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

JEAN-FRANÇOIS CORDEAU

Canada Research Chair in Distribution Management, HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal, Canada H3T 2A7

September 5, 2003

Abstract

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 pickup and delivery and the vehicle routing 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.

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 pre-specified 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 60 minutes, a user wishing to arrive at destination at 9h00 would be picked-up no earlier than 7h45 and dropped-off between 8h45 and 9h00. It should be emphasized that imposing time windows is not sufficient to accurately impose a maximum ride time. In the above example, a pick-up at 7h45 and a drop-off at 9h would exceed the maximum ride time of 60 minutes. However, constraining the pick-up to take place no earlier than 8h would be too restrictive because it would, in particular, prohibit a pick-up at 7h45 and a drop-off at 8h45.

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 since 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 Pick-up and Delivery (VRPPD). Finding a feasible solution for the DARP is itself NP-hard since 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. This algorithm uses adaptations of known valid inequalities for the VRP and VRPPD as well as new inequalities that take advantage of the special structure of the problem.

The remainder of the article 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 Section 5, along with preprocessing techniques. Computational experiments are reported in Section 6 and the conclusion follows in Section 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. Ruland (1995) and Ruland and Rodin (1997) proposed a branch-and-cut algorithm for a special case of the DARP 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 Section 4.

Most of the algorithms for the multiple-vehicle case are heuristics or meta-heuristics. 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.

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).

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 $N = P \cup D \cup \{0, 2n+1\}$, $P = \{1, \dots, n\}$ and $D = \{n+1, \dots, 2n\}$. 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 non-negative service duration d_i such that $q_0 = q_{2n+1} = 0$, $q_i = -q_{n+i}$ $(i = 1, \dots, n)$ 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.

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 formulation is as follows:

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

subject to

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

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

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

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

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

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

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

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

$$B_{2n+1}^k - B_0^k \leq T_k \qquad \forall k \in K \tag{10}$$

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

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

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

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

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. Consistence 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 non-negativity 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 non-linear because of constraints (7) and (8). Introducing constants M_{ij}^k and W_{ij}^k, these constraints can, however, be linearized as follows:

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

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

The validity of these constraints is ensured by setting $M_{ij}^k \geq \max\{0, l_i + d_i + t_{ij} - e_j\}$ and $W_{ij}^k \geq \min\{Q_k, Q_k + q_i\}$. 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

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

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

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

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

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

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

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

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

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:

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

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., $B_i > e_i$) in order 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 $\delta(S) = \delta^+(S) \cup \delta^-(S)$ where $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$. 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:

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

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

These inequalities were used, for example, for solving the Asymmetric Travelling Salesman Problem with Time Windows by branch-and-cut (Ascheuer et al., 2001).

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

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

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

4.2 Subtour elimination constraints

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

Proposition 1. Let $S \subseteq P \cup D$ and $P^+(S) = \{i | i \in S \cap P \text{ and } n+i \notin S\}$. The following inequality is valid for the DARP:

$$x(S) + \sum_{i \in P^+(S)} \sum_{j \in S} x_{n+i,j} \leq |S| - 1. \tag{29}$$

Proof. Because of precedence relationships, set S must be entered at least once before using any of the lifted arcs. As a result, $x(\delta^-(S)) \geq 1 + \sum_{i\in P^+(S)} \sum_{j\in S} x_{n+i,j}$. In addition, $x(\delta^+(S)) = x(\delta^-(S))$ implies that $x(\delta^+(S)) \geq 1 + \sum_{i\in P^+(S)} \sum_{j\in S} x_{n+i,j}$. Since $2x(S) + x(\delta^+(S)) + x(\delta^-(S)) = 2|S|$, one obtains $2x(S) + 2 + 2\sum_{i\in P^+(S)} \sum_{j\in S} x_{n+i,j} \leq 2|S|$ and, finally, $x(S) + \sum_{i\in P^+(S)} \sum_{j\in S} x_{n+i,j} \leq |S| - 1$. $\square$

Example. Consider the node set $S = \{i, j\} \subseteq P$. The resulting lifted subtour elimination constraint is $x_{ij} + x_{ji} + x_{n+i,j} + x_{n+j,i} \leq 1$. This example is illustrated in Figure 1. Observe that variables $x_{n+i,i}$ and $x_{n+j,j}$ need not be introduced in the lifted inequality since the corresponding arcs can be trivially removed from the graph.

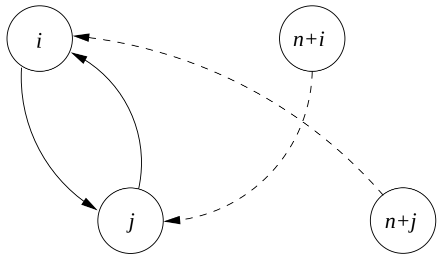

Figure 1: Lifted subtour elimination constraint for $S = \{i, j\} \subseteq P$

Proposition 2. Let $S \subseteq P \cup D$ and $D^-(S) = \{i | n + i \in S \cap D \text{ and } i \notin S\}$. The following inequality is valid for the DARP:

$$x(S) + \sum_{i\in S} \sum_{j\in D^-(S)} x_{ij} \leq |S| - 1. \tag{30}$$

Proof. The proof is similar to that of Proposition 1 by observing that because of precedence relationships, set S must be exited at least once after using any of the lifted arcs and thus $x(\delta^+(S)) \geq 1 + \sum_{i\in S} \sum_{j\in D^-(S)} x_{ij}$. $\square$

Example. Consider the node set $S = \{n + i, n + j\} \subseteq D$. The resulting lifted subtour elimination constraint is $x_{n+i,n+j} + x_{n+j,n+i} + x_{n+i,j} + x_{n+j,i} \leq 1$. This is illustrated in Figure 2.

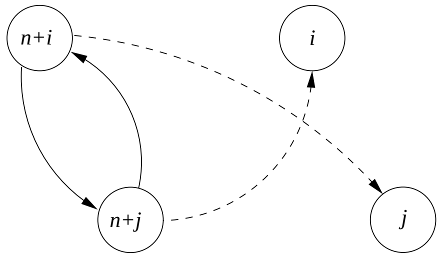

Figure 2: Lifted subtour elimination constraint for $S = \{n + i, n + j\} \subseteq D$

In the case of a directed formulation, one can also lift subtour elimination constraints by taking into account the orientation of the arcs. For a set $S = \{i_1, i_2, \ldots, i_h\} \subseteq N$ with $h \geq 3$ nodes, Grötschel and Padberg (1995) proposed the following inequalities for the asymmetric TSP:

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

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

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 3. Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq P \cup D$ and $P^+(S) = \{i | i \in S \cap P \text{ and } n+i \notin S\}$. The following inequality is valid for the DARP:

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

Proof. Suppose that one arc of the form $(n+i_p, i_1)$ with $i_p \in P^+(S)$ is part of the solution. Then all arcs of the form (i_j, i_1) with $2 \leq j \leq h-1$ 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. But 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 $x_{ij} + x_{jk} + x_{ki} + 2x_{ji} + x_{n+j,i} + x_{n+k,i} \leq 2$. This is illustrated in Figure 3.

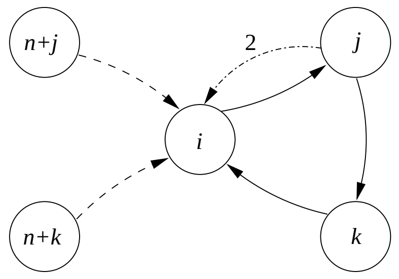

Figure 3: Lifted directed subtour elimination constraint for $S = \{i, j, k\} \subseteq P$

Proposition 4. Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq P \cup D$ and $D^-(S) = \{i | n+i \in S \cap D \text{ and } i \notin S\}$. The following inequality is valid for the DARP:

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=3}^{h} x_{i_1,i_j} + \sum_{j=4}^{h}\sum_{l=3}^{j-1} x_{i_j,i_l} + \sum_{i_p \in D^-(S)} x_{i_1,i_p} \leq h-1. \tag{34}$$

Proof. The proof is similar to that of Proposition 3 by observing that if one arc of the form (i_1, i_p) with $i_p \in D^-(S)$ is part of the solution, then all arcs of the form (i_1, i_j) with $3 \leq j \leq h$ cannot belong to the solution. $\Box$

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 $x_{n+i,n+j} + x_{n+j,n+k} + x_{n+k,n+i} + 2x_{n+i,n+k} + x_{n+i,j} + x_{n+i,k} \leq 2$. This is illustrated in Figure 4.

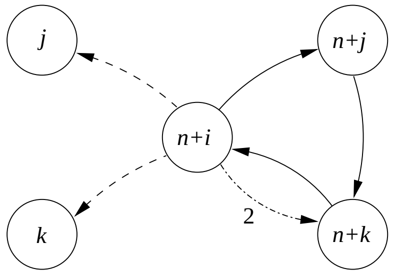

Figure 4: Lifted directed subtour elimination constraint for $S = \{n+i, n+j, n+k\} \subseteq D$

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 $x(\delta(S)) \geq 2R(S)$ 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 $n+i$, $x(S) \leq |S| - 2$ and, equivalently, $x(\delta(S)) \geq 4$. 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 pickup 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:

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

In the directed case, these inequalities can be lifted in two different ways, as shown by the following propositions.

Proposition 5. The following inequality is valid for the DARP:

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

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

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 $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{ij} + x_{i,n+k} \leq 2$. This is illustrated in Figure 5.

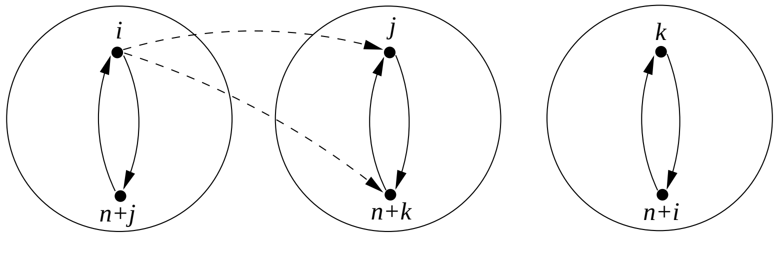

Figure 5: Lifted generalized order constraint with $m = 3$ (first lifting)

Proposition 6. The following inequality is valid for the DARP:

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

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 $x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{n+i,n+j} \leq 2$. 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.

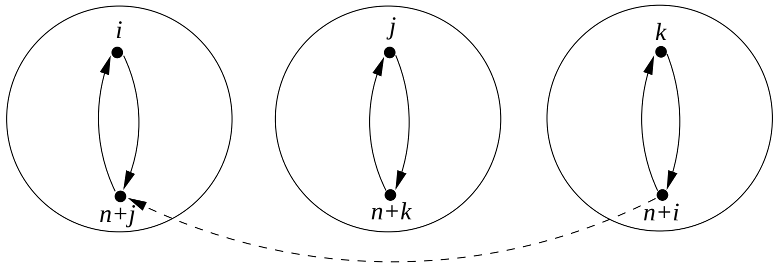

Figure 6: Lifted generalized order constraint with $m = 3$ (second lifting)

4.6 Order matching constraints

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

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

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 7. 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:

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

Proof. First observe that $x(H) \leq |H| - 1$ and $x(T_h) \leq |T_h| - 1$ 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, $x(\delta^+(H)) \geq \alpha$. Since $x(\delta^+(H)) = x(\delta^-(H))$ and $2x(H) + x(\delta^+(H)) + x(\delta^-(H)) = 2|H|$, one obtains that $2x(H) \leq 2|H| - 2\alpha$ and thus $x(H) \leq |H| - \alpha$. Finally, because $x(T_h) \leq |T_h| - 2$ for the remaining $m - \alpha$ sets, one may conclude that $x(H) + \sum_{h=1}^{m} x(T_h) \leq |H| - \alpha + \sum_{h=1}^{m}(|T_h| - 1) - (m - \alpha)$, which simplifies to expression (39). $\square$

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 $x(H)+x(T_1)+x(T_2) \leq 4$ and is illustrated in Figure 7.

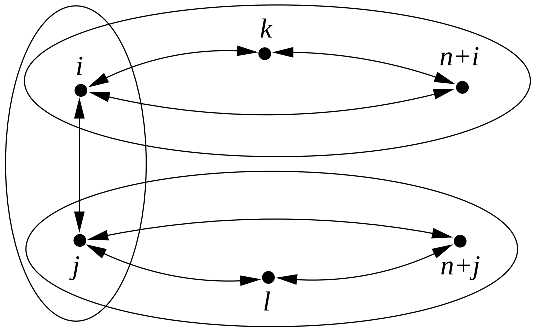

Figure 7: Generalized order matching constraint with $m = 2$

Remark 2. Inequality (39) is stronger than the corresponding TSP comb constraint, defined for $m \geq 3$ and odd, and which is obtained from (39) by replacing the right-hand side with $|H| + \sum_{h=1}^{m} |T_h| - (3m+1)/2$.

4.7 Infeasible path inequalities

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 8. 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:

$$x_{i,k_1} + \sum_{h=1}^{p-1} x_{k_h,k_{h+1}} + x_{k_p,n+i} \leq p - 1. \tag{40}$$

Proof. Suppose that the value of the left-hand side of inequality (40) 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. Since 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. But if 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. $\square$

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; 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 Section 5.1, 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 Section 5.2.

To control the branch-and-cut process, additional variables $y_i^k = \sum_{j \in N} x_{ij}^k$ are introduced in the formulation. At each 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. 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.

Prior to the application of the branch-and-cut algorithm, 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.

Finally, an initial set of valid inequalities is added prior to solving the initial LP relaxation: all bounds on time and load variables, lifted 2-node subtour elimination constraints, generalized order constraints with $m = 2$ and $|U_l| = 2$, and precedence constraints with $|S| = 3$ are introduced in a pool of cuts whose violations are checked at each node of the branch-and-bound tree, including those where not all y variables take integer values.

5.1 Preprocessing

This section describes the time window tightening, arc elimination and variable fixing steps that can be applied prior to 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 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 windows and ride time constraints. Incompatible user pairs $\{i, j\}$ can be identified by checking the feasibility of the following paths: $\{i, j, n+i, n+j\}$, $\{i, j, n+j, n+i\}$, $\{j, i, n+i, n+j\}$, $\{j, i, n+j, n+i\}$, $\{i, n+i, j, n+j\}$, $\{j, n+j, i, n+i\}$.

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:

$$F_i = \min_{i \le j \le q} \left\{ \sum_{i < p \le j} W_p + \min \{ l_j - B_j, L - P_j \} \right\}, \tag{41}$$

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 $n+1 \le i \le 2n$, 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 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 Separation Heuristics

This section describes the separation heuristics used to identify violated inequalities. When all y_i^k variables are integer but there is at least one fractional x_{ij}^k variable, the following heuristics are executed sequentially.

5.2.1 Subtour elimination constraints

The identification of violated inequalities of the form $x(S) \le |S| - 1$ can be achieved by solving a series of maximum flow problems between any node i and all other nodes $j \in N \setminus \{i\}$. 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

$$x(\delta(S)) - 2 \sum_{i \in P^+(S)} \sum_{j \in S} x_{n+i,j} < 2. \tag{42}$$

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 (42). 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 (100 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.

5.2.2 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 $q(S) > Q$. Again, the heuristic runs for 100 iterations and multiple violated inequalities may be identified during a single execution.

5.2.3 Precedence constraints

As explained by Ruland (1995), identifying violated precedence constraints can be done by solving a multi-terminal maximum flow problem for each user. To check whether the precedence constraint for user i is violated, one can compute the maximum flow from the sources 0 and $n+i$ to the sinks i and $2n+1$. If the value of this flow is less than 2, then a precedence constraint $x(S) \leq |S| - 2$ is violated for a set S such that $0, n+i \in S$ and $i, 2n+1 \notin S$. The set S corresponds to one of the shores of the corresponding minimum cut. We have implemented this approach and use the maximum flow algorithm provided in the GTL library (see `http://infosun.fmi.uni-passau.de/GTL`).

5.2.4 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).

5.2.5 Order matching constraints

To identify violated order matching constraints, Ruland (1995) has proposed an algorithm requiring the solution of $O(|N|^2)$ maximum flow problems.

We again resort to a simpler heuristic that aims to identify violations of (39) for the special case where $m = 2$, $|H| = 3$ and $|T_1| = |T_2| = 3$. For each pair of users $i, j \in P$, we first form the subsets $T_1 = \{i, n+i\}$ and $T_2 = \{j, n+j\}$. We then select the nodes k_1 and k_2 such that $x(T_1)$ and $x(T_2)$ are maximized. Finally, we select the node k such $x(H)$ is maximized and check for a violation of the resulting inequality.

5.2.6 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 $\mathcal{P}_i = \{i, k_1, k_2, \dots\}$ 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 (40).

6 Computational Experiments

The branch-and-cut algorithm just described 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 512Mb of memory.

The algorithm was applied to two sets of randomly generated instances comprising up to 32 users. In an instance with n users, where n is even, users $1, \dots, n/2$ are assumed to formulate outbound requests while users $n/2+1, \dots, n$ 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 to each node. For an outbound user i, a time window was generated by first choosing a number e_{n+i} in the interval $[420, 1080]$ (i.e., between 7h00 and 18h00) and then setting $l_{n+i} = e_{n+i} + 15$. For an inbound user, the value of e_i was chosen in the interval $[360, 1020]$ 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 Section 5.1.1.

In the first set of instances, $Q = 3$ for every vehicle, $q_i = 1$ 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 60 minutes. In all cases, we assume that service time is proportional to the number of passengers and $d_i = q_i$. The maximum route duration is set to 720 minutes. 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.

Tables 1 and 2 provide the characteristics of these instances, the number of constraints and variables in the resulting integer programming formulation, the value of the LP relaxation and the cost of a heuristic solution. This solution was obtained by running the tabu search algorithm of Cordeau and Laporte (2003b) for 1000 iterations.

In Tables 3 and 4 we indicate the best lower bounds reached by running the branch-and-cut algorithm on each instance with a maximum CPU time limit of 60 minutes. Column CPLEX indicates the bound obtained by running CPLEX itself with no user-cut generation but after the application of the preprocessing steps of Section 5.1. The next columns report the bounds obtained by generating the lifted bounds on time and load variables (Section 4.1) with only one of the following: subtour eliminations constraints (SEC), capacity constraints (CC), generalized order constraints (GOC), order matching constraints (OMC) and infeasible path constraints (IPC). Average values reported on the last line of each table show that the strength of the various inequalities varies from one set of instances to the other. Furthermore, for the first set of instances, using any kind of inequality alone did not yield significantly

Table 1: Characteristics of the first set of instances

Instance	m	n	Q	L	Cons	Vars	LP	Heuristic
p2-16	2	16	3	30	855	733	270.25	298.00
p2-20	2	20	3	30	1134	1050	300.55	345.14
p2-24	2	24	3	30	1895	1679	304.10	375.15
p2-28	2	28	3	30	2610	2344	337.60	417.12
p2-32	2	32	3	30	3536	3098	350.15	466.53
p3-16	3	16	3	30	920	1020	197.39	258.10
p3-20	3	20	3	30	1520	1776	238.88	282.81
p3-24	3	24	3	30	1957	2311	312.90	374.18
p3-28	3	28	3	30	2609	3059	368.43	478.58
p3-32	3	32	3	30	3676	4593	291.22	423.08
p4-16	4	16	3	30	960	1299	229.12	294.82
p4-20	4	20	3	30	1502	2104	256.25	343.37
p4-24	4	24	3	30	2171	3188	254.20	334.41
p4-28	4	28	3	30	2911	4479	267.28	395.90
p4-32	4	32	3	30	3575	5564	376.65	491.82

Table 2: Characteristics of the second set of instances

Instance	m	n	Q	L	Cons	Vars	LP	Heuristic
g2-16	2	16	6	60	666	600	223.14	269.98
g2-20	2	20	6	60	1232	1176	280.41	374.77
g2-24	2	24	6	60	1379	9379	307.64	429.90
g2-28	2	28	6	60	2448	2163	336.35	427.06
g2-32	2	32	6	60	2747	2395	400.81	544.95
g3-16	3	16	6	60	782	917	207.28	256.70
g3-20	3	20	6	60	1200	1459	251.37	301.41
g3-24	3	24	6	60	1796	2045	319.89	449.88
g3-28	3	28	6	60	2401	2807	325.37	451.09
g3-32	3	32	6	60	2625	3293	391.38	562.73
g4-16	4	16	6	60	875	1220	217.09	273.60
g4-20	4	20	6	60	1146	1642	257.69	363.75
g4-24	4	24	6	60	1753	2639	300.70	423.10
g4-28	4	28	6	60	2484	3399	261.27	363.70
g4-32	4	32	6	60	2880	4221	351.21	516.00

better bounds than using CPLEX directly, except for subtour elimination constraints. This is explained by the fact that the additional CPU time used to identify violated inequalities allows for the exploration of a reduced number of nodes in the search tree. We do not report the results for the generation of precedence constraints since these were significantly worse than the direct CPLEX implementation.

Table 3: Best lower bounds - first set of instances

Instance	CPLEX	SEC	CC	GOC	OMC	IPC
p2-16	298.00	298.00	298.00	298.00	298.00	298.00
p2-20	345.14	345.14	345.14	245.14	345.14	345.14
p2-24	375.15	375.15	375.15	375.15	375.15	375.15
p2-28	403.92	417.05	402.75	417.05	410.24	405.66
p2-32	426.62	441.46	426.31	427.54	427.73	430.53
p3-16	258.10	258.10	258.10	258.10	258.10	258.10
p3-20	282.81	282.81	282.81	282.81	282.81	282.81
p3-24	374.07	374.07	374.07	374.07	374.07	374.07
p3-28	440.69	448.35	433.47	442.73	432.49	442.31
p3-32	343.03	337.49	333.87	333.60	335.11	331.88
p4-16	294.82	294.82	294.82	294.82	294.82	294.82
p4-20	318.89	325.78	316.60	320.27	317.79	316.64
p4-24	291.53	292.54	293.02	295.12	291.07	294.02
p4-28	322.17	320.34	317.51	318.47	320.13	318.99
p4-32	399.74	401.34	401.76	400.73	401.92	402.00
Avg.	344.98	347.50	343.56	338.91	344.30	344.67

Table 4: Best lower bounds - second set of instances

Instance	CPLEX	SEC	CC	GOC	OMC	IPC
g2-16	268.36	268.36	268.36	268.36	268.36	268.36
g2-20	370.77	370.77	370.77	370.77	370.77	370.77
g2-24	397.60	429.27	426.05	429.27	415.52	413.46
g2-28	409.30	418.95	424.27	416.53	418.02	417.29
g2-32	502.84	528.20	523.01	516.85	508.62	510.84
g3-16	255.18	255.18	255.18	255.18	255.18	255.18
g3-20	300.45	300.45	300.45	300.45	300.45	300.45
g3-24	382.67	402.90	411.80	399.11	395.84	393.96
g3-28	375.30	383.36	382.60	381.95	377.08	382.27
g3-32	443.54	453.94	462.18	456.98	452.35	451.84
g4-16	270.24	270.24	270.24	270.24	270.24	270.24
g4-20	346.42	361.56	361.56	361.35	361.56	361.56
g4-24	349.38	360.96	359.82	357.07	358.03	356.16
g4-28	285.90	293.76	295.17	295.42	291.11	294.37
g4-32	390.11	405.24	414.47	411.68	406.33	405.04
Avg.	356.54	366.88	368.40	366.08	363.30	363.45

Combining the different types of inequalities does, however, yield important improvements over the basic version of CPLEX. Tables 5 and 6 reports the results obtained by using the basic version of CPLEX and by using the five types of inequalities concurrently. In both cases, a maximum of 12 hours of CPU time was allowed for the solution of each instance. We

Table 5: Comparisons with CPLEX - first set of instances

	CPLEX			Branch-and-cut			
Instance	Bound	CPU	Nodes	Bound	CPU	Nodes	Cuts
p2-16	*298.00	0.01	54	*298.00	0.03	36	27
p2-20	*345.14	0.11	628	*345.14	0.12	160	77
p2-24	*375.15	5.96	21,844	*375.15	3.12	5,368	203
p2-28	*417.05	188.57	472,114	*417.05	8.59	12,516	359
p2-32	448.70	720.00	909,500	*466.53	142.51	148,501	545
p3-16	*258.10	0.05	288	*258.10	0.07	218	36
p3-20	*282.81	1.37	5,118	*282.81	1.33	3,420	85
p3-24	*374.07	2.04	6,042	*374.07	1.12	1,785	108
p3-28	*462.80	379.36	802,594	*462.80	153.56	240,402	269
p3-32	354.45	720.00	446,400	354.71	720.00	377,030	802
p4-16	*294.82	3.45	22,233	*294.82	1.35	5,764	74
p4-20	*341.93	524.59	1,403,353	*341.93	308.97	765,146	181
p4-24	309.73	720.00	711,000	*332.96	685.54	451,915	1203
p4-28	337.34	720.00	584,900	341.00	720.00	433,158	390
p4-32	412.11	720.00	263,100	418.71	720.00	267,975	471

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

One can observe that the branch-and-cut algorithm is significantly faster than using CPLEX alone. For the 19 instances solved to optimality by both approaches within the time limit, the average CPU time was 118.60 minutes for CPLEX compared with 33.02 minutes for the branch-and-cut algorithm. The average number of nodes explored went down from 351,623 to 61,584. For the eight instances that could not be solved optimally by any of the two approaches, the average lower bound reached by CPLEX was 377.12 compared with 393.36 for the branch-and-cut algorithm. Finally, the branch-and-cut algorithm solved three more instances to optimality.

The small number of cuts generated with respect to the 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. 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 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.

Table 6: Comparisons with CPLEX - second set of instances

	CPLEX			Branch-and-cut			
Instance	Bound	CPU	Nodes	Bound	CPU	Nodes	Cuts
g2-16	*268.36	0.05	483	*268.36	0.08	200	118
g2-20	*370.77	6.95	50,483	*370.77	4.57	6,170	880
g2-24	*429.27	157.16	777,836	*429.27	27.18	36,220	871
g2-28	*424.27	215.48	557,624	*424.27	86.65	40,925	1973
g2-32	*544.86	577.92	1,777,740	*544.86	18.48	20,775	727
g3-16	*255.18	3.26	35,456	*255.18	1.07	4,512	156
g3-20	*300.45	31.70	133,174	*300.45	2.22	4,464	243
g3-24	408.43	720.00	1,356,000	*449.11	205.08	251,378	437
g3-28	390.48	720.00	839,900	413.00	720.00	344,902	1936
g3-32	458.96	720.00	505,300	487.33	720.00	260,305	1853
g4-16	*270.24	28.04	167,427	*270.24	3.64	12,014	167
g4-20	*361.56	127.42	446,344	*361,56	5.21	9,996	145
g4-24	366.52	720.00	929,700	386.89	720.00	652,610	379
g4-28	296.19	720.00	513,100	310.66	720.00	315,316	935
g4-32	400.94	720.00	408,600	434.57	720.00	201,539	1212

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 it is sometimes the case in large cities. It is, however, fast enough to be used as a post-processor to optimize individual routes or small subsets of routes produced by a meta-heuristic. In the presence of ride time constraints, using dynamic programming would be difficult since the states of the dynamic programming algorithm would have to take into account the individual ride time of each user. 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 5.

There are two cases to distinguish.

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

First, we show that there is at least one subset U_k with $1 \leq k \leq l-1$ for which $x(U_k) \leq |U_k|-2$. Suppose this is not true. Then, for every k, there is a path in U_k connecting i_k and $n+i_{k+1}$. Since 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 $k = 2, \dots, l-1$, 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. But 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 $l \leq k \leq m$ for which $x(U_k) \leq |U_k|-2$. Suppose this is not true. Then, for every k, there is a path in U_k connecting i_k and $n+i_{k+1}$. Since 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 $k = l+1, \dots, m$, 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. But 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 $3 \leq l \leq m$ is part of the solution.

First, we show that there is at least one subset U_k with $1 \leq k \leq l-1$ for which $x(U_k) \leq |U_k| - 2$. Suppose this is not true. Since 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. But for $k = l - 1$, this implies that node $n + i_l$ appears before node i_1, which is a contradiction since 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 $l \leq k \leq m$ for which $x(U_k) \leq |U_k| - 2$. Suppose this is not true. Since 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. But for $k = m$, this implies that the path connecting i_m and $n + i_1$ appears before node i_1, which is a contradiction. $\square$

Details of the proof of Proposition 6.

Again, there are two cases to distinguish.

Case 1. An arc $(n + i_1, i_l)$ with $2 \leq l \leq m - 2$ 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}. But 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.

Since 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. But 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 $l + 1 \leq h \leq m - 1$, 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 $2 \leq l \leq m - 1$ 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}. But 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 $l \leq h \leq m - 1$, and thus node $n + i_1$ appears in two different places in the solution. $\square$

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant 227837-00. This support is gratefully acknowledged.

References

N. Ascheuer, M. Fischetti and M. Grötschel. "Solving the Asymmetric Travelling Salesman Problem with Time Windows by Branch-and-Cut." *Mathematical Programming*, **90**:475–506 (2001).

P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán and D. Naddef. "Separating Capacity Inequalities in the CVRP Using Tabu Search." *European Journal of Operational Research*, **106**:546–557 (1999).

L.D. Bodin and T. Sexton. "The Multi-Vehicle Subscriber Dial-a-Ride Problem." *TIMS Studies in Management Science*, **26**:73–86 (1986).

J.-F. Cordeau and G. Laporte. "The Dial-a-Ride Problem (DARP): Variants, Modeling Issues and Algorithms." *4OR - Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, **1**:89–101 (2003a).

J.-F. Cordeau and G. Laporte. "A Tabu Search Heuristic for the Static Multi-Vehicle Dial-a-Ride Problem." *Transportation Research B*, **37**:579–594 (2003b).

G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon and F. Soumis. "VRP with Pickup and Delivery." In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 225–242. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

M. Desrochers and G. Laporte. "Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints." *Operations Research Letters*, **10**:27–36 (1991).

J. Desrosiers, Y. Dumas and F. Soumis. "A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows." *American Journal of Mathematical and Management Sciences*, **6**:301–325 (1986).

Y. Dumas, J. Desrosiers and F. Soumis. "Large Scale Multi-Vehicle Dial-a-Ride Problems." GERAD Technical Report G-89-30, HEC Montréal, 1989.

Y. Dumas, J. Desrosiers and F. Soumis. "The Pickup and Delivery Problem with Time Windows." *European Journal of Operational Research*, **54**:7–22 (1991).

M. Gendreau, G. Laporte and F. Semet. "A Branch-and-Cut Algorithm for the Undirected Selective Traveling Salesman Problem." *Networks*, **32**:263–273 (1998).

M. Grötschel and M.W. Padberg. "Polyhedral Theory." In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, editors, *The Traveling Salesman Problem*, pages 251–305. Wiley, New York, 1995.

J. Jaw, A.R. Odoni, H.N. Psaraftis and N.H.M. Wilson. "A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-a-Ride Problem with Time Windows." *Transportation Research B*, **20**:243–257 (1986).

D.S. Johnson. "Approximation Algorithms for Combinatorial Optimization Problems." *Journal of Computer and System Sciences*, **9**:256–278 (1974).

O.B.G. Madsen, H.F. Ravn and J.M. Rygaard. "A Heuristic Algorithm for a Dial-a-Ride Problem with Time Windows, Multiple Capacities, and Multiple Objectives." *Annals of Operations Research*, **60**:193–208 (1995).

C.E. Miller, A.W. Tucker and R.A. Zemlin. "Integer Programming Formulations and Traveling Salesman Problems." *Journal of the Association for Computing Machinery*, **7**:326–329 (1960).

D. Naddef and G. Rinaldi. "Branch-and-Cut Algorithms for the Capacitated VRP." In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 53–84. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

H.N. Psaraftis. "A Dynamic Programming Approach to the Single-Vehicle, Many-to-Many Immediate Request Dial-a-Ride Problem." *Transportation Science*, **14**:130–154 (1980).

H.N. Psaraftis. "An Exact Algorithm for the Single-Vehicle Many-to-Many Dial-a-Ride Problem with Time Windows." *Transportation Science*, **17**:351–357 (1983).

K.S. Ruland. *Polyhedral Solution to the Pickup and Delivery Problem*. Ph.D. thesis, Sever Institute of Technology, Washington University, 1995.

K.S. Ruland and E.Y. Rodin. "The Pickup and Delivery Problem: Faces and Branch-and-Cut Algorithm." *Computers and Mathematics with Applications*, **33**:1–13 (1997).

M.W.P. Savelsbergh. "Local Search in Routing Problems with Time Windows." *Annals of Operations Research*, **4**:285–305 (1985).

M.W.P. Savelsbergh. "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration." *ORSA Journal on Computing*, **4**:146–154 (1992).

M.W.P. Savelsbergh and M. Sol. "Drive: Dynamic Routing of Independant Vehicles." *Operations Research*, **46**:474–490 (1998).

P. Toth and D. Vigo. "Fast Local Search Algorithms for the Handicapped Persons Transportation Problem." In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 677–690. Kluwer, Boston, 1996.

P. Toth and D. Vigo. "Heuristic Algorithms for the Handicapped Persons Transportation Problem." *Transportation Science*, **31**:60–71 (1997).