Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 35 (2008) 1562 – 1575

computers & operations research

www.elsevier.com/locate/cor

Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem

Marta Mesquita[a,c], Ana Paias[b,c,*]

[a] *Dep. Matemática,Tapada da Ajuda, Instituto Superior de Agronomia, Universidade Técnica de Lisboa, 1349-017 Lisboa, Portugal*
[b] *Universidade de Lisboa, Faculdade de Ciências, DEIO, Bloco C6, piso 4, Cidade Universitária, 1749-016 Lisboa, Portugal*
[c] *Centro de Investigação Operacional da FCUL, Portugal*

Available online 20 October 2006

Abstract

In the integrated vehicle and crew scheduling problem (VCSP) one has to simultaneously assign vehicles to timetabled trips and drivers to vehicles. In this paper, the VCSP is described by an integer linear programming formulation combining a multicommodity network flow model with a set partitioning/covering model.

We propose an algorithm that starts with a pre-processing phase to define the set of tasks and to obtain an initial set of duties. In a second phase, we solve the linear programming relaxation of the models using a column generation scheme. Whenever the resulting solution is not integer, branch-and-bound techniques are used over the set of feasible crew duties generated while solving the linear programming relaxation in order to obtain a feasible solution for the VCSP. We show that, under some conditions, just one subset of the decision variables is required to be integer.

Computational experience, with randomly generated data publicly available for benchmarking on the WEB is reported, and the results show the effectiveness of our approach concerning both, the quality of the solutions and the CPU time needed to obtain them. Moreover, the approach can be applied to large instances.

Keywords: Integer linear programming; Vehicle scheduling; Crew scheduling

1. Introduction

Vehicle and crew scheduling problems are important combinatorial optimization problems that arise in the planning process of mass transit companies. Traditionally, due to their complex nature, these problems are solved separately and sequentially. The strong dependency between these two problems suggests that an integrated approach may lead to important cost reductions. In fact, the vehicle blocks characteristics influence the resulting crew duties and the optimal crew duty set may lead to changes on the original vehicle blocks.

In the integrated vehicle and crew scheduling problem (VCSP) one has to simultaneously assign vehicles to timetabled trips as well as drivers to vehicles.

Some mathematical formulations for the VCSP have been proposed in the literature. For the single depot case, Freling et al. [1,2] presented an integer model that combines a quasi-assignment formulation for vehicle scheduling and a set partitioning formulation for the crew scheduling. They proposed an approach based on a column generation

* Corresponding author. Universidade de Lisboa, Faculdade de Ciências, DEIO, Bloco C6, piso 4, Cidade Universitária, 1749-016 Lisboa, Portugal. Tel.: +351 21 7500408; fax: +351 21 750 00 81.

E-mail addresses: martaoliv@isa.utl.pt (M. Mesquita), ampaias@fc.ul.pt (A. Paias).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.09.001

scheme where the master problem is solved by a Lagrangean relaxation. Extensions to the multi-depot case have been developed, since then, by Huisman et al. [3] and De Groot and Huisman [4]. Borndörfer et al. [5] have used bundle techniques to solve the Lagrangean relaxation proposed by Huisman et al. [3]. Recently, Hollis et al. [6] presented a set covering mathematical formulation to describe the multi-depot VCSP faced by the Australia Post Mail Distribution. They proposed a heuristic algorithm based on column generation.

Two approaches have been proposed for the multi-depot case in a special context, where each vehicle block corresponds to a feasible crew duty. Valouxis and Housos [7] proposed and tested a heuristic to instances arising from several bus companies in Greece and Gaffi and Nonato [8] proposed a solution method based on the Lagrangean relaxation of a network flow model.

For the single depot case, exact methods have been investigated by Haase and Friberg [9] and Haase et al. [10]. The mathematical model proposed in [10] involves set partitioning variables to describe the crew scheduling problem as well as a buses counter variable. The resulting set partitioning formulation includes side constraints on the number of buses in use to guarantee that a feasible vehicle schedule can be obtained afterwards in polynomial time. An optimal solution is obtained by a column generation method embedded in a branch-and-bound procedure.

In this paper, we propose two mathematical formulations for the VCSP. The first one is similar to the one proposed by Huisman et al. [3], but with fewer decision variables and fewer constraints. It combines a multicommodity network flow model for the vehicle scheduling with a set partitioning model for the crew scheduling. For the second one, a subset of the set partitioning constraints is replaced by set covering constraints in order to make the model more flexible.

For real-life applications, the set of feasible crew duties becomes too large to be considered explicitly and so we solve the corresponding linear programming relaxation, exactly or approximately, using a column generation scheme. The columns corresponding to the duties can be seen as paths in an adequate network and are generated as needed by solving shortest path problems with resource constraints. We developed a pre-processing phase, based on the optimal solution of the vehicle scheduling problem without requiring that vehicles return to the source depot. This was in order to construct the set of tasks and the initial set of crew duties.

Whenever the optimal linear programming solution is not integer we obtain a solution for the integer problem using branch-and-bound techniques over a subset of feasible duties for the crews. We show that such solution can be obtained by branching only on a subset of the decision variable set.

Computational experience, with randomly generated data publicly available for benchmarking on the Internet, is reported and shows the effectiveness of the proposed approach for the VCSP.

2. The integrated VCSP

In this section some basic notation and some assumptions for the VCSP are given. Let $T_1, \ldots, T_n$ be a set of timetabled trips to be operated in a planning interval T by vehicles located at depots $D_1, \ldots, D_k$. The vehicles, in number of v_d at each depot D_d, $d = 1, \ldots, k$, are supposed to be identical. Each trip is characterized by a starting time and location and an ending time and location. An ordered pair of trips (T_i, T_j) is said to be compatible if the bus released after the completion of trip T_i can be assigned to trip T_j. The ordered pair (T_i, T_j) defines a deadhead trip, where a bus runs without passengers, between the end location of T_i and the start location of T_j. Deadhead trips which correspond to the movement of a bus from a depot to the start location of a trip are denoted by pull-out trips while deadhead trips corresponding to the movement of a bus from an end location of a trip to a depot are denoted by pull-in trips. Each vehicle returns to the source depot at the end of its duty.

If a vehicle belonging to depot D_d performs the deadhead trip (T_i, T_j) then a cost c_{ij}^d should be paid. Vehicle costs usually take into account travel costs and eventual penalties imposed by the company, for example on idle times. If we also want to minimize the number of vehicles in the schedule then a penalty associated with the use of each new vehicle should be added to the travel costs associated with pull-in trips and pull-out trips.

Concerning crews, a fixed cost is assigned to each one. Each end location of a trip is a potential relief point, where it is allowed to change drivers. Therefore, each task corresponds to a deadhead trip followed by a trip and the crew duties can start (end) at a depot or at an end location of a trip $i \in N$. A crew duty is a combination of pieces of work that respects several constraints such as maximum and minimum spread; maximum working time without a break; break duration; maximum number of changeovers.

Let $N = \{1, \ldots, n\}$ be the set of vertices, where vertex $i \in N$ represents trip T_i and let D denote the set of k depots. For each depot d we consider a vertex $n + d$, $d \in D$. For each $d \in D$ we associate a graph $G^d = (V^d, A^d)$, where

$V^d = N \cup \{n+d\}$ and $A^d = I \cup \{(n+d) \times N\} \cup \{N \times (d+n)\}$. The arc set A^d contains arcs corresponding to compatible pairs of trips, $I \subseteq N \times N$, and arcs related with pull-in trips from depot *d* and pull-out trips from depot *d*. Costs c_{ij}^d, $c_{i,n+d}$ and $c_{n+d,j}$ are associated with the corresponding arcs.

If trips $i \in N$ are ordered by increasing value of their starting time then the arc set *I* contains only arcs (i, j) with $i < j$ and no circuit containing only vertices $i, j \in N$ exists in graph G^d for any *d*.

Let *L* represent the set of feasible duties.

Consider the decision variables,

$$x_{ij}^d = \begin{cases} 1 & \text{if trip } j \text{ follows directly after trip } i \text{ by a vehicle from depot } d, \\ 0 & \text{otherwise,} \end{cases} \quad (i,j) \in I, \ \ d \in D,$$

$$x_{i,n+d} = \begin{cases} 1 & \text{if the bus returns to depot } d \text{ after trip } i, \\ 0 & \text{otherwise,} \end{cases} \quad i \in N, \ \ d \in D,$$

$$x_{n+d,i} = \begin{cases} 1 & \text{if depot } d \text{ directly supplies a bus for trip } i, \\ 0 & \text{otherwise,} \end{cases} \quad i \in N, \ \ d \in D,$$

to describe the vehicle scheduling problem, and the decision variables

$$y_\ell = \begin{cases} 1 & \text{if crew duty } \ell \text{ is in the optimal solution,} \\ 0 & \text{otherwise,} \end{cases} \quad \ell \in L,$$

to represent the crew scheduling problem. Let s_ℓ be the cost associated to dutyℓ and let $L(i, j)$ denote the set of duties covering the deadhead trip from trip *i* to trip *j* and covering trip *j*. Let $DL(j)$ represent the set of duties covering the deadhead trip from any depot to trip *j* and covering trip *j* and let $LD(i)$ denote the set of duties covering the deadhead trip from trip *i* to any depot.

Then, the VCSP can be formulated as follows:

SP-VCSP.

$$\min \quad \sum_{d \in D} \sum_{(i,j) \in I} c_{ij}^d x_{ij}^d + \sum_{d \in D} \sum_{i \in N} (c_{i,n+d} x_{i,n+d} + c_{n+d,i} x_{n+d,i}) + \sum_{\ell \in L} s_\ell y_\ell \tag{2.1}$$

$$\text{s.t.} \quad \sum_{d \in D} \sum_{j:(i,j) \in I} x_{ij}^d + \sum_{d \in D} x_{i,n+d} = 1 \quad \forall i \in N, \tag{2.2}$$

$$\sum_{j:(i,j) \in I} x_{ij}^d + x_{i,n+d} - \sum_{j:(j,i) \in I} x_{ji}^d - x_{n+d,i} = 0 \quad \forall i \in N, \ \ \forall d \in D, \tag{2.3}$$

$$\sum_{i \in N} x_{n+d,i} \leqslant v_d \quad \forall d \in D, \tag{2.4}$$

$$\sum_{\ell \in DL(j)} y_\ell - \sum_{d \in D} x_{n+d,j} = 0 \quad \forall j \in N, \tag{2.5}$$

$$\sum_{\ell \in L(i,j)} y_\ell - \sum_{d \in D} x_{ij}^d = 0 \quad \forall (i,j) \in I, \tag{2.6}$$

$$\sum_{\ell \in LD(i)} y_\ell - \sum_{d \in D} x_{i,n+d} = 0 \quad \forall i \in N, \tag{2.7}$$

$$x_{ij}^d \in \{0,1\} \quad \forall (i,j) \in I, \ \ \forall d \in D, \tag{2.8}$$

$$x_{i,n+d}, x_{n+d,i} \in \{0,1\} \quad \forall i \in N, \ \ \forall d \in D, \tag{2.9}$$

$$y_\ell \in \{0,1\} \quad \forall \ell \in L. \tag{2.10}$$

Objective (2.1) is to minimize a linear combination of vehicle and crew costs.

Constraints (2.2) state that each timetabled trip is performed exactly once. The flow conservation constraints (2.3) guarantee that each vehicle block is assigned to a vehicle that returns to the source depot. Constraints (2.4) are depot capacity constraints. The constraint set $\{(2.5)$–$(2.7)\}$ relates the vehicle scheduling variables with the crew variables and requires that a deadhead trip is covered by a vehicle if and only if it is covered by a crew. According to (2.5) and

(2.6) the crew depot is established by the vehicle depot. Although {(2.5)–(2.7)} are stated just for arcs corresponding to deadhead trips, they are sufficient to guarantee that all deadhead trips and all timetabled trips in a vehicle block are covered by crew duties. This follows from the definition of tasks given before where the deadhead trip, from the end of a trip i (or from the depot) to the start of trip j, and the trip j are performed by the same driver. As a result, SP-VCSP has less constraints than the integer formulation proposed by Huisman et al. [3], where {(2.5)–(2.7)} is stated for each deadhead trip and for each trip $i \in N$.

The mathematical model SP-VCSP includes two types of binary decision variables, one type related to the vehicle schedules and the other type associated to the crew duties. We will show next, that under some assumptions, SP-VCSP reduces to a mixed integer model where just one type of the decision variables is required to be integer.

Definition 2.1. Let SP-VCSP$_{x\geqslant 0}$ be the problem obtained from SP-VCSP by replacing (2.8) and (2.9) by

$$x_{ij}^d \geqslant 0 \quad \forall (i,j) \in I, \ \ \forall d \in D, \tag{2.11}$$

$$x_{i,n+d}, x_{n+d,i} \geqslant 0 \quad \forall i \in N, \ \ \forall d \in D \tag{2.12}$$

and let SP-VCSP$_{y\geqslant 0}$ be the problem obtained from SP-VCSP by replacing (2.10) by

$$y_\ell \geqslant 0 \quad \forall \ell \in L. \tag{2.13}$$

Lemma 2.1. *The mathematical formulation SP-VCSP$_{x\geqslant 0}$ has an integer optimal solution.*

Proof. We first show that the integer decision variables y_ℓ, such that $y_\ell = 1$, define a set of disjoint paths covering all vertices $i \in N$. Each path corresponds to a vehicle block. In order to obtain a feasible solution to SP-VCSP, it suffices to assign each path to a depot.

Let (x, y) be a feasible solution to SP-VCSP$_{x\geqslant 0}$ and consider the following sets:

$$L_1 = \{\ell \in L : y_\ell = 1\}, \quad IDL_1 = \{i : \exists d \in D, DL(i) \cap L_1 \neq \emptyset\}, \quad IL_1 = \{(i,j) \in I : L(i,j) \cap L_1 \neq \emptyset\}$$

and

$$IL_1D = \{i : \exists d \in D, LD(i) \cap L_1 \neq \emptyset\}.$$

Constraints (2.3) can be written as

$$x_{n+d,i} = \sum_{j:(i,j)\in I} x_{ij}^d + x_{i,n+d} - \sum_{j:(j,i)\in I} x_{ji}^d \quad \forall i \in N, \ \ \forall d \in D.$$

For each $i_1 \in IDL_1$, constraints (2.5) guarantee that we have $\sum_{d\in D} x_{n+d,i_1} = 1 \ \forall i_1 \in IDL_1, \forall d \in D$. Consequently, we have $\sum_{j:(j,i_1)\in I} x_{ji_1}^d = 0 \ \forall i_1 \in IDL_1, \forall d \in D$ and, from (2.3), we obtain

$$x_{n+d,i_1} = \sum_{j:(i_1,j)\in I} x_{i_1 j}^d + x_{i_1,n+d} \quad \forall i_1 \in IDL_1, \ \ \forall d \in D.$$

Two cases may, now, occur:

(i) If $i_1 \in IL_1D$ then, by (2.7), we obtain $\sum_{d\in D} x_{i_1,n+d} = 1$, $\sum_{j:(i_1,j)\in I} x_{i_1 j}^d = 0$ and $\nexists j : (i_1, j) \in IL_1$. Thus for $\forall i_1 \in IDL_1 \cap IL_1D$, (2.3) becomes $x_{n+d,i_1} = x_{i_1,n+d} \ \forall d \in D$ (Fig. 1).

(ii) If $i_1 \in IDL_1 \wedge i_1 \notin IL_1D$, there is a crew driving the vehicle from i_1 to another trip $i_2 \in N$. By (2.2) and (2.6), there is only one arc $(i_1, i_2) \in IL_1$. In this case, (2.3) can be written as $x_{n+d,i_1} = x_{i_1,i_2}^d$. Now, consider the vertex i_2, after i_2 the vehicle returns to the depot or performs another trip i_3. That is for vertex i_2, as for vertex i_1, two cases may occur: (i) and (ii). Given that, there are only arcs (i, j) with $i < j$, we have paths starting at a depot $d \in D$, passing through vertices $i_1, i_2, \ldots, i_p$ and ending at the same depot d, such that $\sum_{d\in D} x_{i_h,i_{h+1}}^d = 1, h = 1, \ldots, p-1$ (Fig. 2).

Then, $\forall i_1 \in IDL_1, \exists d \in D$ we have $x_{n+d,i_1} = x_{i_1,i_2}^d = \cdots = x_{i_p,n+d}$.

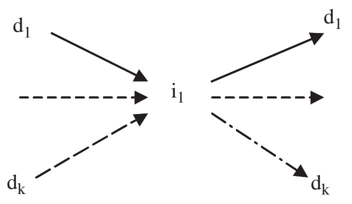

Fig. 1. Case (i).

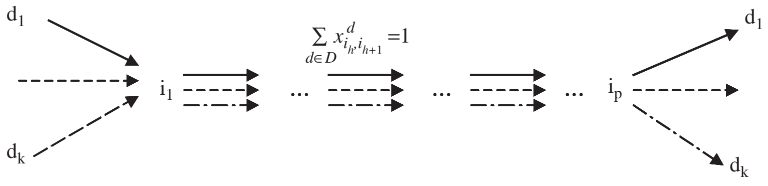

Fig. 2. Case (ii).

Let P be the set of all paths $P^d_{i_1}$, $\forall i_1 \in IDL_1, \forall d \in D$. For each $i \notin IDL_1$, by (2.2) and (2.3), $\exists i_1 \in IDL_1, \exists d \in D : i \in P^d_{i_1}$ and, consequently, P covers all vertices $i \in N$. Define the cost of each path $P^d_{i_1}$, $i_1 \in IDL_1$, $d \in D$, as $C^d_{P_{i_1}} = c_{n+d,i_1} + \sum_{h=1}^{p-1} c^d_{i_h,i_{h+1}} + c_{i_p,n+d}$, which reduces to $C^d_{P_{i_1}} = c_{n+d,i_1} + c_{i_p,n+d}$ in case (i).

Next, we replace, in (2.1)–(2.10), all the nonzero $x^d_{i_h,i_{h+1}}$ and all the nonzero $x_{i_p,n+d}$ by x_{n+d,i_1}. Then, the SP-VCSP$_{x\geqslant 0}$ can be written as follows:

$$
\begin{aligned}
\sum_{\ell\in L_1} s_\ell y_\ell + \min \quad & \sum_{d\in D}\sum_{i\in IDL_1} C^d_{P_i} x_{n+d,i} \\
\text{s.t.} \quad & \sum_{d\in D} x_{n+d,i} = 1, \quad i \in IDL_1, \\
& \sum_{i\in IDL_1} x_{n+d,i} \leqslant v_d, \quad d \in D, \\
& x_{n+d,i} \geqslant 0, \quad i \in IDL_1, \quad d \in D.
\end{aligned}
$$

The previous formulation corresponds to a transportation problem and thus, problem SP-VCSP$_{x\geqslant 0}$, has an integer optimal solution since the parameters v_d, $d \in D$, are integer. □

Lemma 2.1 can be viewed as a generalization for the multiple depot case, of a result obtained by Hasse et al. [9] for the VCSP with one depot.

Next, we prove a similar result where only the decision variables associated to the vehicles are required to be integer.

Lemma 2.2. *If each feasible duty corresponds to a sequence of trips in a vehicle block, then the mathematical formulation SP-VCSP$_{y\geqslant 0}$ has an integer optimal solution.*

Proof. Fixing the vehicle scheduling variables x to feasible 0–1 values, constraints (2.5)–(2.7), associated with nonzero x values, can be viewed as set partitioning constraints. The corresponding constraint matrix, A_y can be ordered in such a way that the nonzero entries in each column appear consecutively. It suffices to consider a vehicle block at a time and give to the rows of A_y the order of the corresponding arcs in that vehicle block (see Figs. 3 and 4).

Therefore, A_y is an interval matrix and, consequently, A_y is totally unimodular. □

As a conclusion, Lemmas 2.1 and 2.2 tell us that SP-VCSP can be used in a branch-and-bound scheme where branching can be made only on a subset of the decision variables.

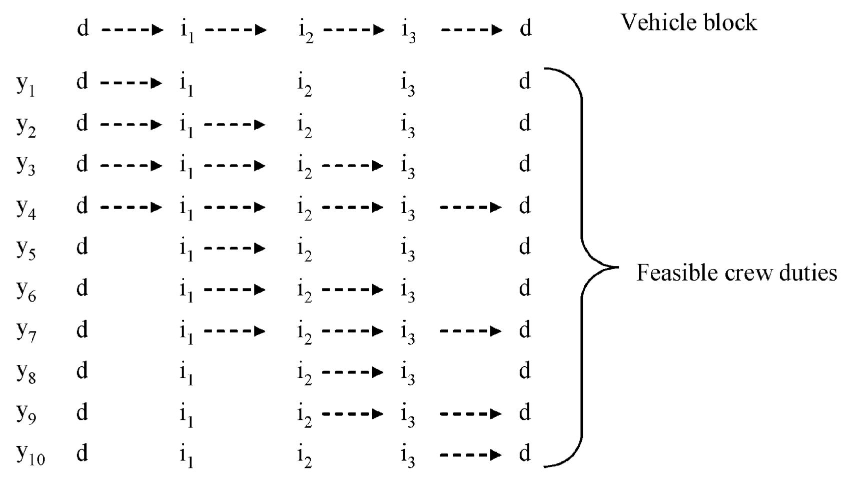

Fig. 3. Feasible crew duties for the SP-VCSP.

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	
1	1	1	1							(d,i_1)
	1	1	1	1	1	1				(i_1,i_2)
		1	1		1	1	1	1		(i_2,i_3)
			1			1		1	1	(i_3,d)

Fig. 4. Constraint matrix.

3. Solution approach

The ILP formulation for the VCSP, derived in the previous section, involves a huge number of variables (vehicle scheduling variables and crew scheduling variables) and it is inefficient to handle these variables explicitly. In this paper, we propose a four-step solution approach to solve the integrated problem. First, a pre-processing procedure is executed to define the set of tasks. Second, we construct an initial set of duties. These two steps are based on the solution of a vehicle scheduling problem without requiring that each vehicle return to the source depot. Third, a near-optimal solution to the linear programming relaxation is obtained by implicit column generation. At each iteration, a set of columns with negative reduced cost is added to the master problem. Finally, if the optimal solution of the linear problem is not integer, then a feasible solution is obtained by branch and bound over the set of duties generated while solving the linear relaxation.

In the next section it is explained how to obtain the set of tasks and an initial set of duties. In Section 3.2, the column generator is described and it is also shown how to use it in order to generate negative reduced cost duties. In Section 3.3 a suboptimal pricing procedure is proposed. In Section 3.4 a mixed partitioning/covering formulation is presented. The main contribution of this new formulation is to allow changeovers to be handled by the master problem instead of being handled by the pricing problem. Finally, in the last section, the branching procedure is discussed.

3.1. Definition of the tasks and generating the initial set of crew duties

In Section 2.1 we stated that each end location of a timetabled trip is a potential relief point. In this case, each task corresponds to a deadhead trip followed by a timetabled trip and the number of tasks is equal to the number of timetabled trips. Usually a task will cover one or few timetabled trips and the set of tasks does not coincide with the set of timetabled trips. Due to their starting and ending time and place, some pairs of trips are expected to be covered, in the optimal solution, by the same vehicle and the same crew and can be included in the same task. We developed a pre-processing procedure based on the solution of a multi-depot vehicle scheduling problem in order to identify such pairs of trips. It is important to include some characteristics of the crew scheduling problem into the vehicle scheduling problem. The easiest way to do this is to include crew features into the objective function of the vehicle scheduling problem. We can assign the corresponding idle time to the cost of each deadhead arc. That is, the cost of a deadhead arc (i, j) is equal to the starting time of trip j minus the ending time of trip i, which corresponds to the time a crew runs without passengers. Besides, in order to minimize the number of vehicles, a big cost is assigned to each new vehicle in the schedule. Thus, the objective is to minimize the number of vehicles in the schedule by linking timetabled trips i and j that have a short idle time between them.

The pre-processing procedure is based on the set of vehicle blocks given by the optimal solution of the multi-depot vehicle scheduling without requiring that each vehicle return to the source depot. This problem can be viewed as an assignment problem or a minimum cost flow problem and solved in polynomial time. The corresponding optimal solution is a set of vehicle blocks covering all timetabled trips. We analyze each vehicle block and merge two consecutive trips whenever the corresponding deadhead cost (idle time) is less then a threshold value and the resulting task does not violate duty constraints. Note that, in a large number of situations where two trips are merged, the ending location of the first trip is the same as the start location of the second one.

In order to obtain the dual variables necessary to start the pricing of the duties in the column generator, we need to obtain an initial set of crew duties covering all tasks. Once more we can do it based on the optimal solution for the multi-depot vehicle scheduling problem without requiring that each vehicle return to the source depot. An efficient way to heuristically build the initial set of duties is using the set of tasks as input for the multi-depot vehicle scheduling problem. Thus, the corresponding optimal solution defines a set of vehicle blocks covering all tasks once. Then, we have to cut each vehicle block into pieces of work trying to satisfy the constraints for the duties. First, one duty is assigned to each vehicle block that satisfies duty constraints. Second, the remaining vehicle blocks are split into pieces of work satisfying only constraints related to maximum spread and maximum working time with or without a break. All duties are checked for the other constraints. A large cost is assigned to nonfeasible duties.

Alternatively, the set of vehicle blocks covering all the timetabled trips or covering all tasks could be obtained by solving a single depot problem where all the depots are replaced by a single artificial depot. Comparing both alternatives we think that the different locations of the depots are better pondered by the optimal solution of the multi-depot problem.

3.2. Column generator

The column generator presented in this section has some similarities with the one proposed by Desrochers and Soumis [11] for solving the linear relaxation of a set covering model for the crew scheduling problem. In fact, each feasible duty can be seen as an adequate path in a network where the feasibility is established by using resources that are consumed along the network or by the definition of the arcs in the network. The resource consumption is restricted by imposing time windows on the vertices of the network. However, the network associated to the column generator proposed in this paper is different from the one proposed by Desrochers and Soumis [11], and is defined next.

The set of vertices consists of a sink node s (a fictitious depot from where the duties for the crews start), two nodes related to the end of the duties, namely vertices t_1 and t_2 and $2n$ vertices related to the trips. In fact, with each trip i we associate two vertices, $2i - 1$ and $2i$.

Several types of arcs are defined as follows:

$(s, 2i - 1)$ sign-on arc representing the start of a duty where the driver picks up a vehicle from the depot and drives it to and along trip i;
$(s, 2i)$ sign-on arc representing the start of a duty where the driver picks up the vehicle at the end location of trip i;
$(2i - 1, t_1)$ sign-off arc corresponding to the end of a duty where the driver takes the vehicle back to the depot after trip i;
$(2i - 1, t_2)$ sign-off arc corresponding to the end of a duty where the driver, after driven the vehicle along trip i, goes back to the depot walking;

$(2i-1, 2i)$ auxiliary arc related to trip i;
$(2i, 2j-1)$ there is a connection arc if it is possible to combine in sequence and in the same duty trip i and trip j;
$(2i, 2j)$ changeover arc. After performing the trip i, the driver leaves the vehicle and walks to the end place of trip j to pick the vehicle that has finished trip j;
(t_1, t_2) auxiliary arc connecting end depots.

For instance, the path $s \rightarrow 2k-1 \rightarrow 2k \rightarrow 2h-1 \rightarrow t_1 \rightarrow t_2$, represents the duty where the driver picks a vehicle from the depot, drives it to and along trip k, performs the deadhead trip between the end location of trip k and the start location of trip h, performs trip h and then drives the vehicle back to the depot. On the other hand, the path $s \rightarrow 2k-1 \rightarrow 2k \rightarrow 2h-1 \rightarrow t_2$ corresponds to a similar duty where the driver returns to the depot walking, after driving the vehicle along trip h, while $s \rightarrow 2k \rightarrow 2h-1 \rightarrow t_2$ indicates that the driver starts his duty by picking the vehicle at the end of trip k. The path $s \rightarrow 2i-1 \rightarrow 2i \rightarrow 2k \rightarrow 2h-1 \rightarrow t_2$, represents the duty where the driver picks up a vehicle from the depot, drives it to and along trip i, and leaves it to change to another vehicle. Then, he drives the new vehicle along the deadhead trip between the end location of trip k and the start location of trip h, performs trip h and returns to the depot walking.

The number of arcs of each type and the type of arcs depend on the particular situation or the company being analyzed. On one hand, we can define arcs $(s, 2i)$ for all trips i, $\forall i \in N$. On the other hand, if there are constraints on the locations where a duty can start, the arcs $(s, 2i)$, are not defined for trips i such that the end location is not feasible for a duty to start. Concerning the arcs $(2i-1, t_2)$, $(s, 2i-1)$ and $(2i-1, t_1)$ similar remarks can be done. Connection arcs, $(2i, 2j-1)$, are defined if trip i and trip j are compatible trips.

Breaks can occur during connection arcs, $(2i, 2j-1)$, whenever the waiting time between the arrival at the starting location of trip j and the starting time for trip j is enough for the driver to take his meal. A break can also occur during a “changeover” arc, $(2i, 2j)$. After taking his meal the driver picks the same or another vehicle.

The number of changeover arcs, $(2i, 2j)$ depends on the conditions defined for the changeover to occur. Such conditions are usually space-related conditions and/or time-related conditions. Usually the number of changeovers is restricted due to insurance reason (for instances in the case of Transportation Companies that use rented vehicles, a higher number of drivers for the same vehicle will imply an increase on the value of the insurance policy), operational reasons, etc.

Changeovers may lead to the inclusion of new arcs in the duty network since even if two trips are incompatible a changeover may occur between them. Consider for instances trip i starting and ending, respectively, at instants 5 and 20 and consider trip j starting and ending at instants 8 and 15, respectively. Obviously both trips cannot be performed by the same vehicle, as they are incompatible. However, if trip i and trip j end at the same place, then the driver that drives the vehicle along trip j can change to the vehicle that performs trip i and drives it after trip i is finished.

Depending on the rules that define the feasibility for the duties, several resources can be defined. For each arc the cost and resource consumption is known. The feasibility of each duty, a path on the network, is established constraining, along that path, the consumption of the resources. For instances, to control the spread of a duty we can define the resource duration. In fact, if the spread of a duty is required to be between min-spread and max-spread, a time window is defined for vertex t_2, [min-spread, max-spread], and the total consumption of the resource duration for a feasible path from s to t_2 must be in that time window.

To control breaks we can define a different resource, the ‘break’ resource. The consumption of this resource is zero for all arcs except the ones corresponding to potential breaks.

Usually, the cost of each duty is given by a fixed cost plus an operational cost. As duties are path in the network described above, the cost of a duty is equal to the cost of the corresponding path which is the sum of the costs of the arcs belonging to that path. The fixed cost is assigned to the arcs $(s, 2i-1)$ and $(s, 2i)$. The operational costs are assigned in accordance of what they depend on.

The reduced cost of a duty is the sum of the reduced cost of the arcs in the path associated to that duty. For the duty network described above, see Fig. 5, the reduced costs for the arcs are listed in the following table, where $\alpha_{d,i}$, $\alpha_{2i,2j-1}$ and $\alpha_{i,d}$ are, respectively, the dual variables for constraints (2.5)–(2.7) (Table 1).

For the remaining five types of arcs, respectively, $(s, 2i)$, $(2i-1, 2i)$, $(2i-1, t_1)$, $(2i, 2j)$ and (t_1, t_2) the reduced cost is equal to the original cost.

Obtaining the feasible path with least negative reduced cost is equivalent to find the minimum cost path with resource constraints in the duty network and using the reduced costs of the arcs as arc costs. This can be done using dynamic programming in the same way as it is described in [11]. The network is acyclic and the vertices can be ordered in such

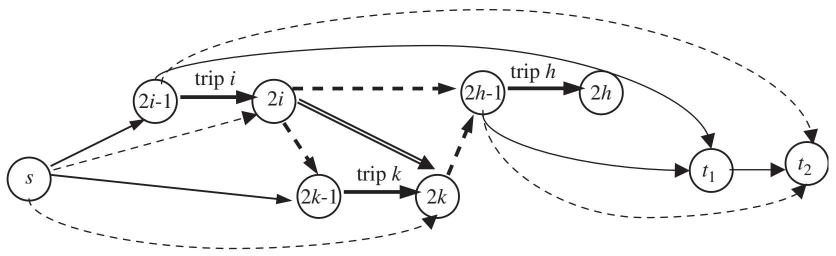

Fig. 5. Example of partial duty network.

Table 1
Reduced costs

Arc	Cost	Reduced cost
$(s, 2i-1)$	$C_{s,2i-1}$	$C_{s,2i-1} - \alpha_{d,i}$
$(2i, 2j-1)$	$C_{2i,2j-1}$	$C_{2i,2j-1} - \alpha_{2i,2j-1}$
$(2i-1, t_1)$	$C_{2i-1,d}$	$C_{2i-1,d} - \alpha_{i,d}$

a way that the dynamic equation is solved straightforwardly. At the final stage of the dynamic program we obtain the states corresponding to all feasible paths and consequently we can, at once, generate several columns with negative reduced cost.

3.3. *Suboptimal pricing*

The complexity of the column generator depends on two issues: the number of feasible paths in the duty network and the type of rules defining the feasibility of the duties. For the same rules, when the number of timetabled trips increases, the number of compatible trips also increases exponentially, leading to an exponential increase to the number of states in each phase of the dynamic program. Therefore, for larger instances, the pricing problem becomes difficult to solve. In such cases, the pricing problem can be solved approximately. That is, we can remove some states that we anticipate might not lead to the optimal final state. Removing these states can be done in the earlier iterations of the column generation scheme, when there are too many columns with negative reduced cost, or in the overall process. Note that, whenever the pricing problem is not solved to optimality in the last iteration, the solution obtained for the corresponding linear problem might not be the optimal solution for the linear relaxation. However an integer solution obtained by branch-and-bound techniques over the subset of the generated columns is still a feasible solution for the VCSP.

The difficulty of such procedure is to decide which states should be discarded and which states should be kept. The decision can be taken using pre-defined rules or can be taken randomly. In our method we have combined both decisions. A state is discarded if its cost is greater than a threshold value. In fact if the cost of a state is too large we do not expect that this state will lead to a path with negative reduced cost and so we discard it. Usually, this rule only discards a few states and in early iterations. So, we are going to combine it with a random-based rule. For this second rule, we pre-define two parameters, namely α and β, such that, α/β is the probability that a state is discarded at each stage of the dynamic program used to solve the pricing problem. Then, at each stage, we randomly generate an integer r in $[1, \beta]$. If $r \leqslant \alpha$, the state is discarded. The first reason for developing such procedure is that in the earlier iterations of the column generation scheme, there are too many paths with negative reduced costs and only a small percentage of them will be included in the linear model at this iteration. The second reason is that the random loss of some states, in one iteration of the dynamic programming, prevents to generate some duties but at the same time it allows to generate alternative ones leading to a greater diversity of columns. The last reason is that if an important state is discarded, we expect it to be generated in future iterations and we do not expect it to be discarded in all of the iterations.

In conclusion, we hope that working with a smaller but varied set of duties will maintain the quality of the linear programming relaxation bound reducing the corresponding CPU time.

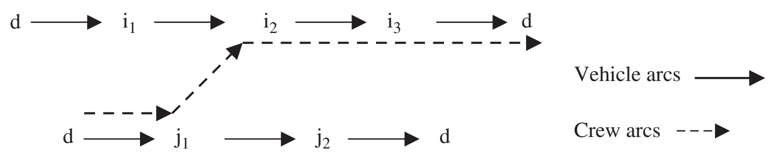

Fig. 6. Example of changeover allowed by the covering model.

3.4. *Mixed partitioning/covering model*

When changeovers are allowed, new arcs and a new resource are included in the duty network. The inclusion of new arcs leads to an exponential increase on the number of states in each phase of the dynamic program. Therefore, the linear programming relaxation becomes difficult to solve. To overcome these disadvantages we propose a slightly different integer formulation from the SP-VCSP to model the VCSP.

Some authors formulate the crew scheduling problem as a set covering model, see for instance [11]. They noticed that the corresponding optimal solution has little over-covering. From a practical point of view, it is not desirable to over-cover deadhead or timetabled trips. Usually assigning just one driver to a piece of work is cheaper than assigning multiple drivers and consequently over-covers only occur whenever they lead to a cheaper solution.

In this section we propose a mixed set partitioning/covering model to describe the VCSP. The set I of arcs (i, j) corresponding to compatible pairs of trips is partitioned into two subsets, I_c and $I\backslash I_c$. The arcs in I_c may be covered by multiple drivers while the arcs in set $I\backslash I_c$may be covered by, at most, a single driver.

Equalities (2.6) of model (SP-VCSP) are going to be split into two sets:

$$\sum_{\ell\in L(i,j)} y_\ell - \sum_{d\in D} x_{ij}^d = 0 \quad \forall (i, j) \in I\backslash I_c, \tag{2.6.1}$$

$$\sum_{\ell\in L(i,j)} y_\ell - \sum_{d\in D} x_{ij}^d \geqslant 0 \quad \forall (i, j) \in I_c. \tag{2.6.2}$$

The resulting integer model will be denoted by SPC-VCSP, since it is a mixed covering/partitioning model. The main advantage of introducing covering constraints in the integrated model is to allow drivers to walk over deadhead arcs that are not included in the vehicle schedule defined by the optimal solution of the VCSP. As a nice consequence changeovers are allowed and the complexity of the pricing problem is not increased. That is, with covering constraints changeovers are handled in the master problem instead of being handled in the pricing problem. Furthermore, this allows more flexibility on the cover of the vehicle schedule by a set of crews. In some cases it may overcome the lack of some columns needed to obtain integer solutions and that were not generated, neither during the branching process, nor in the linear relaxation optimization (Fig. 6).

Usually, the space and time conditions defined for a changeover to occur lead to $|I_c| \ll |I\backslash I_c|$. Consequently, we expect that the optimal solution for the SPC-VCSP will contain very little or no over-covering and will be similar to the SP-VCSP optimal solution.

For model SPC-VCSP, Lemmas 2.1 and 2.2 are not valid. However, as we will see in Section 4, requiring only the integrality of the duty variables was enough to obtain integer solutions.

3.5. *The branching procedure*

If the optimal solution of the linear problem is integer, then it is also the optimal solution for the integer problem. Otherwise, an integer solution is obtained by branch and bound over the set of duties that have been generated while solving the linear programming relaxation.

Usually to obtain high-quality solutions, new columns need to be generated during the branch and bound. However, this could be time consuming since the pricing problem needs to be solved several times during the branching process. The algorithm proposed in this paper looks for an integer solution over a restricted set of columns, generated at the root node. Therefore, we have to obtain, at the root node, a set of columns large enough to include good integer solutions. At each iteration of the pricing problem, we add to the master a relatively large number of columns with negative reduced cost. In the next section we show that, for the tested instances, the subset of the generated duties at the root node was sufficient to obtain integer solutions with high quality.

We noticed that the decision variables corresponding to pull-in and pull-out arcs with value one in the optimal solution of the linear programming relaxation, often have value one in the integer final solution. Consequently, and in order to decrease the computing time consumed by the branch and bound, we decided to fix the pull-out and pull-in decision variables, whenever they took value one in the optimal solution of the linear programming relaxation.

Lemmas 2.1 and 2.2 lead to two different branching strategies. One strategy consists of branching on the vehicle scheduling variables and the other of branching on the crew scheduling variables. Branching decisions do not change the nature of the problem but can reduce the size of the branch-and-bound tree.

4. Computational experience

Our approach was implemented in C + + and we use CPLEX 9.0 to solve the linear programming relaxations and the tree search procedure. In order to obtain an initial set of duties we use a quasi-assignment algorithm [12] to solve the multi-depot vehicle scheduling problem without requiring that each vehicle return to the source depot.

All computational results presented were obtained on PC Pentium IV 3.2 GHz.

4.1. Test data

The approaches proposed in this paper have been tested with randomly generated data problems available at the web page http://www.few.eur.nl/few/people/huisman/instances.htm. A detailed description of these instances is given by Huisman et al. [3]. Briefly, the instances have been classified in two classes according to the travel speed. The travel speed is lower for problems in class B than for problems in class A. Therefore, trips are longer for class B than for class A. We present computational results obtained for class A involving 4 depots and n trips, with $n = 80$, 100, 160, 200, 320 and 400. For each value of n, 10 instances are available. Similar results were obtained for type B and instances with 2 depots, which are easier to solve than the instances we have considered.

In order to find a solution with the minimum number of vehicles a penalty is added to each pull-out and each pull-in trip cost. Concerning the crews, a fixed cost is assigned to each crew in the solution. The amount of time allocated before a duty starts, sign_on time, is 10 min if the crew leaves the depot driving a vehicle and 15 min if the crew leaves the depot walking. The sign_off time is 5 min if the crew reaches the depot driving a vehicle and 15 min if the crew returns to the depot walking.

We considered three types of duty, namely a tripper type and two normal types. The tripper-type duties have a spread between 30 and 300 min. For both normal-type duties, the minimum spread is 30 min, the maximum working time is 540 min, the minimum break length is 45 min and the maximum duration allowed before a break occurs is 300 minutes. The first normal type has a maximum spread of 585 min while the second one has a maximum spread of 720 min.

In our computational tests we are going to consider only model SPC-VCSP since its flexibility allows handling a larger number of situations than model SP-VCSP. Besides, we can always make $I_c = \emptyset$ falling into model SP-VCSP. We decided that a changeover might occur if and only if the location, where the driver leaves the first vehicle, is the same location where he picks up the second one. Thus, for the SPC-VCSP model, covering constraints exist for arcs $(i, j) \in I_c$ such that the end location of trip i is the same as the starting location of j.

According to Lemmas 2.1 and 2.2, concerning the SP-VCSP model, branching can be made either on duty variables or on vehicle variables. However, we did some preliminary computational tests and we have noticed that higher computing time was needed when branching was performed on the vehicle variables (multicommodity flow variables—x_{ij}^d). Note that, when branching on the x_{ij}^d variables the resulting problem on the y_ℓ variables is a special set partitioning model with an interval constraint matrix (see Lemma 2.2.) and when branching on the y_ℓ variables the resulting problem on x_{ij}^d variables is a network flow model. In spite of the fact that, for model SPC-VCSP, Lemma 2.2 may not be valid we noticed that, if branching was made on the duty variables the resulting solutions were always integer. Furthermore, the resulting solutions contain few arcs $(i, j) \in I_c$ and thus, they are almost partition solutions. Note that, usually, in real-life problems, we have $|I_c| \ll |I|$, what can, partially, explain what we have mentioned above. Consequently, for the results presented in the next sections, branch was always performed on the duty variables.

Table 2
Optimal pricing versus suboptimal pricing

Problem	Optimal pricing					Suboptimal pricing				
type	#nv	#nc	tot	cpu_lp	cpu_ip	#nv	#nc	tot	cpu_lp	cpu_ip
80A	9.4	18.8	28.2	187	14	9.4	19.3	28.7	49	24
100A	12.1	23.3	35.4	672	130	12.1	23.4	35.5	331	97

Table 3
Detailed results

Problem type	cpu_lp	#col_gen	nodes	cpu_ip
80A	49	8992	9	24
100A	331	17194	15	97
160A	1631	28617	18	689
200A	1575	30720	46	1489
320A	5081	40577	127	5942
400A	6755	49848	91	6698

4.2. Computational results

In this section we analyze the quality of the solutions obtained by the method proposed in this paper. First, we compare suboptimal pricing with optimal pricing. Second, we detailed some computational results concerning the linear programming relaxation and the branch and bound. Third, the integer solutions we obtained are compared with those obtained by other authors for the same test instances.

Table 2 reports on average results obtained for instances with 80 and 100 trips. For $n \geqslant 160$ the pricing problem becomes difficult to solve to optimality due to the huge number of states.

Columns "#nc" and "#nv" display the number of crews and vehicles, respectively, while "tot" is the sum of these values. The CPU times are given in seconds. Column "cpu_lp" refers to the linear programming CPU times while column "cpu-ip" refers to the branch-and-bound CPU times.

For these size instances, when the pricing sub-problem is not solved to optimality, the cpu_lp reduces drastically leading to a great reduction on the total CPU times (cpu_lp + cpu_ip) without decreasing the quality of the solutions. Therefore, the results presented in the next tables were obtained using suboptimal pricing.

In Table 3, we present some detailed average results for all tested instances. We show the CPU times for solving the linear programming relaxation (cpu_lp), the number of columns generated during the solve of the linear programming relaxation (#col_gen), the number of nodes of the branch-and-bound tree and the computation time for obtaining an integer optimal solution over the columns generated at the root node (cpu_ip).

As expected when the number of trips increases the CPU times as well as the number of nodes of the branching tree increase. For the 320A instances an outlier moves the average number of nodes to a large value. Without this outlier the average number of nodes becomes 73 nodes, which is more consistent with the values obtained for the other instances.

In Table 4, we give an overview of the results obtained by Borndörfer et al. [5] and Huisman et al. [3] for the same test instances and compare them with the results obtained by our approach.

We noticed that the set of duties generated at the root node was "large" enough to obtain good quality integer solutions in terms of both the number of crews and the number of vehicles. Regarding results in [5,3] for the same test instances, our approaches led to a smaller number of crews although, in some cases, a greater number of vehicles. However, we obtained better values for the sum of vehicles and crews. We think that an important improvement over the existing methods is the time consumed by our algorithm to obtain these results. We can not make a direct comparison, since different computers have been used by the different authors. However, we can state that when the size of the problem increases the time spent by our algorithm becomes significantly smaller than the time spent by the algorithm proposed in [5]. From a transportation company point of view, it is an important feature of an algorithm to produce quick and 'good' solutions.

Table 4
Comparison of solutions

Problem type	Approach	Partition/covering			
		#nv	#nc	tot	cpu_tot
80A	Mesquita/Paias	9.4	19.3	28.7	72
	Borndörfer et al.	9.2	20.4	29.6	780[a]
	Huisman et al.	9.2	20.4	29.6	–
100A	Mesquita/Paias	12.1	23.4	35.5	428
	Borndörfer et al.	11.2	24.5	35.7	1260[a]
	Huisman et al.	11.0	25.2	36.2	–
160A	Mesquita/Paias	15.2	30.9	46.1	2436
	Borndörfer et al.	15	32.7	47.7	2640[a]
	Huisman et al.	14.8	34.1	48.9	–
200A	Mesquita/Paias	18.7	38.2	56.9	3064
	Borndörfer et al.	18.5	40.5	59	5280[a]
	Huisman et al.	18.4	41.6	60.0	–
320A	Mesquita/Paias	27.2	55.3	82.5	11023
	Borndörfer et al.	26.7	56.1	82.8	19680[a]
400A	Mesquita/Paias	33.2	68.6	101.8	13453
	Borndörfer et al.	33.1	69.8	102.0	43200[a]

[a]Dell Precision 650 PC with 4 GB of main memory and a dual Intel Xeon 3.0 GHz CPU.

5. Conclusions

In this paper two different mathematical models were proposed for the integrated VCSP, namely the SP-VCSP and the SPC-VCSP. The choice of which mathematical model should be used depends on the particular situation and on the different constraints we want to include in the problem. The advantages of the SPC-VCSP model over the SP-VCSP is the capability of handling some constraints of the crew scheduling problem through the master problem, like for example changeovers, without increasing the complexity of the sub-problem. The only inconvenience is the over-cover of some tasks by the crews. By combining set partitioning with set covering constraints over-cover can only occur on a subset of the tasks fixed for each particular problem. In addition, due to the crew costs structure, few over-covers are expected in the final solutions. Consequently, the mathematical model SPC-VCSP seemed to be well adjustable to the rules that define feasibility for the duties, and this makes it very interesting in modelling the different real situations.

The method proposed to handle these mathematical models starts with a pre-processing procedure to define the set of tasks. This definition is based on the optimal solution of a multi-depot vehicle scheduling problem without requiring that each vehicle return to the source depot where the costs include some features of the crew scheduling problem. Then, in order to obtain an initial set of duties necessary to start the column generator, we again solve a multi-depot vehicle scheduling problem without requiring that each vehicle return to the source depot with the set of tasks as input.

A suboptimal pricing procedure is developed to obtain near-optimal solutions to the linear programming relaxation by a column generation scheme and to deal with large size instances. If the resulting solutions are not integer then branch-and-bound techniques are used to obtain integer solutions.

Compared with previous work presented in the literature, computational results using the SPC-VCSP, indicate that the proposed method seemed to be more effective in terms of the quality of the solutions produced and the time needed to obtain them. Moreover, the SPC-VCSP resulting solutions have few over-covers and are similar to partition-type solutions and this makes them easier to implement in a real situation. In conclusion, the proposed method combined with the SPC-VCSP seems to be a promising tool for dealing with large instances of the integrated VCSP.

References

[1] Freling R, Wagelmans APM, Paixão JMP. An overview of models and techniques for integrating vehicle and crew scheduling. In: Wilson NHM, editor. Computer-aided transit scheduling, Lecture notes in economics and mathematical systems, vol. 471. Berlin, Germany: Springer; 1999. p. 441–60.

[2] Freling R, Huisman D, Wagelmans APM. Models and algorithms for integration of vehicle and crew scheduling. Journal of Scheduling 2003;6: 63–85.

[3] Huisman D, Freling R, Wagelmans APM. Multiple-depot integrated vehicle and crew scheduling. Transportation Science 2005;39(4):491–502.

[4] De Groot S, Huisman D. Vehicle and crew scheduling: solving large real-world instances with an integrated approach, Econometric Institute Report EI2004-13, Erasmus University, Rotterdam, The Netherlands; 2004.

[5] Borndörfer R, Löbel A, Weider S. A bundle method for integrated multi-depot vehicle and duty scheduling public transit. ZIB- Report 04-14 Zuse Institute Berlin; 2004.

[6] Hollis BL, Forbes MA, Douglas BE. Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. European Journal of Operational Research 2006;173:133–50.

[7] Valouxis C, Housos E. Combined bus and driver scheduling. Computers & Operations Research 2002;29:243–59.

[8] Gaffi A, Nonato M. An integrated approach to ex-urban crew and vehicle scheduling. In: Wilson NHM, editor. Computer-aided transit scheduling, Lecture notes in economics and mathematical systems, vol. 471. Berlin, Germany: Springer; 1999. p. 103–28.

[9] Haase K, Friberg C. An exact branch and cut algorithm for the vehicle and crew scheduling. In: Wilson NHM, editor. Computer-aided transit scheduling, Lecture notes in economics and mathematical systems, vol. 471. Berlin, Germany: Springer; 1999. p. 63–80.

[10] Haase K, Desaulniers G, Desrosiers J. Simultaneous vehicle and crew scheduling in urban mass transit systems. Transportation Science 2001;35(3):286–303.

[11] Desrochers M, Soumis F. A column generation approach to the urban transit crew scheduling problem. Transportation Science 1989;23(1): 1–12.

[12] Mesquita M, Paixão J. Multiple depot vehicle scheduling problem: a new heuristic based on quasi-assignment algorithms. In: Desrochers M, Rousseau JM, editors. Computer-aided transit scheduling, Lecture notes in economics and mathematical systems, vol. 386. Berlin, Germany: Springer; 1992. p. 167–80.