-Aided Scheduling of Public Transport, 1997, pp. 53–75.
[34] A. Wren, J. Rousseau, Bus driver scheduling—an overview, in: J.M.P. Paixão, J.R. Daduna, I. Branco (Eds.), Computer-Aided Transit Scheduling, 1995, pp. 173–187.
[35] T.H. Yunes, A.V. Moura, C.C. Souza. Solving large scale crew scheduling problems with constraint programming and integer programming, Technical Report IC 99-19, Institute of Computing, University of Campinas, 1999.
[36] T.H. Yunes, C.C. Souza, A.V. Moura, Modeling and solving a crew rostering problem with constraint logic programming and integer programming, Technical Report IC-00-04, Institute of Computing, University of Campinas, 2000. Available from: <http://goa.pos.dcc.unicamp.br/otimo>.