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

STEFAN ROPKE[*][†]
JEAN-FRANÇOIS CORDEAU[†]
GILBERT LAPORTE[†]

July 14, 2005

Abstract

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 a branch-and-cut algorithm which has been tested on several sets of instances for both the PDPTW and the DARP. Instances with up to eight vehicles and 96 requests (192 nodes) have been solved to optimality.

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

[*]DIKU, University of Copenhagen, Denmark
[†]Canada Research Chair in Distribution Management, HEC Montréal, Canada

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 (see, e.g., Desaulniers et al., 2002; Cordeau et al., 2005). 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 (Psaraftis, 1980, 1983; Desrosiers et al., 1986). For the multiple-vehicle case, column generation approaches have been proposed. The first such method was introduced by Dumas et al. (1991) who addressed 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. (1986) 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 (1998) who used a column management mechanism to reduce the size of the master 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 (1997) who introduced several families of valid inequalities that 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. (1995) and Ascheuer et al. (2000b). A branch-and-cut algorithm for the capacitated multiple-vehicle PDP and PDPTW was later described by Lu and Dessouky (2004). 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 (2005) has developed a branch-and-cut algorithm for the DARP. It is 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 which 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 a new branch-and-cut algorithm for the PDPTW and the closely related DARP. We make three contributions. First, we propose two new formulations for the PDPTW which, unlike the formulation of Cordeau (2005), 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 algorithm which is 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 $N = \{0, \ldots, 2n + 1\}$ and arc set A. Nodes 0 and $2n + 1$ represent the origin and destination depots (which may have the same location) while subsets $P = \{1, \ldots, n\}$ and $D = \{n + 1, \ldots, 2n\}$ 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 $q_0 = q_{2n+1} = 0$, $q_i = -q_{n+i}$ $(i = 1, \ldots, n)$ and $d_0 = d_{2n+1} = 0$. A 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 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 at node i, and Q_i be the load of the vehicle visiting node i, immediately after its departure from this node.

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

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

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 ensured 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 ensure that no subtours exist in the solution. Finally, inequalities (4) are *precedence constraints* (see Ruland and Rodin, 1997) which guarantee that for each user i, node $n+i$ is visited after node i and both nodes are visited by the same vehicle.

By introducing variables L_i representing the ride time of each user i, and denoting by L the maximum ride time, the DARP can be modeled by introducing the following constraints:

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

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

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

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

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

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 (1991), 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:

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

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

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, bounds on the time variables can also be strengthened as follows:

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

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

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

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

$$Q_i \leq \min\{Q, Q + q_i\} - (Q - \max_{j \in N \setminus \{i\}} \{q_j\} - q_i) x_{0i} - \sum_{j \in N \setminus \{i\}} \max\{0, q_j\} x_{ij}. \tag{19}$$

A formulation with fewer variables can be obtained by replacing constraints (5)-(8) with *rounded capacity inequalities* (see, e.g., Naddef and Rinaldi, 2002) and *infeasible path elimination constraints* (see, e.g., Ascheuer et al., 2000a). For any subset $S \subseteq P \cup D$, let $r(S) = \lceil \sum_{i \in S} q_i \rceil$ denote the minimum number of times vehicles must enter and leave S in order to visit all nodes in the set. Denote by $\mathcal{R}$ the set of infeasible paths with respect to time windows, and for each path $R \in \mathcal{R}$, let $A(R) \subset A$ be the set of arcs in this path. With these definitions, the PDPTW can be reformulated as follows:

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

subject to

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

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

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

$$\sum_{i,j \in S} x_{ij} \leq |S| - r(S) \qquad \forall S \subseteq N \setminus \{0, 2n+1\}, |S| \geq 2 \tag{24}$$

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

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

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., Ascheuer et al., 2000a) 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 R is infeasible because of time windows or ride time constraints (and the triangle inequality holds), then the following inequality is valid (see Cordeau, 2005):

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

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

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

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 $Q_1, Q_2, \ldots, Q_m$ are available and let $Q = \max_{1 \leq i \leq m}\{Q_i\}$. One can then define m dummy requests $i = 1, \ldots, m$ 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). Finally, 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 $t_{n+j,2n+1}$.

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). Throughout the remainder of the paper, let $x(S) = \sum_{i,j \in S} x_{ij}$ and $x(S : T) = \sum_{i \in S} \sum_{j \in T} x_{ij}$, where $S, T \subseteq N$.

3.1 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 PDPTW, this inequality can be lifted in many different ways by taking into account the fact that for each user i, node i must be visited before node $n+i$. For any set $S \subseteq P \cup D$, let $\pi(S) = \{i \in P | n+i \in S\}$ and $\sigma(S) = \{n+i \in D | i \in S\}$ denote the sets of *predecessors* and *successors* of S, respectively. BALAS et al. (1995) 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:

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

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

As shown by CORDEAU (2005), the D_k^- and D_k^+ inequalities introduced by GRÖTSCHEL and PADBERG (1985) for the asymmetric TSP can also be lifted by taking precedence relationships into account. Let $S = \{i_1, i_2, \ldots, i_k\} \subseteq P \cup D$ be an ordered set of nodes with $k \geq 3$. The following inequalities are then valid for the PDPTW:

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

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

3.2 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 capacity of the vehicles visiting set S is reduced by the amount corresponding to 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:

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S) + q(U)}{Q} \right\rceil. \tag{34}$$

Proof. Because $q(S) > 0$ and $q(U) \geq 0$, at least $\lceil q(S)/Q \rceil$ paths must visit set S and at least $\lceil q(U)/Q \rceil$ paths must visit sets T and U. 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. Hence, the total number of paths that either leave set S or enter set T is greater than or equal to $\lceil (q(S) + q(U))/Q \rceil$. Taking care not to count twice the paths going directly from S to T, one obtains

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

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

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

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

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 (34).

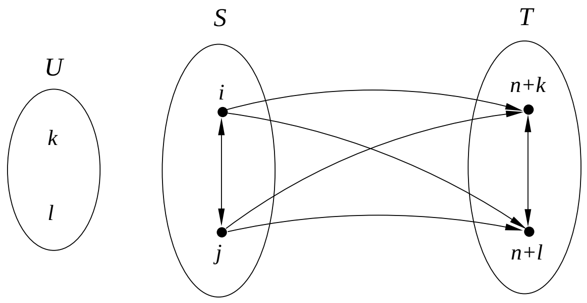

Figure 1: Strengthened capacity constraint where $S = \{i, j\}$, $T = \{n + k, n + l\}$ and $U = \pi(T) \setminus (S \cup T) = \{k, l\}$.

3.3 Generalized order constraints

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

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

Similar inequalities, called *precedence cycle breaking inequalities*, have also been proposed by Balas et al. (1995) for the PCATSP. In the case of a directed formulation, Cordeau (2005) showed that generalized order constraints can be lifted in two different ways as follows:

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

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

3.4 Strengthened infeasible path constraints

Paths that satisfy 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 $\phi_p \in \phi(\pi(R) \setminus R)$ and $\phi_d \in \phi(\sigma(R) \setminus R)$ 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 $S, T_1, \ldots, T_r \subset (P \cup D) \setminus R$ be subsets such that 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. The following inequality is then valid for the PDPTW:

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

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. 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.$\square$

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 $S_1, \ldots, S_r, T \subset (P \cup D) \setminus R$ be subsets such that 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. The following inequality is then valid for the PDPTW:

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

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.

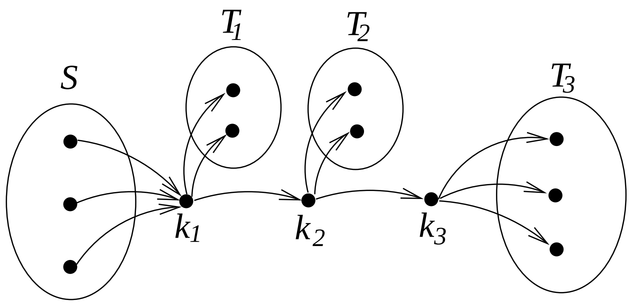

Figure 2: Outfork constraint with $r = 3$

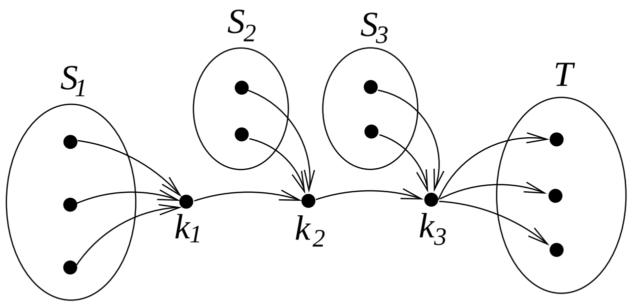

Figure 3: Infork constraint with $r = 3$

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 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 (2004) for the VRP with time windows:

$$x(\delta^-(S) \cap A_T^-) \geq |T| \tag{41}$$
$$x(\delta^+(S) \cap A_T^+) \geq |T|. \tag{42}$$

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 Algorithm

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 branch-and-bound 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 strenghten formulation (PDPTW1) although they are not required to ensure feasiblity.

Taking into account the precedence relationships, time windows and ride time constraints, several arc elimination rules can be used in a pre-processing 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. (1991) and Cordeau (2005).

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. Prior to solving the problem, an upper bound is computed by using either the adaptive large neighbourhood search algorithm of Ropke and Pisinger (2004) for the PDPTW or the tabu search heuristic of Cordeau and Laporte (2003) for the DARP.

We now 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

Violated precedence constraints (4) and (23) can be identified in polynomial time by solving a series of maximum flow problems: for each request $i \in P$, 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. (1990).

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 heuristic introduced by CORDEAU (2005). 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 Infeasible path constraints

To identify infeasible paths violating constraints (25), we use an enumerative procedure. 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.

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.

4.4 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 $j \in N \setminus \{i\}$. 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 (2005).

4.5 Strengthened capacity constraints

To identify sets S and T for which the strengthened capacity constraint is violated, we use 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 calculates

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

and

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

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 (34). 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.

In the root node of the branch and bound search we use a randomized version of this heuristic where noise 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 Generalized order constraints

We use two simple heuristics for the lifted generalized order constraints (36) and (37). 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 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. 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 (37).

4.7 Fork constraints

A partial enumeration procedure is used for fork constraints with $r = 1$. This procedure first enumerates the set H of all infeasible paths containing three nodes. To identify violated outfork constraints, it 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. To identify violated infork constraints, a similar procedure 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 $(k_0, k_1, \ldots, k_l)$ 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 $(k_0, k_1, \ldots, k_l, j)$ 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. In order to keep running times low, only paths containing at most six nodes are considered. Checking if 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 only checked 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 in this path. 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. For 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 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 branch-and-cut algorithms were implemented in C++ by using ILOG Concert 1.3 and CPLEX 9.0. All experiments were performed on a 2.5 GHz Pentium 4 computer with 512MB of memory.

Several sets of instances for the PDPTW and the DARP were used for testing. We first generated some PDPTW instances as suggested by Savelsbergh and Sol (1998). In these instances, the coordinates of each pickup and delivery location are chosen randomly according to a uniform distribution over the $[0, 200] \times [0, 200]$ square. The load q_i of request i is selected randomly from the interval $[5, Q]$, where Q is the vehicle capacity. A planning horizon of length $T = 600$ 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 $[0, T - t_{i,n+i}]$ 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 the following table.

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

In the test instances generated by Savelsbergh and Sol (1998), each vehicle has a different depot whose location is also chosen randomly over the $[0, 200] \times [0, 200]$ square. Because our formulations cannot handle multiple depots directly, we have instead used a single depot located at the middle of the square.

As is apparent from the results reported by Savelsbergh and Sol (1998), using the $[0, 200] \times [0, 200]$ square with $T = 600$ 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.

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 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 2. 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 72.40 for (PDPTW2) compared to 70.52 for (PDPTW1).

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) when considering all families of

Table 2: Lower bounds obtained in the root node as a percentage of the upper bound

	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	83.38	83.35	83.93	83.83	84.00	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	88.50	93.85	99.91	99.96	82,185.31
A60	83.70	92.19	92.19	92.20	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	93.94	93.97	95.60	112,458.28
A75	57.29	70.27	70.27	70.30	70.27	78.46	99.89	99.97	92,528.54
B30	85.21	84.36	84.36	84.36	84.36	100.00	99.99	100.00	51,193.62
B35	67.35	70.74	70.74	83.67	70.74	89.24	91.92	100.00	61,400.07
B40	64.97	65.44	65.45	65.48	65.45	83.58	80.84	85.77	51,421.35
B45	67.29	67.51	67.51	69.74	67.51	99.96	99.92	99.97	61,787.28
B50	51.32	66.52	66.53	66.53	66.51	88.12	99.95	99.98	71,889.75
B55	63.37	58.89	58.88	59.84	58.59	99.99	99.97	100.00	82,080.73
B60	80.37	80.43	80.43	80.43	80.43	91.48	99.99	100.00	102,323.77
B65	85.87	75.38	75.38	75.40	75.38	95.89	99.88	99.92	82,623.98
B70	61.03	67.32	67.32	67.34	67.32	94.58	99.92	99.96	92,641.67
B75	56.32	58.67	60.10	60.12	60.10	85.72	89.23	89.36	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	81.14	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	75.62	75.57	75.59	75.60	75.49	87.77	99.96	100.00	81,405.96
C50	99.40	99.52	99.52	99.54	99.52	99.93	99.87	99.94	61,933.09
C55	67.04	67.20	67.21	67.23	67.21	91.91	99.81	99.92	61,930.55
C60	57.85	68.07	68.07	68.10	68.07	99.86	99.79	99.89	72,104.00
C65	53.96	54.84	54.84	54.85	54.83	77.14	99.70	99.80	82,326.62
C70	56.35	56.47	56.48	56.49	56.48	85.42	89.11	89.23	92,613.68
C75	56.17	67.04	67.04	67.06	67.03	78.34	99.71	99.83	92,711.74
D30	64.68	67.15	67.16	67.18	67.15	89.23	99.95	99.99	61,040.10
D35	46.34	47.19	47.19	47.21	47.20	58.04	99.86	99.93	71,308.04
D40	67.00	67.12	67.12	67.15	67.11	99.87	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	57.96	57.96	58.00	57.96	86.08	99.81	99.91	72,051.95
D60	75.36	75.40	75.40	75.41	75.40	99.98	99.91	99.98	82,306.47
D65	49.05	38.73	38.72	38.77	38.73	93.80	99.72	99.86	82,200.77
D70	55.52	51.12	51.13	51.15	51.13	78.82	99.73	99.84	82,631.56
D75	38.78	37.52	37.52	37.54	37.52	64.18	99.62	99.77	92,970.84
Avg.	70.52	72.40	72.44	72.92	72.43	90.33	97.73	98.15	

valid inequalities. Again, all 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., B55), 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 3, we report the results obtained by considering both formulations with all types of valid inequalities. For each instance that was solved to optimality, we indicate the CPU time (in minutes) needed to prove optimality, the number of nodes explored in the branch-and-bound tree and the total number of cuts generated during the search. When an instance could not be solved to optimality within the maximum CPU time (two hours), we report the value of 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 sligtly better performance: it solved three 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.

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 Cordeau (2005) and are available on the following web site: `http://www.hec.ca/chairedistributique/data/darp`. Their main characteristics are summarized in Table 4. In this table, columns $|K|$ and T 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.

Tables 5 and 6 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 2. This time, however, we also indicate in column LP0 the lower bound obtained with the three-index formulation of Cordeau (2005). 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) do much better than the three-index formulation in terms of the initial lower bound.

Finally, Tables 7 and 8 report the computational statistics collected when solving each instance to optimality with both (PDPTW1) and (PDPTW2). In column (DARP), we also indicate comparable statistics for the three-index DARP fomulation of Cordeau (2005). For the latter formulation, only a small subset of all instances could be solved to optimality. As in Table 3, 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 935.18 minutes compared to 139.48 minutes 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 almost three hours of

Table 3: Computational results for first set of PDPTW instances

		(PDPTW1)				(PDPTW2)			
Instance	U. Bound	Time	Nodes	Cuts	L. Bound	Time	Nodes	Cuts	L. Bound
A30	51,317.40	0.11	0	119		0.11	0	128	
A35	51,343.53	0.21	0	483		0.19	0	593	
A40	61,609.44	0.32	0	446		0.29	0	468	
A45	61,693.01				51,837.31				51,859.47
A50	71,932.03	0.89	0	989		0.94	0	1088	
A55	82,185.31	27.40	242	3457		21.98	165	4072	
A60	92,366.70	1.57	0	911		1.48	0	1018	
A65	82,331.12	5.47	5	1367		6.11	4	1441	
A70	112,458.28	35.16	16	2687		22.36	8	2522	
A75	92,528.54				92,501.89				92,512.92
B30	51,193.62	0.22	0	524		0.21	2	584	
B35	61,400.07	0.38	2	535		0.33	0	713	
B40	51,421.35	1.70	20	1357		1.95	64	1719	
B45	61,787.28	1.93	60	1407		1.68	26	1617	
B50	71,889.75	3.89	12	2295		3.12	6	2378	
B55	82,080.73	2.41	6	1462		1.51	0	1535	
B60	102,323.77	4.38	0	1571		4.61	0	1983	
B65	82,623.98				82,562.35				82,572.12
B70	92,641.67				92,615.75	71.27	149	5230	
B75	92,476.30				82,643.20				82,633.80
C30	51,145.18	0.12	0	120		0.11	0	120	
C35	51,235.64	0.78	8	1242		0.63	5	1172	
C40	61,473.91				51,585.73				51,580.23
C45	81,405.96	1.87	4	1757		1.64	4	2140	
C50	61,933.09	38.05	688	5046		22.93	425	4741	
C55	61,930.55				61,900.79	38.21	163	5689	
C60	72,104.00				72,037.43				72,070.57
C65	82,326.62				82,165.14				82,161.87
C70	92,613.68				82,644.23				82,642.27
C75	92,711.74				92,554.03				92,553.66
D30	61,040.10	0.52	5	969		0.47	11	1088	
D35	71,308.04				71,297.22	47.24	2490	8016	
D40	61,531.68				61,473.86				61,496.58
D45	81,601.52	2.98	22	1177		2.97	29	1359	
D50	71,761.23	7.58	90	2184		4.52	23	2330	
D55	72,051.95				71,988.19				72,025.41
D60	82,306.47	12.74	176	2165		8.68	45	2341	
D65	82,200.77				82,091.47				82,120.80
D70	82,631.56				82,496.68				82,496.12
D75	92,970.84				92,764.15				92,756.57

Table 4: Characteristics of DARP instances

Instance	$\lvert K\rvert$	n	T	Q	L	Instance	$\lvert K\rvert$	n	T	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

CPU time with the DARP formulation (and 126,332 branch-and-bound nodes) while it was solved in the root node with both (PDPTW1) and (PDPTW2). This dramatic improvement results from the improved 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: strenghtened capacity constraints and fork constraints. Computational experiments performed on PDPTW and DARP instances show that both formulations are competitive although the more compact one (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 able to solve much larger instances. The largest instance solved to optimality contains 192 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.

Table 5: Impact of valid inequalities for first set of 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	95.49	98.81	99.03	98.81	98.81	99.83	99.51	99.83	431.12
a3-24	81.07	88.59	95.62	95.63	95.63	95.62	100.00	99.92	100.00	344.83
a3-30	79.16	88.83	94.55	94.55	94.55	94.55	100.00	100.00	100.00	494.85
a3-36	87.67	92.21	96.84	96.84	96.84	96.84	99.28	98.92	99.28	583.19
a4-32	78.12	85.68	92.23	92.22	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	99.60	99.14	99.71	557.69
a4-48	64.63	85.83	91.96	91.96	91.97	91.96	99.74	98.70	99.83	668.82
a5-40	65.25	84.88	93.10	93.16	93.10	93.10	100.00	99.15	100.00	498.41
a5-50	59.92	78.87	88.40	88.44	88.41	88.40	98.92	97.71	99.10	686.62
a5-60	59.68	75.39	87.18	87.23	87.28	87.18	99.81	98.03	99.85	808.42
a6-48	63.36	79.69	87.98	88.03	87.99	87.96	100.00	98.45	100.00	606.06
a6-60	60.11	75.74	86.22	86.25	86.27	86.22	99.67	98.97	99.90	819.25
a6-72	65.42	79.88	89.08	89.27	89.22	89.09	99.31	98.18	99.43	916.05
a7-56	64.08	81.07	88.12	88.27	88.15	88.11	99.33	98.15	99.52	724.04
a7-70	62.49	77.60	85.51	85.53	85.64	85.51	99.84	98.62	99.93	891.55
a7-84	55.81	71.45	82.28	82.35	82.54	82.28	99.39	97.65	99.42	1033.37
a8-64	66.86	74.63	85.40	85.65	85.40	85.41	99.46	98.13	99.72	747.46
a8-80	58.29	69.87	81.10	81.10	81.19	81.10	99.20	96.29	99.33	945.73
a8-96	53.75	66.71	78.45	78.49	78.56	78.45	98.22	94.89	98.69	1234.46
Avg.	70.54	82.81	90.37	90.43	90.42	90.37	99.60	98.59	99.69	

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada under grants 227837-04 and OGP0039682. This support is gratefully acknowledged.

References

N. Ascheuer, M. Fischetti and M. Grötschel. "A Polyhedral Study of the Asymmetric Traveling Salesman Problem with Time Windows." *Networks*, **36**:69–79 (2000a).

N. Ascheuer, M. Jünger and G. Reinelt. "A Branch & Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints." *Computational Optimization and Applications*, **17**:61–84 (2000b).

E. Balas, M. Fischetti and W.R. Pulleyblank. "The Precedence-Constrained Asymmetric Traveling Salesman Polytope." *Mathematical Programming*, **68**:241–265 (1995).

J.-F. Cordeau. "A Branch-and-Cut Algorithm for the Dial-a-Ride Problem." *Operations Research* (2005). Forthcoming.

Table 6: Impact of valid inequalities for second set of DARP instances

	LP0	LP1	LP2	SEC	SCC	GOC	FC	RC	Full	U. Bound
b2-16	91.76	98.89	99.52	99.53	99.52	99.52	99.71	99.53	99.71	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	95.06	95.22	95.06	95.08	99.38	98.65	99.41	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	99.22	99.22	99.22	99.22	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	98.15	98.15	98.15	98.15	100.00	99.55	100.00	656.63
b4-48	81.75	88.92	95.84	95.84	95.93	95.76	99.81	98.86	99.82	673.81
b5-40	70.43	83.16	91.63	91.80	92.04	91.76	99.69	97.20	99.90	613.72
b5-50	70.78	88.19	92.44	92.44	92.96	92.40	99.47	98.54	99.53	761.40
b5-60	70.75	86.55	91.49	91.52	91.98	91.51	99.01	97.55	99.23	902.04
b6-48	75.90	95.06	98.82	98.82	99.07	98.79	100.00	99.88	100.00	714.83
b6-60	68.75	88.49	94.75	94.74	95.29	94.70	100.00	99.65	100.00	860.07
b6-72	69.99	83.97	90.41	90.36	90.78	90.41	98.09	96.90	98.60	978.47
b7-56	64.54	85.32	90.68	90.66	91.36	90.68	98.00	96.23	98.28	823.97
b7-70	68.25	89.43	94.17	94.24	94.41	94.16	99.53	98.27	99.58	912.62
b7-84	61.59	77.55	86.36	86.47	87.15	86.36	99.21	96.99	99.36	1203.37
b8-64	66.98	84.62	90.72	90.81	91.04	90.85	99.18	98.16	99.50	839.89
b8-80	65.85	88.46	92.19	92.25	92.38	92.19	99.83	98.86	99.84	1036.34
b8-96	59.73	77.36	84.75	84.74	85.32	84.79	97.91	94.45	98.64	1185.55
Avg.	76.50	90.26	94.54	94.57	94.81	94.55	99.47	98.53	99.59	

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 (2003).

J.-F. Cordeau, G. Laporte, J.-Y. Potvin and M.W.P. Savelsbergh. "Transportation on demand." In C. Barnhart and G. Laporte, editors, *Transportation*, Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2005. Forthcoming.

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

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).

Table 7: Results on 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	4	46	0.02	0	46	0.02	0	79
a2-20	344.83	0.08	181	93	0.04	0	119	0.04	0	160
a2-24	431.12	0.27	444	140	0.09	2	218	0.09	2	309
a3-24	344.83	4.17	6863	324	0.08	0	291	0.08	0	315
a3-30	494.85	42.39	43632	425	0.15	0	338	0.15	0	509
a3-36	583.19	12.46	7451	423	0.33	7	348	0.32	7	562
a4-32	485.50				0.22	0	577	0.18	0	715
a4-40	557.69				0.71	8	802	0.59	12	965
a4-48	668.82				1.35	4	1306	1.05	4	1829
a5-40	498.41				0.43	0	785	0.34	0	887
a5-50	686.62				2.38	71	1658	1.34	42	1708
a5-60	808.42				4.62	57	2081	2.03	24	2080
a6-48	606.06				1.16	0	1553	0.70	0	1841
a6-60	819.25				3.60	10	1983	1.94	3	2494
a6-72	916.05				8.11	73	2492	4.00	13	3105
a7-56	724.04				4.15	69	1978	2.01	30	2253
a7-70	891.55				7.25	4	2796	3.88	10	3520
a7-84	1033.37				22.17	105	3538	7.64	41	3556
a8-64	747.46				7.02	68	2608	3.08	14	2655
a8-80	945.73				35.94	314	3925	9.32	101	4024
a8-96	1234.46				935.18	5981	8585	139.48	3899	7486

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. 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, 1985.

H. Li and A. Lim. "A Metaheuristic for the Pickup and Delivery Problem with Time Windows." *International Journal on Artificial Intelligence Tools*, **12**:173–186 (2003).

Q. Lu and M. Dessouky. "An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem." *Transportation Science*, **38**:503–514 (2004).

J. Lysgaard. "Reachability Cuts for the Vehicle Routing Problem with Time Windows." Technical Report L-2004-01, Aarhus School of Business, Denmark, 2004.

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.

Table 8: Results on second set of DARP instances

		(DARP)			(PDPTW1)			(PDPTW2)		
Instance	Cost	Time	Nodes	Cuts	Time	Nodes	Cuts	Time	Nodes	Cuts
b2-16	309.41	0.15	487	130	0.02	3	128	0.03	2	164
b2-20	332.64	0.01	2	30	0.03	0	20	0.02	0	74
b2-24	444.71	0.13	146	103	0.07	1	156	0.07	1	195
b3-24	394.51	3.62	8147	252	0.07	1	182	0.07	4	283
b3-30	531.44	6.82	9862	274	0.10	0	194	0.09	0	292
b3-36	603.79	62.07	66079	291	0.17	0	161	0.16	0	261
b4-32	494.82	176.82	126332	375	0.11	0	154	0.10	0	201
b4-40	656.63				0.29	0	370	0.27	0	559
b4-48	673.81				0.80	6	837	0.73	6	1017
b5-40	613.72				0.70	4	1056	0.53	4	1260
b5-50	761.40				1.18	8	1168	0.86	7	1219
b5-60	902.04				4.26	207	1634	2.40	64	1879
b6-48	714.83				0.48	0	586	0.45	0	625
b6-60	860.07				1.16	0	1097	1.04	0	1271
b6-72	978.47				194.05	6162	5368	14.40	477	3824
b7-56	823.97				105.91	3782	8100	13.92	746	4999
b7-70	912.62				5.00	69	1599	3.07	15	1784
b7-84	1203.37				26.83	190	3819	11.74	56	3784
b8-64	839.89				11.41	336	2415	3.93	52	2136
b8-80	1036.34				7.27	93	1681	4.23	12	2032
b8-96	1185.55				603.64	4311	8979	81.06	1823	7017

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).

S. Ropke and D. Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." Technical Report 2004-13, DIKU, University of Copenhagen, 2004.

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