OPERATIONS RESEARCH
Vol. 54, No. 3, May–June 2006, pp. 573–586
ISSN 0030-364X | EISSN 1526-5463 | 06 | 5403 | 0573

DOI 10.1287/opre.1060.0283

A Branch-and-Cut Algorithm for the Dial-a-Ride Problem

Jean-François Cordeau
HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Quebec, Canada H3T 2A7, jean-francois.cordeau@hec.ca

In the dial-a-ride problem, users formulate requests for transportation from a specific origin to a specific destination. Transportation is carried out by vehicles providing a shared service. The problem consists of designing a set of minimum-cost vehicle routes satisfying capacity, duration, time window, pairing, precedence, and ride-time constraints. This paper introduces a mixed-integer programming formulation of the problem and a branch-and-cut algorithm. The algorithm uses new valid inequalities for the dial-a-ride problem as well as known valid inequalities for the traveling salesman, the vehicle routing, and the pick-up and delivery problems. Computational experiments performed on randomly generated instances show that the proposed approach can be used to solve small to medium-size instances.

Subject classifications: transportation: vehicle routing; programming: cutting plane.
Area of review: Transportation.
History: Received September 2003; revision received August 2004; accepted March 2005.

1. Introduction

In the Dial-a-Ride Problem (DARP), users formulate requests for transportation from a specific origin (or *pick-up* point) to a specific destination (or *drop-off* point). Transportation is carried out by vehicles that provide a shared service in the sense that several users may be in a vehicle at the same time. The aim is to design a minimum-cost set of vehicle routes accommodating all requests under a number of side constraints. A common DARP application arises in door-to-door transportation services for the elderly and the disabled. In this context, users often formulate two requests per day: an *outbound* request from home to a destination, and an *inbound* request for the return trip.

Most dial-a-ride services are characterized by the presence of two conflicting objectives: minimizing operating costs and minimizing user inconvenience. Operating costs are mostly related to fleet size and distance traveled, while user inconvenience is often measured in terms of deviations from desired pick-up and drop-off times and in terms of excess ride time (i.e., the difference between the actual ride time of a user and the minimum possible ride time). One way to achieve a balance between these objectives is to treat cost minimization as the primary objective and to impose service quality constraints.

As in the work of Jaw et al. (1986) and Cordeau and Laporte (2003b), we assume here that the user specifies either a desired arrival time at destination (in the case of an outbound request) or a desired departure time from the origin (in the case of an inbound request). In both cases, a time window of a prespecified width is constructed around the desired time. In addition, an upper bound is imposed on the ride time of the user. This approach seems to be in line with the current practice of several North American transporters. For example, in a system with 15-minute time windows and a maximum ride time of 45 minutes, a user wishing to arrive at destination at 9:00 would be picked up no earlier than 8:00 and dropped off between 8:45 and 9:00. It should be emphasized that imposing time windows is not sufficient to accurately impose a maximum ride time. In the above example, a pickup at 8:00 and a drop-off at 9:00 would exceed the maximum ride time of 45 minutes. However, constraining the pickup to take place no earlier than 8:15 would be too restrictive because it would, in particular, prohibit a pickup at 8:00 and a drop-off at 8:45.

Dial-a-ride services may operate according to a *static* or to a *dynamic* mode. In the first case, all requests are known beforehand, while in the second case requests are gradually received throughout the day and vehicle routes are adjusted in real time to meet demand. In practice, pure dynamic problems rarely exist because a large subset of requests is often known in advance. This paper deals with the static variant of the problem.

Because it incorporates time windows and maximum ride-time constraints, the DARP is a difficult problem that generalizes the Vehicle Routing Problem with Pickup and Delivery (VRPPD). Finding a feasible solution for the DARP is itself NP-hard because it also generalizes the Traveling Salesman Problem with Time Windows (TSPTW) (see, e.g., Savelsbergh 1985). As a result, most previous work has concentrated on the development of heuristics. Nevertheless, when the problem is moderately constrained, exact solution approaches can be devised to solve small to medium-size instances.

Our aim is to describe valid inequalities and a branch-and-cut algorithm for the DARP. Branch-and-cut has proved to be an effective technique for solving the TSP, the

VRP, and several other routing problems (see, e.g., Naddef and Rinaldi 2002). The algorithm proposed here uses adaptations of known valid inequalities for the precedence-constrained TSP, the VRP, and the VRPPD, as well as new inequalities that take advantage of the special structure of the problem.

The remainder of this paper is organized as follows. The next section briefly reviews relevant work on the DARP and closely related problems. Section 3 defines the DARP formally and introduces a mixed-integer formulation. Section 4 introduces several families of valid inequalities used in the branch-and-cut algorithm, which is then described in §5, along with separation heuristics and preprocessing techniques. Computational experiments are reported in §6 and the conclusion follows in §7.

2. Literature Review

Early research on the DARP was carried out by Psaraftis (1980, 1983), who developed dynamic programming algorithms for the single-vehicle case. An improved labeling scheme was later proposed by Desrosiers et al. (1986), who were able to solve instances with up to 40 users.

Exact algorithms have also been devised for closely related single-vehicle problems. Ruland (1995) and Ruland and Rodin (1997) proposed a branch-and-cut algorithm for a special case of the pick-up and delivery problem in which there are no capacity or time window constraints. In this case, if distances satisfy the triangle inequality, a single vehicle will serve all users. As a result, the problem reduces to a TSP with the additional constraint that the pick-up node of each user must precede his drop-off node in the vehicle route. These authors provided an integer programming formulation of the problem based on an undirected graph and described four families of valid inequalities: subtour elimination constraints, precedence constraints, generalized order constraints, and order-matching constraints. These inequalities are also valid for the more general problem studied in this paper, and they will be discussed in more depth in §4.

The more general precedence-constrained asymmetric TSP (in which each node may have multiple predecessors and successors) has been studied, among others, by Balas et al. (1995) and Ascheuer et al. (2000), who proposed several families of valid inequalities based, in large part, on the strengthening of existing inequalities for the asymmetric TSP. This problem has applications, for example, in the scheduling of flexible manufacturing systems (see, e.g., Ascheuer 1996). Because the TSP with pickup and delivery is a particular case of the precedence-constrained TSP, some of the proposed inequalities are also useful for solving the DARP. In particular, predecessor and successor inequalities proposed by Balas et al. will be used in our approach.

Most of the algorithms for the multiple-vehicle DARP are heuristics or metaheuristics. An insertion method capable of handling large-scale instances was described by Jaw et al. (1986), while Bodin and Sexton (1986) developed an exchange heuristic that uses Benders decomposition to optimize the individual vehicle routes. Dumas et al. (1989) later presented a clustering and column generation method that can handle instances with several thousand users. More recently, Madsen et al. (1995) proposed an insertion heuristic for the dynamic case, while Toth and Vigo (1996, 1997) used local search and a tabu-thresholding procedure to address a real-life problem arising in Bologna. Finally, Cordeau and Laporte (2003b) described a tabu search heuristic for the variant of the problem addressed in this paper. This heuristic will be used in §6 to compute upper bounds.

Relevant work was also performed on the closely related VRPPD with time windows, arising in contexts such as urban courier services. For this problem, Dumas et al. (1991) presented an exact column generation method using the dynamic programming algorithm of Desrosiers et al. (1986) to solve the pricing subproblem. A similar approach was also described by Savelsbergh and Sol (1998), who proposed several refinements to improve performance.

For recent overviews of the DARP and VRPPD, the reader is referred to the surveys of Cordeau and Laporte (2003a) and Desaulniers et al. (2002), respectively.

3. Formulation

Let n denote the number of users (or requests) to be served. The DARP may be defined on a complete directed graph $G = (N, A)$, where $N = P \cup D \cup \{0, 2n + 1\}$, $P = \{1, \ldots, n\}$, and $D = \{n+1, \ldots, 2n\}$. Subsets P and D contain pick-up and drop-off nodes, respectively, while nodes 0 and $2n + 1$ represent the origin and destination depots. With each user i are thus associated an origin node i and a destination node $n + i$.

Let K be the set of vehicles. Each vehicle $k \in K$ has a capacity Q_k, and the total duration of its route cannot exceed T_k. With each node $i \in N$ are associated a load q_i and a nonnegative service duration d_i such that $q_0 = q_{2n+1} = 0$, $q_i = -q_{n+i}$ $(i = 1, \ldots, n)$, and $d_0 = d_{2n+1} = 0$. A time window $[e_i, l_i]$ is also associated with node $i \in N$, where e_i and l_i represent the earliest and latest time, respectively, at which service may begin at node i. With each arc $(i, j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij}. Finally, denote by L the maximum ride time of a user.

As mentioned in the introduction, we assume here that a time window is specified either for the origin node or for the destination node of a request, but not both. However, a valid time window can always be determined for the other node by taking the direct and the maximum ride times into account. We will explain in §5.1.1 how these time windows can be made as tight as possible.

For each arc $(i, j) \in A$ and each vehicle $k \in K$, let $x_{ij}^k = 1$ if vehicle k travels from node i to node j. For each node $i \in N$ and each vehicle $k \in K$, let B_i^k be the time at which

vehicle k begins service at node i, and Q_i^k be the load of vehicle k after visiting node i. Finally, for each user i, let L_i^k be the ride time of user i on vehicle k.

The DARP can be formulated as the following mixed-integer program:

$$\text{Min} \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ij}^k x_{ij}^k \tag{1}$$

subject to

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1 \quad \forall i \in P, \tag{2}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{n+i,j}^k = 0 \quad \forall i \in P,\ k \in K, \tag{3}$$

$$\sum_{j\in N} x_{0j}^k = 1 \quad \forall k \in K, \tag{4}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = 0 \quad \forall i \in P \cup D,\ k \in K, \tag{5}$$

$$\sum_{i\in N} x_{i,2n+1}^k = 1 \quad \forall k \in K, \tag{6}$$

$$B_j^k \geqslant (B_i^k + d_i + t_{ij})x_{ij}^k \quad \forall i \in N,\ j \in N,\ k \in K, \tag{7}$$

$$Q_j^k \geqslant (Q_i^k + q_j)x_{ij}^k \quad \forall i \in N,\ j \in N,\ k \in K, \tag{8}$$

$$L_i^k = B_{n+i}^k - (B_i^k + d_i) \quad \forall i \in P,\ k \in K, \tag{9}$$

$$B_{2n+1}^k - B_0^k \leqslant T_k \quad \forall k \in K, \tag{10}$$

$$e_i \leqslant B_i^k \leqslant l_i \quad \forall i \in N,\ k \in K, \tag{11}$$

$$t_{i,n+i} \leqslant L_i^k \leqslant L \quad \forall i \in P,\ k \in K, \tag{12}$$

$$\max\{0, q_i\} \leqslant Q_i^k \leqslant \min\{Q_k, Q_k + q_i\} \quad \forall i \in N,\ k \in K, \tag{13}$$

$$x_{ij}^k \in \{0,1\} \quad \forall i \in N,\ j \in N,\ k \in K. \tag{14}$$

The objective function (1) minimizes the total routing cost. Constraints (2) and (3) ensure that each request is served exactly once and that the origin and destination nodes are visited by the same vehicle. Constraints (4)–(6) guarantee that the route of each vehicle k starts at the origin depot and ends at the destination depot. Consistency of the time and load variables is ensured by constraints (7) and (8). Equalities (9) define the ride time of each user, which is bounded by constraints (12). It is worth mentioning that the latter also act as precedence constraints because the nonnegativity of the L_i^k variables ensures that node i will be visited before node $n+i$ for every user i. Finally, inequalities (10) bound the duration of each route, while (11) and (13) impose time windows and capacity constraints, respectively.

This formulation is nonlinear because of constraints (7) and (8). Introducing constants M_{ij}^k and W_{ij}^k, these constraints can, however, be linearized as follows:

$$B_j^k \geqslant B_i^k + d_i + t_{ij} - M_{ij}^k(1 - x_{ij}^k) \quad \forall i \in N,\ j \in N,\ k \in K, \tag{15}$$

$$Q_j^k \geqslant Q_i^k + q_j - W_{ij}^k(1 - x_{ij}^k) \quad \forall i \in N,\ j \in N,\ k \in K. \tag{16}$$

The validity of these constraints is ensured by setting $M_{ij}^k \geqslant \max\{0, l_i + d_i + t_{ij} - e_j\}$ and $W_{ij}^k \geqslant \min\{Q_k, Q_k + q_i\}$. These constraints are very similar to the Miller-Tucker-Zemlin subtour elimination constraints for the TSP (Miller et al. 1960).

One can reduce the number of variables and constraints in Model (1)–(14) by using aggregate time variables B_i at every node except the origin depot 0 and the destination depot $2n+1$. In this case, one replaces (7) and (9) with the constraints

$$B_j \geqslant (B_0^k + d_0 + t_{0j})x_{0j}^k \quad \forall j \in N,\ k \in K, \tag{17}$$

$$B_j \geqslant (B_i + d_i + t_{ij})\sum_{k\in K} x_{ij}^k \quad \forall i \in N,\ j \in N, \tag{18}$$

$$B_{2n+1}^k \geqslant (B_i + d_i + t_{i,2n+1})x_{i,2n+1}^k \quad \forall i \in N,\ k \in K, \tag{19}$$

$$L_i = B_{n+i} - (B_i + d_i) \quad \forall i \in P, \tag{20}$$

to which the same linearization process can be applied. Along the same lines, if the fleet of vehicles is homogeneous in the sense that $Q_k = Q$ for every $k \in K$, one can replace (8) with

$$Q_j \geqslant (Q_0^k + q_j)x_{0j}^k \quad \forall j \in N,\ k \in K, \tag{21}$$

$$Q_j \geqslant (Q_i + q_j)\sum_{k\in K} x_{ij}^k \quad \forall i \in N,\ j \in N, \tag{22}$$

$$Q_{2n+1}^k \geqslant (Q_i + q_{2n+1})x_{i,2n+1}^k \quad \forall i \in N,\ k \in K. \tag{23}$$

Finally, as shown by Desrochers and Laporte (1991), the linearized form of constraints (22) can be lifted as follows by taking the reverse arc (j,i) into account:

$$Q_j \geqslant Q_i + q_j - W_{ij}\left(1 - \sum_{k\in K} x_{ij}^k\right) + (W_{ij} - q_i - q_j)\sum_{k\in K} x_{ji}^k \quad \forall i \in N,\ j \in N,\ k \in K. \tag{24}$$

Observe that in our case an equivalent lifting cannot be performed with constraints (18) because waiting will sometimes take place after the beginning of a time window (i.e., $B_i > e_i$) to reduce the ride time of a user.

4. Valid Inequalities

We now describe several families of valid inequalities for the DARP. All of these inequalities are of course redundant for Model (1)–(14) but can strengthen its LP-relaxation. Because of the structure of the model, analyzing whether these inequalities define facets of the DARP polytope appears to be a rather challenging task. However, their usefulness is demonstrated through computational experiments in the last section.

The following additional notation will be used to describe the valid inequalities. Given a node set $S \subseteq N$, define $\bar{S} = \{i \in N \mid i \notin S\}$ and $\delta(S) = \delta^+(S) \cup \delta^-(S)$, where $\delta^+(S) = \{(i,j) \in A \mid i \in S,\ j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A \mid i \notin S,\ j \in S\}$. For notational convenience, let $x_{ij} = \sum_{k\in K} x_{ij}^k$ denote the total flow on arc (i,j) and define $x(S) = \sum_{i,j\in S} x_{ij}$. Similarly, let $x(A') = \sum_{(i,j)\in A'} x_{ij}$ for any arc set $A' \subseteq A$.

4.1. Bounds on Time and Load Variables

As first suggested by Desrochers and Laporte (1991) in the context of the TSP with time windows, bounds on the time variables B_i can be strengthened as follows:

$$B_i \geqslant e_i + \sum_{j \in N \setminus \{i\}} \max\{0, e_j - e_i + d_j + t_{ij}\} x_{ji}, \tag{25}$$

$$B_i \leqslant l_i - \sum_{j \in N \setminus \{i\}} \max\{0, l_i - l_j + d_i + t_{ij}\} x_{ij}. \tag{26}$$

These inequalities were used, for example, for solving the asymmetric TSP with time windows by branch and cut (Ascheuer et al. 2001).

Similarly, bounds on load variables Q_i can also be strengthened as follows:

$$Q_i \geqslant \max\{0, q_i\} + \sum_{j \in N \setminus \{i\}} \max\{0, q_j\} x_{ji}, \tag{27}$$

$$Q_i \leqslant \min\{Q, Q + q_i\} - \Big(Q - \max_{j \in N \setminus \{i\}} \{q_j\} - q_i\Big) x_{0i} - \sum_{j \in N \setminus \{i\}} \max\{0, q_j\} x_{ij}. \tag{28}$$

4.2. Subtour Elimination Constraints

Consider the simple subtour elimination constraint $x(S) \leqslant |S| - 1$ for $S \subseteq P \cup D$. In the case of the DARP, this inequality can be lifted in many different ways by taking into account the fact that for each user i, node i must be visited before node $n + i$.

For any set $S \subseteq P \cup D$, let $\pi(S) = \{i \in P \mid n + i \in S\}$ and $\sigma(S) = \{n + i \in D \mid i \in S\}$ denote the sets of *predecessors* and *successors* of S, respectively. Balas et al. (1995) have proposed two families of inequalities for the precedence-constrained asymmetric TSP that also apply to the DARP by observing that each node $i \in P \cup D$ is either the predecessor or the successor of exactly one other node.

For $S \subseteq P \cup D$, the following inequality, called a successor inequality (or σ-inequality) is valid for the DARP:

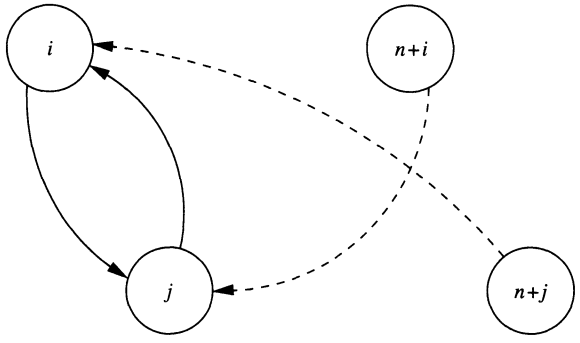
$$x(S) + \sum_{i \in \bar{S} \cap \sigma(S)} \sum_{j \in S} x_{ij} + \sum_{i \in \bar{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} \leqslant |S| - 1. \tag{29}$$

Example. Consider the node set $S = \{i, j\} \subseteq P$. The resulting inequality is $x_{ij} + x_{ji} + x_{n+i,j} + x_{n+j,i} \leqslant 1$. This example is illustrated in Figure 1, where dotted arcs correspond to lifted arcs (i.e., arcs that are not part of the classical subtour elimination constraint). Observe that variables $x_{n+i,i}$ and $x_{n+j,j}$ are not to be introduced in the lifted inequality because the corresponding arcs can be trivially removed from the graph.

Similarly, for any set $S \subseteq P \cup D$, the following predecessor inequality (or π-inequality) is valid for the DARP:

$$x(S) + \sum_{i \in S} \sum_{j \in \bar{S} \cap \pi(S)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in \bar{S} \setminus \pi(S)} x_{ij} \leqslant |S| - 1. \tag{30}$$

Figure 1. Successor inequality for $S = \{i, j\} \subseteq P$.



Example. Consider the node set $S = \{n + i, n + j\} \subseteq D$. The resulting predecessor inequality is $x_{n+i,n+j} + x_{n+j,n+i} + x_{n+i,j} + x_{n+j,i} \leqslant 1$. This is illustrated in Figure 2.

In the case of a directed formulation, one can also lift subtour elimination constraints by taking into account the orientation of the arcs. For an ordered set $S = \{i_1, i_2, \ldots, i_h\} \subseteq N$ with $h \geqslant 3$ nodes, Grötschel and Padberg (1985) proposed the following inequalities, called D_k^+ and D_k^-, respectively, for the asymmetric TSP:

$$\sum_{j=1}^{h-1} x_{i_j, i_{j+1}} + x_{i_h, i_1} + 2 \sum_{j=2}^{h-1} x_{i_j, i_1} + \sum_{j=3}^{h-1} \sum_{l=2}^{j-1} x_{i_j, i_l} \leqslant h - 1, \tag{31}$$

$$\sum_{j=1}^{h-1} x_{i_j, i_{j+1}} + x_{i_h, i_1} + 2 \sum_{j=3}^{h} x_{i_1, i_j} + \sum_{j=4}^{h} \sum_{l=3}^{j-1} x_{i_j, i_l} \leqslant h - 1. \tag{32}$$

Of course, different liftings are obtained by considering different orderings of the nodes in set S. In the case of the DARP, these inequalities can be further lifted by taking precedence relationships into account. This results in the following two propositions.

Proposition 1. *Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq P \cup D$. The following inequality is valid for the DARP:*

$$\sum_{j=1}^{h-1} x_{i_j, i_{j+1}} + x_{i_h, i_1} + 2 \sum_{j=2}^{h-1} x_{i_j, i_1} + \sum_{j=3}^{h-1} \sum_{l=2}^{j-1} x_{i_j, i_l} + \sum_{n+i_p \in \bar{S} \cap \sigma(S)} x_{n+i_p, i_1} \leqslant h - 1. \tag{33}$$

Figure 2. Predecessor inequality for $S = \{n + i, n + j\} \subseteq D$.

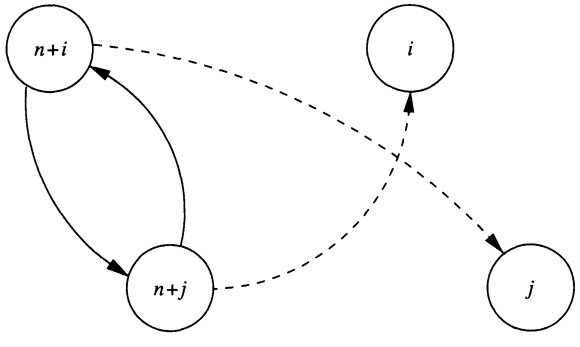


Figure 3. Lifted directed subtour elimination constraint for $S = \{i, j, k\} \subseteq P$.

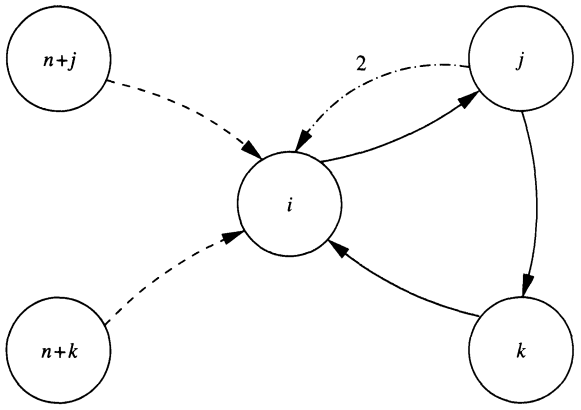
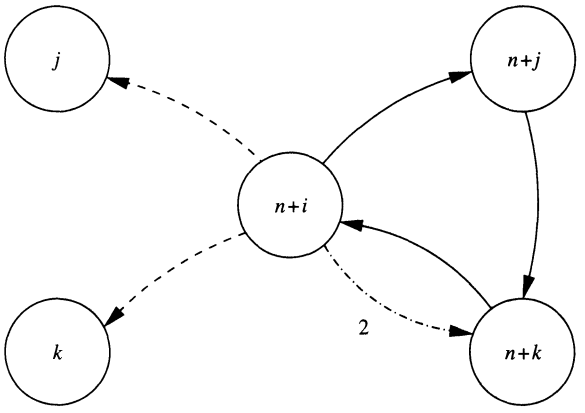


Figure 4. Lifted directed subtour elimination constraint for $S = \{n+i, n+j, n+k\} \subseteq D$.



PROOF. Suppose that one arc of the form $(n + i_p, i_1)$ with $n + i_p \in \bar{S} \cap \sigma(S)$ is part of the solution. Then, all arcs of the form (i_j, i_1) with $2 \leqslant j \leqslant h - 1$ cannot belong to the solution. As a result, if the left-hand side of inequality (33) is larger than $h - 1$, then there exists a subpath linking the h nodes in set S. However, because set S contains the origin node i_p, this subpath, together with the arc $(n + i_p, i_1)$, would violate the precedence constraint for user i_p. □

EXAMPLE. Consider the node set $S = \{i, j, k\} \subseteq P$. One possible lifted directed subtour elimination constraint (obtained with $i_1 = i$, $i_2 = j$, $i_3 = k$) is $x_{ij} + x_{jk} + x_{ki} + 2x_{ji} + x_{n+j,i} + x_{n+k,i} \leqslant 2$. This is illustrated in Figure 3.

PROPOSITION 2. *Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq P \cup D$. The following inequality is valid for the DARP*:

$$\sum_{j=1}^{h-1} x_{i_j, i_{j+1}} + x_{i_h, i_1} + 2\sum_{j=3}^{h} x_{i_1, i_j} + \sum_{j=4}^{h}\sum_{l=3}^{j-1} x_{i_j, i_l} + \sum_{i_p \in \bar{S} \cap \pi(S)} x_{i_1, i_p} \leqslant h - 1. \quad (34)$$

PROOF. The proof is similar to that of Proposition 3 by observing that if one arc of the form (i_1, i_p) with $i_p \in \bar{S} \cap \pi(S)$ is part of the solution, then all arcs of the form (i_1, i_j) with $3 \leqslant j \leqslant h$ cannot belong to the solution. □

EXAMPLE. Consider the node set $S = \{n + i, n + j, n + k\} \subseteq D$. One possible lifted directed subtour elimination constraint (obtained with $i_1 = n + i$, $i_2 = n + j$, $i_3 = n + k$) is $x_{n+i,n+j} + x_{n+j,n+k} + x_{n+k,n+i} + 2x_{n+i,n+k} + x_{n+i,j} + x_{n+i,k} \leqslant 2$. This is illustrated in Figure 4.

For the case $h = 3$, Ascheuer (1996) proposed slightly stronger liftings. These do not, however, generalize to the case $h > 3$.

4.3. Capacity Constraints

For any subset $S \subseteq P \cup D$, let $R(S)$ denote the minimum number of vehicles needed to visit all nodes in S. The constraint $x(\delta(S)) \geqslant 2R(S)$ is then a valid inequality. Although computing $R(S)$ is difficult, a lower approximation is provided by $\lceil q(S)/Q \rceil$, where $q(S) = \sum_{i \in S} q_i$. The resulting inequality is called a *rounded capacity inequality* in the context of the VRP.

While capacity inequalities play an important role in most branch-and-cut algorithms for the VRP (see, e.g., Naddef and Rinaldi 2002), they are not likely to be very strong for the DARP because of the presence of both positive and negative q_i values. In particular, $q(P \cup D) = 0$ by definition and, in the absence of time windows, all nodes can be visited by the same vehicle. Useful inequalities can, however, be obtained by restricting S to be a subset of either P or D and setting $q(S) = |\sum_{i \in S} q_i|$. It should be emphasized that in our context, the quantity $\lceil q(S)/Q \rceil$ estimates the required number of vehicle *passages* in set S rather than the number of vehicles per se. Indeed, by leaving and entering set S more than once, the same vehicle may be able to visit all nodes in the set even though $q(S) > Q$.

4.4. Precedence Constraints

Consider a node set S such that $0 \in S$, $i \notin S$, $n + i \in S$, and $2n + 1 \notin S$ for at least one user i. Because node i must be visited before node $n + i$, $x(S) \leqslant |S| - 2$ and, equivalently, $x(\delta(S)) \geqslant 3$. The same applies to node sets S such that $0 \notin S$, $i \in S$, $n + i \notin S$, and $2n + 1 \in S$ for at least one user i. These inequalities, which are also valid for the DARP, were introduced by Ruland and Rodin (1997) for the pick-up and delivery problem.

4.5. Generalized Order Constraints

Let $U_1, \ldots, U_m \subset N$ be mutually disjoint subsets and let $i_1, \ldots, i_m \in P$ be users such that $0, 2n + 1 \notin U_l$ and i_l, $n + i_{l+1} \in U_l$ for $l = 1, \ldots, m$ (where $i_{m+1} = i_1$). The following inequality, introduced by Ruland and Rodin (1997), is also valid for the DARP:

$$\sum_{l=1}^{m} x(U_l) \leqslant \sum_{l=1}^{m} |U_l| - m - 1. \quad (35)$$

Similar inequalities, called *precedence cycle breaking inequalities*, have also been proposed by Balas et al. (1995).

In the directed case, generalized order constraints can be lifted in two different ways, as shown by the following propositions.

PROPOSITION 3. *The following inequality is valid for the DARP*:

$$\sum_{l=1}^{m} x(U_l) + \sum_{l=2}^{m-1} x_{i_1,i_l} + \sum_{l=3}^{m} x_{i_1,n+i_l} \leqslant \sum_{l=1}^{m} |U_l| - m - 1. \quad (36)$$

PROOF. First observe that in any feasible integer solution, at most one of the lifted arcs may be part of the solution because they all belong to $\delta^+(i_1)$. To demonstrate the validity of the lifting, we show that if any of the lifted arcs are part of the solution, then there are at least two subsets U_l for which $x(U_l) \leqslant |U_l| - 2$. Because $x(U_l) \leqslant |U_l| - 1$ for all other subsets, the validity of the inequality follows directly. The details of the proof are given in the appendix. □

REMARK 1. The numbering of the sets $U_1, \ldots, U_m$ is arbitrary and leads, by symmetry, to $m!$ possibly different liftings.

EXAMPLE. Consider the subsets $U_1 = \{i, n+j\}$, $U_2 = \{j, n+k\}$, and $U_3 = \{k, n+i\}$ with $i_1 = i$, $i_2 = j$, and $i_3 = k$. The lifted generalized order constraint is $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{ij} + x_{i,n+k} \leqslant 2$. This is illustrated in Figure 5.

PROPOSITION 4. *The following inequality is valid for the DARP*:

$$\sum_{l=1}^{m} x(U_l) + \sum_{l=2}^{m-2} x_{n+i_1,i_l} + \sum_{l=2}^{m-1} x_{n+i_1,n+i_l} \leqslant \sum_{l=1}^{m} |U_l| - m - 1. \quad (37)$$

PROOF. The reasoning is similar to that of Proposition 5, and the details of the proof are given in the appendix. □

EXAMPLE. Consider the subsets $U_1 = \{i, n+j\}$, $U_2 = \{j, n+k\}$, and $U_3 = \{k, n+i\}$ with $i_1 = i$, $i_2 = j$, and $i_3 = k$. The lifted generalized order constraint is $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{n+i,n+j} \leqslant 2$. This is illustrated in Figure 6. Observe that in inequality (37), the value of m must be larger than or equal to 4 for the second term of the inequality to have any effect.

Figure 5. Lifted generalized order constraint with $m = 3$ (first lifting).

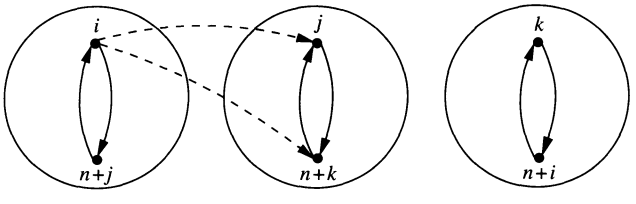
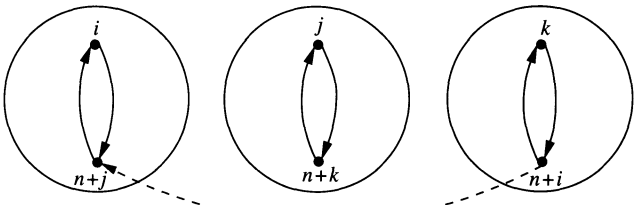


Figure 6. Lifted generalized order constraint with $m = 3$ (second lifting).



4.6. Order-Matching Constraints

Consider two nodes $i, j \in P$ and a subset H such that $\{i, j\} \subseteq H \subseteq N \backslash \{0, n+i, n+j, 2n+1\}$. The following inequality, introduced by Ruland and Rodin (1997), is also valid for the DARP:

$$x(H) + x(\{i, n+i\}) + x(\{j, n+j\}) \leqslant |H|. \quad (38)$$

As suggested by Ruland (1995), this inequality can be lifted by introducing the term $x(\{h, n+h\})$ for every user h such that $h \in H$ and $n+h \notin H$. In the directed case, order-matching constraints are in fact redundant because any arc of the form $(n+h, h)$ can be removed from the graph. As a result, all arcs in the left-hand side of (38) have their origin node in set H. Hence, the sum of the flows on these arcs cannot exceed $|H|$, even in a fractional solution.

Order-matching constraints can, however, be generalized by replacing the arcs $(h, n+h)$ by node sets. This leads to the following proposition.

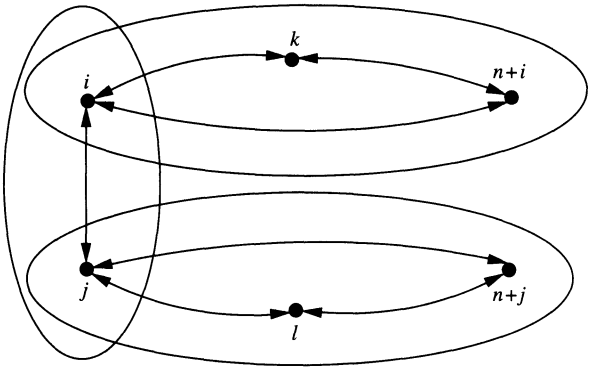
PROPOSITION 5. *Let $i_1, \ldots, i_m$ be m users and let $H \subset P \cup D$ and $T_h \subset P \cup D$, $h = 1, \ldots, m$ be node sets such that $\{i_h, n+i_h\} \subseteq T_h$ and $H \cap T_h = \{i_h\}$. The following inequality is valid for the DARP*:

$$x(H) + \sum_{h=1}^{m} x(T_h) + \leqslant |H| + \sum_{h=1}^{m} |T_h| - 2m. \quad (39)$$

PROOF. First observe that $x(H) \leqslant |H| - 1$ and $x(T_h) \leqslant |T_h| - 1$ for $h = 1, \ldots, m$. If $x(T_h) = |T_h| - 1$ for any given subset T_h, then there exists a path connecting all nodes in T_h, including i_h and $n+i_h$. However, this path cannot finish at node i_h because of the precedence constraint for user i_h. Let α be the number of sets T_h for which $x(T_h) = |T_h| - 1$. Then, $x(\delta^+(H)) \geqslant \alpha$. Because $x(\delta^+(H)) = x(\delta^-(H))$ and $2x(H) + x(\delta^+(H)) + x(\delta^-(H)) = 2|H|$, one obtains that $2x(H) \leqslant 2|H| - 2\alpha$ and thus $x(H) \leqslant |H| - \alpha$. Finally, because $x(T_h) \leqslant |T_h| - 2$ for the remaining $m - \alpha$ sets, one may conclude that $x(H) + \sum_{h=1}^{m} x(T_h) \leqslant |H| - \alpha + \sum_{h=1}^{m} (|T_h| - 1) - (m - \alpha)$, which simplifies to expression (39). □

EXAMPLE. Consider the sets $H = \{i, j\}$, $T_1 = \{i, n+i, k\}$ and $T_2 = \{j, n+j, l\}$ with $i_1 = i$ and $i_2 = j$. The resulting inequality is $x(H) + x(T_1) + x(T_2) \leqslant 4$ and is illustrated in Figure 7.

Figure 7. Generalized order-matching constraint with $m = 2$.



REMARK 2. Inequality (39) is stronger than the corresponding TSP comb constraint, valid for $m \geqslant 3$ and odd, which is defined as $x(H) + \sum_{h=1}^{m} x(T_h) \leqslant |H| + \sum_{h=1}^{m} |T_h| - (3m+1)/2$.

Inequalities (39) can be further lifted by introducing the arcs from any node in H to any node outside of H. Nevertheless, in the presence of lifted subtour elimination constraints (30), these inequalities are redundant, as will be shown in the following proposition.

PROPOSITION 6. *Define* $\tilde{H} = \bigcup_{h=1}^{m} \{i_h\}$. *The following inequality, which is a strengthening of* (39), *is redundant in the presence of inequalities* (30):

$$x(H) + \sum_{h=1}^{m} x(T_h) + \sum_{i \in \tilde{H}} \sum_{j \in \tilde{H} \setminus T_i} x_{ij} + \sum_{i \in H \setminus \tilde{H}} \sum_{j \in \tilde{H}} x_{ij} \leqslant |H| + \sum_{h=1}^{m} |T_h| - 2m. \quad (40)$$

PROOF. Inequality (40) can be written equivalently as

$$\sum_{i \in H} \sum_{j \in N} x_{ij} + \sum_{h=1}^{m} \sum_{i \in T_h \setminus \{i_h\}} \sum_{j \in T_h} x_{ij} \leqslant |H| + \sum_{h=1}^{m} |T_h| - 2m.$$

For every subset T_h, $h = 1, \ldots, m$, one has that $x(T_h \setminus \{i_h\}) + \sum_{i \in T_h \setminus \{i_h\}} x_{i,i_h} \leqslant |T_h| - 2$ because of inequality (30) defined for subset $S = T_h \setminus \{i_h\}$, whose cardinality is $|T_h| - 1$. As a result, $\sum_{i \in T_h \setminus \{i_h\}} \sum_{j \in T_h} x_{ij} \leqslant |T_h| - 2$. In addition, one has that $\sum_{i \in H} \sum_{j \in N} x_{ij} = |H|$ by constraints (2) and (3). The result follows directly from these two observations. □

4.7. Infeasible Path Constraints

Ride-time constraints may give rise to paths that are infeasible in an integer solution, but nonetheless feasible in a fractional solution. Forbidding such paths can be accomplished through the valid inequality introduced in the next proposition.

PROPOSITION 7. *Assume that the triangle inequality holds for travel times. For any directed path* $\mathcal{P} = \{i, k_1, k_2, \ldots, k_p, n+i\}$ *such that* $t_{i,k_1} + d_{k_1} + t_{k_1,k_2} + d_{k_2} + \cdots + t_{k_p,n+i} > L$, *the following inequality is valid for the DARP*:

$$x_{i,k_1} + \sum_{h=1}^{p-1} x_{k_h,k_{h+1}} + x_{k_p,n+i} \leqslant p - 1. \quad (41)$$

PROOF. Suppose that the value of the left-hand side of inequality (41) is equal to p in a feasible integer solution. Then, there is a single arc from path $\mathcal{P}$ not belonging to the solution because the path contains $p+1$ arcs. Because the solution is feasible, it must contain a path from i to $n+i$ in which the missing arc has been replaced by at least two other arcs. However, because the triangle inequality holds for travel times, the path from i to $n+i$ has a larger duration than that of $\mathcal{P}$. As a result, its duration must exceed L, which contradicts the assumption that the solution is feasible. □

REMARK 3. As will be shown in the next section, the case $p = 1$ can be handled directly through the simple elimination of arcs in a preprocessing step.

5. Branch-and-Cut Algorithm

Branch and cut is a popular approach for solving combinatorial problems. It has been applied successfully to several routing problems (see, e.g., Ascheuer et al. 2001, Gendreau et al. 1998, and Naddef and Rinaldi 2002). This section describes the branch-and-cut algorithm used for the DARP. It focuses on the preprocessing techniques developed to reduce problem size and on the separation heuristics used to identify violated inequalities.

After applying the preprocessing steps presented in §5.1 and generating an initial pool of inequalities, the algorithm first solves the LP relaxation of the problem. If the solution to the LP relaxation is integer, an optimal solution has been identified. Otherwise, an enumeration tree is constructed and violated valid inequalities are generated at some nodes of this tree by means of the separation heuristics described in §5.3. Because all inequalities described in §4 are valid for the original formulation, the inequalities added at any node of the tree are also valid for all other nodes. Hence, whenever the LP bound is evaluated at a given node of the tree, the linear program incorporates all cuts generated so far.

To control the branch-and-cut process, additional aggregate variables $y_i^k = \sum_{j \in N} x_{ij}^k$, $i \in P$, are introduced in the formulation to reflect the assignment of requests to vehicles. At a given node of the search tree, if the y_i^k variables are all integer in the current relaxation but there is at least one fractional x_{ij}^k variable, the separation heuristics are executed in the hope of identifying violated valid inequalities (the heuristics are also executed at the root node). If at least one of the heuristics succeeds in finding one or more violated inequalities, the relaxation is solved with all identified

cuts and the heuristics are executed again. The cut generation process at a node terminates when all heuristics fail to find any violated inequality. If the solution to the relaxation is still fractional after the generation of cuts, branching is performed on a fractional y_i^k variable, if there is any, or on a fractional x_{ij}^k variable, otherwise. The variable selected for branching is that whose value is the farthest from the nearest integer. After a node has been processed, the next node selected for cutting and branching is that with the best bound.

In addition to the application of the separation heuristics at some nodes of the branch-and-bound tree, a pool of inequalities is checked exhaustively for violations at each node of the tree, including those where not all y_i^k variables take integer values. The inequalities that belong to this pool are described in §5.2. Finally, prior to solving the problem by branch and cut, an upper bound is computed by using the tabu search heuristic of Cordeau and Laporte (2003b). This upper bound is then used to prune the enumeration tree whenever the solution value at a given node exceeds that of the upper bound.

5.1. Preprocessing

This section describes the time-window tightening, arc elimination, and variable-fixing steps that are performed prior to applying the branch-and-cut algorithm.

5.1.1. Time-Window Tightening. Let T denote the end of the planning horizon. In the case of an outbound user, the time window at the origin node can be tightened by setting $e_i = \max\{0, e_{n+i} - L - d_i\}$ and $l_i = \min\{l_{n+i} - t_{i,n+i} - d_i, T\}$. In the case of an inbound user, the time window at the destination node can be tightened by setting $e_{n+i} = \max\{0, e_i + d_i + t_{i,n+i}\}$ and $l_{n+i} = \min\{l_i + d_i + L, T\}$.

The time window on nodes 0 and $2n+1$ can also be tightened by setting $e_0 = e_{2n+1} = \min_{i \in P \cup D}\{e_i - t_{0i}\}$ and $l_0 = l_{2n+1} = \max_{i \in P \cup D}\{l_i + d_i + t_{i,2n+1}\}$.

5.1.2. Arc Elimination. Formulation (1)–(14) is defined on a complete graph G. However, because of time windows, pairing, and ride-time constraints, several arcs can in fact be removed from the graph as they cannot belong to a feasible solution.

A simple analysis leads to the following observations:

- arcs $(0, n+i)$, $(i, 2n+1)$, and $(n+i, i)$ are infeasible for $i \in P$;
- arc (i, j) with $i, j \in N$ is infeasible if $e_i + d_i + t_{ij} > l_j$;
- arcs (i, j) and $(j, n+i)$ with $i \in P$, $j \in N$ are both infeasible if $t_{ij} + d_j + t_{j,n+i} > L$.

As first proposed by Dumas et al. (1991), combining time windows and pairing constraints leads to even stronger elimination rules:

- arc $(i, n+j)$ is infeasible if path $\mathcal{P} = \{j, i, n+j, n+i\}$ is infeasible;
- arc $(n+i, j)$ is infeasible if path $\mathcal{P} = \{i, n+i, j, n+j\}$ is infeasible;
- arc (i, j) is infeasible if paths $\mathcal{P}_1 = \{i, j, n+i, n+j\}$ and $\mathcal{P}_2 = \{i, j, n+j, n+i\}$ are both infeasible;
- arc $(n+i, n+j)$ is infeasible if paths $\mathcal{P}_1 = \{i, j, n+i, n+j\}$ and $\mathcal{P}_2 = \{j, i, n+i, n+j\}$ are both infeasible.

In the presence of ride-time and capacity constraints, further elimination can be performed by identifying pairs of users that are incompatible (i.e., that cannot be assigned to the same vehicle) because of the interaction between time-window, capacity, and ride-time constraints. Incompatible user pairs $\{i, j\}$ can be identified by checking the feasibility of the following paths: $\{i, j, n+i, n+j\}$, $\{i, j, n+j, n+i\}$, $\{j, i, n+i, n+j\}$, $\{j, i, n+j, n+i\}$, $\{i, n+i, j, n+j\}$, and $\{j, n+j, i, n+i\}$.

If none of these six paths is feasible, then all eight arcs between $\{i, n+i\}$ and $\{j, n+j\}$ can be eliminated. Because of ride-time constraints, checking the feasibility of a path is not always straightforward. In the case of the path $\{i, j, n+i, n+j\}$, for example, setting $B_i = e_i$ may lead to the violation of the ride-time constraint for user i in the event that unnecessary waiting time then occurs at node j. For this reason, the forward time slack should be computed at node i so as to delay the beginning of service as much as possible without violating any of the time windows. The same can be said about node j where the beginning of service should be delayed as much as possible by taking into account the additional constraint that the maximum ride time for user i should not be exceeded. In general, the forward time slack F_i at node i in a path $\{i, i+1, \ldots, q\}$ can be computed as follows:

$$F_i = \min_{i \leqslant j \leqslant q} \left\{ \sum_{i < p \leqslant j} W_p + \min\{l_j - B_j, L - P_j\} \right\}, \tag{42}$$

where W_i denotes the waiting time at node i, P_i denotes the ride time of the user whose destination node is i if $n+1 \leqslant i \leqslant 2n$, and $P_i = -\infty$, otherwise. This definition of the forward time slack generalizes that of Savelsbergh (1992) for the TSP with time windows.

5.1.3. Variable Fixing. The identification of incompatible user pairs can also be used to permanently assign users to specific vehicles. If the fleet of vehicles is homogeneous, one can create a graph $G' = (N', E')$, where $N' = \{1, \ldots, n\}$ and E' contains an edge (i, j) if i and j are incompatible users. Given a clique in G', each user in the clique can be assigned to a different vehicle. In addition, if user i is assigned to vehicle k, constraints can be added to the formulation so as to forbid the assignment to vehicle k of users that are incompatible with i. Finding a clique of maximum cardinality in G' may be very time consuming when n is large. In our implementation, we thus use a greedy heuristic described by Johnson (1974).

5.2. Initial Pool of Inequalities

This section describes the inequalities that are enumerated exhaustively and whose violations are checked individually at every node of the branch-and-bound tree. These

inequalities are chosen to ensure that the size of the pool remains reasonable and the time spent processing a node does not overly increase.

Bounds on time and load variables. For each node $i \in P \cup D$, the four bounding inequalities described in §4.1 are generated and added to the pool.

Subtour elimination constraints. For each pair of users $i, j \in P$, inequality (29) yields the following three particular cases:

- $S = \{i, j\} \Rightarrow x_{ij} + x_{ji} + x_{n+i,j} + x_{n+j,i} \leqslant 1$,
- $S = \{i, n+j\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} \leqslant 1$,
- $S = \{i, n+i, j\} \Rightarrow x_{ij} + x_{ji} + x_{i,n+i} + x_{j,n+i} + x_{n+i,j} + x_{n+j,i} + x_{n+j,n+i} \leqslant 2$.

Similarly, inequality (30) yields the following cases:

- $S = \{n+i, n+j\} \Rightarrow x_{n+i,n+j} + x_{n+j,n+i} + x_{n+i,j} + x_{n+j,i} \leqslant 1$,
- $S = \{i, n+j\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{i,j} \leqslant 1$,
- $S = \{i, n+i, n+j\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{i,n+i} + x_{n+j,n+i} + x_{n+i,j+j} + x_{ij} + x_{n+i,j} \leqslant 2$,

while inequalities (33) and (34) provide the following two cases:

- $S = \{n+i, j, n+i\} \Rightarrow x_{n+i,j} + 2x_{j,n+i} + x_{ji} + x_{i,n+i} + x_{n+j,n+i} \leqslant 2$,
- $S = \{i, n+i, n+j\} \Rightarrow x_{i,n+i} + x_{n+i,n+j} + x_{n+j,i} + 2x_{i,n+j} + x_{ij} \leqslant 2$.

Finally, the four-node subtour inequality $x(S) \leqslant 3$ is generated for every user pair $i, j \in P$ yielding the subset $S = \{i, j, n+i, n+j\}$.

Precedence constraints. For every pair of users $i, j \in P$, the following four particular cases of precedence constraints are generated:

- $S = \{0, i, n+j\} \Rightarrow x_{0i} + x_{i,n+j} + x_{n+j,i} \leqslant 1$,
- $S = \{i, n+j, 2n+1\} \Rightarrow x_{i,n+j} + x_{n+j,i} + x_{n+j,2n+1} \leqslant 1$,
- $S = \{0, i, n+i, n+j\} \Rightarrow x_{0i} + x_{i,n+i} + x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} + x_{n+j,n+i} \leqslant 2$,
- $S = \{i, j, n+j, 2n+1\} \Rightarrow x_{ij} + x_{ji} + x_{i,n+j} + x_{n+j,i} + x_{j,n+j} + x_{n+j,2n+1} \leqslant 2$.

Generalized order constraints. For every pair of requests $i, j \in P$, the following inequality is generated:

$$x_{i,n+j} + x_{n+j,i} + x_{n+i,j} + x_{j,n+i} \leqslant 1.$$

Infeasible path constraints. Finally, the following inequality is generated for every pair of requests $i, j \in P$ such that $t_{ij} + d_j + t_{j,n+j} d_{n+j} + t_{n+j,n+i} > L$:

$$x_{ij} + x_{j,n+j} + x_{n+j,n+i} \leqslant 1.$$

In addition, for each pair of incompatible users $i, j \in P$ and every node $l \in P \cup D$, the following inequalities are generated:

- $x_{il} + x_{li} + x_{lj} + x_{jl} \leqslant 1$,
- $x_{il} + x_{li} + x_{l,n+j} + x_{n+j,l} \leqslant 1$,
- $x_{n+i,l} + x_{l,n+i} + x_{lj} + x_{jl} \leqslant 1$,
- $x_{n+i,l} + x_{l,n+i} + x_{l,n+j} + x_{n+j,l} \leqslant 1$.

5.3. Separation Heuristics

We now describe the separation heuristics used to identify additional violated inequalities. At the root node as well as when all y_i^k variables are integer but there is at least one fractional x_{ij}^k variable at a given node, the following heuristics are executed sequentially.

5.3.1. Subtour Elimination Constraints. The identification of violated inequalities of the form $x(S) \leqslant |S| - 1$ can be achieved by solving a series of maximum-flow problems between any node i and all other nodes $j \in N \backslash \{i\}$. However, in addition to being time consuming, this approach does not take the possible liftings into account. For these reasons, we resort to a simple tabu search heuristic inspired from that proposed by Augerat et al. (1999).

Using the fact that $2x(S) + x(\delta(S)) = 2|S|$ in a feasible integer solution, violations of (29) can be identified by finding node sets S such that

$$x(\delta(S)) - 2 \sum_{i \in \bar{S} \cap \sigma(S)} \sum_{j \in S} x_{ij} - 2 \sum_{i \in \bar{S} \backslash \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} < 2. \quad (43)$$

The heuristic starts with an empty set S. At each iteration, it either adds or removes a node from the set S so as to minimize the left-hand side of (43). Whenever a node is removed from set S, its reinsertion is declared tabu for θ iterations. The heuristic runs for a preset number of iterations (25 in our implementation) and may identify several violated inequalities during a single execution. At each iteration, the current set S is also checked for possible violations of inequality (33). To this purpose, the node with the largest outgoing flow is numbered as i_1, and the other nodes are numbered at random.

A similar heuristic is used to identify violations of inequalities (30) and (34). In the latter case, the node with the largest incoming flow is numbered as i_1.

5.3.2. Capacity Constraints. Again, we use a tabu search heuristic to identify sets S such that $q(S) > Q$ and $x(\delta(S)) < 4$. The heuristic starts either with a random subset $S \subseteq P$ or with a random subset $S \subseteq D$. At each iteration, a node is either removed or added to the set S so as to minimize the value of $x(\delta(S))$, with the constraint that $q(S) > Q$. The heuristic runs for 25 iterations and multiple violated inequalities may be identified during a single execution.

5.3.3. Generalized Order Constraints. Ruland (1995) proposed an approach to identify violated generalized order constraints with $m = 2$. This approach requires the computation of $O(|N|^2)$ maximum flows. Here, we use instead three simple heuristics, the second and third of which take advantage of the lifted forms (36) and (37).

The first heuristic attempts to identify violations of inequalities (35) for the special case where $m = 2$ and $|U_1| = |U_2| = 3$. For each pair of users $i, j \in P$, we first form the subsets $U_1 = \{i, n+j\}$ and $U_2 = \{j, n+i\}$. We

then identify nodes k_1 and k_2 such that $x(U_1)$ and $x(U_2)$ are maximized, and check for a violation of the resulting inequality.

The second and third heuristics are aimed at finding violations of inequalities (36) and (37) for the special case where $m = 3$ and $|U_1| = |U_2| = |U_3| = 2$. The second heuristic finds, for each user i, a user j that maximizes $x_{i,n+j} + x_{n+j,i} + x_{ij}$. It then finds a user k such that the left-hand side of (36) is maximized. Similarly, the third heuristic finds, for each user i, a user j that maximizes $x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j}$, and then finds a user k that maximizes the left-hand side of (37).

5.3.4. Infeasible Path Inequalities. Violated path inequalities are identified by means of a path-construction heuristic applied to each user $i \in P$. The heuristic starts with node i and gradually constructs a path $\mathcal{P}_i = \{i, k_1, k_2, \ldots\}$ by iteratively moving from the current node k_j to the node k_l that maximizes the value of $x_{k_j k_l}$. The process stops if a cycle is formed or if the heuristic reaches either node $n+i$ or node $2n+1$. If the path stops at node $n+i$, it is checked for a violation of inequality (41).

6. Computational Experiments

The branch-and-cut algorithm was implemented in C++ by using ILOG Concert 1.3 and CPLEX 8.1. It was run on a 2.5 GHz Pentium 4 computer with 512 Mb of memory.

The algorithm was applied to two sets of randomly generated instances comprising up to 48 users. In an instance with n users, where n is even, users $1, \ldots, n/2$ are assumed to formulate outbound requests while users $n/2+1, \ldots, n$ formulate inbound requests. In all instances, the coordinates of pick-up and drop-off nodes are randomly and independently chosen in the square $[-10, 10] \times [-10, 10]$ according to a uniform distribution, and the depot is located at the center of this square. For every arc $(v_i, v_j) \in A$, the routing cost c_{ij} and travel time t_{ij} are equal to the Euclidean distance between the two nodes.

A time window $[e_i, l_i]$ is also associated with each node. For an outbound user i, a time window was generated by first choosing a number l_{n+i} in the interval $[60, T]$ and then setting $e_{n+i} = l_{n+i} - 15$, where T denotes the length of the planning horizon. For an inbound user, the value of e_i was chosen in the interval $[0, T-60]$ and the value of l_i was set equal to $e_i + 15$. Time windows on the origin nodes of outbound requests and on the destination nodes of inbound requests are set as explained in §5.1.1.

In the first set of instances, $Q = 3$ for every vehicle, $q_i = 1$ and $d_i = 3$ for every user, and the maximum ride time L is equal to 30 minutes. In the second set, $Q = 6$ for every vehicle and the values of q_i are chosen randomly according to a uniform distribution on the set $\{1, \ldots, Q\}$. The maximum ride time is set equal to 45 minutes. In this case, we assume that service time is proportional to the number of passengers and $d_i = q_i$. The maximum route duration is equal to T. The first set of instances represents the context where cars are used for the transportation of individuals whereas the second set reflects the situation of a transporter using mini-busses for the transportation of individuals or groups of individuals.

All instances used in computational experiments are available on the website

http://www.hec.ca/chairedistributique/data/darp.

Tables 1 and 2 provide the characteristics of these instances, the number of constraints and variables in the resulting mixed-integer programming formulation, and the value of the LP relaxation. The corresponding values are computed with and without the application of the preprocessing techniques described in §5.1. In both cases, the problem reduction step of CPLEX is nevertheless performed. These results show that despite the strength of the CPLEX preprocessing routines, a considerable reduction in problem size (and improvement in the LP bound at the root node) is achieved through the application of the specialized arc elimination and variable-fixing techniques of §5.1. The improvement in the LP lower bound is 19.75% for the first set of instances and 15.17% for the second set. We also

Table 1. Characteristics of the first set of instances.

						Without reductions			With reductions			
Instance	$\|K\|$	n	T	Q	L	Cons	Vars	LP	Cons	Vars	LP	Upper bound
a2-16	2	16	480	3	30	1,289	1,211	219.60	893	665	269.54	294.25
a2-20	2	20	600	3	30	1,858	1,767	251.06	1,392	1,153	301.66	344.83
a2-24	2	24	720	3	30	2,714	2,585	316.38	1,997	1,628	364.22	431.12
a3-18	3	18	360	3	30	1,773	2,281	193.05	1,296	1,258	256.28	300.48
a3-24	3	24	480	3	30	2,898	3,880	221.63	2,244	2,446	260.56	344.83
a3-30	3	30	600	3	30	4,369	5,938	338.36	2,998	3,282	383.10	496.52
a3-36	3	36	720	3	30	6,077	8,329	360.84	3,919	4,171	484.57	600.75
a4-16	4	16	240	3	30	1,657	2,364	175.61	1,193	1,173	207.41	282.68
a4-24	4	24	360	3	30	3,279	5,181	235.68	2,190	2,616	293.07	375.07
a4-32	4	32	480	3	30	5,225	8,977	311.20	3,828	4,941	361.01	486.57
a4-40	4	40	600	3	30	8,104	13,829	366.04	5,535	8,200	408.13	571.07
a4-48	4	48	720	3	30	10,884	19,745	366.04	8,171	12,824	428.60	680.99
Avg.								279.62			334.85	434.10

Table 2. Characteristics of the second set of instances.

						Without reductions			With reductions			
Instance	$\|K\|$	n	T	Q	L	Cons	Vars	LP	Cons	Vars	LP	Upper bound
b2-16	2	16	480	6	45	1,030	993	218.80	834	653	250.55	309.61
b2-20	2	20	600	6	45	1,175	1,241	290.33	768	612	314.05	334.93
b2-24	2	24	720	6	45	1,922	1,873	311.71	1,436	1,189	372.80	445.11
b3-18	3	18	360	6	45	1,334	1,754	218.26	1,068	1,074	256.15	301.80
b3-24	3	24	480	6	45	2,230	2,915	284.78	1,625	1,669	329.14	394.57
b3-30	3	30	600	6	45	3,410	4,457	344.68	2,383	2,626	448.31	536.04
b3-36	3	36	720	6	45	4,200	6,035	437.37	2,984	3,403	493.79	611.79
b4-16	4	16	240	6	45	1,233	1,771	209.25	919	943	223.11	299.07
b4-24	4	24	360	6	45	2,426	4,129	252.87	1,850	2,262	312.28	380.27
b4-32	4	32	480	6	45	3,928	6,405	341.76	2,778	3,719	409.25	500.92
b4-40	4	40	600	6	45	5,361	9,481	477.90	3,797	5,368	525.28	662.91
b4-48	4	48	720	6	45	8,854	15,611	496.37	6,404	9,608	538.43	685.46
Avg.								323.67			372.76	455.21

indicate in the last columns of Tables 1 and 2 the value of an upper bound obtained by running the tabu search heuristic of Cordeau and Laporte (2003b) for 1,000 iterations.

In Tables 3 and 4, we indicate the lower bound obtained at the root node (after applying the problem reduction steps) under seven different scenarios. Column LP indicates the bound obtained without any kind of cut generation. The next four columns indicate the bound obtained by generating violated inequalities of one of the following families: capacity constraints (CC), generalized order constraints (GOC), infeasible path constraints (IPC) and subtour elimination constraint (SEC). Column CPX then reports the bound obtained when activating the automatic cut generation of CPLEX (but not the generation of user cuts). In this case, several types of inequalities (e.g., implied bound cuts, flow cuts, Gomory fractional cuts) are automatically generated by CPLEX at the root node. Finally, column Full indicates the bound obtained when applying both the automatic cut generation of CPLEX and the four separation heuristics of §5.3 (as well as the inequality pool of §5.2). The last column shows the improvement (in percentage) in the lower bound obtained when generating user cuts together with the CPLEX cuts instead of relying only on the latter class (i.e., relative difference between the last two scenarios).

These results show that using any of the four types of inequalities yields a small improvement in the average lower bound obtained. For both sets of instances, the largest improvement is obtained with the generation of infeasible path inequalities, while subtour elimination, capacity, and generalized order constraints have a limited impact. Similar improvements are also obtained by enabling the automatic cut generation of CPLEX. Nevertheless, combining these cuts with our own valid inequalities yields significant improvements. The average improvement exceeds 5% for both sets of instances. It is worth mentioning that similar conclusions are reached when comparing the best bound obtained after running the branch-and-cut algorithm for a certain number of nodes.

Finally, Tables 5 and 6 report the results obtained by using the branch-and-cut algorithm of CPLEX alone and with the concurrent generation of all families of inequalities. In both cases, a maximum of four hours of CPU time were allowed for the solution of each instance. We indicate the best bound obtained, the CPU time (in minutes) used,

Table 3. Initial lower bounds—first set of instances.

Instance	LP	CC	GOC	IPC	SEC	CPX	Full	Imp. (%)
a2-16	269.54	269.54	270.54	278.88	269.66	287.38	290.18	0.97
a2-20	301.66	302.21	304.30	310.73	308.51	321.70	326.13	1.38
a2-24	364.22	365.55	373.30	383.94	365.45	377.20	395.57	4.87
a3-18	256.28	256.28	258.71	260.08	262.43	259.94	269.15	3.54
a3-24	260.56	260.56	260.97	268.45	261.98	276.98	291.00	5.06
a3-30	383.10	386.94	383.26	409.99	384.31	386.62	421.60	9.05
a3-36	484.57	484.57	484.91	516.69	484.87	502.74	529.47	5.32
a4-16	207.41	209.00	207.41	214.45	209.13	212.48	229.19	7.86
a4-24	293.07	293.08	299.58	303.84	293.07	299.52	318.93	6.48
a4-32	361.01	361.01	361.39	368.13	361.24	375.18	390.02	3.96
a4-40	408.13	418.23	412.42	431.30	414.43	421.52	450.43	6.86
a4-48	428.60	431.33	428.61	437.82	429.57	432.09	457.25	5.82
Avg.	334.85	336.53	337.12	348.69	337.05	346.11	364.08	5.10

Table 4. Initial lower bounds—second set of instances.

Instance	LP	CC	GOC	IPC	SEC	CPX	Full	Imp. (%)
b2-16	250.55	250.55	264.13	252.67	251.35	268.08	289.66	8.05
b2-20	314.05	321.24	314.05	314.05	315.15	327.46	332.61	1.57
b2-24	372.80	377.07	380.54	377.29	380.02	389.81	404.80	3.85
b3-18	256.15	256.15	260.86	260.15	256.15	261.82	268.67	2.62
b3-24	329.14	329.14	331.34	338.37	329.38	332.98	344.85	3.56
b3-30	448.31	448.93	453.21	482.79	448.31	465.74	495.56	6.40
b3-36	493.79	495.04	506.92	502.21	493.79	520.59	547.83	5.23
b4-16	223.11	223.11	230.35	228.07	224.17	225.86	247.97	9.79
b4-24	312.28	312.28	315.36	320.69	312.28	320.03	336.40	5.12
b4-32	409.25	409.25	413.98	420.59	412.90	412.77	439.98	6.59
b4-40	525.28	525.28	531.44	544.31	525.28	531.17	561.36	5.68
b4-48	538.43	538.43	541.94	553.12	539.05	547.85	567.06	3.51
Avg.	372.76	373.87	378.68	382.86	373.99	383.68	403.06	5.16

and the number of nodes explored in the branch-and-bound tree. For the branch-and-cut algorithm with user cut generation, we also indicate the total number of cuts generated. An asterisk preceding a lower bound indicates that the instance was solved to optimality.

One can observe that the generation of user cuts significantly improves the performance of the branch-and-cut algorithm. For the 14 instances solved to optimality by both approaches within the time limit, the average CPU time was 25.93 minutes for CPLEX alone compared with 1.81 minutes for CPLEX with user cut generation. The average number of nodes explored went from 95,466 to 3,248. For the five instances that could not be solved optimally by any of the two approaches, the average lower bound reached by CPLEX alone was 500.91 compared with 517.48 for the branch-and-cut with user cut generation. Finally, the latter algorithm solved five more instances to optimality.

The small number of cuts generated with respect to the total number of nodes explored in the branch-and-bound tree is explained by the fact that the separation procedures are executed only when all y_i^k variables are integer, but at least one x_{ij}^k variable is fractional. For most instances, this occurs at approximately 5% of the nodes. Experiments performed with the generation of cuts at every node of the tree have on average produced slightly worse results because of the extra time spent by the separation procedures. It is likely that a larger number of cuts could be generated by using exact or more sophisticated heuristic separation procedures. The development of such procedures is by itself an important area of research that will be addressed in future work.

All instances used for testing are symmetric (i.e., $c_{ij} = c_{ji}$ for every pair of nodes i, j). In additional experiments, we have also tried solving asymmetric instances. Unlike the asymmetric TSP, which is usually easier to solve than its symmetric counterpart, it appears that the DARP is unaffected by the asymmetry of the cost matrix. This is explained by the fact that a while a TSP tour is feasible in both directions, thus leading to highly fractional solutions, a DARP route is feasible in only one direction because of precedence constraints.

The computational results presented in this section show that instances with up to 30 users can be solved optimally in reasonable CPU times. Using column generation to address the VRPPD with time windows, Savelsbergh and Sol (1998) were also able to consistently solve instances with up to 30 requests in short computation times. Experiments performed with the branch-and-cut algorithm on similar

Table 5. Comparisons with CPLEX—first set of instances.

	CPLEX B and C			B and C with user cuts			
Instance	Bound	CPU	Nodes	Bound	CPU	Nodes	Cuts
a2-16	*294.25	0.01	23	*294.25	0.01	4	46
a2-20	*344.83	0.05	313	*344.83	0.08	181	93
a2-24	*431.12	1.42	10,868	*431.12	0.27	444	140
a3-18	*300.48	0.41	3,596	*300.48	0.20	565	172
a3-24	*344.83	76.59	310,667	*344.83	4.17	6,863	324
a3-30	472.17	240.00	515,931	*494.85	42.39	43,632	425
a3-36	570.26	240.00	504,553	*583.19	12.46	7,451	423
a4-16	*282.68	21.49	145,680	*282.68	2.54	6,615	404
a4-24	359.52	240.00	442,000	*375.02	25.19	28,939	351
a4-32	427.65	240.00	189,900	447.66	240.00	105,500	806
a4-40	462.21	240.00	65,000	475.05	240.00	32,200	716
a4-48	466.70	240.00	40,400	486.03	240.00	21,400	1,019

Table 6. Comparisons with CPLEX—second set of instances.

	CPLEX B and C			B and C with user cuts			
Instance	Bound	CPU	Nodes	Bound	CPU	Nodes	Cuts
b2-16	*309.41	0.21	5,815	*309.41	0.15	487	130
b2-20	*332.64	0.01	26	*332.64	0.01	2	30
b2-24	*444.71	2.76	32,399	*444.71	0.13	146	103
b3-18	*301.64	1.29	12,223	*301.64	0.70	2,458	254
b3-24	*394.51	7.27	42,950	*394.51	3.62	8,147	252
b3-30	*531.44	189.74	574,281	*531.44	6.82	9,862	274
b3-36	588.44	240.00	447,474	*603.79	62.07	66,079	291
b4-16	*296.96	2.44	20,189	*296.96	0.79	2,579	196
b4-24	*371.41	59.31	175,495	*371.41	5.85	7,119	255
b4-32	460.78	240.00	222,600	*494.82	176.82	126,332	375
b4-40	570.37	240.00	153,400	591.76	240.00	81,100	632
b4-48	577.64	240.00	50,000	586.91	240.00	21,000	753

instances (obtained by relaxing the ride-time constraints) have, however, shown a deterioration of the algorithm's performance. This is partly explained by the fact that infeasible path inequalities as well as most preprocessing techniques then become irrelevant. In addition, column generation usually provides tighter formulations, leading to stronger LP relaxation values. Nevertheless, it seems that handling ride-time constraints in a column generation approach is far from trivial. Indeed, the states of the dynamic programming algorithm used for pricing would have to take into account the individual ride time of each user. Label elimination based on dominance is then likely to be very weak.

7. Conclusion

We have introduced several new valid inequalities and a branch-and-cut algorithm for the dial-a-ride problem. Although no polyhedral analysis was carried out, the usefulness of these inequalities has been demonstrated through computational experiments. A comparison with version 8.1 of CPLEX shows that the branch-and-cut algorithm proposed here reduces both the CPU time and the number of nodes explored in the branch-and-bound tree. The methodology proposed in this paper obviously cannot be used to solve large-scale instances containing hundreds or thousands of users, as is sometimes the case in large cities. It is, however, fast enough to be used as a postprocessor to optimize individual routes or small subsets of routes produced by a metaheuristic. Future work will concentrate on the development of more refined separation procedures for the various types of inequalities introduced in this paper.

Appendix

Details of the Proof of Proposition 3. There are two cases to distinguish.

Case 1. An arc (i_1, i_l) with $2 \leqslant l \leqslant m-1$ is part of the solution.

First, we show that there is at least one subset U_k with $1 \leqslant k \leqslant l-1$ for which $x(U_k) \leqslant |U_k| - 2$. Suppose this is not true. Then, for every k, there is a path in U_k connecting i_k and $n + i_{k+1}$. Because the arc (i_1, i_l) is in the solution, the path for U_1 must visit node $n + i_2$ before visiting node i_1. In addition, this path must appear immediately before the arc (i_1, i_l) in the solution. Now, for $k = 2, \ldots, l-1$, the path for U_k must appear before that for U_{k-1} in the solution to avoid violating the precedence constraint for user i_k. However, for $k = l-1$, this implies that node $n + i_l$ appears before node i_l, which is a contradiction.

Next, we show that there is at least one subset U_k with $l \leqslant k \leqslant m$ for which $x(U_k) \leqslant |U_k| - 2$. Suppose this is not true. Then, for every k, there is a path in U_k connecting i_k and $n + i_{k+1}$. Because the arc (i_1, i_l) is in the solution, the path for U_l must visit node i_l before visiting node $n + i_{l+1}$. In addition, this path must appear immediately after the arc (i_1, i_l) in the solution. Now, for $k = l+1, \ldots, m$, the path for U_k must appear before that for U_{k-1} in the solution to avoid violating the precedence constraint for user i_k. However, for $k = m$, this implies that node $n + i_1$ appears before node i_1, which is a contradiction.

Case 2. An arc $(i_1, n + i_l)$ with $3 \leqslant l \leqslant m$ is part of the solution.

First, we show that there is at least one subset U_k with $1 \leqslant k \leqslant l-1$ for which $x(U_k) \leqslant |U_k| - 2$. Suppose this is not true. Because the arc $(i_1, n + i_l)$ is in the solution, the path for U_1 must visit node $n + i_2$ before visiting node i_1. In addition, this path must appear immediately before the arc $(i_1, n + i_l)$ in the solution. Now, for $k = 2, \ldots, l-1$, the path for U_k must appear before that for U_{k-1} in the solution to avoid violating the precedence constraint for user i_k. However, for $k = l-1$, this implies that node $n + i_l$ appears before node i_1, which is a contradiction because the arc $(i_1, n + i_l)$ is part of the solution.

Next, we show that for there is at least one subset U_k with $l \leqslant k \leqslant m$ for which $x(U_k) \leqslant |U_k| - 2$. Suppose this is not true. Because the arc $(i_1, n + i_l)$ is in the solution, then for $k = l, \ldots, m$, the path for U_k must appear before the arc $(i_1, n + i_l)$ in the solution to avoid violating the precedence constraint for user i_k. However, for $k = m$, this implies that the path connecting i_m and $n + i_1$ appears before node i_1, which is a contradiction. □

DETAILS OF THE PROOF OF PROPOSITION 4. Again, there are two cases to distinguish.

Case 1. An arc $(n+i_1, i_l)$ with $2 \leqslant l \leqslant m-2$ is part of the solution.

For $k = l-1, \ldots, 1$, the path connecting i_k and $n+i_{k+1}$ must appear after the arc $(n+i_1, i_l)$ in the solution to avoid violating the precedence constraint for user i_{k+1}. However, for $k = 1$ this implies that the path connecting i_1 and $n+i_2$ appears after the arc $(n+i_1, i_l)$, which is a contradiction.

Because the arc $(n+i_1, i_l)$ is in the solution, the subpath for U_l must visit node i_l before visiting node $n+i_{l+1}$. For $k = l+1, \ldots, m$, the path for U_k must appear before that for U_{k-1} to avoid violating the precedence constraint for user i_k. However, for $k = m$, this implies that the path connecting i_m and $n+i_1$ appears before the path for at least one other subset U_h with $l+1 \leqslant h \leqslant m-1$, and thus node $n+i_1$ appears in two different places in the solution.

Case 2. An arc $(n+i_1, n+i_l)$ with $2 \leqslant l \leqslant m-1$ is part of the solution.

For $k = 1, \ldots, l-1$, the path connecting i_k and $n+i_{k+1}$ must appear after the arc $(n+i_1, n+i_l)$, but for $k = 1$, this implies that node i_1 appears after node $n+i_1$, which is a contradiction.

The path for U_l must appear before the arc $(n+i_1, n+i_l)$ and, for $k = l+1, \ldots, m-1$, the path for U_k must appear before that for U_{k-1}. However, for $k = m$, this implies that the path connecting i_m and $n+i_1$ appears before the path for at least one other subset U_h with $l \leqslant h \leqslant m-1$, and thus node $n+i_1$ appears in two different places in the solution. □

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant 227837-00. This support is gratefully acknowledged. Thanks are due to Luigi Moccia for comments on an earlier version of this paper. The author is also thankful to two anonymous referees.

References

Ascheuer, N. 1996. Hamiltonian path problems in the on-line optimization of flexible manufacturing systems. Ph.D. thesis, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany.

Ascheuer, N., M. Fischetti, M. Grötschel. 2001. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Math. Programming* **90** 475–506.

Ascheuer, N., M. Jünger, G. Reinelt. 2000. A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Comput. Optim. Appl.* **17** 61–84.

Augerat, P., J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef. 1999. Separating capacity inequalities in the CVRP using tabu search. *Eur. J. Oper. Res.* **106** 546–557.

Balas, E., M. Fischetti, W. R. Pulleyblank. 1995. The precedence-constrained asymmetric traveling salesman polytope. *Math. Programming* **68** 241–265.

Bodin, L. D., T. Sexton. 1986. The multi-vehicle subscriber dial-a-ride problem. *TIMS Stud. Management Sci.* **26** 73–86.

Cordeau, J.-F., G. Laporte. 2003a. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *4OR—Quart. J. Belgian, French Italian Oper. Res. Soc.* **1** 89–101.

Cordeau, J.-F., G. Laporte. 2003b. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Res. B* **37** 579–594.

Desaulniers, G., J. Desrosiers, A. Erdmann, M. M. Solomon, F. Soumis. 2002. VRP with pickup and delivery. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*. Philadelphia, PA, 225–242.

Desrochers, M., G. Laporte. 1991. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Oper. Res. Lett.* **10** 27–36.

Desrosiers, J., Y. Dumas, F. Soumis. 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *Amer. J. Math. Management Sci.* **6** 301–325.

Dumas, Y., J. Desrosiers, F. Soumis. 1989. Large scale multi-vehicle dial-a-ride problems. Technical Report G-89-30, GERAD, HEC Montréal, Montréal, Quebec, Canada.

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

Gendreau, M., G. Laporte, F. Semet. 1998. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks* **32** 263–273.

Grötschel, M., M. W. Padberg. 1985. Polyhedral theory. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, eds. *The Traveling Salesman Problem*. Wiley, New York, 251–305.

Jaw, J., A. R. Odoni, H. N. Psaraftis, N. H. M. Wilson. 1986. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. *Transportation Res. B* **20** 243–257.

Johnson, D. S. 1974. Approximation algorithms for combinatorial optimization problems. *J. Comput. System Sci.* **9** 256–278.

Madsen, O. B. G., H. F. Ravn, J. M. Rygaard. 1995. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Ann. Oper. Res.* **60** 193–208.

Miller, C. E., A. W. Tucker, R. A. Zemlin. 1960. Integer programming formulations and traveling salesman problems. *J. Association Comput. Machinery* **7** 326–329.

Naddef, D., G. Rinaldi. 2002. Branch-and-cut algorithms for the capacitated VRP. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*. Philadelphia, PA, 53–84.

Psaraftis, H. N. 1980. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Sci.* **14** 130–154.

Psaraftis, H. N. 1983. An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. *Transportation Sci.* **17** 351–357.

Ruland, K. S. 1995. Polyhedral solution to the pickup and delivery problem. Ph.D. thesis, Sever Institute of Technology, Washington University, St. Louis, MO.

Ruland, K. S., E. Y. Rodin. 1997. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Comput. Math. Appl.* **33** 1–13.

Savelsbergh, M. W. P. 1985. Local search in routing problems with time windows. *Ann. Oper. Res.* **4** 285–305.

Savelsbergh, M. W. P. 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA J. Comput.* **4** 146–154.

Savelsbergh, M. W. P., M. Sol. 1998. Drive: Dynamic routing of independent vehicles. *Oper. Res.* **46** 474–490.

Toth, P., D. Vigo. 1996. Fast local search algorithms for the handicapped persons transportation problem. I. H. Osman, J. P. Kelly, eds. *Meta-Heuristics: Theory and Applications*. Kluwer, Boston, MA, 677–690.

Toth, P., D. Vigo. 1997. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Sci.* **31** 60–71.