Multiple-Depot Integrated Vehicle and Crew Scheduling

Dennis Huisman, Richard Freling*and Albert P.M. Wagelmans
Erasmus Center for Optimization in Public Transport (ECOPT) &
Econometric Institute, Erasmus University Rotterdam,
P.O. Box 1738, NL-3000 DR Rotterdam, The Netherlands
E-mail: huisman@few.eur.nl

Econometric Institute Report EI2003-02

Abstract

This paper presents two different models and algorithms for integrated vehicle and crew scheduling in the multiple-depot case. The algorithms are both based on a combination of column generation and Lagrangian relaxation.

Furthermore, we compare those integrated approaches with each other and with the traditional sequential one on random generated as well as real-world data instances for a suburban/extra-urban mass transit system. To simulate such a transit system, we propose a new way of generating randomly data instances such that their properties are the same as for our real-world instances.

1 Introduction

Vehicle and crew scheduling are two main problems arising in public transport scheduling. Mostly, these problems are considered separately, where first the vehicle scheduling problem and afterwards the crew scheduling problem is solved. In this paper we consider the suburban/extra-urban transit system with multiple depots, where we investigate the savings of using an integrated approach instead of a sequential one. It is generally expected that the savings of using an integrated approach in a suburban/extra-urban transit system are much more significant than in an urban mass transit system, since there are much less opportunities to relief one driver for another one in such a way that both drivers can enjoy their break or start/finish their duty. These

*In Memoriam: Richard Freling passed away on January 29, 2002 at the age of 34.

reliefs are only allowed at depots and certain other specified locations, which are much further away from each other than in the urban context. If first an optimal vehicle schedule is constructed, there can be vehicles which do not pass a relief location for hours. Therefore, it is very well possible that there does not exist a feasible crew schedule at all, or more probably, that the crew schedule will be very inefficient.

In this paper we extend the mathematical model and the solution approach which we developed for the single-depot case in Freling, Huisman and Wagelmans (2003) to the multiple-depot setting. This solution approach is based on Lagrangian relaxation in combination with column generation. The column generation is used to generate a set of duties, while Lagrangian relaxation is used to solve the master problem. Finally, Lagrangian heuristics are used to compute feasible solutions. Furthermore, we formulate another model which is an extension of the model for the single-depot case proposed by Haase, Desaulniers and Desrosiers (2001) and we show the relation between both models. We also develop an algorithm for this model, which is based on the same ideas as the algorithm for the first model. However, an important difference between the single-depot and multiple-depot case is that in the latter one the underlying vehicle scheduling problem is NP-hard (see Bertossi, Carraresi and Gallo (1987)), while in the former it can be solved in polynomial time. Of course, the underlying crew scheduling problem is NP-hard in both cases (see Fischetti, Martello and Toth (1989)).

Although a lot of attention has been paid to vehicle and crew scheduling in the literature, only a few papers consider complete integration of vehicle and crew scheduling. However, some papers deal with partial integration: vehicle scheduling during a heuristic approach to crew scheduling or inclusion of crew considerations in the vehicle scheduling process. For an overview on these papers, we refer to Freling, Huisman and Wagelmans (2003). Only very recently, complete integration of vehicle and crew scheduling has been proposed. The first mathematical formulation (for the single-depot case) is proposed by Patrikalakis and Xerocostas (1990), which is slightly changed by Freling, Boender and Paixão (1995). This latter formulation (see also Freling, Wagelmans and Paixão (1999)) is extended in this paper to the multiple-depot case.

In Freling, Huisman and Wagelmans (2003), we have made a comparison between sequential and integrated vehicle and crew scheduling for an urban mass transit system with a single depot. These results are very promising and show that drivers can be saved without using more vehicles. The benefits are especially large in the case that drivers are not allowed to change a vehicle. In the general case, when changeovers are allowed, only small savings can be obtained. Furthermore, we showed that a Lagrangian heuristic based on column generation can produce good solutions within reasonable computation times. In another paper (Freling, Huisman and Wagelmans (2001)),

we showed that this approach can also be applied to solve problems arising from practice, where significantly more complicating constraints are present than usually considered in the literature.

Haase and Friberg (1999) propose an exact algorithm for the single-depot vehicle and crew scheduling problem. Both the vehicle and crew scheduling aspects are modelled by using set partitioning type of constraints. A *branch-and-cut-and-price* algorithm is proposed, that is, column generation and cut generation are combined in a branch-and-bound algorithm. Computational results indicate that only small problems (up to 20 trips) could be solved.

Haase, Desaulniers and Desrosiers (2001) propose an approach which solves a crew scheduling problem that incorporates side constraints for the vehicles. This is done in such a way that the solution of this problem guarantees that an overall optimal solution is found after constructing a compatible vehicle schedule. The solution approach is based on a multi-commodity network flow formulation for the crew scheduling problem with side constraints, which is solved by a branch-and-price algorithm. Computational experiments with random data instances, simulating an urban mass transit environment, show that instances with up to 150 trips can be solved in 82 minutes of CPU time (on average on a SUN ULTRA-10/440 workstation) and with an optimality gap of 0.3% on average and always less than 1.2%. Furthermore, out of these 10 instances, 6 instances could be solved to optimality within 3 hours of CPU time. For larger problem instances a heuristic version of the algorithm is used. With this approach instances up to 350 trips can be solved within 2 hours of CPU time on average. The average (maximum) integrality gap for these instances is 0.3% (1.5%). Currently, they are working on an extension of their approach to the multiple-depot case (see Desaulniers, Cordeau and Desrosiers (2001)). In the airline world, a similar approach is used to integrate aircraft routing and crew scheduling (see Cordeau et al. (2001) and Klabjan et al. (2002)).

To the best of our knowledge, only one paper written by Gaffi and Nonato (1999), deals with integration in the multiple-depot case. Their approach is like ours based on Lagrangian relaxation with column generation. Their mathematical formulation is similar to one of the formulations presented in this paper and their approach is developed for the extra–urban mass transit setting, where crews are tightly dependent on the vehicle activities or dead-heading of crew is highly constrained. Since they consider a particular application, they make some assumptions which are not valid in general. These assumptions are that a driver is assigned to the same vehicle during the whole duty, and that all pieces of work (part of the duty between two breaks) start and end at a depot. Therefore, pieces of work and vehicle blocks coincide, which makes the problem computationally much more attractive than without these assumptions. The authors provide some computational results for Italian public transit operators, which show some improvements over the

results of a sequential approach. CPU times (on a Power PC 604, 180 Mhz) are over 24 hours for cases with up to 257 trips, and on average 2 to 6 hours. However, they do not compute lower bounds, which makes it difficult to give insight in the quality of their approach.

The main contribution of this paper is that the two proposed algorithms are very general (compared to Gaffi and Nonato (1999)) and can thus be used for different applications in the area of suburban and extra-urban mass transit systems. It is even possible to use them for urban transit systems, although there the benefits of using an integrated approach are much smaller as explained earlier. To test the algorithms, we have generated some random data instances for the suburban/extra-urban mass transit system, which are based on our experience with real-world problems. These instances are much more realistic for a suburban/extra-urban mass transit system than randomly generated test instances that have been proposed in the literature before. Furthermore, we compare the two algorithms and we also provide lower bounds for small and medium-sized problem instances. Although we were not able to compute lower bounds for large problem instances, the algorithms can still be used to get feasible solutions, which can be compared with the sequential approach.

The paper is organized as follows. In Section 2 we give a comprehensive problem definition and discuss some of the basic assumptions we make. In Sections 3 and 4, we discuss a mathematical model and an algorithm for the integrated multiple-depot vehicle and crew scheduling problem, respectively. The second formulation and algorithm is given in Section 5. Finally, we conclude the paper with some computational results on real-world and randomly generated data instances in Section 6.

2 Problem Definition

The *multiple-depot vehicle and crew scheduling problem* (MD-VCSP) combines the *multiple-depot vehicle scheduling problem* (MDVSP) and the *crew scheduling problem* (CSP). Given a set of *trips* within a fixed planning horizon, it minimizes the total sum of vehicle and crew costs such that both the vehicle and the crew schedule are feasible and mutually compatible. Each trip has fixed starting and ending times and can be assigned to a vehicle and a crew member from a certain set of depots. Of course, if every trip can only be assigned to a vehicle and a crew member from one depot, the problem decomposes to a number of single-depot vehicle and crew scheduling problems. Furthermore, the traveling times between all pairs of locations are known.

A vehicle schedule is feasible if:

- all trips are assigned to exactly one vehicle;

- each trip is assigned to a vehicle from a depot that is allowed to drive this trip.

The vehicle costs consist of a fixed component for every vehicle and variable costs for idle and travel time. It is allowed that a vehicle returns to a depot between two trips if there is enough time to do this.

From a vehicle schedule it follows which trips have to be performed by the same vehicle and this defines so-called vehicle *blocks*. The blocks are subdivided at *relief points*, defined by location and time, where and when a change of driver may occur and drivers can enjoy their break. A *task* is defined by two consecutive relief points and represents the minimum portion of work that can be assigned to a crew. These tasks have to be assigned to crew members. The tasks that are assigned to the same crew member define a crew *duty*. Together the duties constitute a crew schedule. Such a schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicating constraints corresponding to work load regulations for crews. Typical examples of such constraints are maximum working time without a break, minimum break duration, maximum total working time, and maximum duration. These constraints can differ between different types of duties, e.g., early, split and late duties. The cost of a duty is usually a combination of fixed costs such as wages, and variable costs such as overtime payment. Finally, a *piece (of work)* is defined as a sequence of tasks on one vehicle block without a break that can be performed by a single crew member without interruption.

We make five assumptions, which are discussed one by one below.

1. Each vehicle has its own depot, which means that a vehicle starts and ends in the same depot. The number of vehicles used per depot is unlimited.

2. All crew have their own depot, which means that a duty of a single crew member has only tasks on vehicles from that depot. However, it is not necessary that every duty starts and ends in this depot.

3. The feasibility of a piece only depends on its duration of this sequence, which is limited by a minimum and maximum piece length.

4. There is *continuous attendance*, i.e., there is always a driver present if the bus is outside the depot. However, vehicle attendance at the depot is not necessary.

5. *Changeovers*, which is the change of vehicle of a driver during his break, are allowed.

The last two assumptions imply that if a driver has no changeover, i.e. before and after the break he drives the same vehicle, there should be another driver on this vehicle during the break of the former driver.

We distinguish between two types of tasks, viz., *trip tasks* corresponding to trips, and *dh-tasks* corresponding to deadheading. A *deadhead* is a period that a vehicle is moving to or from the depot, or a period between two trips that a vehicle is outside of the depot (possibly moving without passengers). All trip tasks need to be covered by a crew member, while the covering of dh-tasks depends on the vehicle schedules and determines the compatibility between vehicle and crew schedules. In particular, each dh-task needs to be assigned to a crew if and only if its corresponding deadhead is assigned to a vehicle.

3 Mathematical Formulation

In this section, we propose a mathematical formulation for the MD-VCSP under the assumptions stated in Section 2.

Let $N = \{1, 2, ..., n\}$ be the set of trips, numbered according to increasing starting time, and let $E = \{(i, j) \mid i < j,\ i, j \text{ compatible},\ i \in N, j \in N\}$ be the set of deadheads. Define D as the set of depots and let s^d and t^d both represent depot d. We define the vehicle scheduling network $G^d = (V^d, A^d)$, which is an acyclic directed network with nodes $V^d = N^d \cup \{s^d, t^d\}$, and arcs $A^d = E^d \cup (s^d \times N^d) \cup (N^d \times t^d)$. Note that N^d and E^d are the parts of N and E corresponding to depot d, since it is not necessary that all trips can be served from every depot. Let c^d_{ij} be the vehicle cost of arc $(i, j) \in A^d$, which is usually some function of travel and idle time. Furthermore, a fixed cost for using a vehicle can be added to the cost of arcs (s^d, i) or (j, t^d) for all $i, j \in N^d$.

To reduce the number of constraints, we assume that a vehicle returns to the depot if it has an idle time between two consecutive trips which is long enough to let it return. In that case the arc between the trips is called a *long arc*; the other arcs between trips are called *short arcs*. Denote $A^{sd} \subset A^d$ and $A^{ld} \subset A^d$ as the sets of short and long arcs, respectively.

Furthermore, K^d denotes the set of duties corresponding to depot d and f^d_k denote the crew cost of duty $k \in K^d$, respectively. We assume that deadheads to and from the depot correspond to one dh-task each. Suppose $(i, j) \in A^{ld}$ and let el_i and bl_j denote the ending and starting location of trips i and j, respectively. Then we let $K^d(i, t^d)$ and $K^d(s^d, j)$ denote the set of duties covering dh-task from el_i to depot d and from depot d to bl_j, respectively. Furthermore, $K^d(i)$ denotes the set of duties covering the trip task corresponding to trip $i \in N^d$, which means that we assume that a trip corresponds to exactly one task. $K^d(i, j)$ denotes the set of duties covering

dh-tasks corresponding to deadhead $(i,j) \in A^{sd}$. Decision variable y_{ij}^d indicates whether an arc (i,j) is used and assigned to depot d or not, while x_k^d indicates whether duty k corresponding to depot d is selected in the solution or not. The MD-VCSP can be formulated as follows.

(MD-VCSP1):

$$\min \sum_{d\in D} \sum_{(i,j)\in A^d} c_{ij}^d y_{ij}^d + \sum_{d\in D} \sum_{k\in K^d} f_k^d x_k^d \tag{1}$$

$$\sum_{d\in D} \sum_{\{j:(i,j)\in A^d\}} y_{ij}^d = 1 \qquad \forall i \in N, \tag{2}$$

$$\sum_{d\in D} \sum_{\{i:(i,j)\in A^d\}} y_{ij}^d = 1 \qquad \forall j \in N, \tag{3}$$

$$\sum_{\{i:(i,j)\in A^d\}} y_{ij}^d - \sum_{\{i:(j,i)\in A^d\}} y_{ji}^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{4}$$

$$\sum_{k\in K^d(i)} x_k^d - \sum_{\{j:(i,j)\in A^d\}} y_{ij}^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{5}$$

$$\sum_{k\in K^d(i,j)} x_k^d - y_{ij}^d = 0 \qquad \forall d \in D, \forall (i,j) \in A^{sd}, \tag{6}$$

$$\sum_{k\in K^d\left(i,t^d\right)} x_k^d - y_{it^d}^d - \sum_{\{j:(i,j)\in A^{ld}\}} y_{ij}^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{7}$$

$$\sum_{k\in K^d\left(s^d,j\right)} x_k^d - y_{s^dj}^d - \sum_{\{i:(i,j)\in A^{ld}\}} y_{ij}^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{8}$$

$$x_k^d, y_{ij}^d \in \{0,1\} \qquad \forall d \in D, \forall k \in K^d, \forall (i,j) \in A^d \tag{9}$$

The objective is to minimize the sum of total vehicle and crew costs. The first three sets of constraints, (2)-(4), correspond to the formulation for the MDVSP. In fact, it is not necessary to have both constraints (2) and constraints (3). However, it is useful to have both sets of constraints when we relax constraints (4), as will be done in the next section. Constraints (5) assure that each trip task will be covered by a duty from a depot if and only if the corresponding trip is assigned to this depot. Furthermore, constraints (6), (7) and (8) guarantee the link between dh-tasks and deadheads in the solution, where deadheads corresponding to short and long arcs in A^d are considered separately. In particular, constraints (6) guarantee that each deadhead from i to j is covered by a duty in the set $K^d(i,j)$ if and only if the corresponding short arc is in the vehicle solution. The other two constraint sets, (7) and (8), ensure that the dh-tasks from el_i to t^d and from s^d to bl_j, possibly corresponding to long arc $(i,j) \in A^d$, are both covered by one duty if and only if the corresponding deadheads are in the solution. Note that the structure of these last three sets of constraints is such that each constraint corresponds to the possible selection of one duty from a large set of duties,

where the fact whether or not a duty has to be selected depends on the values of the corresponding y variables. We will refer to these constraints as *partitioning type of constraints.*

4 Algorithm

The algorithm we propose to solve model MD-VCSP1, is a combination of column generation and Lagrangian relaxation and is an extension of the algorithm proposed by us for the single-depot case (see Freling, Huisman and Wagelmans). An outline of the algorithm is shown in Figure 1.

Step 0: Initialization
Solve MDVSP and CSP for every depot and
take as initial set of columns the duties in the CSP-solution.
Step 1: Computation of dual multipliers
Solve a Lagrangian dual problem with the current set of columns.
This gives a lower bound for the current set of columns.
Step 2: Deletion of columns
If there are more columns than a certain minimum amount, then delete columns with positive reduced cost greater than a certain treshold value.
Step 3: Generation of columns
Generate columns with negative reduced cost.
Compute an estimate of a lower bound for the overall problem.
If the gap between this estimate and the lower bound found in Step 1 is small enough (or another termination criterion is satisfied), go to Step 4; otherwise, return to Step 1.
Step 4: Construction of feasible solution
Solve a second Lagrangian dual problem with the set of columns generated in Step 3, where the optimal solution of the subproblem gives feasible vehicle schedules. Solve for each depot the crew scheduling problem corresponding to the feasible vehicle schedules.

Figure 1: Solution method for MD-VCSP1

First, we compute a feasible solution by using the sequential approach, which means we compute the optimal solution of the MDVSP and afterwards, we solve for each depot a CSP given the vehicle schedule for that depot. To solve the MDVSP, we use the model described in Huisman, Freling and Wagelmans (2001) and the all-purpose solver CPLEX. The approach we used to solve the CSP, is described in Freling, Huisman and Wagelmans (2003).

The main part of the algorithm is used to compute a lower bound and we use therefore a column generation algorithm. Traditionally, the *master problem* in a column generation algorithm is solved with LP-relaxation. However,

we found in Freling, Huisman and Wagelmans (2003) very promising results of solving the master problem with Lagrangian Relaxation. Therefore, we use this approach again. The details will be discussed in Subsection 4.1.

Furthermore, we generate the duties in the column generation subproblem (*pricing problem*), which is the topic of Subsection 4.2. Since we do not want to get a very large master problem, columns with high positive reduced costs will be removed. This only happens if there are more columns than a certain minimum number. The deletion of columns is an important difference with the algorithm for the single-depot case, where it was not necessary to delete columns, since the resulting Lagrangian subproblem could be solved faster and the number of generated columns was lower.

Finally, in Step 4 we compute feasible solutions, which will be discussed in detail in Subsection 4.3.

4.1 The Master Problem

To solve the master problem, we use the relaxation of model MD-VCSP1, where the equality signs in the constraints (4)-(8) are first replaced by "greater-than-or-equal" signs, which are subsequently relaxed in a Lagrangian way. That is, we associate non-negative Lagrangian multipliers κ_j^d, λ_i^d, μ_{ij}^d, ν_i^d and ξ_j^d with constraints (4), (5), (6), (7) and (8), respectively. Then the remaining Lagrangian subproblem can be solved by pricing out the x variables and solving a large *single-depot vehicle scheduling problem* (SDVSP) for the y variables.

Furthermore, we need an additional procedure to update the Lagrangian multipliers after solving the Lagrangian relaxation. This is necessary to assure that all duties in the current master problem have non-negative reduced costs so that these duties will not be generated again in the pricing problem. For more details on this procedure, we refer to Freling (1997) and Carraresi, Girardi and Nonato (1995).

An alternative and slightly different approach is to relax constraints (2) instead of (4). Constraints (3) are redundant in this approach. Then the remaining Lagrangian subproblem for the y variables corresponds with solving $|D|$ small SDVSP's instead of a large one. We have compared both approaches and concluded that the initial approach gives slightly better computation times to get the same lower bound (viz., the value of the LP lower bound). Therefore, in the remaining of the paper, we do not consider the alternative again.

4.2 The Column Generation Subproblem

For the MD-VCSP, vehicle blocks are not known and a huge number of feasible pieces of work may exist. Therefore, we propose a two phase procedure

for the column generation pricing problem: in the first phase, a *piece generation network* is used to generate a set of pieces of work which serves as input for the second phase where duties are generated. Since there is no interaction between the different depots in the column generation subproblem, we can solve them separately for every depot.

In every iteration i we compute an estimate LBT_i of the lower bound for the overall problem. Let LBS_i denote the value of the Lagrangian lower bound in iteration i, then the estimate is computed as

$$LBT_i = LBS_i + \sum_{d \in D} \sum_{k \in K_i^d} \overline{f}_k^d, \tag{10}$$

where K_i^d is the set of duties added in iteration i and $\overline{f}_k^d$ is the reduced cost of duty $k \in K^d$, which is defined below.

$$\overline{f}_k^d = f_k^d - \sum_{i \in N(k,d)} \lambda_i^d - \sum_{(i,j) \in A^s(k,d)} \mu_{ij}^d - \sum_{i \in N^t(k,d)} \nu_i^d - \sum_{j \in N^s(k,d)} \xi_j^d, \tag{11}$$

where
$N(k,d)$: set of trips in duty k from depot d,
$A^s(k,d)$: set of short arcs A^s in duty k from depot d,
$N^t(k,d)$: set of trips that have a corresponding task in duty k from depot d with the end of this trip as starting location and the depot as ending location,
$N^s(k,d)$: set of trips that have a corresponding task in duty k from depot d with the depot as starting location and the start of this trip as ending location.

LBT_i is a lower bound for the overall problem if all duties with negative reduced cost are added to the master problem. We terminate Step 3 if the relative difference between LBT_i and LBS_i is small or if a maximum computation time is reached.

4.2.1 Generation of Pieces of Work

Recall that we have defined a piece of work as a continuous sequence of trip tasks and dh-tasks corresponding to (a part of) one vehicle block, and that this sequence of tasks is only restricted by its duration. The network for piece generation is an extension of the network G^d for vehicle scheduling (see Section 3). Let a *start point* (*end point*) be defined as the relief point corresponding to the start (end) of a vehicle trip. We define the network $G^{d\prime} = (V^{d\prime}, A^{d\prime})$, where nodes correspond to the relief points on each trip that can be assigned to a vehicle from depot d, and the source s^d and the sink t^d represent the depot. Arcs in $A^{d\prime}$ correspond to dh-tasks and trip tasks. Notice that $G^{d\prime}$ is acyclic.

Let b_{mn}^d be the cost associated with each arc $(m,n) \in A^{d\prime}$. Recall from Section 4.1 that we associate Lagrangian multipliers λ_i^d, μ_{ij}^d, ν_i^d and ξ_j^d with

constraints (5), (6), (7) and (8), respectively. The reduced cost is then defined as

$$\bar{b}^d_{mn} = \begin{cases} b^d_{mn} - \lambda^d_i, & \text{for each arc } (m,n) \text{ with } m \text{ the start point and } n \text{ the end point of trip } i, \\ b^d_{mn} - \mu^d_{ij}, & \text{for each arc } (m,n) \text{ with } m \text{ the end point of trip } i \text{ and } n \text{ the start point of trip } j, \\ b^d_{mn} - \xi^d_j, & \text{for each arc } (m,n) \text{ with } m = s^d \text{ and } n \text{ the start point of trip } j, \\ b^d_{mn} - \nu^d_i, & \text{for each arc } (m,n) \text{ with } n = t^d \text{ and } m \text{ the end point of trip } i. \end{cases}$$

Thus, the reduced costs on the arcs are defined such that the reduced cost of a path is equal to the reduced cost of the corresponding piece of work. Each path between two nodes u and v in network $G^{d\prime}$ corresponds to a feasible piece of work if its duration is between the minimum and maximum allowed duration of a piece of work. However, it is not necessary to generate all pieces, since we only have to satisfy the column generation optimality condition, that is, there are no duties left with negative reduced costs. The sufficient subset is generated by solving a shortest path problem between each pair of nodes in network $G^{d\prime}$ that satisfy the constraint on the piece duration. For all feasible paths from u to v, three additional paths are considered, namely $s^d, u, ..., v,\ u, ..., v, t^d$ and $s^d, u, ..., v, t^d$. It is easy to see that by generating only this subset of pieces, we assure that the column generation optimality condition is satisfied.

4.2.2 Generation of Duties

Duties have to satisfy certain feasibility conditions. In particular, they consist of a maximum number of pieces. In our case this maximum is equal to 2. This is the reason why we simply enumerate all possible combinations of pieces and check if such a combination is feasible, until we find a specified number of duties with negative reduced costs (or all combinations have been checked). The reduced cost of a duty can be easily computed when the reduced cost of a piece is already known: the reduced cost of a duty is equal to the sum of the reduced cost of the pieces it is built from under the assumption of continuous attendance (as described in Section 2).

4.3 Feasible Solutions

At the end, in Step 4, we only relax constraints (5)-(8), which is again done in a Lagrangian way. Therefore, the solution of the remaining subproblem gives

a feasible vehicle schedule. Notice, that this subproblem is a MDVSP, which is an NP-hard problem. However, we need to solve only a few iterations of the subgradient algorithm to get good solutions, since we start with already good multipliers (the best one from the last iteration in Step 1). After that, for every depot, we compute feasible crew schedules given these (feasible) vehicle schedules. We do this by solving the CSP in the same way as in the initial step. Of course, it is also possible to compute more feasible solutions by solving the CSP not only for the vehicle solution from the last iteration, but also for vehicle solutions which were encountered earlier on. A reason to actually do this could be that the gap between the lower and upper bound is quite large, which is an indication that the upper bound could be improved upon.

5 Alternative Approach

In this section we propose another mathematical formulation for the MD-VCSP which has only variables related to crew duties. The vehicle schedule can be obtained implicitly from the crew schedule. This formulation can be derived from the one previously presented in this paper, but is also equivalent to the formulation of Haase, Desaulniers and Desrosiers (2001) in the case of a single depot. An alternative algorithm based on this model is suggested in Subsection 5.2.

5.1 Mathematical Formulation

The alternative mathematical formulation for the MD-VCSP can be obtained from model (MD-VCSP1) by substituting for the y variables using constraints (6), (7) and (8). The problem can then be formulated as follows.

(MD-VCSP2):

$$\min \sum_{d\in D}\sum_{k\in K^d} g_k^d x_k^d \tag{12}$$

$$\sum_{d\in D}\sum_{\{j:(i,j)\in A^d\}}\sum_{k\in K^d(i,j)} x_k^d = 1 \qquad \forall i \in N, \tag{13}$$

$$\sum_{\{i:(i,j)\in A^d\}}\sum_{k\in K^d(i,j)} x_k^d - \sum_{\{i:(j,i)\in A^d\}}\sum_{k\in K^d(j,i)} x_k^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{14}$$

$$\sum_{k\in K^d(i)} x_k^d - \sum_{\{j:(i,j)\in A^d\}}\sum_{k\in K^d(i,j)} x_k^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{15}$$

$$x_k^d \in \{0,1\} \qquad \forall d \in D, \forall k \in K^d. \tag{16}$$

In this formulation g_k^d is the sum of the costs of duty $k \in K^d$ and the variable vehicle costs corresponding to the arcs in this duty. However, in the

proposed formulation we cannot deal with fixed vehicle costs. We can only introduce them by adding an extra decision variable B to count the number of vehicles and by adding the following set of constraints:

$$\sum_{d \in D} \sum_{k \in K^d(h)} x_k^d \leq B \quad \forall h \in H, \tag{17}$$

where $K^d(h)$ is the set of duties corresponding to depot d where time point h is between the start and end time of one of the tasks of this duty. H is defined as the set of time points at which a vehicle may leave a depot to drive to the start location of a trip, i.e., the start time of the trip minus the driving time from the depot to the start location. It suffices to consider only these time points, since only at these time points the number of vehicles can increase, i.e., a departure may occur. Moreover, if there are two consecutive time points in H between which no arrival at a depot can occur, then the number of vehicles at the latest time point is at least the number of vehicles at the earlier one. This means that the constraint for the earlier time point can be left out.

The main advantage of the formulation above compared to the one in Section 3 is that the number of constraints is much less. In the case that $|N^d| = N$ and $|A^{sd}| = A^s, \forall d \in D$, the number of constraints reduces from $(4|D| + 2)|N| + |D||A^s|$ to $(2|D| + 1)|N| + |H|$. However, we do not have the vehicle schedules explicitly in the model anymore, which means it is less straightforward how to construct feasible solutions by using a Lagrangian heuristic. Therefore, we propose an algorithm that consists of two phases, in which we use both formulations presented in this paper.

5.2 Algorithm

In this subsection we discuss a second algorithm to solve the MD-VCSP which consists of two phases. In the first phase, we compute a lower bound using model MD-VCSP2 by again combining column generation and Lagrangian relaxation. We use the columns generated during the first phase in the second one to find a feasible vehicle schedule and a corresponding crew schedule. The second phase is similar to Step 4 of the previous algorithm described in Section 4. The important differences are thus in computing the lower bound, where we use model MD-VCSP2 instead of MD-VCSP1. These differences are described below.

To compute a lower bound, we relax the constraints (13)-(15) in a Lagrangian way and we do not take the constraints (17) into account, which means we do not consider fixed vehicle costs explicitly. However, we can easily get a lower bound on these costs, since an optimal solution of the MDVSP with only fixed costs, is a lower bound on the fixed vehicle costs. A lower bound for the total problem is thus given by the sum of the Lagrangian lower

bound and the optimal solution of the MDVSP with only fixed costs. Furthermore, we compute given a set of multipliers a new lower bound where we include one of the constraints (17). If we find an improvement, then we use the subgradient algorithm to get a better lower bound. Otherwise, we include the next constraint. In this way, we can subsequently improve the lower bound, since most of the constraints (17) are not binding at all. However, we cannot guarantee that we find the best lower bound in this way.

Notice that we include only one of the constraints (17) at a time, since the Lagrangian subproblem can still be solved in polynomial time. In the case we do not include any of these constraints, the subproblem can even be solved by only pricing out the x variables.

Another important difference compared to the first algorithm is in the definition of the reduced costs of the arcs in the piece generation network. Let μ_i, υ_j^d and λ_i^d be the Lagrangian multiplier corresponding to constraints (13), (14) and (15) in model MD-VCSP2, respectively. The reduced cost of the arcs are then defined as follows:

$$\overline{b}_{mn}^d = \begin{cases} b_{mn}^d - \lambda_i^d, & \text{for each arc } (m,n) \text{ with } m \text{ the start point and } n \text{ the endpoint of trip } i, \\ b_{mn}^d - \mu_i - \upsilon_i^d + \upsilon_j^d, & \text{for each arc } (m,n) \text{ with } m \text{ the end point of trip } i \text{ and } n \text{ the start point of trip } j, \\ b_{mn}^d + \upsilon_j^d, & \text{for each arc } (m,n) \text{ with } m = s^d \text{ and } n \text{ the start point of trip } j, \\ b_{mn}^d - \mu_i - \upsilon_i^d, & \text{for each arc } (m,n) \text{ with } n = t^d \text{ and } m \text{ the end point of trip } i. \end{cases}$$

However, notice that we can still use the all-pairs shortest path algorithm for the generation of the pieces as described in Subsection 4.2.1.

6 Computational Results

In this section we test our algorithms on some real-life datasets from Connexxion and on some random data instances. All tests are executed on a Pentium III 450MHz personal computer (128MB RAM) with the following parameter settings.

1. The objective is to minimize the total sum of vehicles and drivers. For solving the MDVSP in the sequential approach and in the initial step for the integrated approach we use an additional fictitious cost in the variable vehicle costs, viz., for every minute a vehicle is empty outside

the depot a cost equal to 1 is incurred. This is necessary to make a fair comparison between a sequential and an integrated approach.

2. The pricing problems are solved independently for each depot and each type of duty. Moreover, we generate at most 1500 duties for each combination of a depot and type of duty.

3. The maximum number of iterations in the subgradient algorithm to solve the master problem (Step 1) is $500 + 3k$ in the k-th iteration of the column generation algorithm. However, for constructing the feasible solutions in Step 4, the number of iterations is only 10, since in that case the subproblem is NP-hard. Such a small number of iterations is sufficient, since we already start with good multipliers, namely the best ones of the last iteration in the previous step. We construct 10 feasible solutions from which the best one will be selected.

4. The column generation algorithm is stopped if the difference between the current and estimated lower bound is smaller than 0.1% or if the computation time of the lower bound phase is more than 3 hours. Notice that in the latter case we do not have a proven lower bound.

In Subsection 6.1 we describe some properties of the real-world data instances. The results can be found in Subsection 6.2. Furthermore, we propose a new way of generating random data instances to simulate problem instances for an extra-urban bus network in Subsection 6.3. We choose for a different way of generating these random instances, since generators defined in the literature before (see e.g. Dell'Amico, Fischetti and Toth (1993), Freling (1997) and Haase, Desaulniers and Desrosiers (2001)) do not take into account specific properties of an extra-urban bus network. First of all, they choose to take the start times of the trips completely random, which means that the intervals between two trips can vary a lot, e.g., between the first two trips it may be five minutes, while it can be two hours between the second and the third trip. However, in the real world these frequencies are fixed to some extent, e.g., during the peak hours it is 20 minutes and outside the peak hours 30 minutes. Secondly, they have very long travel times such that vehicles can only drive a few trips, while in our real-world instances a vehicle drives about 10 trips on average.

6.1 Properties of the real-world data instances

We test our algorithms on some subsets of a large dataset from Connexxion, which is the largest bus company in the Netherlands. The total set consists of 1104 trips and 4 depots in the area between Rotterdam, Utrecht and Dordrecht, three large cities in the Netherlands. However, it is important to

note that not all trips are allowed to be driven by a vehicle from every depot. In fact, almost half of the trips can only be assigned to one depot and only a very small number can be assigned to all depots. On average, a trip can be assigned to 1.71 depots.

The restrictions that we have taken into account, are as follows. A driver can only be relieved by another driver at the start or end of a trip at certain specified locations or at the depot. There are 8 of these locations, which are all major bus stations. If a driver starts/ends his duty at the depot, there is a sign-on/sign-off time of 10 and 5 minutes, respectively. If a driver starts/ends his duty at another relief location, an extra time of 15 minutes plus the deadhead time between this location and the depot is added to the length of the duty. There are five different types of duties, one tripper type consisting of one piece with a length between 30 minutes and 5 hours, and four normal types consisting of two pieces with the properties described in Table 1.

type	1 (early)		2 (day)		3 (late)		4 (split)	
	min	max	min	max	min	max	min	max
start time			8:00		13:15			
end time		15:30		18:14				19:30
piece length	0:30	5:00	0:30	5:00	0:30	5:00	0:30	5:00
break length	0:45		0:45		0:45		1:30	
duty length		9:45		9:45		9:45		12:00
work time		9:00		9:00		9:00		9:00

Table 1: Properties of the different duty types

6.2 Results real-world data instances

We consider 8 different problem instances for which the number of trips varies between 194 and 653 trips, which have been derived from the large set described in the previous subsection. In Table 2 an overview of the results of the different algorithms is provided for these test problems. For each instance, we give the number of trips and the average number of depots to which a trip may be assigned. Furthermore, we give the number of vehicles, drivers and the total sum of these two for the sequential approach and the two integrated approaches presented earlier in this paper (Section 4 and 5, respectively). Finally, we report the best lower bound given by the two algorithms.

As can be seen from Table 2 both integrated approaches give much better solutions than the sequential one. The number of vehicles does not change, while the number of drivers is reduced significantly. The largest relative improvement is obtained for instance 3, where both algorithms save 10 out

	instance	1	2	3	4	5	6	7	8
	trips	194	210	220	237	304	386	451	653
	depots/trip	1.60	2.47	1.52	2.38	2.48	1.27	1.67	1.74
	vehicles	19	33	27	34	40	32	47	67
seq.	drivers	33	54	51	62	74	59	86	123
	total	52	87	78	96	114	91	133	190
	vehicles	19	33	27	34	40	32	47	67
int. 1	drivers	30	50	41	55	65	58	77	117
	total	49	83	68	89	105	90	124	184
	vehicles	19	33	27	34	40	32	47	67
int. 2	drivers	28	50	41	54	67	58	77	117
	total	47	83	68	88	107	90	124	184
	lower	44*	77	64*	81	95*	-	-	-

Table 2: Results Connexxion data instances

of 51 drivers. Furthermore, we can see that it is difficult to conclude which of the algorithms for the integrated approach is better. Sometimes the first one gives a better solution and sometimes the second one.

We were only able to compute a lower bound for two of these instances given the maximum computation time of 3 hours for the lower bound phase, namely for instances 2 and 4. For instances 1, 3 and 5 the lower bound has been computed without taking a maximum computation time as stop criterion into consideration. For all instances, the best lower bound is obtained by the first algorithm. The computation times to compute these lower bounds vary a lot, e.g., for instance 1 it takes almost 6 hours while for instance 2 it takes only 45 minutes (both for the first algorithm), although instance 2 has more trips and the average of number depots per trip is significantly higher. This difference can be explained by the completely different structure of these instances, namely that the average length of the trips in instance 2 is much higher than in instance 1. This can also be seen from the table, since the number of vehicles and drivers is much lower for instance 1 than for instance 2.

6.3 Generation of random data instances

In this subsection, we give a detailed description of the way the random data instances have been generated. These instances are available at the web page *http://www.few.eur.nl/few/people/huisman/instances.htm*.

The coordinates of the different locations, either depots or start/end points of the lines, are integers generated from a uniform distribution in a 50 by 50 kilometers square. However, there is a minimum distance between each pair of depots and between each pair of start/end points of 10

kilometers.

We consider two different types of instances, which vary in the travel speed. If the travel speed is lower, trips are longer and therefore, less trips will be assigned to one vehicle as well as to one driver. For each of the two types, we have six cases, three in which we have four lines (from A to B, C and D and from B to C) and again three with five lines (the same lines as in the first case plus a line from C to E). All lines are driven in both directions and have the same frequency. Furthermore, we define four different intervals with respect to frequency and travel speed. In Table 3, we denote these frequencies (in the case of 10, 20 and 40 trips per line/direction, respectively) and travel speeds (for type A as well as B) for the different intervals.

interval	frequency (min.)			speed (kms/hour)	
	10-trips	20-trips	40-trips	type A	type B
06:00 - 08:59	80	40	20	28	20
09:00 - 12:59	120	60	30	32	24
13:00 - 18:59	80	40	20	30	23
19:00 - 23:59	240	120	60	35	26

Table 3: Frequencies and travel speeds per interval

The start time of the first trip for each line/direction is uniformly drawn between 06:00 and 07:19, between 06:00 and 06:39 and between 06:00 and 06:19, respectively. The end times are computed as the start time plus the travel time between the locations rounded up to the nearest integer. The travel speed for deadhead trips is 50 kms/hour. We choose this significantly higher than the operational travel times since the bus does not have to stop for passengers entering or leaving, which means the shortest route can be taken.

Finally, we have to choose the relief locations, where a driver can take a break and one driver can be replaced by another one. These locations need some relief facilities like a canteen to take a meal break. It is most likely that these facilities are at the start and end points of the lines. Therefore, we choose A, B, C and D as relief locations in the cases with four lines and these locations plus E in the other ones. We use the same restrictions with respect to the feasibility of the duties as before (see Subsection 6.1).

Notice that in contrary to the real-world instances we assume that vehicles from each depot can carry out all trips.

6.4 Results random data

We have tested our algorithms by generating 10 random instances for the six different cases described in Subsection 6.3 (10, 20 and 40 trips per line/direction with 4 and 5 lines). As a consequence, the total number of trips varies from

80 to 400. These tests are executed with 2 as well as 4 depots. However, with 4 depots we do not consider the two largest cases (with 320 and 400 trips, respectively).

In Tables 4 and 5 we give an overview of the results for instances of type A with 2 and 4 depots, respectively. We give the solution of the traditional sequential approach and of the two integrated approaches described in the Sections 4 and 5, respectively.

	trips	80	100	160	200	320	400
	vehicles	9.2	11.0	14.8	18.4	26.7	32.9
seq.	drivers	23.8	29.0	35.9	44.5	60.8	74.9
	total	33.0	40.0	50.7	62.9	87.5	107.8
	vehicles	9.2	11.0	14.8	18.4	26.7	32.9
int. 1	drivers	20.6	24.8	33.5	40.7	60.1	73.2
	total	29.8	35.8	48.3	59.1	86.8	106.1
	vehicles	9.2	11.0	14.8	18.4	26.7	32.9
int. 2	drivers	20.6	24.6	33.5	41.0	60.0	74.2
	total	29.8	35.6	48.3	59.4	86.7	107.1

Table 4: Average results random data instances - 2 depots - type A

	trips	80	100	160	200
	vehicles	9.2	11.0	14.8	18.4
seq.	drivers	25.8	29.9	38.8	47.1
	total	35.0	40.9	53.6	65.5
	vehicles	9.2	11.0	14.8	18.4
int. 1	drivers	20.5	25.3	34.1	41.6
	total	29.7	36.3	48.9	60.0
	vehicles	9.2	11.0	14.8	18.4
int. 2	drivers	20.4	25.2	34.7	42.0
	total	29.6	36.2	49.5	60.4

Table 5: Average results random data instances - 4 depots - type A

As can been seen from Tables 4 and 5, the total sum of vehicles and drivers can be reduced significantly by using an integrated approach. Furthermore, the difference between the two algorithms is quite small on average for problem instances up to 320 trips. Only in the case of 400 trips (2 depots) and 160 and 200 trips (4 depots), the first algorithm performs significantly better than the second one. Since the maximum computation time for the lower bound phase is fixed, we conclude that only the first algorithm finds good feasible solutions after a few column generation iterations. Finally, notice that the number of drivers in most cases with 4 depots is more than in

the corresponding case with 2 depots. This is conspicuous since all data are the same except the fact that there are two extra depots, which means that the solution in the 2 depot case is also a feasible one in the 4 depot case. Therefore, we can conclude that our algorithms perform worse if there are more depots.

In Tables 6 and 7 we present detailed results for both algorithms for 2 and 4 depots, respectively. In the upper part of the tables we give some statistics with respect to the first algorithm. We denote the average number of iterations of the column generation algorithm and the average computation times for the master problem and the pricing problem, respectively. Furthermore, we give the total average computation time for computing the lower bound and the average computation time for solving the last subgradient algorithm in Step 4. These averages are computed over the instances for which a lower bound is found. Therefore, we also denote the number of instances (out of 10) for which we actually found a lower bound. In the second part of the tables we give the same statistics for the second algorithm. Although, we use a crew scheduling algorithm several times to compute feasible solutions in the final step of the algorithm, we do not mention these cpu times here, since each crew scheduling algorithm can be used and it is not necessary to use the one we suggested.

In the third part, we compare the upper bounds (best feasible solutions) of both algorithms with the best lower bound of the two algorithms, which results in "gap 1" and "gap 2". Notice that sometimes the first algorithm gives the better lower bound and sometimes the second one. Finally, we denote in the bottom part the number of instances (out of 10) that the first algorithm gives a better lower and upper bound, respectively. Behind brackets, we also indicate the number of instances, where these bounds for the two algorithms are equal. By definition, the second algorithm gives the best bound in the remaining cases.

From Tables 6 and 7, we can conclude that only for small instances a lower bound is computed within 3 hours computation time. Furthermore, it is more difficult to find a lower bound in the case of 4 depots. The first algorithm finds the lower bounds more often than the second one. In all cases except two, namely 160 trips for 2 as well as 4 depots, the first algorithm gives the best lower bound of the two or both algorithms give the same lower bound.

If we look at the quality of the solutions found, the first algorithm sometimes performs better, while other times the second one performs better. For instance, in the case of 160 trips and 2 depots the first and the second algorithm give both three times the best solution. However, in the cases with a large number of trips the first algorithm performs significantly better, e.g., in the case of 400 trips and 2 depots the first algorithm gives 5 out of 10 times the best solution, while the second algorithm gives only once the best

	# trips	80	100	160	200	320	400
	# iter.	17.4	25.2	36.8	39.5	-	-
	cpu m.	154.7	403.9	982.8	1641.5	-	-
int. 1	cpu p.	148.7	510.7	3529.8	4769.5	-	-
	cpu t.	317.5	942.3	4721.3	6675.0	-	-
	cpu f.	3.6	5.0	15.3	40.0	-	-
	# found	10	10	4	2	0	0
	# iter.	24.2	22.8	55.5	67.5	-	-
	cpu m.	212.7	224.0	832.3	1251.0	-	-
int. 2	cpu p.	133.7	201.8	4159.5	6128.0	-	-
	cpu t.	354.0	439.3	5140.8	7562.5	-	-
	cpu f.	13.3	20.3	42.0	237.0	-	-
	# found	10	9	4	2	0	0
	upper 1	29.8	35.8	52.2	69.0	-	-
	upper 2	29.8	35.6	52.4	68.5	-	-
	best lower	28.2	33.9	49.2	64.5	-	-
	gap 1 (%)	5.37	5.31	5.75	6.52	-	-
	gap 2 (%)	5.37	4.78	6.11	5.84	-	-
	# lower 1	0 (10)	2 (8)	3 (6)	1 (9)	0 (10)	0 (10)
	# upper 1	1 (8)	1 (6)	3 (4)	4 (3)	3 (4)	5 (4)

Table 6: Detailed results random data instances - 2 depots - type A

solution of the two.

The average gaps between the feasible solutions and the best known lower bound varies between 4 and 9% for the first as well as the second algorithm. These gaps are slightly higher in the cases with 4 depots than in the corresponding case with 2 depots, which confirms our earlier suggestion that the algorithms perform better in the 2 depot case. Furthermore, these gaps do not vary significantly between the two algorithms. Although, the savings of using an integrated approach are quite high, the gaps suggest that there may be some room for further improvement. However, in our opinion the main reason of these gaps is in the fact that the lower bounds are not very strong.

For instances of type B we obtain similar results (see Tables 8, 9, 10 and 11). The main conclusions discussed above about the effectiveness of using an integrated approach and the differences between the two algorithm still hold. The difference between the two types is that the instances of type B are easier to solve than those of type A, since the number of instances for which we found a lower bound is higher, the computation times are lower and the gaps are smaller. Notice that this is also what we expected beforehand.

	# trips	80	100	160	200
	# iter.	27.8	34.0	37.3	-
	cpu m.	316.9	363.2	1278.3	-
int. 1	cpu p.	532.3	853.4	7063.0	-
	cpu t.	875.7	1272.4	8574.7	-
	cpu f.	30.4	172.2	341.7	-
	# found	10	9	2	0
	# iter.	28.6	34.0	52.0	-
	cpu m.	366.4	379.3	1077.0	-
int. 2	cpu p.	285.9	566.4	6276.0	-
	cpu t.	667.0	973.3	7562.5	-
	cpu f.	120.9	392.3	1021.5	-
	# found	7	8	2	0
	upper 1	29.7	37.0	56.7	-
	upper 2	29.6	36.8	57.7	-
	best lower	27.8	34.0	52.7	-
	gap 1 (%)	6.40	8.11	7.06	-
	gap 2 (%)	6.08	7.55	8.67	-
	# lower 1	3 (7)	2 (8)	2 (7)	0 (10)
	# upper 1	1 (7)	4 (2)	5 (5)	4 (4)

Table 7: Detailed results random data instances - 4 depots - type A

7 Conclusion

The results reported in the previous section indicates that medium-sized problem instances with multiple depots can be solved by using an integrated approach for the vehicle and crew scheduling problem. Furthermore, there are significant savings compared to the traditional sequential approach, where first the vehicle scheduling and afterwards the crew scheduling problem is solved. We have suggested two different algorithms which are both Lagrangian heuristics based on column generation. The major difference between these two algorithms is that in the second one the computation of the lower bound is based on a model with only variables related to the crew schedules. However, the lower bounds obtained by this algorithm are rarely stronger than the bounds obtained by the first one and regularly weaker. If the solutions of both algorithms are compared, it is difficult to conclude which algorithm is better, since sometimes the first one gives the best solution and sometimes the second one. However, for the larger random problem instances of both types, the first algorithms performs better.

Finally, we have introduced a new and better way to generate random problem instances to simulate an extra-urban bus network. We hope that also other researchers will use these instances in future experiments.

	trips	80	100	160	200	320	400
	vehicles	11.3	13.8	19.3	24.4	35.8	44.2
seq.	drivers	26.9	32.9	44.4	54.7	79.0	96.8
	total	38.2	46.7	63.7	79.1	114.8	141.0
	vehicles	11.3	13.8	19.3	24.4	35.8	44.2
int. 1	drivers	24.9	29.1	42.3	51.4	77.8	95.0
	total	36.2	42.9	61.6	75.8	113.6	139.2
	vehicles	11.3	13.8	19.3	24.4	35.8	44.2
int. 2	drivers	24.7	29.1	42.6	52.2	78.0	95.6
	total	36.0	42.9	61.9	76.6	113.8	139.8

Table 8: Average results random data instances - 2 depots - type B

	trips	80	100	160	200
	vehicles	11.3	13.8	19.3	24.4
seq.	drivers	28.3	34.1	45.9	56.8
	total	39.6	47.9	65.2	81.2
	vehicles	11.3	13.8	19.3	24.4
int. 1	drivers	25.1	30.3	42.9	52.1
	total	36.4	44.1	62.2	76.5
	vehicles	11.3	13.8	19.3	24.4
int. 2	drivers	24.8	30.0	44.0	53.6
	total	36.1	43.8	63.3	78.0

Table 9: Average results random data instances - 4 depots - type B

Acknowledgments The authors are very grateful to Connexxion for providing the data and supporting this research.

References

[1] A.A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17:271–281, 1987.

[2] P. Carraresi, L. Girardi, and M. Nonato. Network models, Lagrangean relaxation and subgradients bundle approach in crew scheduling problems. In J.R. Daduna, I. Branco, and J.M. Pinto Paixão, editors, *Computer-Aided Transit Scheduling, Proceedings of the Sixth International Workshop*, pages 188–212. Springer Verlag, Berlin, 1995.

[3] J-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35:375–388, 2001.

	# trips	80	100	160	200	320	400
	# iter.	15.9	194	40.0	42.3	-	-
	cpu m.	136.2	243.8	1196.4	1753.0	-	-
int. 1	cpu p.	121.0	266.5	4106.3	4489.0	-	-
	cpu t.	267.6	530.9	5487.3	6321.0	-	-
	cpu f.	4.1	4.3	14.7	20.3	-	-
	# found	10	10	7	3	0	0
	# iter.	19.9	19.1	55.9	55.5	-	-
	cpu m.	162.2	165.1	1044.1	779.5	-	-
int. 2	cpu p.	90.8	116.3	3641.5	1962.5	-	-
	cpu t.	258.8	290.8	4823.3	2809.5	-	-
	cpu f.	12.2	16.6	54.8	86.5	-	-
	# found	10	9	8	2	0	0
	upper 1	36.2	42.9	64.3	86.7	-	-
	upper 2	36.0	42.9	64.6	87.7	-	-
	best lower	34.1	40.9	60.6	82.3	-	-
	gap 1 (%)	5.80	4.66	5.61	5.00	-	-
	gap 2 (%)	5.28	4.66	6.24	6.08	-	-
	# lower 1	0 (10)	1 (8)	3 (6)	2 (8)	0 (10)	0 (10)
	# upper 1	2 (4)	2 (6)	3 (5)	7 (2)	3 (4)	6 (3)

Table 10: Detailed results random data instances - 2 depots - type B

[4] M. Dell'Amico, M. Fischetti, and P. Toth. Heuristic algorithms for the multiple depot vehicle scheduling problem. *Management Science*, 39:115–125, 1993.

[5] G. Desaulniers, J-F. Cordeau, and J. Desrosiers. Simultaneous multi-depot bus and driver scheduling. TRISTAN IV preprints, 2001.

[6] M. Fischetti, S. Martello, and P. Toth. The fixed job schedule problem with working-time constraints. *Operations Research*, 37:395–403, 1989.

[7] R. Freling. *Models and Techniques for Integrating Vehicle and Crew Scheduling*. PhD thesis, Tinbergen Institute, Erasmus University Rotterdam, 1997.

[8] R. Freling, C.G.E Boender, and J.M. Pinto Paixão. An integrated approach to vehicle and crew scheduling. Technical Report 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam, 1995.

[9] R. Freling, D. Huisman, and A.P.M. Wagelmans. Applying an integrated approach to vehicle and crew scheduling in practice. In S. Voß and J.R. Daduna, editors, *Computer-Aided Scheduling of Public Transport*, pages 73–90. Springer, Berlin, 2001.

	# trips	80	100	160	200
	# iter.	24.3	27.6	43.0	-
	cpu m.	287.9	606.1	1548.0	-
int. 1	cpu p.	403.9	1103.7	6538.8	-
	cpu t.	709.9	1753.4	8236.5	-
	cpu f.	36.5	84.2	310.3	-
	# found	10	10	3	0
	# iter.	22.1	43.6	40.0	39.0
	cpu m.	240.1	835.7	1034.8	970.0
int. 2	cpu p.	171.4	534.8	3557.3	4376.0
	cpu t.	421.6	1388.1	4750.8	5583.0
	cpu f.	62.2	217.9	1673.5	3622.0
	# found	9	9	4	1
	upper 1	36.4	44.1	71.5	102.0
	upper 2	36.1	43.8	73.3	103.0
	best lower	33.9	40.6	66.5	92.0
	gap 1 (%)	6.87	7.94	6.99	9.80
	gap 2 (%)	6.09	7.31	9.22	10.68
	# lower 1	2 (7)	2 (8)	3 (6)	0 (9)
	# upper 1	2 (4)	3 (4)	8 (1)	9 (0)

Table 11: Detailed results random data instances - 4 depots - type B

[10] R. Freling, D. Huisman, and A.P.M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6:59–81, 2003.

[11] R. Freling, A.P.M. Wagelmans, and J.M. Pinto Paixão. An overview of models and techniques for integrating vehicle and crew scheduling. In N.H.M. Wilson, editor, *Computer-Aided Transit Scheduling*, pages 441–460. Springer Verlag, Berlin, 1999.

[12] A. Gaffi and M. Nonato. An integrated approach to extra-urban crew and vehicle scheduling. In N.H.M. Wilson, editor, *Computer-Aided Transit Scheduling*, pages 103–128. Springer Verlag, Berlin, 1999.

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

[14] K. Haase and C. Friberg. An exact branch and cut algorithm for the vehicle and crew scheduling problem. In N.H.M. Wilson, editor, *Computer-Aided Transit Scheduling*, pages 63–80. Springer Verlag, Berlin, 1999.

[15] D. Huisman, R. Freling, and A.P.M. Wagelmans. A dynamic approach to vehicle scheduling. Technical Report EI2001-17, Econometric Institute, Erasmus University Rotterdam, Rotterdam, 2001. To appear in Transporation Science under the title "A Robust Solution Approach to the Dynamic Vehicle Scheduling Problem".

[16] D. Klabjan, E.L. Johnson, G.L. Nemhauser, E. Gelman, and S. Ramaswamy. Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36:337–348, 2002.

[17] I. Patrikalakis and D. Xerocostas. A new decomposition scheme of the urban public transport scheduling problem. In M. Desrochers and J.M. Rousseau, editors, *Computer-Aided Transit Scheduling: Proceedings of the Fifth International Workshop*, pages 407–425. Springer Verlag, Berlin, 1992.