Public Transp (2009) 1: 21–37
DOI 10.1007/s12469-008-0005-2

ORIGINAL PAPER

Branching approaches for integrated vehicle and crew scheduling

Marta Mesquita · Ana Paias · Ana Respício

Published online: 6 November 2008

Abstract The integrated multi-depot vehicle and crew scheduling problem simultaneously builds vehicle blocks and crew duties. We present an integer mathematical formulation that combines a multi-commodity flow model with a mixed set partitioning/covering model. We propose solution approaches that start by solving the linear programming relaxation of the model. Whenever the resulting linear programming solution is not integer, three branching alternative strategies can be applied: a branch-and-bound algorithm and two branch-and-price schemes. The branch-and-bound algorithm performs branching over the set of feasible crew duties generated while solving the linear relaxation. In the first branch-and-price scheme the linear programming relaxation is solved approximately, while in the second one it is solved exactly. Computational experience is reported over two different types of problems: randomly generated data publicly available for benchmarking in the Internet and data from a bus company operating in Lisbon.

M. Mesquita
ISA, Dep. Matemática, Technical University of Lisbon, Tapada da Ajuda, 1349-017 Lisbon, Portugal
e-mail: martaoliv@isa.utl.pt

A. Paias (✉)
DEIO, University of Lisbon, Bloco C6, piso 4, Cidade Universitária, 1749-016 Lisbon, Portugal
e-mail: ampaias@fc.ul.pt

A. Respício
DI, University of Lisbon, Bloco C6, piso 3, Cidade Universitária, 1749-016 Lisbon, Portugal
e-mail: respicio@di.fc.ul.pt

M. Mesquita · A. Paias · A. Respício
Centro de Investigação Operacional, Lisbon, Portugal

1 Introduction

The integrated multi-depot Vehicle and Crew Scheduling Problem (VCSP) consists of determining minimal cost vehicle and crew schedules to perform a set of timetabled trips. Usually, mass transit companies tackle these problems separately and solve them in sequence. First, the vehicle scheduling problem is solved obtaining a set of vehicle blocks. Then, drivers are assigned to the vehicle blocks. If necessary, some adjustments are performed over the vehicle blocks to obtain better schedules for the drivers. The strong dependency between these two problems suggests that an integrated approach may lead to important cost reductions.

Since both the multi-depot vehicle scheduling problem and the crew scheduling problem are NP-Hard problems, heuristic approaches are adequate for dealing with the integrated problem. Heuristic algorithms for the multi-depot VCSP have been proposed by Borndörfer et al. (2008), Gaffi and Nonato (1999), Huisman et al. (2005), Gintner et al. (2005) and Valouxis and Housos (2002). In Gintner et al. (2008) the authors present an approach for solving the crew scheduling problem where, instead of using a fixed underlying vehicle schedule, they choose from a set of possibilities the most consistent vehicle schedule with respect to the crew scheduling.

In real life problems, the main goal is to obtain "good" solutions in a short time. Usually, obtaining an optimal solution requires a considerable increase in the CPU time. Additionally, the quality of the solutions sometimes depends on factors that have not been pointed out a-priori by the schedulers, but which are identified, a-posteriori, making some solutions undesirable. Thus it is important to balance the trade-off between the quality of a solution and the computational time required to obtain it.

The research presented in this paper is motivated by some questions that were raised and were left open in a previous work (Mesquita and Paias 2008). In this work (Mesquita and Paias 2008), the VCSP was formulated using an integer linear programming (ILP) model. To solve the model, a non-exact branch-and-bound algorithm was proposed where integer solutions were obtained by branching over the set of columns generated while solving the linear programming (LP) relaxation. The algorithm was tested with random data and the results obtained were promising when compared to the results obtained by other authors for the same test instances. We believe that the reason for this good behavior was the fact that the subset of columns generated at the root node of the branch and bound, although small, had a great diversity. However, this method and its results have led to several questions that remain open, such as:

(i) Would the behavior of the method change for different test data or for different choices of some parameters involved in the branch-and-bound algorithm?

(ii) What would be the effect in the quality of the solutions if new columns were generated during the branching process? And, what will be the increase in the CPU time?

In the current paper we try to answer these questions. Thus, we test the non exact branch-and-bound algorithm presented in Mesquita and Paias (2008) with real data and for different choices of some parameters involved in the algorithm and we compare it with two new algorithms: an exact and a non-exact algorithm, both based on a new branch-and-price scheme.

The paper is organized as follows. In the next section (Sect. 2) we formulate the VCSP as an ILP and discuss the use of a mixed set partitioning/covering model. In Sect. 3, we propose a preprocessing phase to determine the set of tasks. In Sect. 4, we describe a procedure to solve approximately the LP relaxation of the model by a column generation scheme. In Sect. 5, we propose alternative branching schemes to be used when the solution of the LP relaxation is not integer. Computational results, with random and real data are presented and discussed in Sect. 6. Finally, in the last section some conclusions are drawn.

2 The integrated vehicle and crew scheduling problem

First, we introduce some notation necessary to define the multi-depot integrated vehicle and crew scheduling problem.

Consider a set of n timetabled trips, $T_1, \ldots, T_n$, that have to be operated in a planning interval T. The starting time and location, and the ending time and location of each timetabled trip are known. Consider a set of k depots $D_1, \ldots, D_k$. The location and the number of vehicles available, v_d, at depot D_d, $d = 1, \ldots, k$ are also known.

Two trips are *compatible* if the vehicle released after trip T_i completion can be assigned to trip T_j. The corresponding ordered pair (T_i, T_j) is said to be compatible. Associated to each pair of compatible trips (T_i, T_j) a *deadhead trip* may occur, where the vehicle runs without passengers, between the end location of T_i and the start location of T_j.

A *vehicle block* is a sequence of deadhead trips, linking compatible time-tabled trips, which starts and ends at a depot. Deadhead trips from and to a depot are denoted by *pull-out* and *pull-in trips*, respectively. In this paper, we consider each end location of a trip as a potential relief point, where it is permitted to change drivers. Therefore, each *task*, the minimum portion of work that can be assigned to a driver, 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 T_i, $i = 1, \ldots, n$. A *crew duty* is a combination of tasks respecting several constraints such as: maximum and minimum spread; maximum working time without a break; minimum and maximum break duration; maximum number of changeovers.

The integrated multi-depot vehicle and crew scheduling problem determines the vehicle and the crew schedules. It identifies pairs of compatible timetabled trips to group into each vehicle block and tasks to group into crew duties, so that the resulting crew duties cover all vehicle blocks.

Mathematical formulations for the integrated problem can be divided into two types. One type combines a multi-commodity flow model to describe the vehicle scheduling with a set partitioning model to describe the crew scheduling (Borndörfer et al. 2008; Huisman et al. 2005). The other type uses a set partitioning model to describe both problems (Haase and Friberg 1999; Haase et al. 2001). We present a mathematical formulation that combines a multi-commodity flow formulation with a mixed set partitioning/covering formulation.

Let $N = \{1, \ldots, n\}$ denote the set of vertices, where vertex $i \in N$ represents trip T_i and let D denote the set of vertices corresponding to the k depots, $D_1, \ldots, D_k$. There

is a vertex $n + d$, $d \in D$, corresponding to each depot D_d. Let $I \subseteq N \times N$ be the set of compatible pairs of trips. For each $d \in D$ we associate graph $G_d = (V_d, A_d)$, where the set of vertices and the set of arcs are defined by $V_d = N \cup \{n + d\}$ and $A_d = I \cup \{(n+d) \times N\} \cup \{N \times (n+d)\}$, respectively. Let L be the set of all feasible crew duties. The integrated vehicle and crew scheduling problem can be formulated as an ILP model where two types of decision variables are considered. The decision variables related with the scheduling of vehicles $x^d_{i,j}$, $(i, j) \in I, d \in D$, $x_{i,n+d}, x_{n+d,i}$, $i \in N$, $d \in D$, and the decision variables y_ℓ, $\ell \in L$, related with the scheduling of the crews. Let $x^d_{i,j} = 1$ if a vehicle from depot d performs trips i and j in sequence, let $x_{n+d,i} = 1$, $i \in N$, $d \in D$, if a vehicle from depot d runs directly from depot d to trip i and $x_{i,n+d} = 1$, $i \in N$, $d \in D$ if the vehicle returns directly to depot d after performing trip i. Let $y_\ell = 1$, $\ell \in L$, if the crew duty ℓ is in the optimal solution.

The cost associated to deadhead trips c^d_{ij}, $c_{i,n+d}$, $c_{n+d,i}$ represents travel costs related to fuel consumption and vehicle maintenance or penalties imposed by each specific company as for example on idle times. Concerning the crews, let s_ℓ be the cost associated to duty y_ℓ. Cost s_ℓ includes a fixed cost (a driver's salary or a crew unit) and operational costs related to overtime, evening periods, etc.

We define the following sets:

$$L(i, j) = \{\text{duties } \ell \text{ covering the deadhead trip from trip } i \text{ to trip } j \text{ and covering trip } j\},$$

$$DL(j) = \{\text{duties } \ell \text{ covering the deadhead trip from any depot to trip } j \text{ and covering trip } j\},$$

$$LD(i) = \{\text{duties } \ell \text{ covering the deadhead trip from trip } i \text{ to any depot}\}.$$

The VCSP can be formulated through the following ILP model

[*PCVCSP*]

$$\min \sum_{d\in D}\sum_{(i,j)\in I} c^d_{ij}x^d_{ij} + \sum_{d\in D}\sum_{i\in N}(c_{i,n+d}x_{i,n+d} + c_{n+d,i}x_{n+d,i}) + \sum_{\ell\in L} s_\ell y_\ell \tag{1}$$

$$\sum_{d\in D}\sum_{j:(i,j)\in I} x^d_{ij} + \sum_{d\in D} x_{i,n+d} = 1, \quad \forall i \in N \tag{2}$$

$$\sum_{j:(i,j)\in I} x^d_{ij} + x_{i,n+d} - \sum_{j:(j,i)\in I} x^d_{ji} - x_{n+d,i} = 0, \quad \forall i \in N,\ \forall d \in D \tag{3}$$

$$\sum_{i\in I} x_{n+d,i} \le v_d, \quad \forall d \in D \tag{4}$$

$$\sum_{\ell\in DL(j)} y_\ell - \sum_{d\in D} x_{n+d,j} = 0, \quad \forall j \in N \tag{5}$$

$$\sum_{\ell\in L(i,j)} y_\ell - \sum_{d\in D} x^d_{ij} = 0, \quad \forall (i, j) \in I - I_c \tag{6}$$

$$\sum_{\ell \in L(i,j)} y_\ell - \sum_{d \in D} x_{ij}^d \geq 0, \quad \forall (i,j) \in I_c \tag{7}$$

$$\sum_{\ell \in LD(i)} y_\ell - \sum_{d \in D} x_{i,n+d} = 0, \quad \forall i \in N \tag{8}$$

$$x_{ij}^d \in \{0,1\}, \quad \forall (i,j) \in I, \ \forall d \in D \tag{9}$$

$$x_{n+d,i}, x_{i,n+d} \in \{0,1\}, \quad \forall i \in N, \ \forall d \in D \tag{10}$$

$$y_\ell \in \{0,1\}, \quad \forall \ell \in L. \tag{11}$$

Constraints (2) ensure that each trip i is operated by exactly one vehicle. Constraints (3) ensure that each vehicle returns to its source depot. Constraints (4) are depot capacities on the number of available vehicles. Constraints (5) to (8) connect the vehicle and the crews ensuring that each deadhead arc in the solution of the vehicle schedule will be covered by a crew.

Concerning constraints (6) and (7), we have partitioned the arc set I into two subsets, I_c and $I - I_c$. In the arc set I_c we have included the deadhead arcs where changeovers may occur. Constraints (5), (6) and (8), ensure that there is a single driver assigned to each vehicle in a vehicle block. Constraints (7) allow two different situations. On the one hand, it allows deadhead trips to be covered by more than one driver. On the other hand, it allows drivers to walk over deadhead arcs that are not included in a vehicle block. This last situation corresponds to a walk movement of a driver in order to change from the vehicle he was driving to another one. Note that, over-covering only occurs when assigning several drivers to a task is cheaper than assigning a single one. Consequently, the major role of constraints (7) is to allow changeovers to be handled explicitly in the constraint set.

Although constraints (5) to (8) are stated just for arcs corresponding to deadhead trips, according to our task definition—each task includes a deadhead trip, (i, j), followed by a timetabled trip, j—they are sufficient to also guarantee the cover of all timetabled trips in a vehicle block by crew duties.

The next lemma, obtained on a previous work (Mesquita and Paias 2008), shows that, with $I_c = \emptyset$, the PCVCSP reduces to a mixed integer model where just one type of the decision variables is required to be integer.

Lemma 1 *The mathematical formulation PCVCSP with* $I_c = \emptyset$ *and* $x \geq 0$ *has an integer optimal solution.*

This lemma suggests that on a branching scheme just one subset of the decision variables might be considered.

3 Task definition

When the vehicle and the crew scheduling problems are solved sequentially, the set of tasks is defined after the vehicle blocks are known. A task will cover one or a few timetabled trips and the set of tasks usually does not coincide with the set of

timetabled trips. In the integrated problem, vehicle blocks and crew duties are simultaneously built. Vehicle blocks cover deadhead trips and timetabled trips while crew duties cover tasks. Due to their starting and ending times and locations, some pairs of trips are expected to be covered by the same vehicle and the same crew in the optimal solution. These pairs can be, a-priori, included in the same task. The algorithm presented in this paper starts with a preprocessing phase that defines the set of tasks. After that, the algorithm builds an integrated solution based on this set of tasks.

The question is how to know, in advance, "which trips to merge into a task". We must be careful because merging trips has a big effect on the resulting schedule. Merging two particular trips can be efficient from a vehicle scheduling point of view. However it can be very inefficient from a crew scheduling point of view. On the one hand, long tasks might reduce the dimension of the integer problem but might be difficult to combine into feasible duties. On the other hand, tiny tasks are easier to combine but the resulting combinatorial problem is expected to be more difficult to solve.

The proposed preprocessing procedure is based on the optimal solution of a multi-depot vehicle scheduling problem without requiring that each vehicle returns to the source depot. The corresponding optimal solution can be obtained by a polynomial algorithm and it is a set of vehicle blocks covering all timetabled trips. In the task definition, it is important to consider both vehicle and crew features. Crew features were included in the costs of the objective function. The cost of a deadhead arc was set equal to the time spent by the crew to cover that deadhead trip. For all $(i, j) \in I$, $c^d_{i,j}$ was set equal to the starting time of trip j minus the ending time of trip i, which minimizes the time a crew runs a vehicle without passengers (including idle time). Concerning pull-out and pull-in trips, two cases were considered. In the first one, a big cost was assigned to each pull-out and each pull-in deadhead time, to minimize the number of vehicles in the schedule. In the second case, only pull-out and pull-in deadhead times were considered. We have compared both options, from a computational point of view, and we have concluded that better final solutions for the integrated problem were obtained with the first one.

Tasks were obtained by analyzing each vehicle block and merging two consecutive trips whenever the corresponding deadhead cost is smaller than a threshold value, ε, and the resulting task does not violate duty constraints. Note that the dimension of the input of the integrated problem is directly related with the choice of the ε value and can somehow be managed by the scheduler. Large (small) ε values will lead to a small (large) number of tasks.

4 Linear programming relaxation

Our solution approach uses column generation to solve the LP relaxation of model PCVCSP. We consider all vehicle variables, $x^d_{i,j}$, explicitly and all crew variables, y_ℓ, implicitly.

Each feasible crew duty can be seen as an adequate path in a network that takes into account special features of the integrated problem. Feasibility is established by deciding which arcs are included in the network and by using resources that are consumed along the network. Imposing time windows on each (some) vertex restricts the

resource consumption. The pricing problem is a shortest path problem with resource constraints and can be solved using dynamic programming where states are related to crew duties while stages are related to tasks. A complete description of the duty generator can be found in Mesquita and Paias (2008).

If the number of trips/tasks increases, the number of states of the dynamic problem also increases and the pricing problem becomes hard to solve. For problems with a large number of trips/tasks, we propose a heuristic pricing procedure that works with a reduced state space. To obtain this reduced space we propose a combination of two processes expecting to keep some diversity in the set of generated columns. On the one hand, we look for states having a cost larger than a predefined constant δ, and discard them. It is not expected that a state with a large cost will lead to a negative reduced cost path. On the other hand, we construct a filter that discards some states at random. We predefine two integers α and β where the rate $\frac{\alpha}{\beta}$ is the probability of a state to be discarded. At each stage, for each state of the dynamic program we randomly generate an integer $r \in [1, \beta]$ and if $r \leq \alpha$ the state is discarded. If an important state is discarded in one iteration then it is expected to be generated in further iterations and not always discarded. If some columns with a high negative reduced cost have been discarded, then lower reduced cost columns will be added to the master problem, instigating some diversity in the set of added columns. Note that with suboptimal pricing, we might not obtain the optimal solution for the LP relaxation.

Finally, we also define a parameter γ, which is an upper bound on the number of columns added to the master problem at each iteration of the column generator.

An initial set of duties is needed to start the pricing problem resolution. Therefore, we solve a multi-depot vehicle scheduling problem, without requiring each vehicle to return to the source depot. We consider the original deadhead costs and take as input the set of tasks defined in the preprocessing phase. In the optimal solution, each vehicle block is assigned to exactly one depot and is split into duties satisfying a subset of the crew feasibility constraints. The resulting set of duties is checked for the remaining duty constraints and a big cost is assigned to non-feasible duties.

5 Branching procedures

The formulation PCVCSP includes explicitly all possible columns for vehicle variables and includes implicitly all possible columns for crew variables. Thus, this formulation corresponds to the initial Master Problem (MP), which is considered for optimization at the root node of the underlying search tree. A Restricted Master Problem (RMP) consists of explicitly considering only a restricted set of feasible columns in the master. At the root node, the initial set of columns, considered in RMP, contains all vehicle variables and a small number of crew variables generated to start the pricing problem as explained in the last paragraph of Sect. 4. To process a node, the LP relaxation of the corresponding MP, further referred $LP(MP)$, must be solved. The process starts by solving the LP relaxation of the RMP. Then, other columns are priced out by the column generation subproblem. If there exists a feasible attractive column or set of columns (depending on parameter γ), these columns are added to

the RMP, setting up a new one. This process reiterates until the subproblem is unable to find new feasible attractive columns. At this point, the LP relaxation of the current RMP is also optimal for the LP relaxation of the MP in this node. Whenever this solution is not integral, branching occurs giving rise to two new search nodes, and two new MPs, respectively. The RMP in each of these nodes results from adding a branching constraint in the current RMP.

5.1 Branch-and-bound procedure

The first branching approach uses branch-and-bound techniques over the set of feasible crew duties generated while solving the LP relaxation of PCVCSP. The resulting solution is optimal for the set of columns generated at the root node. However, it might not be an optimal solution for the VCSP, although it is still feasible. The main advantage of the branch-and-bound procedure is to avoid the resolution of several pricing problems during the branching process. Aiming at overtaking eventual drawbacks, a set of varied duties is generated at the root node using the heuristic pricing described in Sect. 4. This diversity of generated columns is an important factor to obtain good quality feasible solutions for the integrated problem. Comparing with the branch-and-price procedures, we expect to save some CPU time with the branch-and-bound.

We have noticed that the decision variables corresponding to pull-in and pull-out arcs with value one in the optimal solution of the LP relaxation often have value one in the integer final solution. Hence, to decrease the computing time consumed by the branch-and-bound, we fix the pull-out and pull-in vehicle variables whenever they take value one in the optimal/suboptimal solution of the LP relaxation.

We have compared two branching strategies. In one strategy, we first branch on the crew variables, while on the other we first branch on the vehicle variables. Computational experience has shown that with the first option less computation time was needed to obtain the optimal solution.

5.2 Branch-and-price procedures

The main difference between the branch-and-bound procedure and the branch-and-price procedures is that for the branch-and-bound procedure columns are only generated at the root node, while for the branch-and-price procedures columns are generated during the overall process. Additionally, the branching constraints, which are incrementally added along the path from the root of the search tree, are also different for the referred procedures as explained next. For branch-and-bound, each branching constraint relates to a single crew variable fixing it to 0 or 1. For branch-and-price, before branching on single variables, branching constraints relate to a given subset of columns setting up a new bound over the sum of the corresponding variables. At each node a branching constraint is explicitly added to the current RMP in a way that the structure of the pricing problem remains unchanged, contrary to the usual scenario.

Branching rule If at node u, $LP(MP_u)$ is not integral, branching is performed by looking for an arc (i, j), corresponding to a deadhead trip, such that the sum of the

vehicle variables covering this arc is not integer, respectively

$$0 < \sum_{d \in D} x_{i,j}^{d} < 1. \tag{12}$$

The branching constraints will enforce either

$$\sum_{d \in D} x_{i,j}^{d} = 1 \quad \text{or} \tag{13}$$

$$\sum_{d \in D} x_{i,j}^{d} = 0. \tag{14}$$

Two descendant nodes from u, u_L and u_R, are created and constraint (13) is added to MP_{u_L}, while constraint (14) is added to MP_{u_R}.

Constraint (13) enforces one vehicle covering the fixed arc and, consequently, there will be at least one crew covering it.

Two branching strategies were used, which only differ in the order different candidates for branching are considered. Strategy 1 gives preference to arcs linking timetabled trips, while strategy 2 starts with arcs linking a depot to a timetabled trip or *vice-versa*, as described below.

Branching strategy 1

1. Branch by considering a pair (i, j), such that $(i, j) \in I$ and verify (12);
2. if there are no arcs satisfying the previous condition, branch by considering a pair (i, j) verifying (12) and for which $i \in N$, $j \in D$ or $i \in D$, $j \in N$.

Branching strategy 2

1. Branch by considering a pair (i, j) verifying (12) and for which $i \in N$, $j \in D$ or $i \in D$, $j \in N$;
2. if there are no arcs satisfying the previous condition, branch by considering a pair (i, j), such that $(i, j) \in I$ and verify (12).

Upper bound We propose two methods for obtaining an initial upper bound. In the first one, an initial set of crew duties is built from the optimal solution of a multi-depot vehicle scheduling problem, without requiring each vehicle to return to the source depot. An alternative method to obtain an upper bound is to use branch-and-bound techniques over the set of feasible crew duties generated while solving the LP relaxation of the RMP at the root. In fact, a feasible solution can be obtained in a reasonable time by handling the parameters involved in the branch-and-bound procedure, as we shall see in the next section.

Optimal versus suboptimal pricing Two different branch-and-price procedures are proposed. For larger instances, the pricing problem becomes too difficult to be solved to optimality and we may solve it approximately, as described in Sect. 4, by removing some states of the dynamic state space. In such a case, the solution obtained at each node u of the search tree might not be the optimal solution for the corresponding LP

relaxation of the MP, $LP(MP_u)$. Consequently, the final solution given by the branch-and-price procedure might not be an optimal solution, although it is still feasible for the integrated problem. To overcome this situation, we also propose an exact branch-and-price algorithm where, at each node of the tree, the pricing problem is solved to optimality. In practice, this algorithm has shown the ability to handle only small sized problems.

5.3 Tuning the parameters for non-exact procedures

In the previous sections we have mentioned some parameters whose values have a great influence on the final solution. To achieve good quality solutions, a fine tuning of these parameters is needed. In all, five parameters may be manipulated by the scheduler, ε, $p = \frac{\alpha}{\beta}$, δ and γ. The first one, ε, is directly related with the dimension of the input of the problem. The scheduler can manage the number of tasks of the problem. A large value of ε will result in a small number of tasks with a long duration. In most cases, the resulting integrated problem is easier to solve. The disadvantage of large values of ε is that the resulting integer solution may have a large number of crews, since it is difficult to combine long tasks into feasible duty crews. Consequently, the resulting solutions for the integrated problem may only be feasible if a great number of crews is available.

When suboptimal pricing is used we have to decide the probability p, $p = \frac{\alpha}{\beta}$, to discard a state at each stage. If the value of p is small we will expect the same behavior as if optimal pricing was performed meaning that only few states were discarded. Increasing the value of p will result in discarding more states, which makes the dynamic program easier to solve, but might result in more iterations to solve the LP problem. Another parameter related with the rejection of some states is δ. The choice of value δ must take into account that states which may lead to negative reduced cost paths should not be rejected.

Finally, we have to fix the value of parameter γ, which states the maximum number of columns with negative reduced cost to include in the master problem at each iteration. If few columns are included then more iterations are needed to solve the LP problem. If a large number of columns are included then the LP problem is hard to solve and more CPU time is spent to obtain the optimal solution.

The values of all these parameters should be such that a great diversity occurs in the set of the generated columns, allowing multiple combinations of columns.

Next, we discuss how to choose the values for the different parameters according to some characteristics of the solution we want to achieve. The most relevant aspects we have to take into account are the dimension of the original problem and the purpose for obtaining its solution.

If our intention is to obtain a feasible solution quickly, then we should use a large value of ε. Besides, a small value of p should be considered to avoid discarding too many columns in the pricing subproblem and the parameter γ should take a large value.

If our intention is to obtain a good quality feasible solution in a realistic time then ε should have a value that leads to short tasks, concerning the time duration. The values of p and γ should be chosen in such a way that allow us to obtain a smaller but

varied set of duties, maintaining the quality of the LP relaxation bound but reducing the corresponding CPU time.

In Sect. 6, we show how different choices for the parameter values affect the final solutions and the computing time necessary to obtain them.

6 Computational experience

All the branching approaches were implemented in C. The LP relaxations and the branch-and-bound procedure were solved using CPLEX 9.0 routines. We developed and implemented, in C, the tree search procedure for the branch-and-price approach. We used a quasi-assignment algorithm (Mesquita and Paixão 1992) to solve the multi-depot vehicle scheduling problems without requiring that vehicles return to the source depot.

We have considered different scenarios such as allowing/not allowing generating columns in the branching tree. These scenarios were combined with solving/not solving the pricing problem to optimality. We have compared the quality of the solutions and the time necessary to obtain them.

In the branch-and-bound approach we first branch on the crew variables, y_ℓ. Since, for all the instances we considered, $I_c \neq \emptyset$ Lemma 1 is no longer valid. However, we always obtained the optimal solution without branching on the x_{ij}^d variables. This can be explained by relation $|I_c| \ll |I|$ that usually occurs in practice.

For the branch-and-price approach and concerning the two branching strategies proposed in Sect. 5.2, we were not able to conclude which performs better in general, as they showed similar average results. Consequently, only results obtained with strategy 2 are presented. Moreover, it was sufficient to branch only over sums of vehicle variables.

To obtain a fair comparison with the branch-and-bound approach, we considered for the branch-and price that no upper bound was known in advance.

In order to minimize the number of vehicles a penalty is added to each pull-in and each pull-out trip cost. In the same way, to minimize the number of crews a fixed cost is assigned to each crew. The amount of time allocated before/after a crew duty starts/ends is 10/5 minutes if the crew is driving a vehicle from/to the depot and 15 minutes if the crew leaves or enters the depot walking. A driver may start/end its duty at the end location of each task or at a depot. We have considered that a changeover might occur if and only if the location, where the driver leaves the first vehicle, is the same where he picks up the second one.

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

6.1 Data from CARRIS

We investigated real-world data from CARRIS, a mass transit company operating in the city of Lisbon. The data set corresponds to urban service inside the city of Lisbon and involves four depots and 122, 168, 224, 226 and 238 trips.

According to some rules of the company, we have considered two types of duty crews, namely a tripper type and a normal type. Tripper duties have a spread between

60 and 300 minutes. For normal type duties, the minimum spread is 60 minutes and the maximum working time is 615 minutes. The maximum idle time for a crew at a relief point is 30 minutes. The time window for a break is [60, 140] minutes and the maximum time allowed before a break occurs is 300 minutes.

Our first objective was to analyze the effect of different values of parameters ε, α and β on the final solution of the branch-and-bound procedure. In Table 1, the first four columns report the number of trips, the value of ε, the resulting number of tasks and the value of $p = \frac{\alpha}{\beta}$. The next three columns describe the solution given by the branch-and-bound. More precisely, the number of vehicles, the number of crews and the sum of these two numbers. The last three columns refer to the CPU times, all measured in seconds. In particular, cpu_ l and cpu_ i are the CPU times consumed by the linear relaxation and by the branching tree, respectively, while cpu is the total CPU time.

Previous computational experience led us to fix ε within the set $E = \{5, 7, 10\}$ and p within the set $P = \{1/10, 2/15, 1/6\}$. Our computational experiments were set up over $E \times P$. On the one hand, we noticed that by increasing ε over E decreases the total CPU time. On the other hand, by increasing p over P may lead to an increase of the total CPU time. When the value of p increases more columns with negative reduced cost are discarded, at each iteration of the column generator. Then more iterations may be necessary for solving the LP relaxation and more CPU time is needed. A large value of p might, also, cause an increase in the number of nodes of the branch-and-bound tree.

Next, for each problem, we consider the set of parameters which provided the smallest number of vehicles and crews and the smallest total CPU time. Table 2 displays the performances of the branch-and-bound (BB) and the branch-and-price (BP). For the branch-and-price approach we also compare the performance of the optimal pricing procedure (BP opt) with the performance of the suboptimal pricing procedure (BP sub). Since in the branch-and-price algorithm, columns are generated during the overall branching process, we set the value of parameter γ equal to 500. The value of γ for the branch-and-bound was set equal to 3000, in order to overcome lack of columns.

The solution quality is similar for both approaches. Except for a single instance, the number of vehicles and the number of crews reached were equal. For this exception, the branch-and-bound obtained only one more crew. The total CPU time was much smaller for the branch-and-bound. An exception occurs for the instance with 168 trips where the CPU time is almost the same.

Note that suboptimal pricing and optimal pricing had similar performances concerning the number of vehicles and crews in the final solution. No conclusions can be drawn concerning the CPU time. For three, over five instances, suboptimal pricing was faster than optimal pricing.

6.2 Random data

In this section, we considered randomly generated instances available in the web page http://www.few.eur.nl/few/people/huisman/instances.htm. A detailed description of such instances is given in Huisman et al. (2005). We present computational results

Table 1 Testing different values of ε and p

Trips	ε	Tasks	p	vehicles	crews	total	cpu_1	cpu_i	cpu
122	5	67	1/10	9	18	27	4	4	8
	5	67	2/15	9	17	26	5	2	7
	5	67	1/6	9	17	26	4	2	6
	7	65	1/10	9	17	26	5	0.1	5
	7	65	2/15	9	18	27	4	3	7
	7	65	1/6	9	17	26	5	0.8	6
	10	63	1/10	9	18	27	3	1	4
	10	63	2/15	9	18	27	3	0.4	4
	10	63	1/6	9	18	27	4	1	5
168	5	120	1/10	17	36	53	30	17	47
	5	120	2/15	17	39	56	61	36	97
	5	120	1/6	17	37	54	50	12	62
	7	100	1/10	17	38	55	17	3	20
	7	100	2/15	17	38	55	15	0	15
	7	100	1/6	17	38	55	17	2	19
	10	88	1/10	20	43	63	5	3	8
	10	88	2/15	20	42	62	4	1	5
	10	88	1/6	20	44	64	5	2	7
224	5	126	1/10	17	41	58	105	100	205
	5	126	2/15	17	41	58	131	89	220
	5	126	1/6	17	41	58	86	239	325
	7	122	1/10	18	40	58	47	50	97
	7	122	2/15	17	49	66	58	258	316
	7	122	1/6	18	41	59	61	228	289
	10	118	1/10	18	41	59	40	24	64
	10	118	2/15	18	40	58	44	34	78
	10	118	1/6	18	41	59	44	91	135
226	5	114	1/10	15	34	49	38	73	111
	5	114	2/15	15	35	50	44	241	285
	5	114	1/6	15	36	51	60	68	128
	7	113	1/10	15	34	49	39	45	84
	7	113	2/15	15	37	52	51	72	123
	7	113	1/6	15	35	50	61	30	91
	10	113	1/10	15	34	49	39	45	84
	10	113	2/15	15	37	52	51	72	123
	10	113	1/6	15	35	50	61	30	91
238	5	133	1/10	21	49	70	65	185	250
	5	133	2/15	20	49	69	90	79	169
	5	133	1/6	20	48	68	99	22	121
	7	126	1/10	23	57	80	47	22	69
	7	126	2/15	22	54	76	52	22	74
	7	126	1/6	22	54	76	62	104	166
	10	125	1/10	22	56	78	41	111	152
	10	125	2/15	22	55	77	55	19	74
	10	125	1/6	22	54	76	51	67	118

Table 2 Comparing different branching approaches: Real data

Problem	Approach	vehicles	crews	total	columns	nodes	cpu
122	BP opt	9	17	26	6040	10	17
	BP sub	9	17	26	4397	4	15
	BB sub	9	17	26	3880	0	5
168	BP opt	17	38	55	6190	2	21
	BP sub	17	38	55	2395	0	2
	BB sub	17	38	55	3783	0	15
224	BP opt	18	39	57	10965	106	1884
	BP sub	18	39	57	5760	26	710
	BB sub	18	40	58	6457	0	97
226	BP opt	15	34	49	9869	326	5031
	BP sub	15	34	49	9680	688	11154
	BB sub	15	34	49	6558	30	84
238	BP opt	20	48	68	6250	48	1934
	BP sub	20	48	68	6909	62	2070
	BB sub	20	48	68	6250	0	121

obtained for problems with 80, 100, 160, 200, 320 and 400 trips. In the above referred site, data is grouped into two sets of problems, A and B. We only present results for problems of type A. Similar results were obtained for type B. We have considered vehicles located in four depots as in data from CARRIS (Sect. 6.1).

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

In Table 3 we report average results obtained with the branch-and-price (BP) and the branch-and-bound (BB) approaches. We have tested 10 instances of each problem size. For the 80 and 100 trip problems, we present the results obtained when we solved the pricing problem exactly (opt) and when we solve it approximately (sub). For problems with more than 100 trips, we only solved the pricing problem approximately. For problems with more than 160 trips, only results concerning the branch-and-bound are presented. For these size instances, the average CPU times for branch-and-price exceeded one day of computation time. In Table 3, columns 'vehicles', 'crews' and 'total' report, respectively, average number of vehicles, crews and sum of vehicles and crews in the solution. The columns titled 'columns' and 'nodes' display, respectively, the number of generated duty crews and the number of nodes of the branching tree. Finally, column 'cpu' shows the total computing time, measured in seconds.

The branch-and-price obtained slightly better solutions. In fact, the number of crews, the number of vehicles and the corresponding sum are slightly smaller for

Table 3 Comparing different branching approaches: Random data

Problem	Approach	vehicles	crews	total	columns	nodes	cpu
80A	BP opt	9.4	18.9	28.3	16724	30	4212
	BP sub	9.4	18.9	28.3	10289	27	2434
	BB opt	9.4	18.9	28.3	16545	7	203
	BB sub	9.4	19.3	28.7	8992	9	72
100A	BP opt	12.0	22.8	34.8	35196	34	12939
	BP sub	12.2	22.9	35.1	18795	52	4543
	BB opt	12.1	23.4	35.5	28166	7	802
	BB sub	12.1	23.4	35.5	17194	15	428
160A	BP sub	15.2	30.7	45.9	34864	58	74676
160A	BB sub	15.2	30.9	46.1	28617	18	2436
200A	BB sub	18.7	38.2	56.9	30720	46	3064
320A	BB sub	27.2	55.3	82.5	40557	127	11023
400A	BB sub	33.2	68.6	101.8	49848	91	13453

the branch-and-price approach. However, this improvement required a significant increase in CPU time (ranging from 30 times to almost 60 times the branch-and-bound time). The branch-and-price approach is suitable for handling small to medium sized instances, but consumed too much CPU time for the large sized instances. Concerning the performance of the branch-and-bound 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.

Table 4 depicts a comparison of the sub optimal pricing proposed algorithms with the method presented in Borndörfer et al. (2008), which, to our knowledge, was the best found in the literature for the same data instances. However, for instances with 80 trips, the results in Huisman et al. (2005) were similar to the ones of Borndörfer et al. (2008). In Gintner et al. (2008), although the authors solved the crew scheduling problem for the same test instances, they claim that their results are not comparable with those of the integrated problem as in Huisman et al. (2005). Therefore, they are also not comparable with ours. The results presented in Gintner et al. (2005) are also not comparable, because the authors considered a subset of the instances (100, 160 and 200 trips) with two depots, while ours refer to four depots.

Regarding the results in Borndörfer et al. (2008), our approaches have led to a smaller number of crews although, in some cases, a larger number of vehicles. However, we obtained smaller values concerning the sum of vehicles and crews. An important improvement over existing methods concerns, in our opinion, with the time consumed by the branch-and-bound approach (BB sub). A direct comparison can not be made due to differences in the computers used by the different authors. It is relevant to note that CPU times marked with '*' were obtained in a Dell Precision 650 PC with 4 GB of main memory and a dual Intel Xeon 3.0 Ghz CPU. However, we can state that when the size of the problem increases, the time spent by our branch-and-bound becomes significantly smaller than the time spent by the algorithm proposed in

Table 4 Comparing different methods

Problem	Approach	vehicles	crews	total	cpu
80A	BP sub	9.4	18.9	28.3	2434
	BB sub	9.4	19.3	28.7	72
	Borndörfer et al. (2008)	9.2	20.4	29.6	780 *
100A	BP sub	12.2	22.9	35.1	4543
	BB sub	12.1	23.4	35.5	428
	Borndörfer et al. (2008)	11.2	24.5	35.7	1260 *
160A	BP sub	15.2	30.7	45.9	74676
	BB sub	15.2	30.9	46.1	2436
	Borndörfer et al. (2008)	15.0	32.7	47.7	2640 *
200A	BB sub	18.7	38.2	56.9	3064
	Borndörfer et al. (2008)	18.5	40.5	59.0	5280 *
320A	BB sub	27.2	55.3	82.5	11023
	Borndörfer et al. (2008)	26.7	56.1	82.8	19680 *
400A	BB sub	33.2	68.6	101.8	13453
	Borndörfer et al. (2008)	33.1	69.8	102.0	43200 *

Borndörfer et al. (2008). From a transportation company point of view, an important feature of an algorithm concerns the ability of producing 'good' solutions in short time.

7 Conclusions

This paper focuses on the integrated vehicle and crew scheduling problem. The problem is described by an integer linear programming formulation that combines a multi-commodity flow model with a mixed set partitioning/covering model. Two main contributions have been presented. The first one concerns the proposal of a branch-and-price approach to the integrated problem. Another contribution is the evaluation of a specialized branch-and-bound algorithm, which has revealed a good trade-off between the quality of near-optimal solutions and computing time.

Three solution approaches were presented, compared and discussed: an exact branch-and-price algorithm, a heuristic branch-and-price algorithm and an approximated branch-and-bound algorithm. All of them rely on branching strategies along a search tree combined with column generation to solve the linear programming relaxation of the model. These approaches start with a preprocessing phase, where the set of tasks is defined.

On the heuristic approaches, the pricing problem is solved approximately. In such cases, the scheduler can manage given parameters to control the cardinality and some diversity of the set of columns, generated by pricing and further added to the master problem. For the exact branch-and-price, the subproblem is optimally solved and all the attractive columns are considered for possible introduction in the master.

Branch-and-bound only performs column generation at the root node, though producing a large and diverse set of columns over which branching is enforced afterwards over a single variable at each time. For both branch-and-price proposed algorithms branching concerns sum of variables and branching constraints are added explicitly to the master. Computational results have shown that exact pricing was not able to solve the linear programming relaxation of the "larger" sized problems, within reasonable computing time. Concerning the branch-and-bound performance, with a fine tuning of the parameters, this approach handled these "larger" sized problems, giving good quality feasible solutions in a short time.

Acknowledgements This work has been partially supported by POCTI-ISFL-1-152 and by POCI/MAT/57893/2004.

References

Borndörfer R, Löbel A, Weider S (2008) A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. In: Hickman M, Mirchandani P, Voß S (eds) Computer-aided systems in public transport. Lecture notes in economics and mathematical systems, vol 600. Springer, Berlin, pp 3–24

Gaffi A, Nonato M (1999) An integrated approach to ex-urban crew and vehicle scheduling. In: Wilson NHM (ed) Computer-aided transit scheduling. Lecture notes in economics and mathematical systems, vol 471. Springer, Berlin, pp 103–128

Gintner V, Steinzen I, Suhl L (2005) A variable fixing heuristic for the multiple-depot integrated vehicle and crew scheduling. In: Jaszkiewicz A, Kaczmarek M, Zak J, Kubiak M (eds) Advanced OR and AI methods in transportation. Publishing House of Poznan University of Technology, Poznan, pp 547–552

Gintner V, Kliewer N, Suhl L (2008) A crew scheduling approach for public transit enhanced with aspects from vehicle scheduling. In: Hickman M, Mirchandani P, Voß S (eds) Computer-aided systems in public transport. Lecture notes in economics and mathematical systems, vol 600. Springer, Berlin, pp 25–42

Haase K, Friberg C (1999) An exact branch and cut algorithm for the vehicle and crew scheduling. In: Wilson NHM (ed) Computer-aided transit scheduling. Lecture notes in economics and mathematical systems, vol 471. Springer, Berlin, pp 63–80

Haase K, Desaulniers G, Desrosiers J (2001) Simultaneous vehicle and crew scheduling in urban mass transit systems. Transp Sci 35(3):286–303

Huisman D, Freling R, Wagelmans APM (2005) Multiple-depot integrated vehicle and crew scheduling. Transp Sci 39:491–502

Mesquita M, Paixão J (1992) Multiple depot vehicle scheduling problem: a new heuristic based on quasi-assignment algorithms. In: Desrochers M, Rousseau JM (eds) Computer-aided transit scheduling. Lecture notes in economics and mathematical systems, vol 386. Springer, Berlin, pp 167–180

Mesquita M, Paias A (2008) Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. Comput Oper Res 35(5):1562–1575. (Available online 20 October 2006)

Valouxis C, Housos E (2002) Combined bus and driver scheduling. Comput Oper Res 29:243–259