Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems

Knut Haase • Guy Desaulniers • Jacques Desrosiers
Institut für Betriebswirtschaftslehre, Universität Hohenheim, Stuttgart, Germany
École Polytechnique and GERAD, 3000 chemin de la Côte-Ste-Catherine, Montréal, Québec, Canada, H3T 2A7
École des Hautes Études Commerciales and GERAD, 3000 chemin de la Côte-Ste-Catherine, Montréal Québec, Canada, H3T 2A7
haasek@uni-hohenheim.de • guy@crt.umontreal.ca • jacques.desrosiers@hec.ca

This paper presents an exact approach for solving the simultaneous vehicle and crew scheduling problem in urban mass transit systems. We consider the single depot case with a homogeneous fleet of vehicles. This approach relies on a set partitioning formulation for the driver scheduling problem that incorporates side constraints for the bus itineraries. The proposed solution approach consists of a column generation process (only for the crew schedules) integrated into a branch-and-bound scheme. The side constraints on buses guarantee that an optimal vehicle assignment can be derived afterwards in polynomial time. A computational study shows that this approach out-performs the previous methods found in the literature for a set of randomly generated instances. A heuristic version of the solution approach is also proposed and tested on larger instances.
(*Transportation; Vehicle Scheduling; Crew Scheduling; Column Generation*)

An important problem in urban mass transit systems is the scheduling at minimum cost of vehicles and crews to serve trips defined by a timetable. The usual planning procedure consists in solving the vehicle and the crew scheduling problems sequentially, i.e., buses are first assigned to trips, then drivers are assigned to buses. This approach is strongly criticized in Ball et al. (1983), since crew costs dominate vehicle costs in most cases encountered in the urban transportation context. Therefore, it should be useful to address crew scheduling at the same time as vehicle scheduling. The combination of these two problems is simply called the vehicle and crew scheduling problem (VCSP).

It is well known that the single-depot, homogeneous fleet vehicle scheduling problem is a polynomially solvable minimum cost flow problem, while the multiple depot and heterogeneous versions are NP-hard (Bertossi et al. 1987). Furthermore, due to complex constraints arising from wage agreements and internal regulations, the single-base crew scheduling problem is already NP-hard (see Fischetti et al. 1989). Clearly, simultaneously scheduling vehicles and crews is also an NP-hard problem, independently of the number of depots.

From a computational point of view, very large multiple-depot vehicle scheduling problems can be solved to optimality in reasonable times, using highly sophisticated optimization methods. For example, practical instances with up to 20,000 trips were solved optimally by Löbe1 (1999). Also, large crew scheduling problems arising in practice can be solved to optimality in urban mass transit (see Desrochers and Soumis 1989, Mingozzi et al. 1999) and in air transportation (see Hoffman and Padberg 1993, Desaulniers et al. 1997). Unfortunately, it seems that the VCSP is much harder to solve.

0041-1655/01/3503/0286$05.00
1526-5447 electronic ISSN

In this paper, we consider the version of the VCSP that involves a single depot and a homogeneous fleet of vehicles. Although this version is not the most common in practice, it is still NP-hard and used here to introduce a promising solution approach (or part of it) for the more general versions of the problem. For the remainder of the text, the acronym VCSP will correspond to this version of the problem unless otherwise specified.

Several heuristics for the VCSP have been proposed in the literature; for instance, those of Ball et al. (1983), Tosini and Vercellis (1988), and Patrikalakis and Xerocostas (1992). These heuristics can be classified in one of the following two categories: vehicles are scheduled while solving the crew scheduling problem by a heuristic approach; vehicles are scheduled first taking into account crew considerations, then crew schedules are computed.

Alternatively, exact formulations for the VCSP have been suggested together with optimization-based heuristic approaches. Freling et al. (1995) have introduced an integer linear programming formulation with three components: a quasiassignment structure for vehicle scheduling; set partitioning constraints for crew scheduling; and a set of constraints linking vehicle and crew schedules. Two algorithms based on the Lagrangian relaxation of the set partitioning and the linking constraints are presented to compute the linear relaxation bound of their formulation. The Lagrangian subproblem separates into two parts: the first determines the vehicle schedules while the second selects the crew schedules. In the first algorithm, all crew schedules are given a priori while, in the second, they are constructed as needed, using a column generation technique. The follow-up to this preliminary research is presented in Freling's dissertation (1997), where the emphasis is on the second algorithm, which includes at this point a heuristic procedure to derive integer solutions. Computational experiments were conducted on randomly generated problems with up to 120 trips and some real-world problems with up to 148 trips. The objective considered was twofold: first, minimize the number of buses and drivers required; second, minimize bus operational costs. This methodology has produced average optimality gaps, ranging from 6% to 26% for the random problems and from 0% to 7.6% for the real-world ones. The proposed formulation and solution methodology, as well as the results for the real-world problems, can also be found in Freling et al. (1999).

The first exact solution approach for the VCSP was introduced by Haase and Friberg (1999). The underlying mathematical description combines the approaches of Desrochers and Soumis (1989) and Ribeiro and Soumis (1994), i.e., the VCSP is formulated as a set partitioning problem with additional constraints in which a column represents either a schedule for a crew or for a vehicle. The additional constraints are introduced to connect both schedule types. To derive optimal solutions, a branch-and-price-and-cut algorithm is proposed in which column generation is performed to generate both vehicle and crew schedules. In order to improve the LP relaxation of the set partitioning formulation, the clique cuts proposed by Hoffman and Padberg (1993) are considered for the vehicles. Despite the use of this highly sophisticated methodology, only small instances involving up to 20 trips could be solved to optimality within reasonable CPU time (less than 3 hours).

The main contribution of this paper on the simultaneous vehicle and crew scheduling problem in urban mass transit systems is twofold: First, we present a new formulation for the single-depot, homogeneous fleet case; second, we propose a crew-first, vehicle-second approach to solve it.

The formulation, which involves crew scheduling variables only, is a set partitioning model with side constraints. These side constraints guarantee that a feasible vehicle schedule can be derived afterwards in polynomial time. Furthermore, the inclusion of vehicle costs in this extended crew scheduling formulation ensures the overall optimality of the two-phase approach. We propose a column generation method embedded in a branch-and-bound procedure to solve the proposed formulation. This exact solution methodology can also be easily modified to obtain heuristic solutions. Finally, we show through a computational study that larger instances than before can be solved to optimality using this new approach and that the heuristic version of the algorithm yields very good quality solutions for even larger instances. At this point, we must mention that a similar formulation has also been proposed by Klabjan (2000) in the airline area for simultaneously scheduling aircraft and crews. Both research works have been done independently.

The remainder of the paper is as follows. The VCSP in the context of urban mass transit systems is defined in §1. Section 2 presents the extended crew scheduling model containing side constraints that ensure vehicle schedule feasibility. Section 3 describes branch-and-bound strategies for solving this model, where lower bounds are computed via a column generation method. It also specifies how these strategies can be modified to derive heuristic results. Finally, §4 provides computational results used to evaluate the efficiency of the proposed approach.

1. Problem Description

It is not an easy task to provide a general definition for the VCSP. Depending on the real-life situation considered, the cost structure is influenced by many factors, and numerous constraint types can be taken into account while constructing vehicle and crew schedules. In other words, there may exist substantial differences in the planning process of different transportation companies. Hence, the following problem description should be understood as an exemplary planning situation for a practical application, which shows that our approach is sufficiently general to handle the most typical and important constraints arising in practice.

In this paper, we focus on VCSPs arising at a planning level in a company that offers transportation service along predetermined bus lines in a city. Bus and driver schedules must be computed over a 1-day horizon. For the vehicle aspect of the problem, we consider the single-depot, homogeneous fleet case. For its crew aspect, the single-base, multiple-crew-type case is retained: the crew base corresponds to the vehicle depot, and crew types differ with respect to the workday type they can be assigned to, not to their individual characteristics. Therefore, we assume that all drivers are identical but can be assigned to work days of different types.

Each *line* of the bus system is defined by a start location, an end location, and several intermediate stops where passengers can get on and off the bus. Some of these locations (among which the start and the end locations) are also considered as *relief points*, that is, an exchange of drivers may occur there. Note that the depot is also a relief point. Depending on passenger demand, lines are served at different frequencies, especially during peak hours where frequencies are higher. For each line, there exists a timetable defining a set of trips to be operated. For each *trip*, the timetable specifies a start time, an end time, and arrival times at intermediate stops. Each of these trips has to be serviced by exactly one bus. Therefore, no bus exchanges are possible at intermediate stops. Without loss of generality and to ensure bus flow conservation in our model, we assume that at most one trip can start from (respectively end at) the same location at the same time. If it is not the case, start and end locations can be duplicated to artificially simulate this situation.

From a driver point of view, a trip is divided into a sequence of consecutive segments, called *d-trips*, which are defined according to the relief points along the trip. A single driver must be assigned to each d-trip, and driver exchanges are allowed at the start and the end of each d-trip.

Often, buses need to move without passengers to reach the start of a trip or to return to the depot. These empty bus moves are called *deadheads*. As in Freling (1997), we assume that a deadhead between two relief points cannot be part of a solution if it is less expensive to go through the depot where the driver can take a break. In fact, such a situation is plausible since breaks are usually not paid. Drivers may also need to move on their own (without driving a bus) to reach a relief point or to return to the depot. In this case, we say that the driver is *walking* (although this is not necessarily the case in real life).

The workday of a driver is referred to as a *duty*. Various duty types with regards to labor regulations can be assigned to drivers. For instance, a first type may impose that a driver remains on the same bus all along its duty while a second type may allow up to two bus changes during the same duty. The set of consecutive tasks (d-trips and deadheads) performed by a driver on the same bus is called a *piece of work*, also abbreviated by *p_of_w*. Of course, drivers are entitled to break periods when they are assigned to long duties. A *break* must be granted between two pieces of work and must take place at the depot or any other relief point. In the former case, it is possible to define a deadhead to return to the depot, where a different driver (or even the same driver) takes the bus to reach the start location of its next trip. In the latter case, we assume that the break is taken after walking to the next location to minimize delays caused by longer than expected walks (breaks can always be shortened). Nevertheless, the model can also be adapted when this assumption does not hold.

Each bus schedule and each driver's duty must start and end at the depot. Duties must also begin and end with sign-on and sign-off periods of predetermined durations. Furthermore, depending on its type, a duty is subject to certain restrictions regarding: its length (spread time); the length of its pieces of work and breaks; the number of pieces of work it contains; and the total work time, that is, the time spent driving a bus or attending to it outside of the depot. In particular, the length of all breaks must fall within the interval $[b_{\min}, b_{\min}]$. The specific rules considered in this paper are discussed in §4.

Operational costs are incurred for buses and drivers, including, for instance, fuel, bus inspection and servicing, bus running repairs, and drivers' salaries. These costs are often proportional to the mileage traveled by the buses and to the time worked by the drivers. We assume that operational costs can be determined separately for each d-trip, deadhead, or walking movement. The cost structure may additionally involve bus and driver fixed costs.

The single-depot, homogeneous fleet VCSP in urban mass transit systems can be stated as follows. Given a set of trips divided into d-trips, find a set of minimum cost bus and driver schedules such that each trip is covered by exactly one bus; each d-trip and each bus deadhead used are assigned to exactly one driver; and all work rules are satisfied. Note that in this version of the problem, we also assume that the numbers of available drivers and buses are unbounded. However, such constraints can easily be included in the proposed formulation and treated by the solution approach.

2. A Mathematical Formulation for the VCSP

This section presents a mathematical formulation for the VCSP that relies on the following network structure.

2.1. Driver Network Structure

Two network structures usually underly the simultaneous vehicle and crew scheduling problem for urban bus systems—one for the buses and another one for the drivers. However, the formulation we propose is based solely on the second structure that incorporates part of the bus network structure. Since there are several driver duty types, a specific network is defined for each of them. This section describes the network structure of these driver networks.

Let U, indexed by u, be the set of duty types. Let $G^u = (N^u, A^u)$, $u \in U$, be the network representing implicitly all feasible duties of type u, where N^u and A^u denote its node and arc sets, respectively. The components of a typical driver network G^u are illustrated in Figure 1 and described in the following paragraphs. In such a time-space network, a node corresponds to a specific time at a specific location while an arc represents a movement in time, in space, or in both dimensions. In the following, let τ_i and λ_i be the time and location associated with node i, respectively, and denote by b_{ij} and w_{ij} the travel times between locations λ_i and λ_j when driving a bus and walking, respectively. We assume that $w_{ij} \geq b_{ij}$ for all relevant pairs of locations λ_i and λ_j.

As shown in Figure 1, there are eight node types in N^u—*source, sink, start_of_trip, end_of_trip, relief, start_of_break, end_of_break, and min_break*. There is a single *source* node o^u and a single *sink* node d^u. They correspond to the start and the end of a duty, respectively, and their associated times are given by the earliest feasible departure time from the depot and the latest feasible arrival time at the depot, respectively. A pair of *start_of_trip* and *end_of_trip* nodes is associated with each trip to be accomplished. They represent the start and the end of the corresponding trip, respectively, and their associated times are defined according to the timetable. A *relief* node is defined for

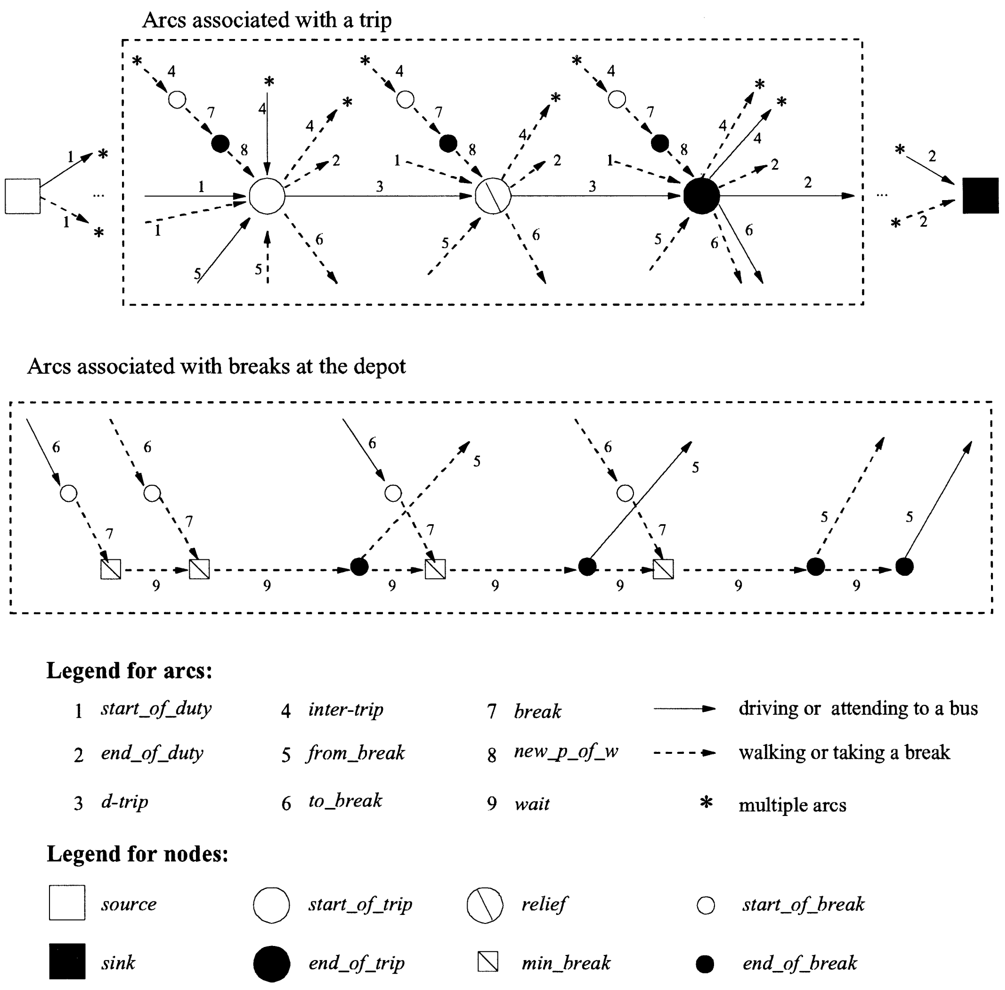

Figure 1 Network Components

each relief point (other than the start and end locations) along a trip. These nodes provide the possibility to exchange drivers in the middle of a trip and their associated times are also defined according to the timetable. Finally, *start_of_break* and *end_of_break* nodes indicate the start and the end of breaks while a *min_break* node specifies that the minimum break time $b_{\min}$ has elapsed.

Several of the *start_of_break*, *end_of_break*, and *min_break* nodes are associated with a trip to model breaks at the depot. In that case, the time τ_j associated with the *start_of_break* node j is equal to $\tau_i + \tau_{ij}$, where i is the corresponding *end_of_trip* node and τ_{ij} is equal to b_{ij} or w_{ij} depending on whether the driver drives or walks to the depot; the time τ_i associated with the *min_break* node j is equal to $\tau_i + b_{\min}$, where i is the corresponding *start_of_break* node; and the time τ_i associated with the *end_of_break* node i is given by $\tau_j - \tau_{ij}$, where j is the corresponding *start_of_trip* node and τ_{ij} is equal to b_{ij} or w_{ij} depending on whether the driver is driving or walking out of the depot. Pairs of *start_of_break* and *end_of_break* nodes are also defined to model breaks at relief points. The time τ_j associated with such a *start_of_break* node j is given by $\tau_i + w_{ij}$, where τ_i is the time when the driver finished its last piece of work at location λ_i. The time τ_i associated with such an *end_of_break* node i corresponds to the time when the driver starts its next piece of work at location λ_i.

There are nine arc types in A^u—*start_of_duty, end_of_duty, d-trip, inter-trip, from_break, to_break, break, new_p_of_w*, and *wait* (see Figure 1). A *start_of_duty* arc (o^u, j) links the source node o^u to a *start_of_trip*, a *relief*, or an *end_of_trip* node j, while an *end_of_duty* arc (i, d^u) links a *start_of_trip*, a *relief*, or an *end_of_trip* node i to the sink node d^u. These arcs correspond to a driver's first and last movements of the day, during which he might be driving a bus or walking. *Start_of_duty* and *end_of_duty* arcs also include sign-on and sign-off periods, respectively. Each d-trip in each trip is represented by a *d-trip* arc (i, j), which links a pair of consecutive *start_of_trip, relief*, or *end_of_trip* nodes i and j associated with the same trip. All *d-trip* arcs correspond to bus driving movements.

Intertrip arcs are defined to represent a driver moving from one relief point to another at a specific time. For a bus driving movement, such an arc (i, j) links an *end_of_trip* node i to a *start_of_trip* node j whenever $\tau_j - \tau_i \geq w_{ij}$. For a walking movement that is followed by a break, such an arc (i, j) links a *start_of_trip*, a *relief*, or an *end_of_trip* node i to a *start_of_break* node j whenever $\tau_m - \tau_i - b_{ij} \in [b_{\min}, b_{\max}]$, where τ_m is the time associated with the end of the break. Note that *inter-trip* arcs are created only when it is less expensive to go directly from the first relief point to the second than to go through the depot, where breaks are generally cheaper if not without costs.

To_break and *from_break* arcs are also defined for bus driving and walking movements. A *to_break* arc (i, j) connects a *start_of_trip*, a *relief*, or an *end_of_trip* node i to a *start_of_break* node j and corresponds to a driver moving from a relief point to the depot where he takes a break. Such an arc (i, j) exists only if the driver can be assigned to a subsequent d-trip or deadhead, i.e., if there exists a *start_of_trip*, a *relief*, or an *end_of_trip* node k such that $\tau_k - \tau_i \geq \tau^1_{ij} + \tau^2_{jk} + b_{\min}$, where τ^1_{ij} is equal to b_{ij} if it corresponds to a driving movement and w_{ij} otherwise, and τ^2_{jk} is given by b_{jk} if k is a *start_of_trip* node and w_{kj} otherwise. Symmetrically, a *from_break* arc (i, j) goes from an *end_of_break* node i to a *start_of_trip*, a *relief* or an *end_of_trip* node j and corresponds to a driver moving from the depot after a break to a relief point. Such an arc exists only if the driver could have been assigned to a preceding d-trip or deadhead, i.e., if there exists a *start_of_trip*, a *relief*, or an *end_of_trip* node k such that $\tau_j - \tau_k \geq \tau^1_{ki} + \tau^2_{ij} + b_{\min}$, where τ^1_{ki} is equal to b_{ki} if k is an *end_of_trip* node and w_{ki} otherwise, and τ^2_{ij} is given by b_{ij} if it is a driving movement and w_{ij} otherwise.

A *break* arc (i, j) is defined for each *start_of_break* node i. When linked to a *min_break* node j, it represents a break of minimum duration taken at the depot. When directly linked to an *end_of_break* node, it corresponds to a full break between two pieces of work. In that case, a *new_p_of_w* arc, indicating the beginning of a new piece of work, connects this *end_of_break* node to the *start_of_trip, end_of_trip*, or *relief* node corresponding to the driver's location at the end of its break. Such *end_of_break* nodes and *new_p_of_w* arcs are used to validate the piece of work minimum length restriction, as explained in §4.2.

Finally, a sequence of *wait* arcs is defined at the depot to represent a driver extending its break over the minimum break duration. Such an arc (i, j) links a pair of consecutive (in time) nodes i and j, each of them being of type *min_break* or *end_of_break*. Note that if *min_break* nodes and *end_of_break* nodes are associated with the same time, the former nodes are ordered before the latter ones.

A cost c_{ij} is defined for each arc $(i, j) \in A^u$, $u \in U$. This cost is charged for each unit of flow that passes through the arc. It usually includes vehicle and driver operational costs but also it may include driver fixed costs (for instance, on *start_of_duty* arcs). Vehicle fixed costs are handled differently since they cannot be taken into account by arc costs in the proposed model. Details are provided in the next section. We assume that idle buses at the depot incur no costs.

The concept of resources is used to model at the network level some of the constraints defining the feasibility of drivers' duties (see Desrochers and Soumis 1989, and Desrochers et al. 1992). For instance, a resource can be used to restrict the time spent on a piece of work. Let R^u be the set of resources considered for duty type $u \in U$ and associate with each arc $(i, j) \in A^u$, $u \in U$, a resource consumption t^{ur}_{ij} for each resource $r \in R^u$. Such a consumption usually corresponds to the amount of resource r consumed by the activity represented by arc (i, j). We also associate with each node $i \in N^u$, $u \in U$, a resource interval $[a^{ur}_i, b^{ur}_i]$ for each resource $r \in R^u$. Such an interval limits the set of feasible values that can be taken by resource r at node i while constructing a duty of type u. Note that a resource cannot take a value greater than the upper bound, but if it takes a value less than the lower bound, it is raised to that lower bound. However, if the lower bound is a hard constraint, an additional resource is required to strictly enforce it. This new resource is the negative of the one used to satisfy the upper bound. Mathematically speaking, it means that an interval $[a, b]$ restricting a variable v must be treated as two upper bounds, $v \le b$ and $-v \le -a$ (see Gamache et al. 1998). Resources are further discussed in §4.2.

The network structure described above can be simplified for the one-piece duty case. Indeed, since no breaks are allowed in such duties, all *start_of_break*, *min_break*, and *end_of_break* nodes can be removed, as well as all *break, to_break, from_break, new_p_of_w*, and *wait* arcs. Moreover, all *intertrip* arcs representing a walking movement can be discarded.

2.2. A Crew-based Model

The VCSP formulation presented below is not complete in itself since it is only based on crew networks and not on a vehicle network. Consequently, it can yield crew schedules but cannot provide vehicle schedules right away. In fact, these schedules can be computed in a polynomial postprocessing phase. Nevertheless, sufficient vehicle information is integrated in the crew scheduling phase to ensure global optimality of crew and vehicle schedules. The proposed formulation corresponds to a set partitioning model with side constraints. It requires the following additional notation.

Let V, indexed by v, and W, indexed by w, be the sets of d-trips and bus trips, respectively, and let H, indexed by h, be the set of times at which a bus must leave the depot to arrive exactly on time at the start of each trip. Let Ω^u, indexed by ρ, be the set of feasible paths (or schedules) in G^u, $u \in U$. For each path ρ in Ω^u, denote its cost by c_ρ and define the following binary parameters: e_ρ^v takes value 1 if path ρ contains d-trip $v \in V$, and 0 otherwise; f_ρ^w takes value 1 if path ρ contains a driving movement ending at the start location of trip $w \in W$, and 0 otherwise; g_ρ^w takes value 1 if path ρ contains a driving movement starting from the end location of trip $w \in W$, and 0 otherwise; and q_ρ^h takes value 1 if path ρ contains a driving movement or a bus attendance task starting at time $h \in H$ or before and ending after time h, and 0 otherwise. Finally, denote by c the fixed cost paid for each bus used during a day.

The following model involves two types of variables—a set of path flow variables and one bus variable. A binary path flow variable θ_ρ^u, $\rho \in \Omega^u$, $u \in U$, takes value 1 if a driver is assigned to duty ρ, and 0 otherwise. The nonnegative integer bus variable B is simply used to compute the number of buses required to cover all the trips.

In order to derive globally optimal crew schedules for the VCSP in the context of urban mass transit systems, we propose the following set partitioning model with side constraints:

$$\text{Minimize} \quad cB + \sum_{u \in U} \sum_{\rho \in \Omega^u} c_\rho \theta_\rho^u \tag{1}$$

subject to:

$$\sum_{u \in U} \sum_{\rho \in \Omega^u} e_\rho^v \theta_\rho^u = 1 \qquad \forall v \in V, \tag{2}$$

$$\sum_{u \in U} \sum_{\rho \in \Omega^u} f_\rho^w \theta_\rho^u = 1 \qquad \forall w \in W, \tag{3}$$

$$\sum_{u \in U} \sum_{\rho \in \Omega^u} g_\rho^w \theta_\rho^u = 1 \qquad \forall w \in W, \tag{4}$$

$$\sum_{u \in U} \sum_{\rho \in \Omega^u} q_\rho^h \theta_\rho^u \le B \qquad \forall h \in H, \tag{5}$$

$$\theta_\rho^u \in \{0, 1\} \qquad \forall u \in U, \forall \rho \in \Omega^u. \tag{6}$$

The objective function (1) aims at minimizing total costs including bus and driver operational and fixed costs. Constraints (2) indicate that each d-trip must be assigned to exactly one driver. At the same time, these constraints also serve as covering constraints for the bus trips: Given that no arcs representing a bus movement other than *d-trip* arcs are incident to *relief* nodes, a single bus is assigned to each trip. Constraints (3) ensure that a bus arrives at the start location of each trip, while constraints (4) ensure that a bus leaves the end location of each trip. Therefore, the assignment-type constraint sets (3) and (4), together with the d-trip covering constraint set (2), guarantee bus flow conservation outside of the depot. Relations (5) state that the number of buses in use (either on a d-trip or a deadhead) at each relevant time of departure from the depot must be smaller than or equal to B, the total number of buses needed to cover all trips. Given a positive bus fixed cost (i.e., c > 0), these relations allow one to find the exact value of B. Finally, binary requirements on path flow variables are given by (6). Note that B is restricted to take a nonnegative integer value according to (5)–(6).

3. Solution Approach

As suggested in the algorithmic framework proposed by Desaulniers et al. (1998), the set partitioning type formulation (1)–(6) can be solved by a column generation approach embedded in a branch-and-bound procedure. Branching decisions are then taken as in Desrochers and Soumis (1989) for an exact approach and also on path flow variables θ_p^u, $p \in \Omega^u$ and $u \in U$ for a heuristic approach. Moreover, easily identifiable cutting planes may be added to reinforce linear relaxations throughout the search tree when the number of buses and/or drivers has to be minimized. Sections 3.1 and 3.2 give an overview of these aspects of the solution approach (for further details, see Desrochers and Soumis 1989, Desrosiers et al. 1995, and Desaulniers et al. 1998). Finally, §3.3 explains how vehicle schedules can be derived in a postprocessing phase.

3.1. Lower Bound Evaluation

At each node of the branch-and-bound search tree, a lower bound is computed by solving the linear relaxation of (1)–(6), which contains in practice a huge number of variables that cannot be enumerated explicitly. To overcome this difficulty, we use an iterative column generation method that generates feasible driver duties as needed. This method relies on a master problem and a set of subproblems.

The master problem is simply the linear relaxation of (1)–(6), together with the branching decisions and cutting planes applicable at the current node. During the iterative process, the master problem is restricted to a subset of its path variables (columns)—namely, those generated at previous iterations. In our implementation, these restricted master problems are solved using the primal simplex algorithm. The role of the restricted master problem is twofold: first, finding the best solution from the restricted subset of path variables, and second, computing corresponding dual variable values. These values are needed at the subproblem level to identify path variables with a negative reduced cost.

The role of the subproblems consists in generating negative reduced cost path variables when such a variable exists. To do so, a subproblem is defined for each driver network G^u, $u \in U$, with the objective of finding the feasible duty of the type u with the least reduced cost. Since all feasible duties of type u are implicitly defined in the network G^u as paths between source node o^u and sink node d^u, the subproblem for duty type u corresponds to a resource-constrained shortest path problem on network G^u.

In each subproblem, the reduced cost of a path variable is computed as the sum of the reduced costs of the arcs composing that path. Given that $\boldsymbol{\alpha} = \{\alpha^v \mid v \in V\}$, $\boldsymbol{\beta} = \{\beta^w \mid w \in W\}$, $\boldsymbol{\gamma} = \{\gamma^w \mid w \in W\}$, and $\boldsymbol{\delta} = \{\delta^h \mid h \in H\}$ are the vectors of dual variables associated with constraint sets (2)–(5), respectively, the reduced cost of arc $(i, j) \in A^u$, $u \in U$, is expressed as:

$$\bar{c}_{ij} = c_{ij} - e_{ij}^v \alpha^v - f_{ij}^w \beta^w - g_{ij}^w \gamma^w - q_{ij}^h \delta^h. \tag{7}$$

Two types of variables are required to formulate the subproblem on network G^u, $u \in U$. First, a binary arc flow variable X_{ij}^u is defined for each arc $(i, j) \in A^u$ to indicate whether or not the arc (i, j) is part of the subproblem solution. Second, a continuous resource variable T_i^{ur} is defined for each node $i \in N^u$ and resource $r \in R^u$ to cumulate the amount of resource r used from source node o^u to node i. Using this notation, the subproblem for driver duty type $u \in U$ can be formulated as follows:

$$\text{Minimize} \quad \sum_{(i,j)\in A^u} \bar{c}_{ij} X_{ij}^u \tag{8}$$

subject to:

$$\sum_{j:(o^u,j)\in A^u} X_{o^u,j}^u \le 1, \tag{9}$$

$$\sum_{i:(i,j)\in A^u} X_{ij}^u - \sum_{i:(j,i)\in A^u} X_{ji}^u = 0 \qquad \forall j \in N^u \backslash \{o^u, d^u\}, \tag{10}$$

$$\sum_{i:(i,d^u)\in A^u} X_{i,d^u}^u \le 1, \tag{11}$$

$$X_{ij}^{u}\left(T_i^{ur}+t_{ij}^{ur}-T_j^{ur}\right)\le 0 \qquad \forall (i,j)\in A^u, \forall r\in R^u, \quad (12)$$

$$a_i^{ur}\le T_i^{ur}\le b_i^{ur} \qquad \forall i\in N^u, \forall r\in R^u, \quad (13)$$

$$X_{ij}^{u}\in\{0,1\} \qquad \forall (i,j)\in A^u. \quad (14)$$

The objective function (8) minimizes the sum of the reduced costs of the arcs used in the subproblem solution. Relations (10) are the usual network flow conservation constraints while constraints (9) or (11) limit the flow between the source node o^u and the sink node d^u to a maximum of one unit. Constraints (12) define the resource extension process on the arcs of network G^u. Such a constraint, indexed by $(i,j)\in A^u$ and $r\in R^u$, imposes that, whenever there is a positive flow on arc (i,j) (i.e., $X_{ij}^u=1$), T_j^{ur}, the value of resource r at node j be greater than or equal to $T_i^{ur}+t_{ij}^{ur}$. When no flow passes through arc (i,j), this constraint has no effect at all. Relations (13) restrict the set of feasible values that can be taken by the resource variables at the nodes of network G^u. Finally, binary requirements on arc flow variables are given by (14). Each subproblem can be transformed into a shortest path problem with resource variables by converting inequalities into equalities in constraints (9) and (11) by adding the zero-cost arc (o^u, d^u). Note that the resource-constrained shortest path problem can be solved by the dynamic programming algorithm of Desrochers (1986), which is also described in the survey paper by Desrosiers et al. (1995).

3.1.1. Accelerating Strategies. As discussed in the following paragraphs, five different strategies can be implemented to speed up the solution process: omission of redundant constraints in the master problem; node aggregation within the subproblems; dynamic generation of the bus count constraints (5); substitution of the covering constraints (2) and (4); and early LP termination.

The first strategy consists in omitting redundant constraints of set (5). These redundant constraints correspond to those associated with a departure time from the depot immediately followed by another such departure time. Indeed, only the constraints associated with a departure time from the depot immediately followed by an arrival time at the depot are needed, since the number of buses used at that departure time is always greater than or equal to the number of buses used at earlier departure times when no arrivals occurred in the meantime.

The node aggregation strategy concerns the *min_break* and *end_of_break* nodes associated with the depot. In each network G^u, $u\in U$, these nodes can be aggregated as follows: All consecutive *min_break* nodes and all following consecutive *end_of_break* nodes can be replaced by a single aggregated node whose associated time corresponds to that of the first *end_of_break* node in that sequence of nodes. Such a node aggregation requires the modification of the resource consumptions on the arcs incident to the newly aggregated node.

The subsystem of the bus count constraints (5) may dramatically increase the density of the proposed augmented crew scheduling model (1)–(6). Indeed, a path flow variable associated with a type II duty may cover up to 50% of these rows. However, only a subset of the binding constraints from this subsystem need to be present in the formulation. Hence, these constraints can be introduced in the model as needed during the solution process, that is, when they are violated at the end of the solution of a linear relaxation. Since at least one of these constraints is needed, the solution process can start with a few of these constraints at peak hours (a single one in our implementation). Computational results are reported in §4.3.1 to assess the efficiency of this strategy.

The fourth strategy is derived from the observation that a driver assigned to a bus remains, in general, on that bus for several d-trips. Therefore, to further reduce the density of the columns generated, constraints (2) can be replaced by flow conservation constraints, stipulating that the number of drivers walking to the start location of a d-trip must be equal to the number of drivers walking away from this location. Since constraints (3) ensure that a driver drives a bus to the beginning of the first d-trip of a trip, these flow conservation constraints will ensure that a driver will cover the d-trips of this trip. Similarly, constraints (4) can also be replaced by such flow conservation constraints.

Flow conservation constraints replacing (2) and (4) can be formulated as follows:

$$\sum_{u\in U}\sum_{\rho\in\Omega^u} \underline{e}^v_\rho \theta^u_\rho - \sum_{u\in U}\sum_{\rho\in\Omega^u} \bar{e}^v_\rho \theta^u_\rho = 0 \quad \forall v \in V, \qquad (15)$$

$$\sum_{u\in U}\sum_{\rho\in\Omega^u} \underline{g}^w_\rho \theta^u_\rho - \sum_{u\in U}\sum_{\rho\in\Omega^u} \bar{g}^w_\rho \theta^u_\rho = 0 \quad \forall w \in W, \qquad (16)$$

where, for each path ρ in Ω^u, $u \in U$, parameter $\underline{e}^v_\rho$ (resp. $\bar{e}^v_\rho$) takes value 1 if path ρ contains a walking movement ending at (resp. starting from) the start location of d-trip $v \in V$, and 0 otherwise; parameter $\underline{g}^w_\rho$ (resp. $\bar{g}^w_\rho$) takes value 1 if path ρ contains a walking movement ending at (resp. starting from) the end location of trip $w \in W$, and 0 otherwise. Section 4.3.1 also reports computational results on the use of this accelerating strategy.

The last accelerating strategy, only used in the heuristic version of the approach, consists of terminating prematurely the solution of the linear relaxations. In fact, the column generation process is stopped if a certain decrease in the objective function value has not been attained in a certain number of consecutive iterations. In our implementation, a minimum objective value improvement of 10 was required in the last 5 iterations for pursuing the solution process (except for the first linear relaxation, which was solved to optimality). This strategy is well known (see, for instance, Barnhart et al. 1998, Gamache et al. 1999) and reduces the tailing-off effect of the column generation method.

3.2. Branching Strategies and Cutting Planes

An exact and a heuristic version of the solution approach have been designed to obtain the results presented in §4. In both versions all linear relaxations are solved to optimality. However, given the size of the master problem to be solved by the heuristic approach, right-hand side perturbation, as discussed below, is introduced in constraints (2)–(4) to reduce degeneracy during the master problem solution. In addition to master problem perturbation, the two versions of the solution approach only differ by the branching scheme used to derive integer solutions.

In the exact version of the solution approach, the branch-and-bound search tree is explored using a depth-first procedure until a good integer solution is found. Then a best-first procedure is adopted to complete the exploration. A solution is said to be good when its value is within a certain percentage of the value of the linear relaxation at the root node of the search tree. This percentage was set to 5% for the computational results presented in §4.

Given a fractional linear relaxation solution, branching decisions are taken on the so-called intertask values. This is the strategy used in Desrochers and Soumis (1989), a special case of the Ryan and Foster (1981) branching scheme developed for set partitioning problems. In the VCSP considered, tasks are defined on arcs $(i,j) \in A^u$, $u \in U$, for which the binary parameters e^v_{ij}, f^w_{ij}, or g^w_{ij} take value 1, i.e., when an arc corresponds to a d-trip $v \in V$, a bus driving movement ending at the start location of a trip $w \in W$, or a bus driving movement starting from the end location of a trip $w \in W$, respectively. Intertask values represent the flow of drivers between two consecutive tasks. Since all tasks must be covered exactly once as imposed by constraints (2)–(4), all intertask values are subject to binary requirements. Therefore, given a solution with a fractional intertask value, two branching nodes are created—one where the intertask value is set to 1 and another where it is set to 0. In both branches, the decision is imposed at the subproblem level by creating different lists of labels in the dynamic programming algorithm of Desrochers (1986). Each list is associated with a task and a label is put in the list associated with the last task performed along the corresponding partial path. The dynamic programming algorithm is valid only when label dominance is performed list by list (see Dumas et al. 1991, for such an implementation in the context of pickup and delivery problems).

To solve larger VCSP instances, we also developed a heuristic approach that differs from the exact one in several ways. First, the branch-and-bound search tree is explored using a depth-first procedure where no backtracks are allowed. Second, two heuristic branching strategies compete via a scoring system to determine the branching decisions to be imposed at each node of the search tree that cannot be pruned. Both strategies create a single branch node and have the possibility of taking several decisions at the same time. When the first strategy is selected, the highest fractional intertask values are fixed to 1. When the second strategy is favored, the path flow variables with the highest fractional values are set to 1. In this case, decisions are imposed by removing the corresponding θ_p^u variables from the master problem, modifying its right-hand side accordingly, removing from the networks the arcs that bear the covered task, and finally, removing the corresponding task covering constraints from the master problem. In our implementation, a maximum of 10 decisions can be imposed at the same time for both strategies.

Finally, ϵ-bounded slack and surplus variables are used to perturb constraints (2)–(4) and thereby stabilize dual variables, as explained in Du Merle et al. (1999). This perturbation remains fixed during the solution process until no fractional intertask values nor path flow variables are greater than given thresholds. At that point, perturbation is removed. For our computational experiments, these thresholds were set to 0.9 and 0.8, respectively.

Two types of cutting planes are added to the master problem to reinforce linear relaxations throughout the search tree. The first type is only valid when all arc costs c_{ij} and the bus fixed cost c are integer. In that case, given a solution with a fractional value z, the following cutting plane is added to the master problem:

$$cB + \sum_{u\in U} \sum_{p\in\Omega^u} c_p \theta_p^u \geq \lceil z \rceil, \qquad (17)$$

where $\lceil z \rceil$ denotes the smallest integer greater than or equal to z. The second type of cutting plane is applicable when the primary objective is to minimize the number of buses and drivers needed to perform all trips. In that case, given a solution with n_b buses and n_d drivers with $n_b + n_d$ fractional, the following cutting plane is added to the master problem:

$$B + \sum_{u\in U} \sum_{p\in\Omega^u} \theta_p^u \geq \lceil n_b + n_d \rceil. \qquad (18)$$

Note that similar cuts can be devised when only the number of buses or the number of drivers needs to be minimized. Obviously, when adding any of these cutting planes, the dual variables associated with them are also transferred appropriately to the subproblem objective functions. The reader is referred to Desrosiers et al. (l984) for an earlier work on the impact of these cuts on the solution process.

3.3. Vehicle Schedules

Given a solution to the VCSP formulation (1)–(6), one can easily find corresponding optimal vehicle schedules in polynomial time. Indeed, constraints (2)–(4) have ensured flow conservation for vehicles outside the depot. Therefore, following driving movements in the VCSP solution, one can retrieve vehicle tours starting and ending at the depot. In order to obtain a minimal number of buses, a simple first-in, first-out procedure can be used to combine these vehicle tours while preserving overall optimality. Therefore, given a crew solution and flow conservation for buses outside of the depots, assigning buses to fixed scheduled tours becomes a simple application of the Dilworth's chain decomposition theorem for partially ordered sets, as explained in Ford and Fulkerson (1962, p. 67).

4. Computational Experimentation

This section presents the computational experimentation that was conducted in order to assess the efficiency of the formulation and approach proposed to model and solve the VCSP in the context of urban mass transit systems. The experimentation consisted in solving randomly generated VCSP instances that reflect the main features of real-life VCSPs. In order to be able to compare our results with Freling's results (1997), the generation process was designed to yield instances with characteristics similar to those of Freling's randomly generated instances. Furthermore, we adopted all work rules proposed by Freling. Section 4.1 describes how the problems were generated, while §4.2 specifies the work rules defining the set of duty types. The latter also identifies the resources needed to model these rules. Finally, §4.3 presents the computational results obtained.

4.1. Generation of Instances

All VCSP instances used to test the proposed solution approach were randomly generated as described in this section. These data sets are available from the authors upon request.

Freling (1997) has used the instance generator of Dell'Amico et al. (1993) to generate bus trips.

This generator does not rely on the specific structure of a bus line system and is not suitable in our opinion to define adequate line intersections. Consequently, in order to provide a more realistic experimental investigation, we have chosen to randomly generate VCSP instances using the bus line system illustrated in Figure 2. This system is composed of eight lines defined in pairs (lines I–II, III–IV, V–VI, and VII–VIII), where lines of a pair correspond to the same route but in opposite directions. Each line contains nine relief points, including its starting and ending locations. However, depending on the scenario tested, some of these relief points might not be considered. Note that four of these relief points are at the intersection of bus lines, thus allowing for quick driver exchange.

Given a number of trips to operate along that line system, the timetable of each trip was generated as follows. First, a line on which this trip operates is chosen from the set of possible lines according

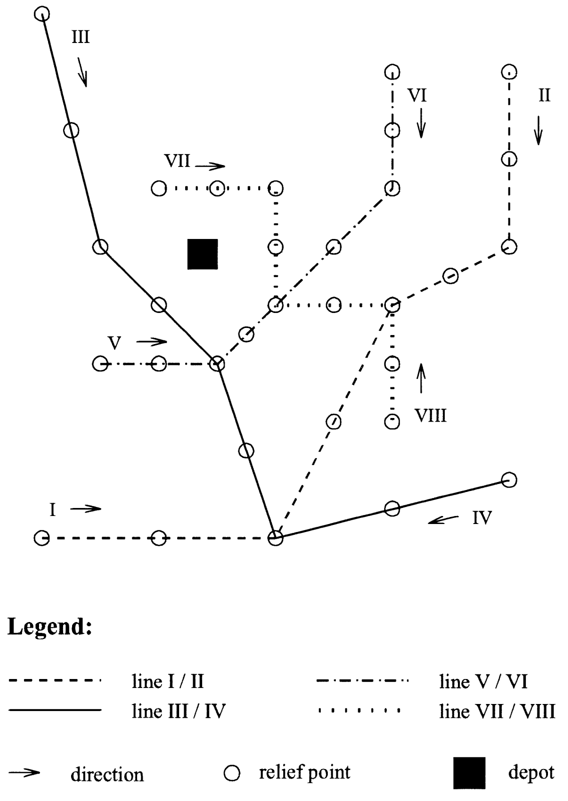

Figure 2 Bus Line System

to a uniform distribution. Then, its starting hour is determined according to the probability distribution given in Table 1. Note that it is assumed that peak hours occur from 7:00 to 9:59 in the morning and from 4:00 to 6:59 in the afternoon. Finally, the trip starting minute is selected from a uniform distribution over $\{0, 1, \ldots, 59\}$.

The duration of a trip is calculated as the sum of the durations from one relief point, considered or not, to the next along that trip. The duration of a bus movement (measured in minutes) from one location to another (either along a trip, along an intertrip, from or to the depot) is equivalent to the Euclidean distance between these locations truncated to the nearest integer. As in Freling (1997), we assume that walking always takes 10 minutes more than driving and that sign-on and sign-off periods last 10 minutes.

Again, as in Freling (1997), the objective of the problem is to minimize first the sum of buses and drivers required to cover all trips and d-trips and then the bus operational costs. Therefore, a fixed cost of 50,000 units is imposed for each bus and each driver used, and a unit cost is charged for each minute that a bus is away from the depot.

4.2. Modeling Work Rules Using Resource Variables

This section presents the work rules defining the duty types, as well as the resources required to model them.

In accordance with Freling (1997), we consider two types of duty differing mainly by the number of pieces of work a duty contains. The work rules of these two duty types are described in Tables 2 and 3. Minimum and maximum values are given for the number of pieces of work contained in a duty, the length of a duty, the length of each piece of work, the length of a break, and the total work time in a duty that corresponds to the time spent driving or attending to a bus. Therefore, only two networks are

Table 1 Trip Starting Hour Probability Distribution

Hours (a.m.)	0	1	2	3	4	5	6	7	8	9	10	11
Prob. (%)	0	0	1	1	1	3	5	9	10	8	5	4
Hours (p.m.)	0	1	2	3	4	5	6	7	8	9	10	11
Prob. (%)	3	3	4	5	9	10	8	5	3	2	1	0

Table 2 Work Rules for Duty Type I

	Minimum	Maximum
No. of pieces	1	1
Duty length (min.)	15	300

necessary—one to generate duties of type I and the other to generate duties of type II. These networks are denoted by $G^I = (N^I, A^I)$ and $G^{II} = (N^{II}, A^{II})$, respectively, that is, $U = \{I, II\}$.

Seven resources are needed to model the work rules: *min_p_of_w_nb* and *max_p_of_w_nb* to set the exact number of pieces of work in a duty; *max_duty_length*, *max_break_length*, and *max_work_time* to bound from above the length of a duty, of a break, and the time worked in a duty, respectively; *min_p_of_w_length* and *max_p_of_w_length* to restrict the length of a piece of work. Note that no resources are needed to model the minimum break length, since it is directly included in the network structure. Furthermore, since no duties consisting only of walking movements can be part of an optimal solution (such a duty would have a positive cost while contributing to none of the constraints), all optimal duties of type I necessarily contain a piece of work and those of type II, two pieces of work and a break. Therefore, the minimum duty length and work time constraints are redundant with the minimum piece of work length and minimum break length constraints and do not need to be taken into account.

G^I requires only the *max_duty_length* resource, numbered 3 in the following. The resource interval $[a_i^{I,3}, b_i^{I,3}]$ is the same for all nodes $i \in N^I$, namely [0, 300]. The resource consumption $t_{ij}^{I,3}$ along arc $(i, j) \in A^I$ is given by $\tau_j - \tau_i$, except when (i, j) is a *start_of_duty* or an *end_of_duty* arc, in which case it is given by b_{ij} or w_{ij}, depending on whether (i, j) represents a driving or a walking movement.

Table 3 Work Rules for Duty Type II

	Minimum	Maximum
No. of pieces	2	2
Duty length (min)	45	600
Piece length (min)	15	300
Break length (min)	15	90
Work time (min)	30	480

G^{II} needs all resources to model the work rules applicable to duty type II. Table 4 provides the resource interval $[a_i^{II,r}, b_i^{II,r}]$ for each resource $r \in R^{II}$ and each node $i \in N^{II}$ grouped by node type. Related resource consumptions for each arc type are presented in Table 5, where indices i and j refer to an arc $(i, j) \in A^{II}$. In this table, τ_{ij} is equal to b_{ij} if (i, j) corresponds to a driving movement, and w_{ij} otherwise; β_{ij} is equal to 15, the minimum break time if (i, j) represents a break at the depot, and 0 otherwise; ω_{ij} is equal to b_{ij} if (i, j) corresponds to a driving movement, and 0 otherwise; and ω_{ij}^2 is equal to $\tau_j - \tau_i$ if (i, j) represents a driving movement or a bus attendance task, and 0 otherwise.

4.3. Computational Results

All randomly generated VCSP instances considered to analyze the performance of our approach were solved using GENCOL software (developed at the GERAD research center in Montréal, Canada), which relies on the optimizer CPLEX to solve linear programs. This software package uses the methodology described in §3 to solve various types of deterministic vehicle routing and crew scheduling problems with resource variables. All computational results presented hereafter were obtained on a SUN ULTRA-10/440 workstation (22.7 SPECfp95, 18.1 SPECint95) using a single processor. The maximum memory required was 125 Mb for the 350-trip instances.

Test experiments were conducted in three steps. First, linear relaxations of VCSPs were solved to confirm that some of the strategies presented in §3.1.1 were indeed accelerating computational times, Second, the exact solution approach was tested on VCSPs involving various numbers of trips and d-trips. Last, the heuristic version of the algorithm was tested on larger VCSPs.

For all VCSPs considered, 10 instances were randomly generated as described in §4.1 Therefore, all results correspond to averages over sets of 10 VCSP instances. In all cases, a maximum of 3 hours of CPU time was allowed to solve each instance.

4.3.1. Linear Relaxation Results. A first series of tests was conducted to confirm that the dynamic generation of bus count constraints and the substitution

Table 4 **Resource Intervals for Each Node Type in Network G^{II}**

	min p_of_w nb	max p_of_w nb	max duty length	max break length	max work time	min p_of_w length	max p_of_w length
source	[0, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[0, 0]	[0, 300]
sink	[−2, −2]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[−15, −15]	[0, 300]
start_of_trip	[−2, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[−15, 0]	[0, 300]
end_of_trip	[−2, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[−15, 0]	[0, 300]
relief	[−2, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[−15, 0]	[0, 300]
start_of_break	[−2, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[−15, −15]	[0, 300]
min_break	[−2, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[−15, 0]	[0, 300]
end_of_break	[−2, 0]	[0, 2]	[0, 600]	[0, 90]	[0, 480]	[0, 0]	[0, 300]

for covering constraints (2) and (4) by flow conservation constraints (15) and (16) are indeed accelerating computational times. Since both strategies aim at reducing the time needed to solve the restricted master problem at each column generation iteration, the tests were performed only on the linear relaxations of the VCSPs. Therefore, the benchmark consisted in solving the linear relaxations of VCSPs using four variants of the exact solution approach: (i) without these two strategies (STD); (ii) with only the flow conservation constraints (FC); (iii) with only the bus dynamic constraints (DC); and (iv) with both accelerating strategies (FC & DC). Two sets of VCSP instances were used—VCSPs containing 50 trips with eight d-trips per trip and VCSPs containing 150 trips with two d-trips per trip. The first set should highlight the efficiency of the FC strategy since its instances contain a large number of task-covering constraints to be replaced by flow conservation constraints. The second set should reveal the positive impact of the DC strategy on solution time since its instances contain a large number of bus dynamic constraints.

Table 6 presents the results obtained from these tests. For each variant of the exact solution approach and for each set of VCSPs, we report the average total CPU time (in minutes), the average CPU time per column generation iteration (in minutes), the average number of nonzeros in the constraint coefficient matrix, and, when applicable, the average number of dynamic constraints generated during the solution process (all bus constraints were initially included in the problems when the DC strategy was not used). The average numbers of bus count constraints were 18 and 57 for the first and second sets of VCSP instances, respectively.

As one can see, using the FC strategy is always profitable, especially for the 50-trip VCSPs with eight d-trips per trip, where it yields an average gain of more than 25% in solution time. As expected, the DC

Table 5 **Resource Consumptions for Each Arc Type in Network G^{II}**

	min p_of_w nb	max p_of_w nb	max duty length	max break length	max work time	min p_of_w length	max p_of_w length
start_of_duty	−1	1	$\tau_{ij}+10$	0	ω^1_{ij}	$-\omega^1_{ij}$	ω^1_{ij}
end_of_duty	0	0	$\tau_{ij}+10$	0	ω^1_{ij}	$-\omega^1_{ij}$	ω^1_{ij}
d-trip	0	0	$\tau_j-\tau_i$	0	b_{ij}	$-b_{ij}$	b_{ij}
inter-trip	0	0	$\tau_j-\tau_i$	0	ω^2_{ij}	$-\omega^2_{ij}$	ω^2_{ij}
from_break	0	0	$\tau_j-\tau_i$	0	ω^1_{ij}	$-\omega^1_{ij}$	ω^1_{ij}
to_break	0	0	$\tau_j-\tau_i$	−90	ω^1_{ij}	$-\omega^1_{ij}$	ω^1_{ij}
break	−1	1	$\tau_j-\tau_i$	β_{ij}	0	0	−300
new_p_of_w	0	0	0	0	0	0	0
wait	0	0	$\tau_j-\tau_i$	$\tau_j-\tau_i$	0	0	0

Table 6 Linear Relaxation Results

50 Trips and 8 d-Trips per Trip	STD	FC	DC	FC & DC
LP CPU time (min)	17.8	13.3	19.2	13.0
CPU time per iteration (min)	2.1	1.4	2.1	1.3
No. of nonzeros per column	15.8	8.8	12.3	5.5
No. of dynamic constraints generated (/18)	—	—	0.7	0.6
150 Trips and 2 d-Trips per Trip	**STD**	**FC**	**DC**	**FC & DC**
LP CPU time (min)	8.8	7.4	6.9	5.7
CPU time per iteration (min)	4.5	3.7	3.1	2.6
No. of nonzeros per column	24.5	21.6	9.3	6.6
No. of dynamic constraints generated (/57)	—	—	2.8	2.6

strategy speeds up the average solution time for the 150-trip VCSPs, with an average gain of more than 21%. However, this strategy alone is not efficient, on average, for the 50-trip VCSPs since the number of bus constraints is rather low. Note that in this case, the overall solution time is increased since additional iterations are required after the introduction of violated dynamic constraints. Finally, note that combining both strategies yields a much less dense coefficient matrix (the average number of nonzero coefficients decreases by 65% and 73% on the first and second sets, respectively), resulting in reduced solution times.

4.3.2. Exact Approach Results. The exact solution approach was tested on VCSPs involving between 50 and 150 trips, each of them containing two d-trips, and on 50-trip VCSPs with an increasing number of d-trips. The goal of these tests is to study the impact of the number of trips and the number of d-trips per trip on solution time.

Tables of results are presented in this subsection and the next one. Each table is divided into four parts. The first part provides information regarding problem size: the numbers of trips and d-trips; the average numbers of nodes and arcs contained in network G^{II}; the number of tasks to cover, i.e., the number of constraints (2)–(4); the average number of bus constraints (5); and the average total number of constraints contained in the master problem (including the dynamic bus constraints). Next, the second part presents information regarding the solution process: the average numbers of column generation iterations, columns generated, branch-and-bound nodes, bus constraints added dynamically, cuts on total cost (17), and the total number of buses and drivers (18). The third part reports various solution times in minutes: the average CPU times needed to solve the first linear relaxation, to find the best feasible solution, and to complete the solution process, as well as the minimum and maximum total CPU times over the instances solved. Finally, the last part provides information on the quality of the computed solutions: the average numbers of buses and drivers required in the solution; the average integrality gap; the maximum integrality gap over the instances solved; the average optimality gap; the maximum optimality gap over the instances solved; and the number of instances solved to optimality within the 3-hour CPU time limit. These last three statistics are given only for the exact approach results. The integrality gap is computed as the difference between the value of the best solution found and that of the first linear relaxation solution, divided by the latter. The optimality gap is computed as the difference between the value of the best solution found and the smallest bound associated with an unexplored node in the search tree, divided by that bound.

Note that, given the high fixed cost imposed for using a bus or a driver, the gap values essentially reflect this aspect of the problem, as in Freling (1997).

Table 7 shows the results obtained using the exact approach on problems involving between 50 and 150 trips, each of them containing two d-trips. Observe that for the larger instances, the network size contains almost 30,000 nodes and over 45,000 arcs, while the master problem involves more than 650 rows. An integer solution, not necessarily optimal, was found within 3 hours of CPU time for all instances treated.

Table 7 Exact Approach Results (Two D-Trips per Trip)

No. of trips	50	75	100	125	150
No. of d-trips	100	150	200	250	300
No. of nodes	3,454	7,765	13,532	20,787	29,421
No. of arcs	5,950	12,696	21,961	33,343	46,810
No. of tasks	200	300	400	500	600
No. of bus constraints	20	31	40	51	59
No. of MP constraints	220	331	440	551	659
No. of CG iterations	1,029	4,368	6,510	3,875	1,727
No. of columns generated	31,747	205,974	437,409	290,945	184,517
No. of B & B nodes	99	260	441	290	190
No. of bus constraints generated	1.1	1.9	3.6	1.7	2.7
No. of cuts on cost	3.7	42.5	6.8	4.2	3.9
No. of cuts on buses & drivers	0.7	0.4	0.4	0.5	0.7
LP CPU time (min)	0.1	0.4	1.3	2.9	5.6
Best solution CPU time (min)	0.4	2.9	5.8	25.7	18.4
Total CPU time (min)	1.6	21.1	91.0	98.5	81.9
Min. total CPU time (min)	0.1	0.5	1.1	3.1	7.2
Max. total CPU time (min)	9.1	180.0	180.0	180.0	180.0
No. of buses	14.2	20.4	24.9	30.2	35.1
No. of drivers	24.1	34.0	43.4	52.6	62.3
Integrality gap (%)	0.6	0.3	0.3	0.1	0.5
Max. integrality gap (%)	1.8	2.4	1.5	0.7	1.9
Optimality gap (%)	0	0.2	0.1	<0.05	0.3
Max. optimality gap (%)	0	1.9	1.3	<0.05	1.2
Number solved to optimality	10	9	5	5	6

The optimality gap does not exceed 1.9%, with averages ranging from 0% to 0.3%. Note also that the best feasible solutions are found, on average, early in the solution process.

Results derived from applying the exact solution approach to 50-trip problems involving between 100 and 400 d-trips are presented in Table 8. As anticipated, these results show that the number of d-trips per trip affects solution time. In particular, the time needed to solve the first linear relaxation increases quite rapidly with the number of d-trips per trip. Again, note that the best feasible solution is found rapidly on average.

4.3.3. Heuristic Approach Results. The heuristic version of the algorithm was tested on VCSPs involving up to 350 trips, each of them containing two d-trips. The results of these tests are presented in Table 9. Note that on average, much less CPU time is needed to solve the first linear relaxation of the 150-trip problems compared to the time needed for the exact approach (3.9 minutes compared to 5.6 seconds taken from Table 7). This large reduction in CPU time is solely due to the use of master problem perturbation, which helps in reducing degeneracy and stabilizing dual variables. However, as it is actually implemented in GENCOL, perturbation turns out to be conflicting with efficient branching strategies for small-sized problems. For this reason, it was not included in the exact approach. Note also that, when compared to the exact solution approach, the heuristic approach exhibits less variance of the total computational times.

Given the size of the problems treated and the times needed to compute the solutions, the quality of the heuristic solutions is very satisfying. In fact, the average integrality gap ranges from 0.3% to 1.3% with an overall maximum of 5.7%. When compared to Freling's results (1997) for the randomly generated problems (gaps ranging between 3.8% and 25.9% for 40–120-trip problems, where each trip contains a single d-trip), these results are clearly superior.

Table 8 **Exact Approach Results (Varying Number of D-Trips per Trip)**

No. of trips	50	50	50	50
No. of d-trips	100	200	300	400
No. of nodes	3,454	9,218	17,800	29,090
No. of arcs	5,950	14,970	28,215	45,540
No. of tasks	200	300	400	500
No. of bus constraints	20	20	20	20
No. of MP constraints	220	320	420	520
No. of CG iterations	1,029	208	2,160	1,861
No. of columns generated	31,747	18,071	175,196	177,910
No. of B & B nodes	99	33	89	114
No. of bus constraints generated	1.1	0.6	0.6	0.6
No. of cuts on cost	3.7	0	2.4	0.5
No. of cuts on buses & drivers	0.7	0.1	0.1	0.2
LP CPU time (min)	0.1	0.9	3.4	13.1
Best solution CPU time (min)	0.4	1.4	7.3	25.0
Total CPU time (min)	1.6	1.4	25.0	38.8
Min. total CPU time (min)	0.1	0.6	1.9	4.8
Max. total CPU time (min)	9.1	2.5	180.0	180.0
No. of buses	14.2	14.1	14.0	14.2
No. of drivers	24.1	22.6	22.8	21.7
Integrality gap (%)	0.6	0.2	0.2	0.2
Max. integrality gap (%)	1.8	1.9	2.0	2.1
Optimality gap (%)	0	0	<0.05	<0.05
Max. optimality gap (%)	0	0	<0.05	<0.05
Number solved to optimality	10	10	9	9

Table 9 **Heuristic Approach Results (Two D-Trips per Trip)**

No. of trips	150	200	250	300	350
No. of d-trips	300	400	500	600	700
No. of nodes	29,421	52,095	80,793	115,785	157,347
No. of arcs	46,810	81,950	126,259	180,124	243,982
No. of tasks	600	800	1,000	1,200	1,400
No. of bus constraints	59	78	97	121	136
No. of MP constraints	659	878	1,097	1,321	1,536
No. of CG iterations	221	317	357	386	476
No. of columns generated	11,806	17,326	22,175	30,371	35,537
No. of B & B nodes	60	96	127	122	171
No. of bus constraints generated	3.8	6.9	13.3	11.1	15.0
No. of cuts on cost	6.2	10.7	13.3	13.1	21.1
No. of cuts on buses & drivers	0.8	1.6	0.6	0.9	0.7
LP CPU time (min)	3.9	10.0	20.0	41.8	66.4
Total CPU time (min)	5.3	14.5	33.0	62.1	117.1
Min. total CPU time (min)	3.8	8.9	21.1	35.1	85.9
Max. total CPU time (min)	8.5	20.3	49.8	93.5	181.2
No. of buses	35.1	44.8	54.2	63.7	66.6
No. of drivers	62.4	80.5	99.4	115.7	121
Integrality gap (%)	0.6	1.3	0.3	0.4	0.3
Max. integrality gap (%)	2.1	5.7	1.0	1.1	1.5

References

Ball, M., L. Bodin, R. Dial. 1983. A matching based heuristic for scheduling mass transit crews and vehicles. *Transportation Sci.* **17** 4–31.

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **46** 316–329.

Bertossi, A. A., P. Carraresi, G. Gallo. 1989. On some matching problems arising in vehicle scheduling models. *Networks* **17** 271–281.

Dell'Amico, M., M. Fischetti, P. Toth. 1993. Heuristic algorithms for the multiple depot vehicle scheduling problem. *Management Sci.* **39** 115–125.

Desaulniers, G., J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, F. Soumis. 1997. Crew Pairing at Air France. *Eur. J. Oper. Res.* **97** 245–259.

———, ———, I. Ioachim, M. M. Solomon, F. Soumis, D. Villeneuve. 1998. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. T. G. Crainic, G. Laporte, eds. *Fleet Management and Logistics*. Kluwer, Norwell, MA, 57–93.

Desrochers, M. 1986. La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes. Ph.D. thesis, Université de Montréal, Montréal, Canada. (In French.)

———, J. Gilbert, M. Sauvé, F. Soumis. 1992. CREW-OPT: Subproblem modeling in a column generation approach to urban crew scheduling. M. Desrochers, J.-M. Rousseau, eds. *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems 386*. Springer-Verlag, Berlin, Germany, 395–406.

———, F. Soumis. 1989. A column generation approach to the urban transit crew scheduling problem. *Transportation Sci.* **23** 1–13.

Desrosiers, J., Y. Dumas, M. M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M. O. Ball et al., eds. *Network Routing. Handbooks in Operations Research and Management Science 8*, Elsevier Science, Amsterdam, The Netherlands, 35–139.

———, F. Soumis, M. Desrochers. 1984. Routing with time windows by column generation. *Networks* **14** 545–565.

Dumas, Y., J. Desrosiers, F. Soumis. 1991. The pickup and delivery problem with time windows. *Eur. J. Oper. Res.* **54** 7–22.

du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen. 1999. Stabilized column generation. *Discrete Math.* **194** 229–237.

Fischetti, M., S. Martello, P. Toth. 1989. The fixed job schedule problem with working-time constraints. *Oper. Res.* **37** 395–403.

Ford, L., D. R. Fulkerson. 1962. *Flows in Networks*. Princeton University Press, Princeton, NJ.

Freling, R. 1997. Models and techniques for integrating vehicle and crew scheduling. *Tinbergen Inst. Res. Ser.* 157, Thesis Publishers, Amsterdam.

———, G. Boender, A. Paixão. 1995. An integrated approach to vehicle and crew scheduling. Report 9503/A, Erasmus University, Rotterdam, The Netherlands.

———, A. P. M. Wagelmans, A. Paixão. 1999. An overview of models and techniques for integrating vehicle and crew scheduling. N. H. M. Wilson, ed. *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems 471*. Springer, Berlin, Germany, 441–460.

Gamache, M., F. Soumis, G. Marquis, J. Desrosiers. 1999. A column generation approach for large scale aircrew rostering problems. *Oper. Res.* **47** 247–263.

———, ———, D. Villeneuve, J. Desrosiers, E. Gélinas. 1998. The preferential bidding system at Air Canada. *Transportation Sci.* **32** 246–255.

Haase, K., C. Friberg. 1999. An exact branch and cut algorithm for the vehicle and crew scheduling problem. N. H. M. Wilson, ed. *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems 471*. Springer, Berlin, Germany, 63–80.

Hoffman, K. L., M. Padberg. 1993. Solving airline crew scheduling problems by branch-and-cut. *Management Sci.* **39** 657–682.

Klabjan, D. 2000. Topics in airline crew scheduling and large optimization. Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA.

Löbel, A. 1999. Recent computational developments for large-scale multiple-depot vehicle scheduling problems. N. H. M. Wilson, ed. *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems 471.* Springer, Berlin, Germany, 193–220.

Mingozzi, A., M. Boschetti, S. Ricciardelli, L. Bianco. 1999. A set partitioning approach to the crew scheduling problem. *Oper. Res.* **47** 873–888.

Patrikalakis, I., D. Xerocostas. 1992. A new decomposition scheme of the urban public transport scheduling problem. M. Desrochers, J.-M. Rousseau, eds. *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems 386*. Springer-Verlag, Berlin, Germany, 407–425.

Ribeiro, C. C., F. Soumis. 1994. A column generation approach to the multiple-depot vehicle scheduling problem. *Oper. Res.* **42** 41–52.

Ryan, D. M., B. A. Foster. 1981. An integer programming approach to scheduling. A. Wren, ed. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*. North-Holland, Amsterdam, The Netherlands, 269–280.

Tosini, E., C. Vercellis. 1988. An interactive system for extra-urban vehicle and crew scheduling problems. J. R. Daduna, A. Wren, eds. *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems 308*. Springer-Verlag, Berlin, Germany, 41–53.

Received: October 1998; revisions received: October 1999, February 2001; accepted: March 2001.