Models and Branch-and-Cut Algorithms for Pickup and Delivery Problems with Time Windows

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

Jean-François Cordeau
Canada Research Chair in Logistics and Transportation, HEC Montréal 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

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

In the pickup and delivery problem with time windows (PDPTW), capacitated vehicles must be routed to satisfy a set of transportation requests between given origins and destinations. In addition to capacity and time window constraints, vehicle routes must also satisfy pairing and precedence constraints on pickups and deliveries. This paper introduces two new formulations for the PDPTW and the closely related dial-a-ride problem (DARP) in which a limit is imposed on the elapsed time between the pickup and the delivery of a request. Several families of valid inequalities are introduced to strengthen these two formulations. These inequalities are used within branch-and-cut algorithms which have been tested on several instance sets for both the PDPTW and the DARP. Instances with up to eight vehicles and 96 requests (194 nodes) have been solved to optimality. © 2007 Wiley Periodicals, Inc. NETWORKS, Vol. 49(4), 258–272 2007

Keywords: pickup and delivery; time windows; valid inequalities; branch-and-cut

1. INTRODUCTION

In the *Pickup and Delivery Problem* (PDP), capacitated vehicles must be routed to satisfy a set of transportation requests between given origins and destinations. Each route must start and finish at a common depot and satisfy pairing and precedence constraints: for each request, the origin must precede the destination, and both locations must be visited by the same vehicle. The PDP arises naturally in several contexts such as urban courier services and door-to-door transportation systems for the elderly and the disabled. In most applications, time windows restrict the time at which each pickup and delivery location may be visited by a vehicle. This gives rise to the *PDP with Time Windows* (PDPTW). In the case of passenger transportation, additional constraints may also be present to reduce customer dissatisfaction. In particular, ride time constraints are often imposed to limit the time spent by a passenger in the vehicle. The resulting problem is called the *Dial-a-Ride Problem* (DARP).

Both the PDP and PDPTW are generalizations of the classical *Vehicle Routing Problem* (VRP) and are thus $\mathcal{NP}$-hard. As a result, the development of solution methods for these problems has focused on heuristics [8, 10]. Nevertheless, when the problem is sufficiently constrained, it is possible to obtain optimal solutions within reasonable computation time. For instance, dynamic programming has been used successfully to solve the single-vehicle PDP with or without time windows [12, 19, 20]. For the multiple-vehicle case, column generation approaches have been proposed. The first such method was introduced by Dumas et al. [13] for the PDPTW. Their set partitioning formulation is solved by a branch-and-price method in which columns of negative reduced cost are generated by a dynamic programming algorithm similar to that of Desrosiers et al. [12] for the single-vehicle case. The method has been successful in solving instances with tight capacity constraints and a small number of requests per route. Several arc elimination rules have also been proposed to reduce the size of the problem. A similar approach was later developed by Savelsbergh and Sol [23] who used a column management mechanism to reduce the size of the master

Received July 2005; accepted November 2006
Correspondence to: S. Ropke; e-mail: sropke@diku.dk
Contract grant sponsor: Natural Sciences and Engineering Research Council of Canada; Contract grant numbers: 227837-04 and OGP0039682
DOI 10.1002/net.20177
Published online in Wiley InterScience (www.interscience.wiley.com).

problem, and construction and improvement heuristics to accelerate the solution of the pricing subproblem.

Another solution methodology that has proven successful for solving the PDP is branch-and-cut. The single-vehicle case without time windows was first studied by Ruland and Rodin [22] who introduced several families of valid inequalities which are also valid for the PDPTW and will thus be described in more detail in Section 3. Branch-and-cut has also been used to solve the more general *Precedence-Constrained Asymmetric Traveling Salesman Problem* (PCATSP) in which each node may have multiple predecessors. Valid inequalities and a branch-and-cut algorithm for this problem have been developed, respectively, by Balas et al. [5] and Ascheuer et al. [3]. A branch-and-cut algorithm for the capacitated multiple-vehicle PDP and PDPTW was later described by Lu and Dessouky [15]. Their formulation contains a polynomial number of constraints and uses two-index flow variables, but relies on extra variables to impose pairing and precedence constraints. Instances with up to five vehicles and 25 requests were solved optimally with this approach. More recently, Cordeau [6] has developed a branch-and-cut algorithm for the DARP, based on a three-index formulation with a polynomial number of constraints. It uses several families of valid inequalities that are either adaptations of existing inequalities for the TSP and the VRP, or new inequalities that take advantage of the structure of the problem. Most of these inequalities are valid for the PDPTW and will also be described in Section 3. This approach was capable of solving instances with up to four vehicles and 32 requests.

In this paper, we introduce new branch-and-cut algorithms for the classical version of the PDPTW, as defined in [10], and the closely related DARP. We make three contributions. First, we propose two new formulations for the PDPTW which, unlike the formulation of Cordeau [6], have an exponential number of constraints, but lead to more efficient solution algorithms because they contain fewer variables and provide tighter bounds. Second, we introduce new valid inequalities combining the pickup and delivery structure of the problem with either the vehicle capacity constraints or the time window constraints. Third, we report computational experiments on several sets of test instances and show that our approach is capable of solving some instances with up to eight vehicles and 96 requests.

The remainder of the paper is organized as follows. Section 2 formally defines the PDPTW and introduces two formulations of the problem. Section 3 describes the valid inequalities used in the branch-and-cut algorithms which are then introduced in Section 4. Computational results are reported in Section 5, followed by conclusions in the last section.

2. FORMULATIONS OF THE PDPTW

Let n denote the number of requests to satisfy. The PDPTW can be defined on a directed graph $G = (N, A)$ with node set $N = \{0, \dots, 2n+1\}$ and arc set A. Nodes 0 and $2n+1$ represent the origin and destination depots (which may have the same location) while subsets $P = \{1, \dots, n\}$ and $D = \{n+1, \dots, 2n\}$ represent pickup and delivery nodes, respectively. With each request i are thus associated a pickup node i and a delivery node $n+i$. With each node $i \in N$ are associated a load q_i and a non-negative service duration d_i satisfying $d_0 = d_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$, and for $i = 1, \dots, n$, $q_i \geq 0$ and $q_{n+i} = -q_i$. An unlimited fleet of identical vehicles with capacity Q is available to serve the requests. With each arc $(i,j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij}. A time window $[e_i, l_i]$ is also associated with every node $i \in P \cup D$, where e_i and l_i represent the earliest and latest time, respectively, at which service may start at node i. The depot nodes may also have time windows $[e_0, l_0]$ and $[e_{2n+1}, l_{2n+1}]$ representing the earliest and latest times, respectively, at which the vehicles may leave from and return to the depot. We assume that the triangle inequality holds both for routing costs and travel times. Finally, to impose pairing and precedence constraints, it is convenient to define the set $\mathcal{S}$ of all node subsets $S \subseteq N$ such that $0 \in S$, $2n+1 \notin S$ and there is at least one request i for which $i \notin S$ and $n+i \in S$.

For any node subset $S \subseteq N$, define its complement $\bar{S} = N \setminus S$. Let also $\delta(S) = \delta^+(S) \cup \delta^-(S)$, where $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$. Finally, define $x(S) = \sum_{i,j \in S} x_{ij}$ and $x(S : T) = \sum_{i \in S} \sum_{j \in T} x_{ij}$, where $S, T \subseteq N$.

For each arc $(i,j) \in A$ let x_{ij} be a binary variable equal to 1 if and only if a vehicle travels directly from node i to node j. For each node $i \in P \cup D$, let B_i be the time at which service begins, and Q_i be the vehicle load upon departure.

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

$$\text{(PDPTW1)} \qquad \text{Minimize} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in P \cup D \tag{2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in P \cup D \tag{3}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{4}$$

$$B_j \geq (B_i + d_i + t_{ij}) x_{ij} \qquad \forall i \in N, \ j \in N \tag{5}$$

$$Q_j \geq (Q_i + q_j) x_{ij} \qquad \forall i \in N, \ j \in N \tag{6}$$

$$e_i \leq B_i \leq l_i \qquad \forall i \in N \tag{7}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \qquad \forall i \in N \tag{8}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in N, \ j \in N. \tag{9}$$

The objective function (1) minimizes the total routing cost. Constraints (2) and (3) require each node to be visited exactly once. Consistency of the time and load variables is ensured through constraints (5) and (6). The respect of time windows and vehicle capacity is then enforced through constraints (7) and (8). Under the assumption that $d_i + t_{i,n+i} > 0$ for every request i, constraints (5) and (7) also guarantee that no subtours exist in the solution.

Finally, inequalities (4) are *precedence constraints* [22], which state that for each user i, node $n+i$ is visited after node i and both nodes are visited by the same vehicle. These constraints were originally proposed in a single-vehicle context but they apply directly to the multivehicle case because route feasibility conditions are the same in both cases. In multivehicle problems constraints (4) not only enforce precedence relations but, because of the definition of $\mathcal{S}$, also ensure that the two nodes of the same request are on the same route. Indeed, suppose that a feasible integer solution contains a path $(0, k_1, \ldots, k_r, n+i)$ where $k_j \neq i$ for all j, i.e., a path connecting the origin depot to node $n+i$ without visiting node i. In this case, the set $S = \{0, k_1, \ldots, k_r, n+i\}$ clearly belongs to $\mathcal{S}$ and leads to a violation of the associated inequality (4). It is also worth pointing out that because $x(S) = |S| - 1 - x(\delta^-(S))$ for sets S such that $0 \in S$, inequalities (4) can be written equivalently as $x(\delta^-(S)) \geq 1$ for all $S \in \mathcal{S}$.

Introducing variables L_i to represent the ride time of each user i, and denoting by L the maximum ride time, the DARP can be modeled through the following constraints:

$$L_i = B_{n+i} - (B_i + d_i) \quad \forall i \in P \tag{10}$$

$$t_{i,n+i} \leq L_i \leq L \quad \forall i \in P. \tag{11}$$

Formulation (1)–(9) is nonlinear because of constraints (5) and (6). Introducing constants M_{ij} and W_{ij}, these constraints can be linearized as follows:

$$B_j \geq B_i + d_i + t_{ij} - M_{ij}(1 - x_{ij}) \quad \forall i \in N, \ j \in N \tag{12}$$

$$Q_j \geq Q_i + q_j - W_{ij}(1 - x_{ij}) \quad \forall i \in N, \ j \in N. \tag{13}$$

The validity of these constraints is ensured by setting $M_{ij} \geq \max\{0, l_i + d_i + t_{ij} - e_j\}$ and $W_{ij} \geq \min\{Q, Q + q_i\}$. As shown by Desrochers and Laporte [11], constraints (12) and (13), for a given pair $i, j \in N$, can be lifted as follows by taking the reverse arc (j, i) into account:

$$\begin{aligned} B_j \geq B_i + d_i + t_{ij} - M_{ij}(1 - x_{ij}) & \\ + (M_{ij} - d_i - t_{ij} - \max\{d_j + t_{ji}, e_i - l_j\})x_{ji} & \quad (14) \end{aligned}$$

$$Q_j \geq Q_i + q_j - W_{ij}(1 - x_{ij}) + (W_{ij} - q_i - q_j)x_{ji}. \tag{15}$$

In the case of the DARP, lifting (14) is, however, invalid because of constraints (10) and (11) which put additional restrictions on the time variables B_i.

As suggested by Desrochers and Laporte [11], bounds on the time variables can also be strengthened as follows:

$$B_i \geq e_i + \sum_{j \in N \setminus \{i\}} \max\{0, e_j - e_i + d_j + t_{ji}\}x_{ji} \tag{16}$$

$$B_i \leq l_i - \sum_{j \in N \setminus \{i\}} \max\{0, l_i - l_j + d_i + t_{ij}\}x_{ij}. \tag{17}$$

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

$$Q_i \geq \max\{0, q_i\} + \sum_{j \in N \setminus \{i\}} \max\{0, q_j\}x_{ji} \tag{18}$$

$$\begin{aligned} Q_i \leq \min\{Q, Q + q_i\} - \left(Q - \max_{j \in N \setminus \{i\}} \{q_j\} - q_i\right) x_{0i} & \\ - \sum_{j \in N \setminus \{i\}} \max\{0, q_j\}x_{ij}. & \quad (19) \end{aligned}$$

A formulation with fewer variables can be obtained by replacing constraints (5)–(8) with *rounded capacity inequalities* (see, e.g., [17]) and *infeasible path elimination constraints* (see, e.g., [1]). For any subset $S \subseteq P \cup D$, define $q(S) = \sum_{i \in S} q_i$. A lower bound on the number of times vehicles must enter and leave S in order to visit all nodes in the set is then provided by $\max\{1, \lceil |q(S)|/Q \rceil\}$. Denote by $\mathcal{R}$ the set of infeasible paths with respect to time windows, and let $A(R)$ and $N(R)$ be the arc set and the node set of R. With these definitions, the PDPTW can be reformulated as follows:

$$\text{(PDPTW2)} \qquad \text{Minimize} \sum_{i \in N} \sum_{j \in N} c_{ij}x_{ij} \tag{20}$$

subject to

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in P \cup D \tag{21}$$

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in P \cup D \tag{22}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \quad \forall S \in \mathcal{S} \tag{23}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - \max\{1, \lceil |q(S)|/Q \rceil\} \quad \forall S \subseteq N \setminus \{0, 2n+1\}, \ |S| \geq 2 \tag{24}$$

$$\sum_{(i,j) \in A(R)} x_{ij} \leq |A(R)| - 1 \quad \forall R \in \mathcal{R} \tag{25}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in N, \ j \in N. \tag{26}$$

With formulation (PDPTW2), the DARP can be modeled by simply introducing in set $\mathcal{R}$ the paths violating the ride time constraints.

Constraints (25) can in fact be strengthened into so-called tournament constraints (see, e.g., [1]) as follows. If $R = (k_1, \ldots, k_r)$ is an infeasible path, then the following inequality is valid:

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} x_{k_i,k_j} \leq |A(R)| - 1. \tag{27}$$

Infeasible path constraints can also be strengthened when they link a node pair $(i, n+i)$. Consider a path $R = (i, k_1, \ldots, k_r, n+i)$. If the triangle inequality holds and R is infeasible because of time windows or ride time constraints, then the following inequality is valid (see [6]):

$$x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + x_{k_r,n+i} \le |A(R)| - 2. \tag{28}$$

Finally, if both the path $R = (k_1, \dots k_r)$ and the reverse path $R' = (k_r, \dots, k_1)$ are infeasible, then the following symmetric inequality is clearly valid:

$$\sum_{i=1}^{r-1} (x_{k_i,k_{i+1}} + x_{k_{i+1},k_i}) \le r - 1. \tag{29}$$

Although formulations (PDPTW1) and (PDPTW2) assume identical vehicles, vehicles of different capacities can be handled through the introduction of dummy requests. Suppose that m vehicles of capacity $Q^1, \dots, Q^m$ are available and let $Q = \max_{1 \le i \le m}\{Q^i\}$. One can then define m dummy requests $i = 1, \dots, m$ with $d_i = d_{n+i} = 0$ and $q_i = -q_{n+i} = Q - Q^i$. Each dummy pickup node should be reachable only from the origin depot, while each dummy delivery node should connect only to the destination depot (both with cost and travel time equal to 0). The arc from a dummy pickup node to a normal pickup node j should have a cost c_{0j} and a travel time t_{0j}, while the arc from a normal delivery node $n + j$ to a dummy delivery node should have a cost $c_{n+j,2n+1}$ and a travel time $t_{n+j,2n+1}$. Finally, the corresponding values should be zero for all arcs between dummy pickup and delivery nodes.

Finally, note that, as observed by Ascheuer et al. [2] in the context of the TSP with time windows, model (PDPTW1) is more flexible than model (PDPTW2) in the sense that it can accommodate a more general objective function involving time and load variables. For example, one could minimize the makespan or a weighted sum of waiting times.

3. VALID INEQUALITIES

We now describe several families of valid inequalities for the PDPTW. These inequalities can be used to strengthen both (PDPTW1) and (PDPTW2). The first two families, subtour elimination constrains and generalized order constraints are borrowed from [6]. The next three families, strengthened capacity constraints, strengthened infeasible path constraints, and fork constraints, are new. The last family, reachability constraints, are adapted from existing inequalities for the VRPTW [16].

3.1. Subtour Elimination Constraints

Consider the simple subtour elimination constraint $x(S) \le |S| - 1$ for $S \subseteq P \cup D$. In the case of the PDPTW, this inequality can be lifted in several ways by taking into account the fact that for each request i, node i must be visited before node $n + i$. For any set $S \subseteq P \cup D$, let $\pi(S) = \{i \in P | n + i \in S\}$ and $\sigma(S) = \{n + i \in D | i \in S\}$ denote the sets of *predecessors* and *successors* of S, respectively. Balas et al. [5] have proposed two families of inequalities for the PCATSP which also apply to the PDPTW because 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 predecessor and successor inequalities are valid for the PDPTW:

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

$$x(S) + \sum_{i \in \bar{S} \cap \sigma(S)} \sum_{j \in S} x_{ij} + \sum_{i \in \bar{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} \le |S| - 1. \tag{31}$$

As shown by Cordeau [6], the D_k^- and D_k^+ inequalities introduced by Grötschel and Padberg [14] for the asymmetric TSP can also be lifted by taking precedence relationships into account. Let $S = \{i_1, \dots, i_k\} \subseteq P \cup D$ be an ordered set of nodes with $k \ge 3$. The following inequalities are then valid for the PDPTW:

$$\sum_{j=1}^{k-1} x_{i_j,i_{j+1}} + x_{i_k,i_1} + 2\sum_{j=3}^{k} x_{i_1,i_j} + \sum_{j=4}^{k} \sum_{l=3}^{j-1} x_{i_j,i_l} + \sum_{h \in \bar{S} \cap \pi(S)} x_{i_1,h} \le k - 1 \tag{32}$$

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

3.2. Generalized Order Constraints

Let $U_1, \dots, U_s \subset N$ be mutually disjoint subsets and let $i_1, \dots, i_s \in P$ be requests such that $0, 2n+1 \notin U_l$ and $i_l, n + i_{l+1} \in U_l$ for $l = 1, \dots, s$ (where $i_{s+1} = i_1$). The following inequality, introduced by Ruland and Rodin [22], is also valid for the PDPTW:

$$\sum_{l=1}^{s} x(U_l) \le \sum_{l=1}^{s} |U_l| - s - 1. \tag{34}$$

Similar inequalities, called *precedence cycle breaking inequalities*, have also been proposed by Balas et al. [5] for the PCATSP. In the case of a directed formulation, Cordeau [6] has shown that generalized order constraints can be lifted in two different ways as follows:

$$\sum_{l=1}^{s} x(U_l) + \sum_{l=2}^{s-1} x_{i_1,i_l} + \sum_{l=3}^{s} x_{i_1,n+i_l} \le \sum_{l=1}^{s} |U_l| - s - 1 \tag{35}$$

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

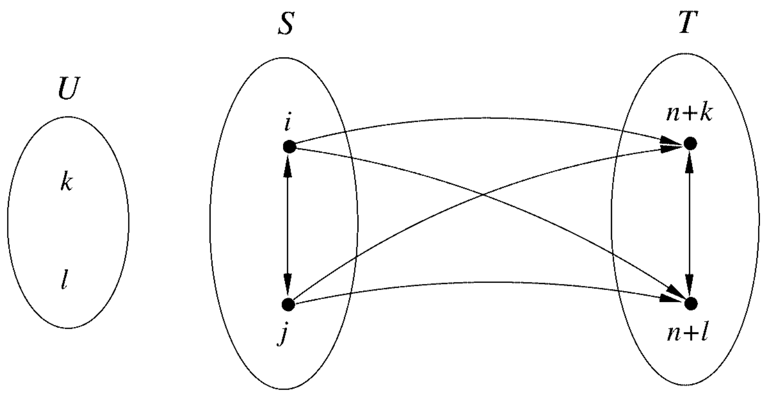

FIG. 1. Strengthened capacity constraint where $S = \{i, j\}$, $T = \{n+k, n+l\}$ and $U = \pi(T) \setminus (S \cup T) = \{k, l\}$.

3.3. *Strengthened Capacity Constraints*

Capacity constraints can be strengthened by considering node pairs $(k, n+k)$ such that the pickup node k is visited before entering set S, while the delivery node $n+k$ is visited after leaving this set. In this case, the number of vehicles visiting set S increases to accommodate the demand of all such node pairs. This yields the following result.

Proposition 1. *Let $S, T \subset P \cup D$ be two disjoint sets such that $q(S) > 0$. Also define $U = \pi(T) \setminus (S \cup T)$. The following inequality is then valid for the PDPTW:*

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S) + q(U)}{Q} \right\rceil. \tag{37}$$

Proof. Because $q(S) > 0$ and $q(U) \geq 0$, $x(\delta^+(S)) \geq \lceil q(S)/Q \rceil$ and $x(\delta^-(T)) \geq \lceil q(U)/Q \rceil$. If a path uses an arc from the set $(S:T)$ and reaches a node $n+k \in T$ with $k \in U$, without leaving set T, then node k must have been visited by that path before entering set S. Taking care not to count double the paths that leave set S and enter set T, one thus obtains

$$x(\delta^+(S)) + x(\delta^-(T)) - x(S:T) \geq \left\lceil \frac{q(S) + q(U)}{Q} \right\rceil.$$

Because $x(S) + x(\delta^-(S)) = x(S) + x(\delta^+(S)) = |S|$, this is equivalent to

$$|S| - x(S) + |T| - x(T) - x(S:T) \geq \left\lceil \frac{q(S) + q(U)}{Q} \right\rceil,$$

which yields the desired result after properly rearranging the terms. ∎

One may observe that inequalities (37) constitute a strengthening of the rounded capacity inequalities. From the inequalities

$$x(S) \leq |S| - \left\lceil \frac{q(S)}{Q} \right\rceil$$

and

$$x(T) + X(S:T) \leq x(T) + \delta^-(T) = |T|,$$

one obtains

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S)}{Q} \right\rceil,$$

which is indeed weaker than (37) if $q(U) > 0$.

Figure 1 depicts an example for which at most two arcs can be used if $q_i = q_j = q_k = q_l = 1$ and the vehicle capacity is $Q = 2$. Arcs in the figure are those on the left-hand-side of (37).

3.4. *Strengthened Infeasible Path Constraints*

Paths satisfying time windows can sometimes be eliminated by taking precedence relationships into account. Consider for instance the path $R = (i, n+j, k)$. Obviously, node j must be visited before R, while nodes $n+i$ and $n+k$ must be visited after R. Hence, if both $(j, i, n+j, k, n+i, n+k)$ and $(j, i, n+j, k, n+k, n+i)$ are infeasible, then R cannot belong to a feasible solution. More generally, let $\phi(S)$ denote the set of all permutations of nodes in S. If R is a feasible path in G but (ϕ_p, R, ϕ_d) is infeasible for all $\phi_p \in \phi(\pi(R) \setminus N(R))$ and $\phi_d \in \phi(\sigma(R) \setminus N(R))$ then R cannot belong to a feasible solution and it can thus be eliminated by (27).

3.5. *Fork Constraints*

Infeasible paths can also be eliminated in a different way by considering groups of infeasible paths sharing some common arcs. For instance, if the path $R = (k_1, \ldots, k_r)$ is feasible, but the path (i, R, j) is infeasible for every $i \in S$ and $j \in T$ with $S, T \subset N$, then the following inequality is clearly valid:

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_r,j} \leq r. \tag{38}$$

This inequality can be strengthened by associating to each intermediate node $k_2, \ldots, k_{r-1}$ a set of nodes leading to infeasible paths. This results in the following *outfork inequality*.

Proposition 2. *Let $R = (k_1, \ldots, k_r)$ be a feasible path in G and $S, T_1, \ldots, T_r \subset (P \cup D)$ be subsets such that $k_j \notin T_{j-1}$ for $j = 2, \ldots, r$. If for any integer $h \leq r$ and any node pair $i \in S, j \in T_h$, the path $(i, k_1, \ldots, k_h, j)$ is infeasible, then the following inequality is valid for the PDPTW:*

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{h=1}^{r} \sum_{j \in T_h} x_{k_h,j} \leq r. \qquad (39)$$

Proof. Assume that the inequality is violated in a feasible integer solution. Then, among the arcs belonging to the inequality, $r + 1$ must have been selected. Because of the degree constraints, there must be one arc from S to k_1, one outgoing arc from each node $k_1, \ldots, k_{r-1}$, and one arc from k_r to T_r. As a result, the path originating in S reaches one of the nodes in the sets $T_h, 1 \leq h \leq r$, and must thus be infeasible. ∎

The outfork inequality is illustrated in Figure 2 for the case $r = 3$. Similar inequalities, called *infork inequalities* and illustrated in Figure 3 for the case $r = 3$, are obtained by reversing the orientation of the arcs reaching path R. These lead to the following proposition.

Proposition 3. *Let $R = (k_1, \ldots, k_r)$ be a feasible path in G and $S_1, \ldots, S_r, T \subset (P \cup D)$ be subsets such that $k_j \notin S_{j+1}$ for all $j = 1, \ldots, r-1$. If for any integer $h \leq r$ and any node pair $i \in S_h, j \in T$, the path $(i, k_h, \ldots, k_r, j)$ is infeasible, then the following inequality is valid for the PDPTW:*

$$\sum_{h=1}^{r} \sum_{i \in S_h} x_{i,k_h} + \sum_{h=1}^{r-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_r,j} \leq r. \qquad (40)$$

It is worth pointing out that fork constraints can be used in any routing problem where the concept of infeasible paths is well defined, for instance the vehicle routing problem with time windows.

3.6. Reachability Constraints

For any node $i \in N$, let $A_i^- \subset A$ be the minimum arc set such that any feasible path from the origin depot 0 to node i uses only arcs from A_i^-. Let also A_i^+ be the minimum arc set such that any feasible path from i to the destination depot $2n + 1$ uses only arcs in A_i^+. Consider a node set T such

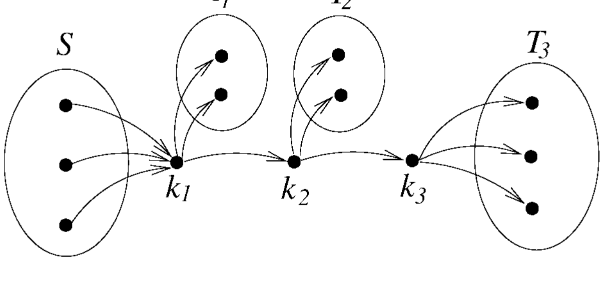

FIG. 2. Outfork constraint with $r = 3$.

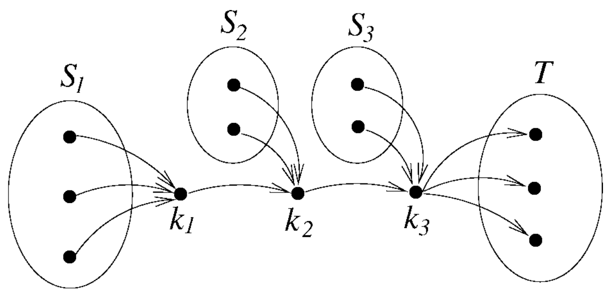

FIG. 3. Infork constraint with $r = 3$.

that each node in T must be visited by a different vehicle. This set is said to be *conflicting*. For any conflicting node set T, define the *reaching arc set* $A_T^- = \cup_{i \in T} A_i^-$ and the *reachable arc set* $A_T^+ = \cup_{i \in T} A_i^+$. For any node set $S \subseteq P \cup D$ and any conflicting node set $T \subseteq S$, the following two valid inequalities were introduced by Lysgaard [16] for the VRP with time windows:

$$x(\delta^-(S) \cap A_T^-) \geq |T| \qquad (41)$$

$$x(\delta^+(S) \cap A_T^+) \geq |T|. \qquad (42)$$

These inequalities are obviously also valid for the PDPTW. In this problem, however, nodes can be conflicting not only because of time windows but also because of the precedence relationships and the capacity constraints. In the case of the DARP, the ride time constraints should also be taken into account when checking whether a pair of requests is conflicting.

4. BRANCH-AND-CUT ALGORITHMS

We have implemented two branch-and-cut algorithms for the PDPTW: one with formulation (PDPTW1) and one with formulation (PDPTW2). In both algorithms, an attempt is made to generate violated valid inequalities at each node of the search tree. With formulation (PDPTW1), precedence inequalities (4) must be generated to ensure feasibility. With formulation (PDPTW2), feasibility is ensured by generating not only the precedence inequalities (23), but also the capacity inequalities (24) and infeasible path inequalities (25). In both formulations, the additional inequalities described in the previous section can be used to improve the LP relaxation obtained at each node of the branch-and-bound tree. In addition, inequalities (24) and (25) can be used to strengthen formulation (PDPTW1) although these are not required to ensure feasibility.

Taking into account the precedence relationships, time windows and ride time constraints, several arc elimination rules can be used in a preprocessing step to reduce the size of the problem. In addition, time windows can often be tightened. Details on these preprocessing steps can be found in the papers of Dumas et al. [13] and Cordeau [6].

In both branch-and-cut algorithms, the LP relaxations are solved by the simplex algorithm. Branching is performed on the x_{ij} variables by choosing, at each node of the enumeration tree, the variable whose value is the farthest from the nearest integer. The search is performed by applying the best-bound strategy. Before solving the problem, an upper bound is computed by using either the adaptive large neighbourhood search algorithm of Ropke and Pisinger [21] for the PDPTW or the tabu search heuristic of Cordeau and Laporte [7] for the DARP.

We first describe the separation procedures used to generate the precedence, capacity and infeasible path inequalities. We then describe procedures for the additional inequalities introduced in Section 3.

4.1. *Precedence Constraints*

As mentioned earlier, precedence constraints (4) and (23) can be written equivalently in the form $x(\delta^-(S)) \geq 1$ for all $S \in \mathcal{S}$. Violated inequalities of this form can be identified in polynomial time by solving a series of maximum flow problems: for each request i, one can compute the maximum flow from nodes i and $2n+1$ to nodes 0 and $n+i$ in G, with arc capacities given by the values of the x_{ij} variables. If the value of this flow is less than 1, then a precedence constraint 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 procedure by using the Ford-Fulkerson algorithm described by Cormen et al. [9].

4.2. *Capacity Constraints*

Two heuristics are used for the identification of violated capacity constraints. The first one is a randomized construction heuristic which starts from a given node $i \in P \cup D$ and gradually adds nodes to S by considering, at each iteration, the nodes connected to S with some flow. The choice of the node being added to S from the set of potential nodes is done randomly (with each node having a probability of being selected proportional to the flow on the corresponding arc). The procedure is repeated several times for each start node. If a capacity constraint is violated in an integer solution, the violation will clearly be detected by this procedure since it will add, at each iteration, the only node connected to the previously added node. At some point during the process, the set S will thus satisfy $q(S) > Q$.

The second heuristic is a simple tabu search procedure described by Cordeau [6] and inspired by that originally proposed by Augerat et al. [4]. This heuristic starts with either a random subset $S \subseteq P$ or a random subset $S \subseteq D$. At each iteration, a node is either removed or added to S so as to minimize the value of $x(\delta(S))$ while satisfying $q(S) > Q$.

4.3. *Subtour Elimination Constraints*

It is well known that the separation problem for subtour elimination constraints is solvable in polynomial time by computing the maximum flow between each node i and all other nodes $j \in N \setminus \{i\}$ (see [18]). This procedure, however, does not take into account the various liftings proposed in inequalities (30)–(33). Hence, we resort here to a simple tabu search heuristic very similar to the one used for capacity constraints and also described in more detail by Cordeau [6].

4.4. *Generalized Order Constraints*

We use two simple heuristics for the lifted generalized order constraints (35) and (36). These heuristics consider the case where $m = 3$ and $|U_1| = |U_2| = |U_3| = 2$. The first heuristic identifies, for each user i, a user j maximizing $x_{i,n+j} + x_{n+j,i} + x_{ij}$. It then finds a user k such that the left-hand side of (35) is maximized. The second heuristic identifies, for each user i, a user j maximizing $x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j}$ and then a user k maximizing the left-hand side of (36).

4.5. *Strengthened Capacity Constraints*

To identify sets S and T for which the strengthened capacity constraint is violated, we apply a construction heuristic similar to that used for the capacity constraints. This procedure starts from a set S containing a single pickup node and gradually augments this set by adding one node at a time. Before augmenting the set, the procedure determines

$$b_p \in \arg \max_{i \in P \setminus S} \{x(S : i) + x(i : S)\}$$

and

$$b_d \in \arg \max_{i \in D \setminus S} \{x(S : i) + x(i : S)\}$$

which we consider to be the best pickup (resp. delivery) node to add to the set. We prefer to add a pickup node to S in order to increase $q(S)$ on the right-hand side of inequality (37). Node b_d is only added if $x(S : b_p) + x(b_p : S) < x(S : b_d) + x(b_d : S)$, $q(S \cup \{b_d\}) > 0$ and either $x(S : b_d) + x(b_d : S) \geq 1$ or $x(S : i) + x(i : S) = 0$ for all $i \in P \setminus S$. Each time a node is added to S, the set T is reconstructed by using a similar construction heuristic where the roles of pickups and deliveries are interchanged. Only nodes from $N \setminus S$ are added to T.

At the root node of the search tree, we use a modified version of this heuristic where a random perturbation $\epsilon \sim U[0, 0.5]$ is added to the evaluation of $x(S : i) + x(i : S)$ and the heuristic is restarted several times from each pickup node.

4.6. *Strengthened Infeasible Path Constraints*

To identify infeasible paths violating constraints (25), we use an enumerative procedure similar to that of Ascheuer et al. [2]. In this procedure, every node $i \in P \cup D$ is in turn considered as a start node from which a tree of paths with positive flow is constructed. Each path is extended as long as a violation along this path is still possible (i.e., as long as the total flow on the arcs in path R is strictly greater than $|A(R)| - 1$ and the path has not reached node $2n+1$). Each time an infeasible path is identified, the corresponding tournament constraint (27) is generated.

TABLE 1. Characteristics of the Savelsbergh and Sol PDPTW instances.

Class	Q	W
A	15	60
B	20	60
C	15	120
D	20	120

A very similar procedure is used to identify violated strengthened infeasible path constraints for the DARP. In this case, however, each node $i \in P$ is considered as a start node and the extension of a path also stops if it reaches node $n+i$, at which point it is checked for feasibility with respect to the time windows and the ride time constraint for user i.

4.7. Fork Constraints

Observe that for $r = 1$, infork and outfork constraints coincide. A partial enumeration procedure is used to separate these particular fork constraints. This procedure first enumerates the set H of all infeasible paths containing three nodes. It then starts from an arc (i,j) and constructs the set S by identifying all paths of the form (h,i,j) belonging to H. Finally, the set T_1 is constructed by identifying all nodes k such that $(h,i,k) \in H$ for every node $h \in S$. This procedure is repeated for every arc (i,j) for which $x_{ij} > 0$ in the current solution. A similar procedure then starts from an arc (i,j) and constructs a set T containing all nodes k such that $(i,j,k) \in H$. The set S_1 is then constructed by identifying all nodes h such that $(h,j,k) \in H$ for every node $k \in T$.

For $r \geq 2$ a different heuristic is used. The heuristic iteratively uses every node $k_0 \in P \cup D$ as a seed node. From k_0, feasible paths $(k_0, k_1, \dots, k_l)$ are gradually constructed by extending existing paths along arcs with positive flow. For every path, one then checks if a violated fork constraint can be found with the path as a backbone. First, the set T is constructed such that $(k_0, k_1, \dots, k_l, j)$ is infeasible for all $j \in T$. Then, the set $S \ni k_0$ is constructed such that all paths $(i, k_1, \dots, k_l, j), i \in S, j \in T$ are infeasible. The two sets S and T and the path $(k_1, \dots, k_l)$ define a simple fork inequality (38). If this inequality is not violated, the procedure attempts to lift it into an outfork or an infork inequality. To lift the inequality into an outfork inequality, one adds as many nodes as possible to the sets $T_1, \dots, T_l$. A similar approach is used to lift the inequality into an infork. To keep running times low, only paths containing at most six nodes are considered. Checking whether a path is infeasible can be time consuming as many permutations have to be examined as described in Section 3.4. To alleviate this problem the feasibility of a path is checked only once, and the result of the query is stored in a hash table from which it can be quickly retrieved.

4.8. Reachability Constraints

Our procedure first computes, for each node $i \in P \cup D$, the sets A_i^+ and A_i^-. When doing this, precedence relationships must be taken into account. For example, when checking whether an arc $(i, n+j)$ belongs to the set A_{n+k}^-, one must check the existence of a path containing this arc and such that k is visited before $n+k$, j is visited before $n+j$, and $n+i$ is visited after i. The procedure then identifies, by complete enumeration, all sets of conflicting requests with a cardinality smaller than or equal to a given threshold. Each set of conflicting requests gives rise to several sets of conflicting nodes. For a set of k conflicting requests, 2^k sets of conflicting nodes exist. When k is greater than a parameter τ we do not generate all conflicting node sets, but only those two consisting of either the pickups or the deliveries of the conflicting requests. At a fractional solution, one then considers each conflicting node set T and solves a maximum flow problem between the node 0 and the set T by considering only the arcs in A_T^-. If the capacity of the corresponding minimum cut is smaller than $|T|$, then a violation of a reachability cut has

TABLE 2. Solution values and computing times for the PDPTW heuristic.

Instance	U. Bound	Time	Density (%)
A30	51,317.40	0.45	14.9
A35	51,343.53	0.55	18.4
A40	61,609.44	0.69	15.9
A45	61,693.01	0.83	25.2
A50	71,932.03	0.98	17.5
A55	82,185.31	1.07	19.2
A60	92,366.70	1.28	16.9
A65	82,331.12	1.46	17.3
A70	112,458.28	1.64	16.3
A75	92,529.42	1.88	20.6
B30	51,193.62	0.47	21.9
B35	61,400.07	0.54	20.4
B40	51,421.35	0.74	20.4
B45	61,787.28	0.82	21.5
B50	71,889.75	0.98	20.9
B55	82,080.73	1.07	18.6
B60	102,323.77	1.23	14.7
B65	82,623.98	1.42	19.7
B70	92,647.75	1.68	18.9
B75	92,476.30	1.88	20.1
C30	51,145.18	0.47	14.4
C35	51,235.64	0.57	17.6
C40	61,473.91	0.72	18.6
C45	81,408.89	0.83	21.1
C50	61,936.27	1.06	20.1
C55	61,930.55	1.19	20.9
C60	72,104.00	1.38	18.0
C65	82,326.62	1.50	24.0
C70	92,613.68	1.70	19.0
C75	92,711.74	1.88	21.4
D30	61,040.10	0.46	22.9
D35	71,308.04	0.56	25.6
D40	61,531.68	0.72	25.5
D45	81,601.63	0.80	17.7
D50	71,761.23	1.00	20.9
D55	72,051.95	1.15	21.7
D60	82,308.08	1.31	18.0
D65	82,200.77	1.50	24.9
D70	82,631.56	1.70	19.2
D75	92,970.84	1.83	21.6

TABLE 3. Root node lower bounds as a percentage of the upper bound (PDPTW instances).

	LP1	LP2	SEC	SCC	GOC	FC	RC	FULL	U. Bound
A30	99.95	99.98	99.98	99.98	99.98	100.00	99.99	100.00	51,317.40
A35	99.68	99.84	99.85	99.86	99.84	100.00	99.99	100.00	51,343.53
A40	97.42	99.85	99.85	99.86	99.85	100.00	100.00	100.00	61,609.44
A45	83.21	83.35	83.35	83.39	83.35	83.90	83.83	83.99	61,693.01
A50	78.09	72.20	72.20	74.53	72.20	93.13	99.99	100.00	71,932.03
A55	71.49	88.50	88.50	88.52	88.17	93.85	99.91	99.95	82,185.31
A60	83.70	92.19	92.19	92.21	92.19	100.00	100.00	100.00	92,366.70
A65	89.27	89.23	89.23	89.24	89.23	99.99	99.97	100.00	82,331.12
A70	82.03	91.00	91.00	91.01	91.00	93.93	93.98	95.60	112,458.28
A75	57.29	70.27	70.27	70.30	70.27	78.44	99.89	99.97	92,525.46
B30	85.21	84.36	84.36	84.36	84.36	99.99	99.99	99.99	51,193.62
B35	67.35	70.74	70.74	72.89	70.74	89.24	91.92	91.93	61,400.07
B40	64.97	65.45	65.45	65.48	65.45	83.55	80.86	85.72	51,421.35
B45	67.29	67.51	67.51	69.73	67.51	99.96	99.92	99.97	61,787.28
B50	51.32	66.52	66.52	66.53	66.52	87.90	99.95	99.98	71,889.75
B55	63.37	58.88	58.89	59.85	58.89	99.98	99.97	99.99	82,080.73
B60	80.37	80.43	80.43	80.43	80.43	90.58	99.99	100.00	102,323.77
B65	85.88	75.39	75.39	75.40	75.39	94.41	99.89	99.93	82,617.22
B70	61.03	67.32	67.32	67.34	67.32	94.57	99.92	99.96	92,641.67
B75	56.32	58.66	60.10	59.05	58.93	85.46	89.23	89.35	92,476.30
C30	90.17	90.28	90.28	90.28	90.28	100.00	99.99	100.00	51,145.18
C35	80.24	80.33	80.33	80.35	80.33	80.72	99.94	99.98	51,235.64
C40	67.20	67.32	67.33	67.34	67.32	83.80	83.78	83.83	61,473.91
C45	50.74	75.51	75.58	75.61	75.60	87.76	99.96	100.00	81,405.96
C50	99.40	99.52	99.52	99.53	99.52	99.92	99.86	99.94	61,933.09
C55	67.04	67.20	67.21	67.23	67.20	91.90	99.81	99.92	61,930.55
C60	57.85	68.07	68.07	68.10	68.07	99.86	99.80	99.89	72,100.68
C65	53.96	54.84	54.84	54.86	54.84	76.57	99.70	99.79	82,326.62
C70	56.35	56.47	56.48	56.48	56.48	84.96	89.11	89.22	92,613.68
C75	56.17	67.03	67.04	67.06	67.03	78.33	99.71	99.82	92,711.74
D30	64.68	67.16	67.16	67.18	67.15	88.45	99.95	99.99	61,040.10
D35	46.34	47.20	47.19	47.21	47.20	58.04	99.86	99.93	71,308.04
D40	67.00	67.11	67.12	67.15	67.11	99.86	99.79	99.87	61,531.68
D45	87.76	87.56	87.56	87.56	87.56	99.98	99.98	99.99	81,601.52
D50	54.60	57.99	57.99	58.03	57.99	86.14	99.92	99.99	71,761.23
D55	52.59	57.95	57.95	58.00	57.96	86.06	99.80	99.90	72,051.95
D60	75.36	75.40	75.40	75.41	75.40	99.97	99.91	99.98	82,306.47
D65	49.05	38.71	38.72	38.78	38.73	93.77	99.72	99.85	82,200.77
D70	55.52	51.12	51.13	51.14	51.12	79.38	99.73	99.83	82,631.56
D75	38.78	34.84	34.84	34.89	37.51	63.49	99.62	99.76	92,970.84
Avg.	*69.90*	*72.33*	*72.37*	*72.55*	*72.40*	*90.20*	*97.73*	*97.95*	

been found. The same is done by considering A_T^+ and solving a maximum flow problem between set T and the destination depot $2n+1$.

5. COMPUTATIONAL EXPERIMENTS

The two branch-and-cut algorithms were implemented in C++ by using ILOG Concert 1.3 and CPLEX 9.0. All experiments were performed on an AMD Opteron 250 computer (2.4 GHz). Several sets of instances for the PDPTW and the DARP were used for testing. All instances are available on `http://www.hec.ca/chairedistributique/data`.

5.1. Results for the PDPTW

We first generated some PDPTW instances as suggested by Savelsbergh and Sol [23]. In these instances, the coordinates of each pickup and delivery location are randomly chosen according to a uniform distribution over the $[0, 200] \times [0, 200]$ square. The load q_i of request i is randomly selected from the interval $[5, Q]$, where Q is the vehicle capacity. A planning horizon of length $H = 600$ is considered and each time window has width W. The time windows for request i are constructed by first randomly selecting e_i in the interval $[0, H - t_{i,n+i}]$ and then setting $l_i = e_i + W$, $e_{n+i} = e_i + t_{i,n+i}$ and $l_{n+i} = e_{n+i} + W$. In all instances, the primary objective consists of minimizing the number of vehicles, and a fixed cost of 10^4 is thus imposed on each outgoing arc from the depot. Four classes of instances are obtained by varying the values of Q and W, as indicated in Table 1.

In the test instances generated by Savelsbergh and Sol [23], each vehicle has a different depot whose location is also randomly chosen over the $[0, 200] \times [0, 200]$ square. Because our formulations cannot directly handle multiple depots, we

TABLE 4. Computational results for PDPTW instances.

		(PDPTW1)				(PDPTW2)			
Instance	U. Bound	Time	Nodes	Cuts	L. Bound	Time	Nodes	Cuts	L. Bound
A30	51, 317.40	0.05	0	78		0.05	0	140	
A35	51, 343.53	0.11	0	407		0.10	0	602	
A40	61, 609.44	0.16	0	382		0.16	0	510	
A45	61, 693.01				51, 824.39				51, 842.89
A50	71, 932.03	0.52	0	838		0.41	0	988	
A55	82, 185.31	18.30	855	3877		4.75	91	3704	
A60	92, 366.70	0.86	0	766		0.81	0	1071	
A65	82, 331.12	2.15	2	1325		1.75	0	1423	
A70	112, 458.28				109, 244.57	4.74	11	2492	
A75	92, 525.46				92, 503.50	36.82	438	6414	
B30	51, 193.62	0.10	2	426		0.10	7	549	
B35	61, 400.07	0.18	2	462		0.15	2	743	
B40	51, 421.35	1.26	70	1638		0.52	17	1134	
B45	61, 787.28	1.53	98	1636		0.87	31	1814	
B50	71, 889.75	6.19	603	2134		1.32	8	2441	
B55	82, 080.73	1.06	2	1138		1.07	2	1684	
B60	102, 323.77	2.26	4	1891		2.17	6	1975	
B65	82, 617.22				82, 573.37	77.00	1884	10505	
B70	92, 641.67	109.08	3012	5777		13.97	158	4578	
B75	92, 476.30				82, 649.55				84, 105.71
C30	51, 145.18	0.06	0	103		0.06	0	158	
C35	51, 235.64	0.32	5	920		0.26	6	1059	
C40	61, 473.91				51, 565.97				51, 628.73
C45	81, 405.96	0.94	4	1515		0.69	6	1994	
C50	61, 933.09	40.64	1541	5637		7.49	228	3989	
C55	61, 930.55	96.36	2529	8311		19.32	345	6188	
C60	72, 100.68				72, 053.40	76.54	3294	10498	
C65	82, 326.62				82, 159.87				82, 163.46
C70	92, 613.68				82, 666.21				86, 645.63
C75	92, 711.74				92, 555.04				92, 554.26
D30	61, 040.10	0.30	12	896		0.23	18	1166	
D35	71, 308.04				71, 299.29	33.07	3155	8618	
D40	61, 531.68				61, 463.09				61, 493.63
D45	81, 601.52	1.16	34	1167		0.82	27	1270	
D50	71, 761.23	4.22	204	2100		1.61	16	2216	
D55	72, 051.95				72, 001.09				72, 034.13
D60	82, 306.47	5.36	144	2098		3.25	51	2589	
D65	82, 200.77				82, 091.05				82, 122.81
D70	82, 631.56				82, 493.27				82, 514.97
D75	92, 970.84				92, 751.85				92, 751.63

have instead used a single depot located in the middle of the square.

It is apparent from the results reported by Savelsbergh and Sol [23] that using the $[0, 200] \times [0, 200]$ square with $H = 600$ yields instances in which it is difficult to serve more than two or three requests in the same route. In addition, the long travel times make it difficult to stop at an intermediate location between the pickup of a request and its delivery. As a result, all instances generated in this way could be solved at the root node by our algorithms. To obtain harder instances, we have decreased the size of the square from which the locations are chosen. By choosing coordinates from the set $[0, 50] \times [0, 50]$, travel times become smaller and it is then possible to serve more requests in each route. Furthermore, it becomes easier to produce a sequence of several successive pickups followed by the corresponding deliveries. In each of the four problem classes, we have generated ten instances by considering values of n between 30 and 75. The name of each instance (e.g., A50) indicates the class to which it belongs and the number of requests it contains.

We first present in Table 2 the solution values and computing times (in minutes) of the Ropke and Pisinger [21] adaptive large neighbourhood search heuristic for the PDPTW. The table also indicates the graph density after preprocessing, calculated as $100|A|/(2n + 2)^2$, where $(2n + 2)^2$ is the number of arcs in the complete graph, including loop-arcs.

To evaluate the strength of formulations (PDPTW1) and (PDPTW2), we have first solved the LP relaxation of both formulations by considering the minimal sets of inequalities required for feasibility. Hence, violated

TABLE 5. Characteristics of DARP instances.

Instance	$\lvert K\rvert$	n	H	Q	L	Instance	$\lvert K\rvert$	n	H	Q	L
a2-16	2	16	480	3	30	b2-16	2	16	480	6	45
a2-20	2	20	600	3	30	b2-20	2	20	600	6	45
a2-24	2	24	720	3	30	b2-24	2	24	720	6	45
a3-24	3	24	480	3	30	b3-24	3	24	480	6	45
a3-30	3	30	600	3	30	b3-30	3	30	600	6	45
a3-36	3	36	720	3	30	b3-36	3	36	720	6	45
a4-32	4	32	480	3	30	b4-32	4	32	480	6	45
a4-40	4	40	600	3	30	b4-40	4	40	600	6	45
a4-48	4	48	720	3	30	b4-48	4	48	720	6	45
a5-40	5	40	480	3	30	b5-40	5	40	480	6	45
a5-50	5	50	600	3	30	b5-50	5	50	600	6	45
a5-60	5	60	720	3	30	b5-60	5	60	720	6	45
a6-48	6	48	480	3	30	b6-48	6	48	480	6	45
a6-60	6	60	600	3	30	b6-60	6	60	600	6	45
a6-72	6	72	720	3	30	b6-72	6	72	720	6	45
a7-56	7	56	480	3	30	b7-56	7	56	480	6	45
a7-70	7	70	600	3	30	b7-70	7	70	600	6	45
a7-84	7	84	720	3	30	b7-84	7	84	720	6	45
a8-64	8	64	480	3	30	b8-64	8	64	480	6	45
a8-80	8	80	600	3	30	b8-80	8	80	600	6	45
a8-96	8	96	720	3	30	b8-96	8	96	720	6	45

precedence constraints were generated for (PDPTW1), while for (PDPTW2) we have also generated violated capacity constraints and infeasible path constraints. These results are reported in Table 3. For each instance, we indicate in columns LP1 and LP2 the value of the lower bound computed at the root node as a percentage of the upper bound indicated in the rightmost column of the table. This upper bound is either the optimal value of the problem, if the instance could be solved to optimality, or an upper bound computed by a heuristic, otherwise. One can see that for most instances (PDPTW2) provides a tighter lower bound, with an average of 72.33 for (PDPTW2) compared to 69.90 for (PDPTW1). In Tables 3 and 4, the number of vehicles in the solution is equal to $\lfloor U/10^4 \rfloor$, where U is the upper bound.

To measure the strength of each type of inequality introduced in Section 3, we then solved the LP relaxation of (PDPTW2) by separately considering each type of inequality: subtour elimination constraints (SEC), strengthened capacity constraints (SCC), generalized order constraints (GOC), fork constraints (FC), and reachability constraints (RC). Finally, column "Full" reports the lower bound obtained with (PDPTW2) with all families of valid inequalities. Again, all

TABLE 6. Solution values and computing times for the DARP heuristic.

Instance	U. Bound	Time	Density (%)	Instance	U. Bound	Time	Density (%)
a2-16	294.25	0.05	31.2	b2-16	309.61	0.05	28.1
a2-20	344.83	0.12	32.1	b2-20	334.93	0.10	18.9
a2-24	431.12	0.21	32.6	b2-24	445.11	0.21	23.2
a3-24	344.83	0.12	35.2	b3-24	394.57	0.11	23.3
a3-30	496.52	0.24	31.6	b3-30	536.04	0.24	23.5
a3-36	600.75	0.48	31.2	b3-36	611.79	0.46	21.8
a4-32	486.57	0.20	32.9	b4-32	500.92	0.17	20.8
a4-40	571.07	0.38	32.3	b4-40	662.91	0.38	20.2
a4-48	680.99	0.75	34.8	b4-48	685.46	0.77	27.0
a5-40	507.59	0.28	33.5	b5-40	619.09	0.26	27.2
a5-50	699.86	0.58	34.8	b5-50	777.20	0.55	23.5
a5-60	825.57	1.20	32.8	b5-60	923.07	1.08	27.6
a6-48	618.00	0.37	34.2	b6-48	727.06	0.31	21.7
a6-60	847.19	0.75	33.1	b6-60	888.28	0.77	23.3
a6-72	946.41	1.50	33.8	b6-72	1007.99	1.44	24.3
a7-56	745.08	0.52	32.3	b7-56	844.54	0.45	24.0
a7-70	936.96	1.00	33.4	b7-70	939.10	0.89	21.7
a7-84	1069.77	1.87	33.6	b7-84	1255.10	1.78	25.3
a8-64	770.52	0.69	33.9	b8-64	865.65	0.54	23.7
a8-80	992.52	1.22	35.0	b8-80	1085.91	1.33	21.3
a8-96	1289.59	2.55	33.5	b8-96	1236.42	2.36	26.0

TABLE 7. Root node lower bounds as a percentage of the upper bound (DARP instances).

	LP0	LP1	LP2	SEC	SCC	GOC	FC	RC	FULL	U. Bound
a2-16	98.42	99.14	99.93	99.93	99.93	99.93	100.00	100.00	100.00	294.25
a2-20	93.38	95.49	99.31	99.47	99.31	99.31	100.00	100.00	100.00	344.83
a2-24	87.51	95.25	98.81	99.03	98.81	98.81	99.80	99.51	99.80	431.12
a3-24	81.07	88.58	95.62	95.63	95.63	95.62	100.00	99.92	100.00	344.83
a3-30	79.16	88.84	94.55	94.55	94.55	94.52	100.00	100.00	100.00	494.85
a3-36	87.67	92.21	96.85	96.85	96.85	96.85	99.29	98.92	99.29	583.19
a4-32	78.12	85.69	92.23	92.23	92.32	92.23	100.00	100.00	100.00	485.50
a4-40	76.36	92.00	95.64	95.67	95.64	95.64	99.22	99.15	99.32	557.69
a4-48	64.63	86.12	91.96	91.96	91.97	91.96	99.38	98.75	99.62	668.82
a5-40	65.25	84.89	93.10	93.16	93.10	93.10	100.00	99.14	100.00	498.41
a5-50	59.92	78.87	88.40	88.44	88.41	88.40	98.79	97.71	99.04	686.62
a5-60	59.68	75.39	87.18	87.22	87.28	87.18	99.43	98.03	99.48	808.42
a6-48	63.57	79.95	88.25	88.31	88.26	88.26	99.97	98.75	100.00	604.12
a6-60	60.11	75.74	86.22	86.29	86.27	86.22	99.37	98.89	99.61	819.25
a6-72	65.42	79.88	89.08	89.27	89.22	89.09	99.14	98.18	99.36	916.05
a7-56	64.08	81.07	88.12	88.27	88.15	88.12	99.02	98.15	99.21	724.04
a7-70	62.66	77.81	85.74	85.76	85.87	85.74	99.57	98.78	99.75	889.12
a7-84	55.81	71.45	82.27	82.35	82.53	82.28	99.05	97.66	99.20	1033.37
a8-64	66.86	74.63	85.40	85.65	85.40	85.41	99.09	98.13	99.51	747.46
a8-80	58.29	69.87	81.10	81.10	81.19	81.10	99.04	96.17	99.18	945.73
a8-96	53.83	66.81	78.57	78.60	78.67	78.57	97.85	95.03	98.49	1232.61
Avg.	*70.56*	*82.84*	*90.40*	*90.46*	*90.45*	*90.40*	*99.43*	*98.61*	*99.56*	

lower bounds are expressed as a percentage of the upper bound reported in the last column of the table. These results show that fork constraints and reachability constraints have the largest impact, with all other types of inequalities playing only a minor role in the improvement of the lower bound. It is worth pointing out that for some instances (e.g., A55), the lower bound obtained with one type of inequality is sometimes worse than that obtained with just the basic formulation (column LP2). This is explained by the fact that we use a heuristic separation procedure for capacity constraints, which may lead to the generation of a different set of inequalities.

In Table 4, we report the results obtained by considering both formulations with all types of valid inequalities. For each instance solved to optimality, we indicate the CPU time in minutes (including the preprocessing time) needed to prove optimality, the number of nodes explored in the search tree and the total number of cuts generated during the search. When an instance could not be solved to optimality within the prescribed CPU time of two hours, we report the value of

TABLE 8. Root node lower bounds as a percentage of the upper bound (DARP instances).

	LP0	LP1	LP2	SEC	SCC	GOC	FC	RC	FULL	U. Bound
b2-16	91.76	97.19	99.52	99.53	99.52	99.52	99.59	99.53	99.59	309.41
b2-20	99.90	98.61	100.00	100.00	100.00	100.00	100.00	100.00	100.00	332.64
b2-24	89.30	96.42	99.24	99.24	99.35	99.24	99.96	99.96	99.96	444.71
b3-24	89.65	92.21	95.08	95.08	95.08	95.06	99.33	98.65	99.60	394.51
b3-30	87.91	98.86	99.91	99.91	99.91	99.91	100.00	100.00	100.00	531.44
b3-36	87.72	97.93	99.23	99.23	99.23	99.23	100.00	100.00	100.00	603.79
b4-32	83.82	99.34	100.00	100.00	100.00	100.00	100.00	100.00	100.00	494.82
b4-40	81.25	96.20	98.14	98.14	98.14	98.14	100.00	99.55	100.00	656.63
b4-48	81.75	88.92	95.84	95.82	95.94	95.84	99.67	98.87	99.70	673.81
b5-40	70.43	83.15	91.80	91.67	92.06	91.80	98.99	97.19	99.57	613.72
b5-50	70.78	88.19	92.57	92.44	93.07	92.49	99.20	98.48	99.30	761.40
b5-60	70.75	86.55	91.48	91.52	91.97	91.49	98.88	97.54	99.10	902.04
b6-48	75.90	95.05	98.85	98.82	99.10	98.83	100.00	99.88	100.00	714.83
b6-60	68.75	88.48	94.74	94.75	95.29	94.73	100.00	99.65	100.00	860.07
b6-72	69.99	83.97	90.31	90.36	90.71	90.41	97.76	96.90	98.41	978.47
b7-56	64.54	85.32	90.66	90.67	91.31	90.67	97.56	96.10	98.09	823.97
b7-70	68.25	89.43	94.16	94.24	94.41	94.17	99.44	98.27	99.43	912.62
b7-84	61.59	77.54	86.33	86.47	87.10	86.31	98.78	96.99	99.19	1203.37
b8-64	66.98	84.62	90.83	90.81	91.06	90.86	99.08	98.15	99.44	839.89
b8-80	65.85	88.46	92.18	92.24	92.43	92.20	99.61	98.87	99.63	1036.34
b8-96	59.73	77.36	84.71	84.75	85.30	84.79	97.56	94.36	98.30	1185.55
Avg.	*76.50*	*90.18*	*94.55*	*94.56*	*94.81*	*94.56*	*99.30*	*98.52*	*99.49*	

TABLE 9. Computational results for first set of DARP instances.

		DARP			PDPTW1			PDPTW2		
Instance	Cost	Time	Nodes	Cuts	Time	Nodes	Cuts	Time	Nodes	Cuts
a2-16	294.25	0.01	11	39	0.01	0	35	0.01	0	99
a2-20	344.83	0.05	203	91	0.02	0	84	0.02	0	183
a2-24	431.12	0.12	412	131	0.05	3	165	0.04	3	315
a3-24	344.83	1.31	3090	279	0.04	0	233	0.03	0	330
a3-30	494.85	15.48	22250	369	0.08	0	275	0.08	0	564
a3-36	583.19	8.13	8178	415	0.18	5	273	0.18	9	604
a4-32	485.50				0.12	0	482	0.09	0	759
a4-40	557.69				0.51	6	729	0.32	8	896
a4-48	668.82				1.01	12	1120	0.56	4	1848
a5-40	498.41				0.24	0	612	0.17	0	937
a5-50	686.62				2.67	209	1520	1.04	59	1875
a5-60	808.42				2.59	15	1648	1.55	9	2203
a6-48	604.12				0.90	0	1379	0.44	0	1893
a6-60	819.25				3.94	86	1913	1.69	13	2793
a6-72	916.05				8.16	143	2360	3.31	25	3144
a7-56	724.04				5.36	141	2194	1.72	67	2338
a7-70	889.12				6.13	15	2516	3.49	20	3655
a7-84	1033.37				29.43	315	3625	8.23	48	3970
a8-64	747.46				6.48	70	2443	3.61	86	2995
a8-80	945.73				26.14	305	3519	13.22	308	4360
a8-96	1232.61				1210.56	7758	9336	70.55	1652	7436

the current lower bound at the end of the computation (i.e., the lower bound associated with the best pending node). These results show that formulation (PDPTW2) provides a slightly better performance: it solved five more instances to optimality and for those instances that were solved by both formulations, (PDPTW2) required on average less CPU time, fewer nodes and fewer cuts. Finally, when neither model could reach an optimal solution, the latter usually provided a higher lower bound.

It is worth pointing out that some instances with a large initial integrality gap (such as instance B40) could be solved optimally while others with a small gap (such as instance C65) could not be solved. This is in large part explained by the fact that the gap often comes from the number of vehicles.

TABLE 10. Computational results for second set of DARP instances.

		(DARP)			(PDPTW1)			(PDPTW2)		
Instance	Cost	Time	Nodes	Cuts	Time	Nodes	Cuts	Time	Nodes	Cuts
b2-16	309.41	0.04	296	109	0.01	3	101	0.01	4	170
b2-20	332.64	0.01	0	30	0.01	0	12	0.01	0	80
b2-24	444.71	0.06	263	107	0.03	2	120	0.03	1	225
b3-24	394.51	0.55	2372	165	0.04	2	147	0.03	3	291
b3-30	531.44	3.20	6267	253	0.05	0	150	0.05	0	313
b3-36	603.79	37.57	44881	246	0.08	0	125	0.08	0	341
b4-32	494.82	75.42	44877	313	0.05	0	136	0.05	0	253
b4-40	656.63				0.15	0	315	0.14	0	609
b4-48	673.81				0.63	9	802	0.50	15	1072
b5-40	613.72				0.59	25	1064	0.28	8	1243
b5-50	761.40				0.92	28	1064	0.66	18	1326
b5-60	902.04				1.96	37	1404	1.81	97	1913
b6-48	714.83				0.27	0	478	0.24	0	613
b6-60	860.07				0.86	0	962	0.68	0	1237
b6-72	978.47				127.71	4769	4864	17.07	989	4278
b7-56	823.97				56.65	1959	7741	13.46	1120	5148
b7-70	912.62				4.66	157	1463	2.12	17	1892
b7-84	1203.37				11.51	77	3036	6.61	58	3463
b8-64	839.89				8.11	405	2155	2.50	66	2165
b8-80	1036.34				4.04	8	1455	3.05	8	1994
b8-96	1185.55				847.41	8246	10171	120.09	2746	8431

When the cuts generated at the root node do not force an outgoing flow from the depot node equal to the number of vehicles used in an optimal solution, the initial lower bound is weak and a large gap is obtained. However, the lower bound may quickly improve after a few branching nodes have been explored.

5.2. *Results for the DARP*

We have then tested our approach on two sets of randomly generated Euclidean DARP instances comprising up to 96 requests. These instances have narrow time windows of 15 minutes. In the first set ("a" instances), $q_i = 1$ for every request i and the vehicle capacity is $Q = 3$. In the second set ("b" instances), q_i belongs to the interval $[1, 6]$ and $Q = 6$. These data are described in detail in [6] and their main characteristics are summarized in Table 5. In this table, columns $|K|$ and H indicate, respectively, the number of available vehicles and the length of the planning horizon in which time windows are generated. The constraint on the number of vehicles is easily imposed in our formulations as a bound on the total outgoing flow from the origin depot. Finally, L denotes the maximum ride time.

We present in Table 6 the solution values and computing times for the Cordeau and Laporte [7] tabu search heuristic for the DARP. The table also shows the graph density after preprocessing.

Tables 7 and 8 show the strength of the lower bounds obtained with the different types of valid inequalities. These tables can be interpreted in the same way as Table 3. This time, however, we also indicate in column LP0 the lower bound obtained with the three-index formulation of Cordeau [6]. Again, formulation (PDPTW2) provides better bounds than (PDPTW1) while fork constraints and reachability constraints are the most useful. One can also see that both (PDPTW1) and (PDPTW2) perform much better than the three-index formulation in terms of the initial lower bound.

Finally, Tables 9 and 10 report the computational statistics collected when solving each instance to optimality with both (PDPTW1) and (PDPTW2), using again all types of inequalities. In column (DARP), we also indicate comparable statistics for the three-index DARP formulation of Cordeau [6]. For the latter formulation, only a small subset of all instances could be solved to optimality. As in Table 4, one can see that formulation (PDPTW2) usually requires less computation time and a smaller number of branch-and-bound nodes than (PDPTW1). The largest CPU time for (PDPTW1) is 1210.56 min compared to 120.09 min for (PDPTW2). Comparisons with the three-index DARP formulation show that the latter is totally dominated by the two new formulations. For example, instance b4-32 required more than one hour of CPU time and 44,877 branch-and-cut nodes with the three-index formulation, while it was solved in the root node with both (PDPTW1) and (PDPTW2). This dramatic improvement results from the better lower bound provided by the tighter (PDPTW1) and (PDPTW2) formulations, and from the new inequalities introduced in this paper.

6. CONCLUSION

By using appropriate inequalities, we have introduced two new formulations for the PDPTW which do not require the use of a vehicle index to impose pairing and precedence constraints, as is the case in three-index formulations. In addition to adapting infeasible path constraints and reachability constraints to take advantage of the structure of the problem, we have also introduced two new families of inequalities: strengthened capacity constraints and fork constraints. Computational experiments performed on PDPTW and DARP instances show that both formulations are competitive, although the more compact of the two (in terms of variables) has a slight advantage. In the case of the DARP, comparisons with a previously introduced three-index formulation show that the two new formulations are capable of solving much larger instances. The largest instance solved to optimality contains 194 nodes. Given the current state of the art for the exact solution of vehicle routing problems with time windows, it seems fair to say that these are large instances.

Acknowledgments

We are thankful to David Pisinger and to three anonymous referees for their valuable comments.

REFERENCES

[1] N. Ascheuer, M. Fischetti, and M. Grötschel, A polyhedral study of the asymmetric traveling salesman problem with time windows, Networks 36 (2000), 69–79.

[2] N. Ascheuer, M. Fischetti, and M. Grötschel, Solving the asymmetric travelling salesman problem with time windows by branch-and-cut, Math Program 90 (2001), 475–506.

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

[4] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef, Separating capacity inequalities in the CVRP using tabu search, Eur J Operational Res 106 (1999), 546–557.

[5] E. Balas, M. Fischetti, and W.R. Pulleyblank, The precedence-constrained asymmetric traveling salesman polytope, Math Program 68 (1995), 241–265.

[6] J.-F. Cordeau, A branch-and-cut algorithm for the dial-a-ride problem, Operations Res 54 (2006), 573–586.

[7] J.-F. Cordeau and G. Laporte, A tabu search heuristic for the static multi-vehicle dial-a-ride problem, Transport Res B 37 (2003), 579–594.

[8] J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M.W.P. Savelsbergh, "Transportation on demand," Transportation, Handbooks in Operations Research and Management Science, C. Barnhart and G. Laporte (Editors), Elsevier, Amsterdam, 2007, pp. 429–466.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, Mass., 1990.

[10] G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, and F. Soumis, "Vrp with pickup and delivery," The Vehicle

Routing Problem, P. Toth and D. Vigo (Editors), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002, pp. 225–242.

[11] M. Desrochers and G. Laporte, Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints, Operations Res Lett 10 (1991), 27–36.

[12] J. Desrosiers, Y. Dumas, and F. Soumis, A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows, Am J Math Manage Sci 6 (1986), 301–325.

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

[14] M. Grötschel and M.W. Padberg, “Polyhedral theory,” The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (Editors), Wiley, Chichester, 1985, pp. 251–305.

[15] Q. Lu and M. Dessouky, An exact algorithm for the multiple vehicle pickup and delivery problem, Transportation Sci 38 (2004), 503–514.

[16] J. Lysgaard, Reachability cuts for the vehicle routing problem with time windows, Eur J Operational Res 175 (2006), 210–223.

[17] D. Naddef and G. Rinaldi, “Branch-and-cut algorithms for the capacitated vrp,” The Vehicle Routing Problem, P. Toth and D. Vigo (Editors), SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002, pp. 53–84.

[18] M.W. Padberg and M. Grötschel, “Polyhedral computations,” The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (Editors), Wiley, Chichester, 1985, pp. 307–360.

[19] H.N. Psaraftis, A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem, Transport Sci 14 (1980), 130–154.

[20] H.N. Psaraftis, An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows, Transport Sci 17 (1983), 351–357.

[21] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, Transport Sci 40 (2006), 455–472.

[22] K.S. Ruland and E.Y. Rodin, The pickup and delivery problem: Faces and branch-and-cut algorithm, Computers Math Appl 33 (1997), 1–13.

[23] M.W.P. Savelsbergh and M. Sol, DRIVE: Dynamic routing of independent vehicles, Operations Res 46 (1998), 474–490.