European Journal of Operational Research 54 (1991) 7-22
North-Holland

Theory and Methodology

The pickup and delivery problem with time windows

Yvan Dumas, Jacques Desrosiers and François Soumis
GERAD, 5255 avenue Decelles, Montréal, Canada

Received May 1989; revised February 1990

Abstract: The vehicle routing problem (VRP) involves the design of a set of minimum cost routes for a fleet of vehicles which services exactly once a set of customers with known demands. The pickup and delivery problem with time windows (PDPTW) is a generalization of the VRP which is concerned with the construction of optimal routes to satisfy transportation requests, each requiring both pickup and delivery under capacity, time window and precedence constraints. This paper presents an exact algorithm which solves the pickup and delivery problem when transporting goods. This algorithm uses a column generation scheme with a constrained shortest path as a subproblem. The algorithm can handle multiple depots and different types of vehicles.

Keywords: Transportation, routing, scheduling, column generation

1. Introduction

The vehicle routing problem (VRP) involves the design of a set of minimum cost routes, originating and terminating at a depot, for a fleet of vehicles which services exactly once a set of customers with known demands. The VRP area has been intensively studied in the literature. We refer the reader to [1,18,20] for comprehensive surveys of the VRP and its variations which also describe the many practical occurences of the problem. In the VRP with time windows (VRPTW), the above issues have to be dealt with under the added complexity of allowable delivery times, or time windows. The service of a customer can begin within the time window defined by the earliest time and the latest time which the customer will allow the service to start. Note that the times at which services begin are decision variables. The pickup and delivery problem with time windows (PDPTW) is a generalization of the VRPTW which is concerned with the construction of optimal routes to satisfy transportation requests, each requiring pickup at the origin and delivery at the destination under capacity, time window and precedence constraints. In addition, each route satisfies pairing constraints since corresponding pickup and delivery locations must be serviced by the same vehicle. Note that the VRPTW is the particular case of PDPTW where the destinations are all to the common depot. We refer the reader to [6,29] for recent surveys on time window constrained routing and scheduling problems. Note however that while heuristics have been found very effective in solving practical size VRPTW, optimal approaches have lagged far behind. The only exact method [17] for the VRPTW extend the shortest q-path relaxation algorithm [4] to the problem with time windows. The largest problems solved to optimality involved 4 vehicles servicing 14 customers and 3 vehicles servicing 15 customers with tight time windows.

Most of the literature on pickup and delivery problems has appeared for the dial-a-ride problem (DARP). It can be seen that the time constrained single vehicle (DARPTW) is a constrained version of the classical travelling salesman problem. Optimal dynamic programming algorithms [10,22,23] and a heuristic procedure based on Benders' decomposition [26–28] have been used to solve this single vehicle problem. For the multi-vehicle DARPTW version, many heuristics have been designed to successively solve very large scale problems of over 2000 requests. These approximation methods are based on parallel insertion procedures [16,24,25], traditional 'cluster first, route second' approach [2] and finally, mini-clustering first and optimal routing second solved by column generation [11].

The aim of this paper is to present an exact algorithm to solve *some instances* of the PDPTW when transporting goods. This algorithm works well for problems for which the demand at each customer is large, i.e., when the capacity constraints are restrictive. It is not designed to solve large scale dial-a-ride problems. This algorithm uses a column generation scheme described below with a constrained shortest path as a subproblem. This algorithm can handle multiple depots and different types of vehicles.

To simplify the presentation, consider a homogeneous fleet of vehicles where all vehicles have the same characteristics. The following notation is used:

n : the number of transportation requests;
V : the set of available vehicles;
Ω : the set of admissible routes;
X_r : a binary variable; 1 if route r is used, 0 otherwise;
a_{ir} : a binary constant; 1 if route r includes request i, 0 otherwise;
c_r : the cost of route r.

When each transportation request is satisfied exactly once, the problem can be formulated as follow:

$$\text{Min} \quad \sum_{r\in\Omega} c_r X_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r\in\Omega} a_{ir} X_r = 1, \quad i = 1, 2, \ldots, n, \tag{2}$$

$$\sum_{r\in\Omega} X_r = |V|, \tag{3}$$

$$X_r \text{ binary}, \quad r \in \Omega. \tag{4}$$

The column vector $[a_{1r}, a_{2r}, \ldots, a_{nr}, 1]^T$ corresponds to an admissible route. The number of columns in Ω is generally too great to allow exhaustive enumeration. A column generation method is used to solve the linear relaxation of this set partitioning type problem; the bound obtained in this way is generally excellent and constitutes the starting point for an enumeration tree. Columns in Ω are generated as needed by solving the constrained shortest path problem which defines an admissible route.

When the constraints are restrictive, the column generation scheme has been successfully used to solve vehicle routing [11,13] and crew scheduling [9,19] problems. In each case, a different constrained shortest path problem is solved. The shortest path problem with time windows (SPPTW) has been extensively studied using dynamic programming algorithms: a Ford–Bellman type algorithm [12], a Dijkstra type algorithm [8], and a primal–dual reoptimization algorithm [7]. Finally, this above primal–dual approach has been generalized in [5] to the shortest path problem with multiple resource constraints. If one considers the multiple resource constraints as being vehicle capacity and time window constraints, this constrained shortest path algorithm can be used to solve the VRPTW by column generation.

Notice that all the exact methods presented in the literature review use dynamic programming algorithms either for obtaining an integer solution of a subproblem or for directly solving the global problem. We also propose a dynamic programming algorithm to solve the shortest path problem with pickup and delivery subject to capacity and time windows. In the following presentation, we emphasize the numerous state elimination criteria which result from the multiple constraints.

2. The pickup and delivery problem with time windows

We give the formulation for the PDPTW involving only a single depot and a homogeneous fleet. However, a formulation for more general variants has been considered in [14]. Note also that for a given depot and a given type of vehicles, the embedded constrained shortest path is the same as the one described later on. The notation and for-

mulation for the PDPTW will closely follow the one given in [29].

2.1. Notation

Let there be n customers indexed by i. Associate to the pickup location of customer i a node i and to his delivery location a node $n+i$. Also associate to the depot, nodes 0 and $2n+1$. This creates a clear distinction between customers, their associated locations and the nodes of the network. Note however that different nodes may correspond to the same physical location. Therefore, $N = \{0, 1, 2, \ldots, n, n+1, n+2, \ldots, 2n, 2n+1\}$ is the node set for our network, $P^+ = \{1, 2, \ldots, n\}$, $P^- = \{n+1, n+2, \ldots, 2n\}$ and $P = P^+ \cup P^-$ is the set of nodes other than the depot nodes.

Customer i demands that d_i units be shipped from node i to node $n+i$. Next, let $[a_i, b_i]$ denote the pickup time window for customer i and let $[a_{n+i}, b_{n+i}]$ denote his delivery time window. Let $V = \{1, 2, \ldots, |V|\}$ be the set of vehicles to be routed and scheduled; index this set by v. Let D be the capacity of each vehicle. Let also $[a_0, b_0]$ denote the vehicles' departure time window from the depot and $[a_{2n+1}, b_{2n+1}]$ their time window for arrival back at the depot. For each distinct i, j in N, let t_{ij} and c_{ij} represent the travel time and the travel cost from i to j, respectively, while s_i is the service time at node i. Retain only the arcs (i, j) which satisfy a priori capacity and time constraints, as given in the network construction (Subsection 2.4). Finally, let Γ be the fixed cost associated with each vehicle, incurred if this vehicle is utilized.

Three types of variables are used in the mathematical formulation: binary flow variables X_{ij}^v, $v \in V$, $i, j \in N$, $i \neq j$, time variables T_i, $i \in P$ and T_0^v, T_{2n+1}^v, $v \in V$, and load variables Y_i, $i \in P$. Let the binary decision variable be X_{ij}^v equal to one if vehicle v travels from node i to node j, and equal to zero otherwise, $v \in V$, and $i, j \in N$, $i \neq j$. Let T_i be the time at which service at node i begins, $i \in P$. Let also T_0^v be the time at which vehicle v leaves the depot and T_{2n+1}^v be the time at which it returns to the depot, $v \in V$. Next, let Y_i be the total load on the vehicle just after it leaves node i, $i \in P$. It is assumed that the vehicles depart empty from the depot, i.e., $Y_0 = 0$. These are decision variables with nonnegativity constraints.

Finally, let $g(Y_i) > 0$ denote a nondecreasing function of the total load transported on the vehicle just after it leaves node i, $i \in P$. This function will act as a penalty factor on the travel cost. We are now in position to present the mathematical formulation of the pickup and delivery problem with time windows.

2.2. PDPTW formulation

$$\text{Min} \sum_{v \in V} \sum_{i \in N} \sum_{j \in N} g(Y_i) \cdot c_{ij} \cdot X_{ij}^v \tag{5}$$

subject to

$$\sum_{v \in V} \sum_{j \in N} X_{ij}^v = 1, \quad i \in P^+, \tag{6}$$

$$\sum_{j \in N} X_{ij}^v - \sum_{j \in N} X_{ji}^v = 0, \quad i \in P, \ v \in V, \tag{7}$$

$$\sum_{j \in P^+} X_{0j}^v = 1, \quad v \in V, \tag{8}$$

$$\sum_{i \in P} X_{i,2n+1}^v = 1, \quad v \in V, \tag{9}$$

$$\sum_{j \in N} X_{ij}^v - \sum_{j \in N} X_{j,n+i}^v = 0, \quad i \in P^+, \ v \in V, \tag{10}$$

$$T_i + s_i + t_{i,n+i} \leqslant T_{n+i}, \quad i \in P^+, \tag{11}$$

$$X_{ij}^v = 1 \Rightarrow T_i + s_i + t_{ij} \leqslant T_j, \quad i, j \in P, \ v \in V, \tag{12}$$

$$X_{0j}^v = 1 \Rightarrow T_0^v + t_{0j} \leqslant T_j, \quad j \in P^+, \ v \in V, \tag{13}$$

$$X_{i,2n+1}^v = 1 \Rightarrow T_i + s_i + T_{i,2n+1} \leqslant T_{2n+1}^v, \quad i \in P^-, \ v \in V, \tag{14}$$

$$a_i \leqslant T_i \leqslant b_i, \quad i \in P, \tag{15}$$

$$a_0 \leqslant T_0^v \leqslant b_0, \quad v \in V, \tag{16}$$

$$a_{2n+1} \leqslant T_{2n+1}^v \leqslant b_{2n+1}, \quad v \in V, \tag{17}$$

$$X_{ij}^v = 1 \Rightarrow Y_i + d_j = Y_j, \quad i \in P, \ j \in P^+, \ v \in V, \tag{18}$$

$$X_{ij}^v = 1 \Rightarrow Y_i - d_{j-n} = Y_j, \quad i \in P, \ j \in P^-, \ v \in V, \tag{19}$$

$$X_{0j}^v = 1 \Rightarrow Y_0 + d_j = Y_j, \quad j \in P^+, \ v \in V, \tag{20}$$

$$Y_0 = 0, \qquad d_i \leqslant Y_i \leqslant D, \quad i \in P^+, \tag{21}$$

$$X_{ij}^v \text{ binary}, \quad i, j \in N, \ v \in V. \tag{22}$$

We seek to minimize the sum of the total travel cost. Constraints (6)–(9) and (21) form a multicommodity flow problem. Constaints (10) ensure that the same vehicle v visits both i and $n+i$. Constraints (11) are precedence constraints which force node i to be visited before node $n+i$. Next, constraints (12)–(14) describe the compatibility requirements between routes and schedules, while constraints (15)–(17) are the time windows constraints. Finally, constraints (18)–(20) express the compatibility requirements between routes and vehicle loads, while constraints (21) are the capacity constraints.

To also consider the minimization of the fleet size, one must add to the objective function the term $\sum_{v \in V} \sum_{j \in P^+} \Gamma \cdot X_{0j}^v$ and replace the equality sign with a less than or equal to sign in constraints (8) and (9) (also in constraints (3), in the set partitioning type formulation). Note that the cost Γ is usually incorporated into the values c_{0j}, $j \in P^+$.

The formulation includes a route duration restriction $(b_{2n+1} - a_0)$. Note also that along with the time window constraints, constraints (12)–(14) allow waiting time before visiting a node. This waiting time is not penalized and a possible evaluation of the times of arrival at the node of a route is given by

$$X_{ij}^v = 1 \Rightarrow T_j = \max\left[a_j,\ T_i + s_i + t_{ij}\right] \quad i,\ j \in P. \tag{23}$$

Finally, constraints (12)–(14) impose increasing times at the nodes of the route. Thus, any route will be *elementary*, i.e. without cycles. In fact, in linear form, these constraints generalize the subtour elimination constraints of Miller, Tucker and Zemlin [21] for the travelling salesman problem.

2.3. The shortest path problem with pickup, delivery and time window constraints

Solving the PDPTW using the set partitioning type formulation (1)–(4) requires admissible routes or columns. For a given vehicle $v \in V$, these are solutions of constraints (7)–(22) for which the marginal cost of a route is minimized. Note that for a given vehicle $v \in V$, the index v can be dropped everywhere. Constraints (7)–(22) describe a path starting and ending at the depot (7)–(9), under pairing constraints (10), precedence constraints (11), time constraints (12)–(17) and capacity constraints (18)–(21).

By solving the linear relaxation of the set partitioning type formulation, we obtain the dual variables π_i, $i = 1, \ldots, n$ and π_0 associated respectively to constraints (2) and (3). These dual variables are transferred to the nodes of the network for the generation of new admissible columns by associating to pickup nodes the values $\sigma_i = \pi_i$, $i = 1, \ldots, n$, and to delivery nodes the value $\sigma_{n+i} = 0$, $i = 1, \ldots, n$. In the same way, the dual variable associated with the departure node takes the value $\sigma_0 = \pi_0$ while $\sigma_{2n+1} = 0$ at the arrival node. This assignment of the dual variables to the nodes satisfy $\sigma_i + \sigma_{n+i} = \pi_i$ and $\sigma_0 + \sigma_{2n+1} = \pi_0$. This is consistent with the pairing constraints (10) which ensure that the same vehicle visits both nodes i and $n+i$. In the same manner, constraints (8) and (9) ensure that the vehicle starts at node 0 and ends at node $2n+1$.

Given the dual variables, the travel costs and the penalty factor on the travel cost, the objective function which minimizes the marginal cost of a path or admissible route is given by

$$\text{Min} \sum_{i \in N} \sum_{j \in N} \bar{c}_{ij}(Y_i) \cdot X_{ij}, \tag{24}$$

where the marginal cost of arc (i, j) is defined as

$$\bar{c}_{ij}(Y_i) = g(Y_i) \cdot c_{ij} - \sigma_i \quad i,\ j \in N. \tag{25}$$

The next subsection describes in detail the network construction, while Section 3 of the paper is devoted to the solution approaches of relations (24), (7)–(22) describing the shortest path problem with pickup and delivery subject to time window constraints.

2.4. Network construction

The network's set N of nodes is made up of pickup and delivery nodes as well as the vehicle departure and arrival nodes: $N = \{0, 1, \ldots, 2n+1\}$. The set of admissible arcs in the network is a subset of N^2; it is made up of arcs which satisfy a priori certain constraints of the problem. The network construction is a preprocessing step.

A preliminary step in the determination of the admissible arcs is the shrinking of the time windows associated with the pickup and delivery nodes. This is done by reducing the upper bounds

of the time windows so that for $i = 1, \ldots, n$ the partial paths $n+i \to 2n+1$ and $i \to n+i \to 2n+1$ are admissible for all values $T_i \in [a_i, b_i]$ and $T_{n+i} \in [a_{n+i}, b_{n+i}]$. The upper and lower bounds are successively redefined as

$$b_{n+i} := \min\{b_{n+i}, b_{2n+1} - t_{i,2n+1}\} \quad \text{and}$$

$$b_i := \min\{b_i, b_{n+i} - t_{i,n+i}\}, \ i = 1, \ldots, n;$$

$$a_i := \max\{a_i, a_0 + t_{0,i}\} \quad \text{and}$$

$$a_{n+i} := \max\{a_{n+i}, a_i + t_{i,n+i}\}, \ i = 1, \ldots, n.$$

The constraints are used to eliminate the following inadmissible arcs:

(a) *priority.* Arcs $(0, n+i)$, $(n+i, i)$, $(2n+1, 0)$, $(2n+1, i)$ and $(2n+1, n+i)$, $i = 1, \ldots, n$ are eliminated.

(b) *pairing.* Arcs $(i, 2n+1)$, $i = 1, \ldots, n$ are eliminated.

(c) *vehicle capacity.* If $d_i + d_j > D$, $i, j \in \{1, \ldots, n\}$, $i \neq j$, then the following arcs are eliminated: (i, j), (j, i), $(i, n+j)$, $(j, n+i)$, $(n+i, n+j)$ and $(n+j, n+i)$.

(d) *time windows.* If $a_i + s_i + t_{ij} > b_j$, $i, j \in \{1, \ldots, 2n\}$, then the arc (i, j) is eliminated.

(e) *time windows and pairing of requests.* If the travel times satisfy the triangle inequality, then for two requests $i, j \in \{1, \ldots, n\}$, $i \neq j$, some arcs are eliminated if they cannot be part of any path including both the pickup and delivery nodes for those requests, i.e. if, when the time of arrival at each node is evaluated according to relation (22), the time window constraints are not satisfied.

- Arc $(i, n+j)$ is eliminated if the path $j \to i \to n+j \to n+i$ is infeasible with $T_j = a_j$.
- Arc $(n+i, j)$ is eliminated if the path $i \to n+i \to j \to n+j$ is infeasible with $T_i = a_i$ (a special case of (d).
- Arc (i, j) is eliminated if the paths $i \to j \to n+i \to n+j$ and $i \to j \to n+j \to n+i$ are infeasible with $T_i = a_i$.
- Arc $(n+i, n+j)$ is eliminated if the paths $i \to j \to n+i \to n+j$ and $j \to i \to n+i \to n+j$ are infeasible with $T_i = a_i$, and with $T_j = a_j$.

(f) *same location.* If the travel costs satisfy the triangle inequality and if $c_{ij} = 0$, then an arc $(i, j) \in \{(u, w), (n+u, n+w), (u, n+w) \mid u, w \in \{1, \ldots, n\}, u \neq w\}$ is eliminated if $b_i + s_i + t_{ij} \geq b_j + s_j + t_{ji}$ and $a_i \geq a_j$ with at least one strict inequality, or if $b_i + s_i = b_j + s_j$, $a_i = a_j$ and $i < j$. This criterion allows the a priori imposition of a servicing order for the nodes i and j at the same location. The proof is presented in [10].

3. Solution of the constrained shortest path problem

The technique used to solve the constrained shortest path problem is a forward dynamic programming algorithm. This technique provides an integer solution and takes advantage of the numerous additional constraints. The more constraints you have the more efficient it is, given you cut away the state space.

3.1. Definition of the labels

There exists many paths arriving at a node. We denote by $\mathscr{P}_l^k$ the path number k from the departure of node 0 to node l. This path is admissible if it respects the time window, priority and capacity constraints, and if it satisfies the pairing constraints when $l = 2n+1$. A label (S_l^k, T_l^k, Z_l^k) is associated with the path $\mathscr{P}_l^k$, $l \in N$, where:
$S_l^k \subset N$ is the set of nodes visited on the path k,
$T_l^k \in [a_l, b_l]$ is the time of service at node l on the path k,
$Z_l^k \in \mathbb{R}$ is the sum of the costs on the path k.

This information is sufficient to calculate on this path the load Y_l^k of the vehicle at node l, since $Y_l^k = \sum_{i \in S_l^k} d_i$.

For the construction of vehicle routes using a set partitioning formulation, the columns correspond to admissible routes, and it is not necessary that the path be elementary. If a path r includes a cycle, thus satisfying a transportation request i more than once, the coefficient a_{ir} corresponding to this request in the column r will be an integer value greater than 1. This column cannot therefore become part of the integer solution of the set partitioning problem and will be rejected in an enumeration tree [13].

Let $R(S_l^k)$ be the subset of S_l^k containing the pickup nodes which have been visited, but whose corresponding delivery nodes have not been visited.

$$R(S_l^k) = \{i \in P^+ \mid i \in S_l^k \cap P^+ \text{ and } n+i \notin S_l^k \cap P^-\}.$$

A label $(R(S_l^k), T_l^k, Z_l^k)$ is then associated with the admissible path $\mathscr{P}_l^k$. The associated label contains sufficient information to allow the verification of the constraints on time windows, precedence, pairing and also capacity since $Y_l^k = \sum_{i \in R(S_l^k)} d_i$. However, it is not sufficient to construct elementary paths as $R(S_l^k)$ contains no information on the transportation requests whose pickup and delivery nodes have both been visited.

The dynamic programming approach considers, for each label, the state $(R(S_l^k), T_l^k)$ with a cost of Z_l^k. An attempt may be made to extend a path $\mathscr{P}_l^k$ to a node j if arc (l, j) exists: one then obtains a new path $\mathscr{P}_j^{k'}$ with associated label $(R(S_j^{k'}), T_j^{k'}, Z_j^{k'})$. This label is calculated as follows:

$$S_j^{k'} = S_l^k \cup \{j\} \tag{26}$$

$$R(S_j^{k'}) = \begin{cases} R(S_l^k) \cup \{j\} \\ \quad \text{if } j \in P^+ \text{ and } j \notin R(S_l^k), \\ R(S_l^k) \setminus \{n-j\} \\ \quad \text{if } j \in P^- \text{ and } n-j \in R(S_l^k) \end{cases} \tag{27}$$

$$T_j^{k'} = \max\{a_j, T_l^k + s_j + t_{lj}\} \quad \text{if } T_j^{k'} \leqslant b_j, \tag{28}$$

$$Z_j^{k'} = Z_j^k + \bar{c}_{lj}(Y_l^k). \tag{29}$$

To be admissible, this path must also satisfy the capacity constraint for a pickup point

$$Y_j^{k'} = Y_l^k + d_j \leqslant D \quad \text{if } j \in P^+. \tag{30}$$

The algorithm is initialized by the path $\mathscr{P}_0^1$ whose label is $(\emptyset, a_0, 0)$. The optimal solution of the shortest path (with the possibility of cycles) will be given by the path $\mathscr{P}_{2n+1}^{k^*}$ with a minimal cost value $Z_{2n+1}^{k^*}$: its label will be of the form $(\emptyset, T_{2n+1}^{k^*}, Z_{2n+1}^{k^*})$.

A path with a cycle will be obtained only if the network contains a negative cost cycle satisfy the same transportation requests twice while respecting the time windows for pickup and delivery. This kind of network is not common if the time windows associated with the nodes are small in comparison with the travel times between nodes.

3.2. Label elimination

In the following paragraphs, we present three state elimination methods based respectively on the notion of a non-post-feasible label, on the notion of dominance between labels, and on the order of treatment of the labels. The elimination of a label results in the elimination of the associated path.

A label associated with a path $\mathscr{P}_l^k$, admissible from node 0 to node l, which cannot be extended from node l to the return node $2n+1$ while respecting the time window and pairing constraints is called a *non post-feasible label*; such a label can be eliminated. This concept was introduced in [10], in the transportation of the handicapped.

Propositions 1 and 2 which follow are more powerful than criteria used for the a priori elimination of arcs in the network construction since the arrival time T_l^k at node l is known during the solution process.

Proposition 1. *A label $(R(S_l^k), T_l^k, Z_l)$ such that $R(S_l^k) \neq \emptyset$, $i \in R(S_l^k)$ and $T_l^k > a_i$ is eliminated if the path extension $l \to n+i \to 2n+1$ is infeasible.*

Proposition 2. *A label $(R(S_l^k), T_l^k, Z_l)$ such that $R(S_l^k) \neq \emptyset$, $i, j \in R(S_l^k)$ and $T_l^k > a_i$ is eliminated if both path extensions $l \to n+i \to n+j \to 2n+1$ and $l \to n+j \to n+i \to 2n+1$ are infeasible.*

Labels are tested for non post-feasibility when the vehicle load is non zero, i.e. when pickup nodes have been visited but the corresponding delivery nodes have not yet been visited. Ideally, this test should be carried out including all the delivery nodes which must be visited in the extension of $\mathscr{P}_l^k$. To do this, a travelling salesman problem with time window constraints on the set of nodes to be visited would have to be solved. In practice, the test is limited to a subset of at most two delivery nodes. However, it is not necessary to carry out a test with the arrival node $2n+1$, as this was done a priori during the construction of the network.

Label elimination due to dominance is carried out as a function of time and cost using the sets obtained with $R(\cdot)$.

Proposition 3. *If two labels are such that $R(S_l^k) = R(S_l^{k'})$, $T_l^k \leqslant T_l^{k'}$ and $Z_l^k \leqslant Z_l^{k'}$, then the label $(R(S_l^{k'}), T_l^{k'}, Z_l^{k'})$ can be eliminated.*

The proof is based on the fact that every feasible path extension of $\mathscr{P}_l^{k'}$ is also feasible for the

path $\mathscr{P}_l^k$ at a lesser cost. Various applications of this kind of dominance between labels as a function of time and cost values can be found in the literature [5,7,8,10,12,17], although the S_l^k dimension is not present.

By imposing additional conditions on the travel costs c_{ij}, further label eliminations are carried out as indicated by Proposition 4.

Proposition 4. *If two labels are such that* $R(S_l^k) \subset R(S_l^{k'})$, $T_l^k \leqslant T_l^{k'}$ *and* $Z_l^k \leqslant Z_l^{k'}$, *and if the cost values* c_{ij}, $i, j \in N$, *satisfy the triangle inequality, then the label* $(R(S_l^{k'}), T_l^{k'}, Z_l^{k'})$ *can be eliminated.*

Proof. Let $\mathscr{P}'_{l,2n+1}$ be the feasible path from l to $2n+1$ optimally extending the path associated with the label $(R(S_l^{k'}), T_l^{k'}, Z_l^{k'})$; if this extension does not exist, we can eliminate this label. A feasible extension of the path associated with the label $(R(S_l^k), T_l^k, Z_l^k)$ denoted by $\mathscr{P}_{l,2n+1}$ is obtained by visiting the nodes of $\mathscr{P}'_{l,2n+1}$ in the same order but excluding the first visit to the delivery nodes of the set $Q = \{q \in \mathscr{P}^- \mid q - n \in R(S_l^{k'}) \setminus R(S_l^k)\}$. Because of the pairing constraints, we remove the delivery nodes which cannot be visited in the extension $\mathscr{P}_{l,2n+1}$ as the corresponding pickup nodes are not included.

Consider the sequence of nodes $p \to q \to r$ visited in the extension $\mathscr{P}'_{l,2n+1}$ with $q \in Q$. In order for the elimination to be valid, it is sufficient to ensure that the marginal costs satisfy the inequality

$$\bar{c}_{pr}(Y_p) \leqslant \bar{c}_{pq}(Y'_p) + \bar{c}_{qr}(Y'_q),$$

where Y_p, Y'_p and Y'_q represent respectively the vehicle loads at node p in the extension $\mathscr{P}_{l,2n+1}$ and at nodes p and q in extension $\mathscr{P}'_{l,2n+1}$. In extension $\mathscr{P}'_{l,2n+1}$, we always have $Y'_p > 0$, as the delivery node q remains to be visited. It is easily deduced that $Y_p \leqslant Y'_q < Y'_p$. As the function $g(Y) > 0$ is non-decreasing and as the cost values satisfy the triangle inequality, then

$$g(Y_p) \cdot c_{pr} \leqslant g(Y'_g) \cdot c_{pq} + g(Y'_q) \cdot c_{qr}.$$

With $q \in Q$, we have $\sigma_q = 0$ and the marginal costs satisfy

$$g(Y_p) \cdot c_{pr} - \sigma_p \leqslant g(Y'_p) \cdot c_{pq} - \sigma_p + g(Y'_q) \cdot c_{qr} - \sigma_q,$$

which allows the elimination of the label $(R(S_l^{k'}), T_l^{k'}, Z_l^{k'})$. □

Proposition 4 can also be proven under the hypothesis that the costs $\bar{c}_{ij}(Y_j)$ satisfy the triangle inequality; however, it is stronger in the above form and it is in this form that we use it in the algorithm.

The third state elimination method is based on the order of treatment of the labels. Given the label $(\emptyset, a_0, 0)$ at node 0 at the first iteration, at iteration $K \geqslant 2$ of the algorithm we construct paths visiting exactly K nodes based on the paths generated in the preceding iteration. Once the labels from iteration $K-1$ have been treated, it is not necessary to store them to generate new labels and they are eliminated. For paths finishing at the arrival node $2n+1$ we store only the label associated with the least cost admissible path with a cardinality less than or equal to K: this path may constitute the optimal solution to the shortest path problem. Finally, labels are not treated one by one. The labels at a node with the same set $R(\cdot)$ are grouped together, thus facilitating the application of Proposition 3. The labels are sorted in increasing time order, which results in a decreasing cost order. The paths are extended simultaneously. This accelerates the verification of the capacity, priority, and pairing constraints and the elimination criteria. In the case of Proposition 4, the number of labels to be compared may be very large. Our program handles this difficulty by limiting the tests on the equal sets $R(Sk_l)$ and $R(S_l^{k'})$ (as in Proposition 3) and on the sets $R(S_l^k) \subset R(S_l^{k'})$ such that the cardinality of $R(S_l^k)$ is at most one element.

In most algorithms for shortest path problems, the paths are not stored explicitly but are reconstructed from a list of labels with pointers. However, this procedure requires the labels from previous iterations to be stored. In our application, the number of labels is generally very large and occupies considerable memory space. It is therefore preferable to eliminate the labels from previous iterations while retaining the paths.

4. Solution of the master problem

The set partitioning type problem defined by (1)–(4) is called the master problem while the

shortest path column generator (24), (7)–(22) is called the subproblem. In addition, let the restricted master problem be the linear relaxation of problem (1)–(4) defined on a subset $\Omega' \subset \Omega$ of columns.

4.1. Obtaining the linear relaxation solution

The master problem is solved by a column generation algorithm and a branch-and-bound exploration tree. Given a set of columns or admissible paths, the restricted master problem is solved using the simplex algorithm. This algorithm gives the dual variables necessary for column generation and lends itself well to reoptimization each time new columns are generated. The branch-and-bound tree is described in Subsection 4.2.

Initially, the restricted master problem contains exactly n columns, that is one vehicle for each request. At the start out, it is also possible to include a set of good columns formed with two requests by solving a matching problem. A new column in next generated by the subproblem and added to the restricted master problem which is in turn solved by the simplex algorithm. When no shortest paths can be generated with negative marginal cost, the optimal solution of the restricted master problem is also optimal for the linear relaxation of the master problem. This solution is a primal lower bound on the optimal integer solution. The next step is to augment it before using the branch-and-bound enumeration tree.

In our test problem, the fixed cost per vehicle, Γ, is large enough so that the number of vehicles used must be minimized. When the number of vehicles used, given by $\sum_r X_r = m$, is not integer in the optimal solution of the restricted master problem, a global cut is added to round up this number to at least the next integer:

$$\sum_{r \in \Omega} X_r \geqslant \lceil m \rceil. \tag{31}$$

The restricted master problem is then reoptimized by the dual simplex algorithm and new columns are generated as needed. The objective function of the subproblem is modified by subtracting the dual variable α associated to constraint (31).

$$\text{Min} \quad -\alpha + \sum_{i \in N} \sum_{j \in N} \left(g(Y_i) \cdot c_{ij} - \sigma_i \right) X_{ij}. \tag{32}$$

If the solution is still non-integer, and if the total travel cost, given by $\sum_r c_r X_r$ is fractional, a second cut is added to round up this quantity to the next integer. As in the previous case, the objective function of the subproblem is also modified by subtracting the dual variable associated to the second cut, and new columns are generated. At the end of the column generation scheme, including the use of the two additional cuts, the procedure gives out Z_{inf} which is a primal lower bound on the objective function value.

4.2. Obtaining integer solutions

The branch-and-bound enumeration tree must be done in such a way that it is possible to transfer, from the restricted master problem to the subproblem, the branching information so as to generate new admissible paths as needed. Note that it is impossible to directly fix variables X_r at 0 or 1 without any changes in the subproblem structure.

One possible branching strategy sets at 0 or 1 the variables X_{ij} associated with the arcs which make up a path [3,13,18]. Suppose that variable X_r is fractional and that the associated path is given by $0 \to 1 \to 2 \to n+2 \to n+1 \to 2n+1$. Six branches are created, as follows:

$$X_{0,1} = X_{1,2} = X_{2,n+2} = X_{n+2,n+1} = X_{n+1,2n+1} = 1;$$

$$X_{0,1} = X_{1,2} = X_{2,n+2} = X_{n+2,n+1} = 1,$$

$$X_{n+1,2n+1} = 0;$$

$$X_{0,1} = X_{1,2} = X_{2,n+1} = 1, \quad X_{n+2,2n+1} = 0;$$

$$X_{0,1} = X_{1,2} = 1, \quad X_{2,n+1} = 0;$$

$$X_{0,1} = 1, \quad X_{1,2} = 0;$$

$$X_{0,1} = 0.$$

This kind of information is easy to transfer to the subproblem network. However, the branching enumeration tree rapidly grows since for a path r satisfying n_r requests, there are $2(n_r + 1)$ branches created! We propose a new branching strategy which is applied directly to the requests, more precisely to the pickup sequence. Define order variables O_{ij}, $i, j \in P^+ \cup \{0, 2n+1\}$. For a path r satisfying n_r requests, only $(n_r + 2)$ branches are

created similarly to the previous strategy. For the preceding example, four branches are created:

$O_{0,1} = O_{1,2} = O_{2,2n+1} = 1;$

$O_{0,1} = O_{1,2} = 1, \quad O_{2,2n+1} = 0;$

$O_{0,1} = 1, \quad O_{1,2} = 0;$

$O_{0,1} = 0.$

In the second branch, the first pickup node visited is node 1, then pickup node 2 and there must be at least another pickup node visited after node 2. This information is easily transferable to the subproblem if the dynamic programming labels are modified as follows: add a fourth component p which represents the last pickup node visited (or $p = 0$ at start or $p = 2n + 1$ at the end). A label at node l, associated with the path $\mathscr{P}_l^k$ is now given by $(R(S_l^k), T_l^k, Z_l^k, p_l^k)$. New conditions are imposed on the shortest path algorithm so as to satisfy constraints on this component. The subproblem is not harder to solve since this fourth component is closely related to the information contained in the first one.

4.3. Implementation strategies

In this application, several strategies were used to reduce execution time. Under the hypothesis that cost values satisfy the triangle inequality, each subpath of a feasible path is also feasible at a lesser cost. Equation (2) of the form $(=)$ can be transformed into inequality $(\geqslant)$, thus obtaining better numerical stability for the simplex algorithm.

Given a set of multipliers, a strategy which generates several columns was adopted. From the shortest path algorithm, it is possible to obtain the n best paths from node 0 to node $2n + 1$ for which the last delivery node is different. All those paths for which the marginal cost is negative are simultaneously added to the restricted master problem. On the other hand, columns with a large marginal cost are removed from the restricted master problem to accelerate the simplex algorithm.

Given a subset Ω' of feasible columns of Ω, the restricted master problem is solved. Denote by $Z(\Omega')$, the optimal value of the objective function and by $\Pi(\Omega')$, the optimal simplex multipliers. $Z(\Omega')$ is a *primal upper bound* on $Z(\Omega)$, the optimal value of the linear relaxation of the master problem. It is also possible to evaluate a *dual lower bound* given be $Z(\Omega') + v_{\text{sup}} \cdot \bar{c}(\Pi(\Omega'))$, where v_{sup} is the smallest upper bound on the number of vehicles and $\bar{c}(\Pi(\Omega')) \leqslant 0$ is the marginal cost of the shortest path given the simplex multipliers $\Pi(\Omega')$. Given these two bounds and considering only the vehicle fixed cost contribution, it is possible to judiciously impose the cut on the number of vehicles far before the optimal linear solution of the master problem is reached. For example, if the primal bound is at 12.9 vehicles and the dual bound is at 12.1 vehicles, a cut with at least 13 vehicles is valid. This strategy eliminates the final and slower iterations of the column generation scheme.

The next accelerating procedures are the elimination of arcs and nodes. Rather than always solving constrained shortest path problems on the entire network, we have also used a small partial network with 30% of the best arcs and an intermediate one with 50% of the best arcs. Elimination of nodes is also done heuristically. Given the current set of multipliers, if π_i is small enough, request i (nodes i and $n + i$) can be temporarily eliminated for the following reason. Suppose that a feasible path satisfies request i. This path makes a detour $c \geqslant 0$ with an additional cost, say $(c - \pi_i)$ to visit both nodes i and $n + i$. If $\pi_i < c$, this path will not be the shortest one. Details of these heuristic procedures are given in [14]. It should also be noted that, in the branch-and-bound tree, new columns are generated on the entire network. The branch-and-bound tree is explored depth first in order to rapidly obtain a feasible integer solution and exploration terminates when the number of vehicles is optimal and when the gap with respect to the primal bound is small.

5. Computational results

We investigated the computational performance of the column generation algorithm just described, using eight problems that ranged from 19 to 55 requests (40 to 112 nodes, including depot nodes). Test problems were solved on the University of Montreal CYBER 855 computer (but reported CPU times are CYBER 173 equivalent) using the FORTRAN 5 (with optimized code) compiler.

Problems A19 and A30 are real life problems

Table 1
Tests problems

	A19	A30	B30	C20	C30	D40	D50	D55
Route duration (hrs)	30	10	10	9	11	12	12	12
Number of arcs	687	821	1176	534	1041	1299	1462	1996
Z(LP): nb. of vehicles	6	12	10	8	11	15	20	22
travel cost	3131	2283	1800	1092	2121	3391	4271	4767
Z(ILP): travel cost	3131	2283	1811	1100	2137	3426	4364	4918
GAP on travel cost (%)	0	0	0.6	0.7	0.7	1.0	2.2	3.2
Z(LP) cpu time (sec.)	92	47	112	28	111	66	95	204
Z(ILP) cpu time (sec.)	95	51	142	51	169	172	215	313
Best B.-B. node	1	1	2	2	9	2	8	42
Number of nodes explored	1	1	6	7	9	10	22	43
Columns generated	99	92	115	86	135	139	155	220
Number of states (max.)	1225	801	1326	326	1182	760	838	1840

from a French carrier [15]. Problem B30 differs from problem A30 due to the fact that vehicles are not restricted to return to the departure depot. Problems C20 and C30 have been generated from the 67 cities served by the French carrier while problems D40, D50 and D55 have been generated from a random network of 200 cities. It is not common to find the same location for different requests in problems of type D.

For all problems in Table 1, the load demand distribution is approximately 40% of the full load, 40% of the half load and 20% of one third of the vehicle capacity. Also, there are no specific time windows associated with the nodes, except the maximum route duration associated to the depot nodes. The travel time between two nodes is proportional to the Euclidean distance between them with a speed of 65 km/hour, and the cost load factor $g(Y)$ is given by 32 litres/100 km if the vehicles are empty and 44 litres/100 km otherwise. Finally, the service times s_i, $i \in P$ are calculated in minutes by $5 + 25 * (d_i/D)$, so that it takes 30 minutes to pickup or deliver a full vehicle. Table 1 also indicates the number of arcs in the network following the construction described in Subsection 2.4. The reader can then see the difference between problems A30 and B30.

Some detailed results are given for each problem. The first series indicates the optimized number of vehicles and the travel cost (primal lower bound), the value of travel cost for the best integer solution (the number of vehicles was always opti-

Table 2
Variation of the route duration for problem B30

Route duration (hrs)	10	11	12	13
Number of arcs	1176	1384	1545	1653
Z(LP): nb. of vehicles	10	9	8	7
travel cost	1800	1780	1748	1721
Z(ILP): travel cost	1811	1792	1785	1734
GAP on travel cost (%)	0.6	0.7	2.1	0.8
Z(LP) cpu time (sec.)	112	148	237	323
Z(ILP) cpu time (sec.)	142	258	690	603
Best B.-B. node	2	3	3	2
Number of nodes explored	6	7	21	7
Columns generated	115	133	223	174
Number of states (max.)	1326	1921	2428	2774

Table 3
Variation of load distribution for problem B30

Distribution [a]	40/40/20	40/60/0	50/50/0	60/40/0	100/0/0
Number of arcs	1176	1343	1075	845	536
Z(LP): nb. of vehicles	10	11	11	12	14
travel cost	1800	1916	2064	2024	2248
Z(ILP): travel cost	1811	1932	2064	2074	2248
GAP on travel cost (%)	0.6	0.8	0	2.5	0
Z(LP) cpu time (sec.)	112	173	85	36	14
Z(ILP) cpu time (sec.)	142	181	88	101	17
Best B.-B. node	2	1	1	29	1
Number of nodes explored	6	4	1	29	1
Columns generated	115	111	95	151	86
Number of states (max.)	1326	2185	1198	525	157

[a] The load demand distribution is given in percentage, for full load, half load and one third of the vehicle capacity.

mal) and the gap (%) on travel cost. The second series provides the computational time in seconds while the third one indicates the branch-and-bound node where the best integer solution was found and the total number of nodes explored. The last series presents the total number of columns generated and the maximum number of states kept in the shortest path algorithm.

Both real problems A19 and A30 were solved to optimality. Furthermore, an integer optimal solution was found at the very first node which corresponds to the linear relaxation of the set partitioning type formulation. For all the other problems, the number of vehicles is always minimal after the addition of a cut (at the first node); the gap on travel cost ranges from 0.6% to 3.2%. Problems become harder to solve as the number of requests increases. However, a very good solution was obtained within the exploration of a few nodes of the branch-and-bound tree (less than ten), except for the biggest problem of 55 requests.

The next three tables of results present independent variations of the original problem B30: route duration, load demands distribution and time windows. In Table 2 the route duration increases from 10 to 13 hours and problems become harder to solve. The reader can also observe the number of states kept in the shortest path algorithm, the number of columns generated and the CPU time increase. The number of vehicles (which decreases) is again optimal and the best integer solution found within a few nodes of the branch-and-bound remains close to the primal

Table 4
Variation on time windows for problem B30

Time windows	∞	10h.	6h.40	3h.20	0h.
Number of arcs	1176	1175	1175	1175	1175
Z(LP): nb. of vehicles	10	10	10	12	13
travel cost	1800	1843	1908	1965	2239
Z(ILP): travel cost	1811	1843	1908	1981	2239
GAP on travel cost (%)	0.6	0	0	0.8	0
Z(LP) cpu time (sec.)	112	92	44	30	7
Z(ILP) cpu time (sec.)	142	97	50	66	13
Best B.-B. node	2	1	1	7	1
Number of nodes explored	6	1	1	7	1
Columns generated	115	102	92	110	54
Number of states (max.)	1326	1170	621	281	44

lower bound given by the linear relaxation of the set partitioning type formulation. The load variation (Table 3) considers distributions of full and half load demands. Two problems have been solved to optimality, and the reader should notice that the algorithm is also applicable to full truck load problems. In Table 4, a heuristic solution has been found for problem B30 and time windows were defined around this solution. As expected, problems with smaller time windows are easier to solve, as indicated by the CPU time, the branch-and-bound characteristics and the maximum number of states in the shortest path algorithm.

Finally, the column generation algorithm has been generalized to solve problems with more than one depot. The special feature is that there is a subproblem for each depot. Table 5 presents variations of problem A30, from one to five depots whose locations were randomly selected from the network. Two problems were solved to optimality while solutions for problems with four and five depots have a gap of less than one percent on the travel cost. Computational time seems to increase less than linearly with the number of depots.

These experiments reveal that time windows and the distribution of load demands are the parameters that have the most significant influence on the running time of this column generation algorithm. If these parameters are such that the number of requests satisfied by each vehicle is at most five (ten nodes), the shortest path algorithm is an efficient way to generate feasible routes. The set partitioning type formulation solved by column generation is particularly efficient for the strongly constrained problems. The linear programming relaxation provides an excellent bound and allows to rapidly obtain a good integer solution. Combined to accelerating techniques, this primal scheme is well designed to produce near optimal solutions, even for large pickup and delivery problems.

Appendix A

This section illustrates, on a four client problem, some detailed steps of the dynamic programming algorithm for the generation of new columns. It first gives the original data (Table A.1), and after the reduction of the time windows and the arc elimination procedure based on capacity and time window criteria, it presents the resulting reduced data (Table A.2).

For a better visual aspect, pickup and delivery node sets are respectively given by $\{1^+, 2^+, 3^+, 4^+\}$ and $\{1^-, 2^-, 3^-, 4^-\}$. Origin and destination depots are denoted by s and t. Finally, vehicle capacity equals 100, while on each arc, cost and travel time are the same. Vehicle cost is equal to 1000. In this simplified example, vehicle costs are independent of the vehicle load.

From Table A.1 to Table A.2, time windows have been reduced by calculating (at each node) the earliest feasible arrival time and the latest feasible departure time. Next, the set of arcs $\{(i^+, j^+), (i^+, j^-), (i^-, j^-)$: $i = 1, 2$, and $j = 3\}$ is removed by the vehicle capacity constraint. Otherwise, the elimination of arcs involves time

Table 5
Variation on the number of depots for problem A30

Number of depots	1	2	3	4	5
Number of arcs	821	1593	1658	1725	1794
Z(LP): nb. of vehicles	12	12	12	12	12
travel cost	2283	2267	2265	2264	2259
Z(ILP): travel cost	2283	2267	2265	2267	2272
GAP on travel cost (%)	0	0	0	0.1	0.6
Z(LP) cpu time (sec.)	47	87	107	124	154
Z(ILP) cpu time (sec.)	51	92	113	130	212
Best B.-B. node	1	1	1	1	2
Number of nodes explored	1	1	1	2	6
Columns generated	92	116	126	125	150
Number of states (max.)	801	791	789	793	755

Table A.1
Original data table

	Travel costs (travel times) on arcs											
nodes	s	1^+	2^+	3^+	4^+	1^-	2^-	3^-	4^-	t	Demands	Time windows
s	–	27	51	42	100	–	–	–	–	–	–	360, 720
1^+	–	–	29	18	74	63	95	111	152	–	60	540, 600
2^+	–	29	–	28	46	40	69	80	128	–	40	540, 600
3^+	–	18	28	–	69	52	86	103	139	–	70	360, 650
4^+	–	74	46	69	–	26	28	38	88	–	30	580, 650
1^-	–	–	40	52	26	–	30	51	89	95	−60	600, 720
2^-	–	95	–	86	28	30	–	18	60	64	−40	600, 720
3^-	–	111	80	–	38	51	18	–	55	54	−70	360, 720
4^-	–	152	128	139	–	89	60	55	–	14	−30	600, 720
t	–	–	–	–	–	–	–	–	–	–	–	360, 720

window restrictions only. For example, arc $(1^+, 4^+)$ has been discarded since both paths $(1^+, 4^+, 1^-, 4^-)$ and $(1^+, 4^+, 4^-, 1^-)$ are infeasible. Detailed service times at each node are the following: $1^+(540) \rightarrow 4^+(614) \rightarrow 1^-(640$: infeasible) and $1^+(540) \rightarrow 4^+(614) \rightarrow 4^-(702) \rightarrow 1^-(782$: infeasible).

Next, we present detailed steps of the dynamic programming algorithm at the first call. Since the set partitioning problem is initialized with one vehicle for each request (an identity matrix), the simplex multipliers are given by $\Pi = c_B B^{-1} = c_B$ = (1185, 1184, 1199, 1202). For example, the cost of the first column is given by the cost of a vehicle plus the total travel time on the path $(s, 1^+, 1^-, t)$: 1000 + 27 + 63 + 95 = 1185.

Selected notes on Table A.3

(1) The cost of label 1.1 is the cost of the arc $(s, 1^+)$: 1000 + 27.

(2) The cost of label 2.1 is the cost of label 1.1 plus the cost of the arc $(1^+, 2^+)$ minus the simplex multiplier on the node (1^+): 1027 + 29 − 1185 = −129.

(3) This label is rejected since customer 1 is not currently being served in the vehicle.

(4) The start of service at node 4^+ is equal to the lower bound of the time window since waiting time is permitted.

(5) This label is rejected since vehicle capacity is exceeded at the arrival node 4^+.

(6) This label is rejected since it is impossible to visit node 1^- later on.

(7) This label is rejected since customer 2^+ is not currently being served in the vehicle.

(8) This path ending at node t is discarded since the marginal cost is equal to zero.

(9) This label is rejected since it is non post-feasible for node 2^-.

(10) Label 3.3 is not dominated by label 3.2 since they don't have the same arrival node.

Table A.2
Reduced data table

	Travel costs (travel times) on arcs											
nodes	s	1^+	2^+	3^+	4^+	1^-	2^-	3^-	4^-	t	Demands	Time windows
s	–	27	51	42	100	–	–	–	–	–	–	360, 521
1^+	–	–	29	–	–	63	–	–	–	–	60	540, 562
2^+	–	–	–	–	46	40	69	–	–	–	40	540, 587
3^+	–	–	–	–	69	–	–	103	–	–	70	402, 563
4^+	–	–	–	–	–	–	28	38	88	–	30	580, 618
1^-	–	–	–	–	–	–	30	–	–	95	−60	603, 625
2^-	–	–	–	–	–	–	–	–	60	64	−40	609, 656
3^-	–	–	80	–	38	–	–	–	55	54	−70	505, 666
4^-	–	–	–	–	–	–	–	–	–	14	−30	668, 706
t	–	–	–	–	–	–	–	–	–	–	–	559, 720

(11) Label 3.4 and the associated partial path is not rejected even if the actual marginal cost is positive. It may lead to a feasible path with a negative marginal cost.
(12) Labels 4.2, 4.3, 4.4 are all partial paths ending at node 4^- with no clients being actually served in the vehicle. However, no label dominates the others. The reader can observe that for increasing time values, the costs are decreasing.
(13) This label and label 4.1 have the same arrival

Table A.3
Dynamic programming algorithm steps

Preceding label	Last node visited	Potential arrival nodes	Clients being served	Start of service at arrival node	Vehicle load at arrival node	Cumulative cost	Label number	Selected notes [a]
Initialization								
–	–	s	$\emptyset$	360	0	0	0	
Iteration 1								
0	s	1^+	$\{1^+\}$	540	60	1027	1.1	(1)
		2^+	$\{2^+\}$	540	40	1051	1.2	
		3^+	$\{3^+\}$	402	70	1042	1.3	
		4^+	$\{4^+\}$	580	30	1100	1.4	
Iteration 2								
1.1	1^+	2^+	$\{1^+, 2^+\}$	569	100	−129	2.1	(2)
		1^-	$\emptyset$	603	0	−95	2.2	
1.2	2^+	4^+	$\{2^+, 4^+\}$	586	70	−87	2.3	
		1^-	rejected: $1^+ \notin \{2^+\}$					(3)
		2^-	$\emptyset$	609	0	−64	2.4	
1.3	3^+	4^+	$\{3^+, 4^+\}$	580	100	−88	2.5	(4)
		3^-	$\emptyset$	505	0	−54	2.6	
1.4	4^+	2^-	rejected: $2^+ \notin \{4^+\}$					
		3^+	rejected: $3^+ \notin \{4^+\}$					
		4^-	$\emptyset$	668	0	−14	2.7	
Iteration 3								
2.1	2^+	4^+	$\{1^+, 2^+, 4^+\}$	615	130 rejected			(5)
		1^-	$\{2^+\}$	609	40	−1273	3.1	
		2^-	$\{1^+\}$	638 rejected: non post-feasible for 1^-				(6)
2.2	1^-	2^-	rejected: $2^+ \notin \emptyset$					(7)
		t	$\emptyset$	698	0	0	–	(8)
2.3	4^+	2^-	$\{4^+\}$	614	30	−1261	3.2	
		3^-	rejected: $3^+ \notin \{2^+, 4^+\}$					
		4^-	$\{2^+\}$	674 rejected: non post-feasible for 2^-				(9)
2.4	2^-	4^-	rejected: $4^+ \notin \emptyset$					
		t	$\emptyset$	673	0	0	–	(8)
2.5	4^+	2^-	rejected: $2^+ \notin \{3^+, 4^+\}$					
		3^-	$\{4^+\}$	618	30	−1252	3.3	(10)
		4^-	$\{3^+\}$	668 rejected: non post-feasible for 3^-				
2.6	3^-	2^+	$\{2^+\}$	585	40	26	3.4	(11)
		4^+	$\{4^+\}$	580	30	−16	3.5	(4)
		4^-	rejected: $4^+ \notin \emptyset$					
		t	$\emptyset$	559	0	0	–	(8)
2.7	4^-	t	$\emptyset$	682	0	0	–	(8)

Table A.3 (continued)

Preceding label	Last node visited	Potential arrival nodes	Clients being served	Start of service at arrival node	Vehicle load at arrival node	Cumulative cost	Label number	Selected notes
Iteration 4								
3.1.	1^-	2^-	$\emptyset$	639	0	−1243	4.1	
		t	$\{2^+\}$ rejected: the vehicle is not empty					
3.2	2^-	4^-	$\emptyset$	674	0	−1201	4.2	(12)
		t	$\{4^+\}$ rejected: the vehicle is not empty					
3.3	3^-	2^+	$\{2^+, 4^+\}$	698 rejected: time window exceeded				
		4^+	$\{4^+\}$ rejected: node 4^+ has already been served					
		4^-	$\emptyset$	673	0	−1197	4.3	(12)
		t	$\{4^+\}$ rejected: the vehicle is not empty					
3.4	2^+	4^+	$\{2^+, 4^+\}$	631 rejected: time window exceeded				
		1^-	rejected: $1^+ \notin \{2^+\}$					
		2^-	$\emptyset$	654	0	−1089	rejected	(13)
3.5	4^+	2^-	rejected: $2^+ \notin \{4^+\}$					
		3^-	rejected: $3^+ \notin \{4^+\}$					
		4^-	$\emptyset$	668	0	−1130	4.4	(12)
Iteration 5								
4.1	2^-	4^-	rejected: $4^+ \notin \emptyset$					
		t	$\emptyset$	703	0	−1179	$t.1$	(14)
4.2	4^-	t	$\emptyset$	688	0	−1187	$t.2$	(14)
4.3	4^-	t	$\emptyset$	687	0	−1183	$t.3$	(14)
4.4	4^-	t	$\emptyset$	682	0	−1116	$t.4$	(14)

[a] See subsection for the selected notes.

node (node 2^-) and the same set of clients being served in the vehicle (actually the empty set) but they are different for the time and cost values. However, the current label considered has a greater time value ($654 > 639$) and a greater cost value ($-1089 > -1243$). It is then dominated by label 4.1: any extension to label 4.1 is always better than any extension of the current label.

(14) Four paths with negative marginal cost.

The best path is given by lable $t.2$ and column $(0, 1, 0, 1)^T$ should enter the basis at the next iteration of the simplex algorithm (see Table A.4). In our multiple column generation scheme, the first three columns are introduced in the set partitioning formulation and the last one is discarded since it is identical to the one provided by label $t.3$.

Adding these 3 new columns to the initial ones, a new solution to the linear relaxation of the set partitioning formulation is found, as well as new simplex multipliers. If the subproblem does not generate additional columns with negative margi-

Table A.4

Label	Path	Column	Marginal cost	Real cost
$t.1$	$(s, 1^+, 2^+, 1^-, 2^-, t)$	$(1, 1, 0, 0)^T$	−1179	1190
$t.2$	$(s, 2^+, 4^+, 2^-, 4^-, t)$	$(0, 1, 0, 1)^T$	−1187	1199
$t.3$	$(s, 3^+, 4^+, 3^-, 4^-, t)$	$(0, 0, 1, 1)^T$	−1183	1218
$t.4$	$(s, 3^+, 3^-, 4^+, 4^-, t)$	$(0, 0, 1, 1)^T$	−1116	1285

nal cost, the LP solution of the set partitioning is optimal. If the LP solution is not integer, branch-and-bound is needed, as well as column generation at each node of the exploration tree.

Acknowledgement

This research was supported by the Quebec Government (FCAR grant) and by the Federal Government of Canada (NSERC grants).

References

[1] Bodin, L., Golden, B., Assad, A., and Ball, M., "The routing and scheduling of vehicles and crews – The state of the art", *Computers and Operations Research* 10 (1983) 62–212.

[2] Bodin, L., and Sexton, T., "The multi-vehicle subscriber dial-a-ride problem", *TIMS Studies in the Management Sciences* 22 (1986) 73–86.

[3] Carpaneto, G., and Toth, P., "Some new branching and bounding criteria for the asymmetric travelling salesman problem", *Management Science* 26 (1980) 736–743.

[4] Christofides, N., Mingozzi, A., and Toth, P., "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations", *Mathematical Programming* 20 (1981) 255–282.

[5] Desrochers, M., "An algorithm for the shortest path problem with resource constraints", Publication 421A, Centre de Recherche sur les Transports, Université de Montréal, 1986.

[6] Desrochers, M., Lenstra, J.K., Savelsberg, M.W.P., and Soumis, F., "Vehicle routing with time windows: Optimization and approximation", *TIMS Studies in the Management Science* 16 (1988) 65–84.

[7] Desrochers, M., and Soumis, F., "A reoptimization algorithm for the shortest path problem with time windows", *European Journal of Operational Research* 35 (1988) 242–254.

[8] Desrochers, M., and Soumis, F., "A generalized permanent labeling algorithm for the shortest path problem with time windows", *INFOR* 26 (1988) 193–214.

[9] Desrochers, M., and Soumis, F., "A column generation approach to the urban transit crew scheduling problem", *Transportation Science* 23 (1989) 1–13.

[10] Desrosiers, J., Dumas, Y., and Soumis, F., "A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows", *The American Journal of Mathematical and Management Sciences* 6 (1986) 301–325.

[11] Desrosiers, J., Dumas, Y., and Soumis, F., "The multiple vehicle dial-a-ride problem", in: J.R. Daduda and A. Wren (eds), *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical System 308, Springer, Berlin, 1988, 15–27.

[12] Desrosiers, J., Pelletier, P., and Soumis, F., "Plus court chemin avec contraintes d'horaires", *RAIRO Operations Research* 17 (1983) 357–377.

[13] Desrosiers, J., Soumis, F., and Desrochers, M., "Routing with time windows by column generation", *Networks* 14 (1984) 545–565.

[14] Dumas, Y., "Confection d'itinéraires de véhicles en vue du transport de plusieurs origines à plusieurs destinations", Publication 434, Centre de Recherche sur les Transports, Université de Montréal, 1985.

[15] Guinet, A., "Le système T.I.R.: un système d'établissement de tournées industrielles routières", Thèse de doctorat en informatique et automatique appliquées, Université Claude Bernard, Lyon, 1984.

[16] Jaw, J., Odoni, A., Psaraftis, H., and Wilson, N., "A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with windows", *Transportation Research B*20 (1986) 243–257.

[17] Kolen, A., Rinnooy Kan, A., and Trienekens, H., "Vehicle routing with time windows", *Operations Research* 35 (1987) 266–273.

[18] Laporte, G., and Nobert, Y., "Exact algorithms for the vehicle routing problems", *Annals of Discrete Mathematics* 31 (1987) 147–184.

[19] Lavoie, S., Minoux, M., and Odier, E., "A new approach for crew pairing problems by column generation with application to air transportation", *European Journal of Operational Research* 35 (1988) 45–48.

[20] Magnanti, T., "Combinatorial optimization and vehicle fleet planning: Perspectives and prospects", *Networks* 11 (1981) 179–214.

[21] Miller, C., Tucker, A., and Zemlin, R., "Integer programming formulations and traveling salesman problems", *Journal of the ACM* 7 (1960) 326–329.

[22] Psaraftis, H., "A dynamic programming solution to the single-vehicle, many-to-many, immediate request dial-a-ride problem", *Transportation Science* 14 (1980) 130–154.

[23] Psaraftis, H., "An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows", *Transportation Science* 17 (1983) 351–357.

[24] Roy, S., Rousseau, J.-M., Lapalme, G., and Ferland, J.A., "Routing and scheduling for the transportation of disabled persons – The algorithm", Report TP 5596E, Transport Canada, Transport Development Center, Montreal, 1984.

[25] Roy, S., Rousseau, J.-M., Lapalme, G., and Ferland, J.A., "Routing and scheduling for the transportation of disabled persons – The tests", Report TP 5598E, Transport Canada, Transport Development Center, Montreal, 1984.

[26] Sexton, T., and Bodin, L., "Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling", *Transportation Science* 19 (1985) 378–410.

[27] Sexton, T., and Bodin, L., "Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing", *Transportation Science* 19 (1985) 411–435.

[28] Sexton, T., and Choi, Y., "Pick-up and delivery of partial loads with time windows", *The American Journal of Mathematical and Management Sciences* 6 (1986) 369–398.

[29] Solomon, M., and Desrosiers, J., "Time window constrained routing and scheduling problems", *Transportation Science* 22 (1988) 1–13.