# A Branch-and-Cut Algorithm for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading

Jean-François Cordeau* Manuel Iori† Gilbert Laporte*
Juan José Salazar González‡

August 13, 2007

**Abstract**

In the *Traveling Salesman Problem with Pickup and Delivery* (TSPPD) a single vehicle must serve a set of customer requests, each defined by an origin location where a load must be picked up, and a destination location where the load must be delivered. The problem consists of determining a shortest Hamiltonian cycle through all locations while ensuring that the pickup of each request is performed before the corresponding delivery. This paper addresses a variant of the TSPPD in which pickups and deliveries must be performed according to a Last-In First-Out (LIFO) policy. We propose three mathematical formulations for this problem and several families of valid inequalities which are used within a branch-and-cut algorithm. Computational results performed on test instances from the literature show that most instances with up to 17 requests can be solved in less than 10 minutes, while the largest instance solved contains 25 requests.

Keywords: traveling salesman problem, pickup and delivery, LIFO, branch-and-cut.

## 1 Introduction

In the *Traveling Salesman Problem with Pickup and Delivery* (TSPPD) a single vehicle must serve a set of customer requests, each defined by an origin location where a load must be picked up, and a destination location where the load must be delivered. The problem consists of determining a shortest Hamiltonian cycle through all locations while ensuring that the pickup of any given request is performed before the corresponding delivery. The vehicle may be capacitated or uncapacitated. The TSPPD has several practical applications in freight and passenger transportation. It arises, for example, in urban courier service operations,

*CIRRELT, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada.
†DISMI, Università di Modena e Reggio Emilia, Via Amendola 2, 42100 Reggio Emilia, Italy.
‡DEIOC, Universidad de La Laguna, 38271 La Laguna, Tenerife, Spain.

in less-than-truckload transportation, and in door-to-door transportation services for the elderly and the disabled (see, e.g., Cordeau et al. [7]).

We address a variant of the TSPPD in which pickups and deliveries must be performed in Last-In First-Out (LIFO) order. Under this condition, a load being picked up is always placed at the rear of the vehicle, while a delivery can be performed only if the associated load is currently at the rear. This problem is referred to as the *TSPPD with LIFO Loading* (TSPPDL).

The TSPPDL arises naturally in the routing of vehicles that have a single access point for the loading and unloading of freight. Avoiding load rearrangements is also particularly important in the case of rear-loading vehicles transporting large, heavy or fragile items, or hazardous materials. Another application arising in industrial contexts is the routing of automated guided vehicles (AGVs) which typically use a stack to move items between workstations.

Vehicle routing problems with pickup and delivery have been studied extensively. For general surveys, we refer the reader to Cordeau et al. [7], Desaulniers et al. [8] and Savelsbergh and Sol [28]. One of the most studied pickup and delivery problems is the TSPPD which, because of its difficulty, has been solved mainly by means of heuristics. Notable exceptions are the branch-and-bound approach of Kalantari et al. [17] and the branch-and-cut algorithms of Ruland and Rodin [27] and Dumitrescu et al. [10].

The TSPPDL has, however, received far less attention. To the best of our knowledge, the first mention of this problem was made by Ladany and Mehrez [18] who studied a real-life delivery problem in Israel. These authors have provided a description of the problem but no mathematical formulation. A similar problem was later investigated by Pacheco [25, 26] who proposed a heuristic algorithm with Or-opt exchanges for the uncapacitated TSPPDL, and reported results on randomly generated instances involving up to 120 requests. The TSPPDL was also recently studied by Cassani [5] who developed greedy heuristics and a variable neighborhood descent algorithm combining four types of exchanges. Results were presented on instances with up to 100 customers. Finally, new exchange operators and a variable neighborhood search heuristic were described by Carrabs et al. [4] who reported results on instances with up to 375 requests.

Three exact algorithms have been proposed for the solution of the TSPPDL. All are branch-and-bound methods that rely on relaxations of the TSP. The first one was developed by Pacheco [23, 24] and is an adaptation of the TSPPD branch-and-bound approach of Kalantari et al. [17] which is itself based on the branch-and-bound algorithm of Little et al. [22] for the TSP. The second one was proposed by Cassani [5] and uses lower bounds based on minimum spanning tree and assignment problem relaxations. The largest instances solved by these methods contain 11 requests. Very recently, another branch-and-bound algorithm was introduced by Carrabs et al. [3]. This algorithm uses additive lower bounds based on assignment problem and shortest spanning $r$-arborescence problem relaxations. Several filters are also applied at each node of the enumeration tree to eliminate arcs that cannot belong to feasible solutions. This algorithm is capable of solving most instances with 15 requests and some instances with up to 21 requests.

A related stream of research has addressed the scheduling of the execution of computer programs, using LIFO stack storage structures. Volchenkov [29] showed that this problem can be reduced to finding a planar layout in a given oriented graph. He proposed an algorithm based on the enumeration of normal covering trees. This approach was later improved by Levitin [20] through considerations on the permutation structures. Levitin and Abezgaouz [21] pursued the work of Volchenkov [29] and Levitin [20] in a different context: the routing of multiple-load AGVs for the distribution of raw materials or semi-manufactured products to workstations in industrial plants. They considered the case of a single AGV operating in a LIFO fashion and having infinite capacity. They formulated the conditions for the existence of routes in which each workstation is visited once and the LIFO constraint is satisfied. Finally, they proposed an algorithm for finding the shortest route and reported computational results on randomly generated instances involving up to 50 requests.

A form of LIFO constraints has also been considered in the context of the capacitated vehicle routing problem by Iori et al. [16] who studied the case where customer demands consist of lots of two-dimensional weighted items. In this case, one must ensure the existence for each vehicle route of a feasible placement of the items satisfying the last-in-first-out policy. These authors have proposed a branch-and-cut algorithm in which the loading phase is solved through a nested branch-and-bound search. Tabu search algorithms for this problem were also described by Gendreau et al. [12, 13]. Finally, LIFO constraints were considered by Xu et al. [30] in a practical pickup and delivery problem involving multiple vehicles, time windows, and capacity constraints.

The remainder of the paper is organized as follows. The next section describes the TSPPDL formally and investigates its underlying structure. Section 3 introduces three mathematical formulations for the problem. The first one is derived from the classical two-index model for the TSP, with additional inequalities imposing the LIFO constraints. The next two formulations use different approaches which lead to stronger linear programming relaxations. Valid inequalities that strengthen these formulations are then described in Section 4. These inequalities are then used within a branch-and-cut algorithm which is described in Section 5 and whose computational performance is studied in Section 6.

# 2 Problem Description

Let $n$ denote the number of requests to be satisfied. The TSPPDL can be defined on a complete 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. Each request $i$ is associated with a pickup node $i$ and a delivery node $n + i$. Each arc $(i, j) \in A$ has a routing cost $c_{ij}$.

The TSPPDL consists of finding a minimum cost route starting from the origin depot 0, visiting every node in $P \cup D$ exactly once, and finishing at the destination depot $2n + 1$. In addition, this route must satisfy precedence and LIFO constraints. Precedence constraints require that for any given request the pickup node of the request must be visited before

the delivery node. LIFO constraints may be defined formally by considering a stack and enforcing the following discipline:

a) when visiting a pickup node, the load being picked up is placed at the top of the stack;

b) visiting a delivery node is possible only if the load to be delivered is located at the top of the stack.

The TSPPDL is clearly $\mathcal{NP}$-hard since the well-known *Asymmetric TSP* (ATSP) can be reduced to the TSPPDL by associating each ATSP customer $i$ with two nodes $i$ and $n+i$ and using a modified cost matrix $(d_{ij})$, where $d_{i,n+i} = 0, d_{n+i,i} = \infty$, for all $i \in P$, and $d_{ij} = d_{i,n+j} = d_{n+i,n+j} = \infty$, $d_{n+i,j} = c_{ij}$, for all $i, j \in P, i \neq j$.

As in the work of Pacheco [23, 24, 25, 26], Cassani [5] and Carrabs et al. [4], we focus here on the uncapacitated case.

## 2.1 The structure of the TSPPDL

To investigate the structure of the problem, we first introduce the following definitions: a PP arc is an arc $(i, j)$ connecting two pickup nodes; a PD arc is an arc $(i, n+i)$ connecting a pickup node to a delivery node; a DP arc is an arc $(n+i, j)$ connecting a delivery node to a pickup node; a DD arc is an arc $(n+i, n+j)$ connecting two delivery nodes. With respect to PD arcs, one can first observe that no arc of the form $(i, n+j)$ with $j \neq i$ can be used in a feasible solution. Moreover, for the connectivity of the solution, at least one PD arc must be used.

Now consider the number of PD arcs in a feasible solution to the TSPPDL. The case where *exactly* $n$ PD arcs are used is known as the *Full-Truck-Load Pickup and Delivery Problem* (see, e.g., Savelsbergh and Sol [28]) and can be reduced to an ATSP in which each node represents a pickup and delivery pair. We now focus on the case where *exactly one* PD arc is used. In this situation the problem reduces to an ATSP on a graph with $n$ nodes. Indeed, all pickups are performed before any of the deliveries, and the delivery nodes are visited in the opposite order from the pickup nodes. The sequence of customers along the route can then be read as a *palindrome*, i.e., the pickup and the delivery nodes form a mirror structure with respect to the PD arc. This particular structure generalizes to the case where more than one PD arc is used. In the general case the route will form a sequence of *nested palindromes* (see Figure 1) such that removing each nested palindrome yields a simple palindrome structure.

As mentioned in the introduction, the structure of graphs satisfying LIFO constraints was first investigated by Volchenkov [29] for the problem of managing stack structures in computers. Volchenkov showed that finding a feasible ordering of write (i.e., pickup) and read (i.e., delivery) operations on a stack is equivalent to finding a planar representation of a graph in which an edge is inserted between each pickup and delivery pair (see Figure 2). This consideration directly applies to our problem: placing a pickup node $j$ (resp. delivery node $n+j$) between a node pair $i, n+i$, without placing the corresponding delivery node

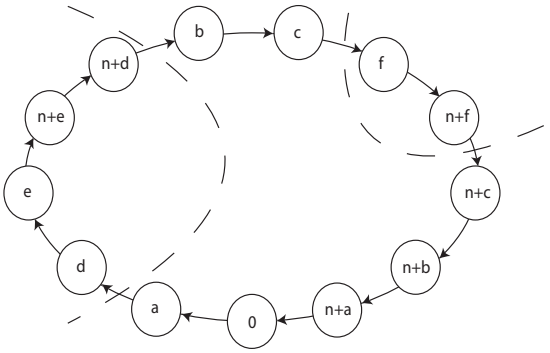


Figure 1: *A sequence of nested palindromes.*

$n + j$ (resp. pickup node $j$) yields a violation of the LIFO constraint when trying to unload the demand of request $i$ (resp. request $j$). This is equivalent to having two crossing edges in the graph of Figure 2.

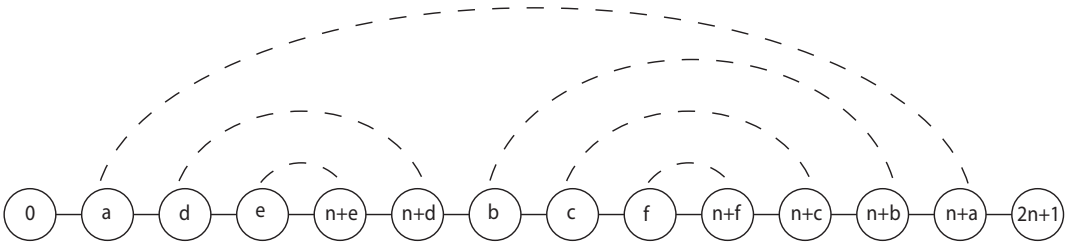


Figure 2: *Planar graph guaranteeing the LIFO constraints.*

## 2.2 The number of feasible routes

Ruland and Rodin [27] have shown that for the TSPPD with $n$ requests, the number $K_n$ of feasible routes is given by the following recursion:

$$K_1 = 1 \tag{1}$$
$$K_n = n(2n-1)K_{n-1} \qquad (n > 1). \tag{2}$$

For the TSPPDL one can also determine the number of feasible routes through a recursive formula. To this end, we first define an *n-request feasible route structure* as a sequence containing $n$ pickups and $n$ deliveries in an order compatible with the LIFO constraints. For example, with $n = 2$ only the following two route structures are feasible: (P,D,P,D) and (P,P,D,D). For each $n$-request feasible route structure, there exist $n!$ different TSPPDL routes obtained by permuting the pickup nodes. Denote by $f(n, m)$ the number of $n$-request

feasible route structures starting with *exactly* $m$ pickups. Then the total number of feasible routes with $n$ requests and starting with exactly $m$ pickup nodes is $n!f(n,m)$. Consequently, the number of feasible TSPPDL solutions with $n$ requests is

$$L_n = n! \sum_{m=1}^{n} f(n,m), \tag{3}$$

where the value of $f(n,m)$ can be computed recursively as shown by the following result.

**Proposition 1.** The value of $f(n,m)$ is given by the following recursion:

$$f(1,1) = 1 \tag{4}$$

$$f(n,m) = \sum_{\ell=m-1}^{n-1} f(n-1,\ell) \qquad (m=1,\ldots,n), \tag{5}$$

where $f(n,0)=0$.

**Proof.** It is easy to see that $f(1,1)=1$. Suppose that the route starts with $m$ pickups and then performs a delivery. The request just completed can be removed from further consideration. The vehicle then contains the first $m-1$ pickups and there are two possible ways to extend the route:

a) Perform a delivery: the number of ways to extend the route in this way is $f(n-1,m-1)$ because $m-1$ pickups have been performed and the corresponding deliveries have not yet been performed.

b) Perform one or more pickups. The number of ways to extend the route in this way is $\sum_{\ell=m}^{n-1} f(n-1,\ell)$.

Hence

$$f(n,m) = f(n-1,m-1) + \sum_{\ell=m}^{n-1} f(n-1,\ell) = \sum_{\ell=m-1}^{n-1} f(n-1,\ell). \ \square$$

Table 1 reports the number of feasible ATSP, TSPPD and TSPPDL solutions for different values of $n$. For the ATSP, we indicate the number of feasible solutions in an instance with $2n+1$ nodes (including the depot). Carrabs et al. [3] have proposed a different (but equivalent) recursion for computing the number of feasible routes in the TSPPDL.

# 3 Mathematical Formulations

In this section we first recall a classical formulation for the TSPPD. We then introduce three different formulations for the (uncapacitated) TSPPDL.

To formulate the TSPPD, we associate to each arc $(i,j) \in A$ a binary variable $x_{ij}$ taking value 1 if and only if node $j$ is visited immediately after node $i$. As usual, we define

Table 1: The number of feasible solutions with $n$ requests

| $n$ | ATSP | TSPPD | TSPPDL |
|---|---|---|---|
| 1 | 2 | 1 | 1 |
| 2 | 24 | 6 | 4 |
| 3 | 720 | 90 | 30 |
| 4 | 40,320 | 2,520 | 336 |
| 5 | 3,628,800 | 113,400 | 5,040 |
| 6 | 479,001,600 | 7,484,400 | 95,040 |
| 7 | 87,178,291,200 | 681,080,400 | 2,162,160 |
| 8 | 20,922,789,888,000 | 81,729,648,000 | 57,657,600 |

$\bar{S} = N \setminus S$, $x(S) = \sum_{i,j \in S} x_{ij}$, $x(\delta^+(S)) = \sum_{i \in S, j \notin S} x_{ij}$ and $x(\delta^-(S)) = \sum_{i \notin S, j \in S} x_{ij}$. For any node $i \in N$, let also $x(i, S) = \sum_{j \in S} x_{ij}$ and $x(S, i) = \sum_{j \in S} x_{ji}$. We also define the collection $\mathcal{S}$ of all node subsets $S \subset N$ such that $0 \in S, 2n+1 \notin S$ and there exists a node $i$ such that $i \notin S$ and $n + i \in S$.

Using this notation, the TSPPD can then be formulated as the following integer program:

$$\text{(TSPPD)} \qquad \text{Minimize} \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{6}$$

subject to

$$x(\delta^+(i)) = 1 \qquad \forall i \in P \cup D \cup \{0\} \tag{7}$$

$$x(\delta^-(i)) = 1 \qquad \forall i \in P \cup D \cup \{2n+1\} \tag{8}$$

$$x(S) \leq |S| - 1 \qquad \forall S \subseteq P \cup D, |S| \geq 2 \tag{9}$$

$$x(S) \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A. \tag{11}$$

The objective function (6) minimizes the total routing cost. Constraints (7) and (8) ensure that each pickup and delivery node is visited exactly once. Constraints (9) ensure the connectivity of the solution while constraints (10) impose the precedence relationships between pickups and deliveries. Precedence constraints (10) were introduced by Balas et al. [2] in the context of the TSP with precedence constraints, and by Ruland and Rodin [27] in the context of the TSPPD. Since the TSPPDL is a restriction of the TSPPD, these constraints are also valid for the former problem.

The TSPPDL can be formulated by introducing additional sets of constraints in model (6)-(11). We now describe three such sets of constraints, giving rise to as many different formulations. The first two formulations require a polynomial number of additional variables and constraints while the third one requires an exponential number of constraints but no new variable.

To describe the first and second formulations (which can handle both the capacitated and uncapacitated versions of the problem), we associate each node $i \in N$ with a demand parameter $q_i \neq 0$, such that $q_i = -q_{n+i}$ for $i = 1, \ldots, n$. The demand $q_i$ is positive for every pickup node $i$ and is equal to $-q_i$ for the corresponding delivery node $n + i$. If the problem is uncapacitated, one may simply set $q_i = 1, \forall i \in P$ and $q_i = -1, \forall i \in D$. We assume $q_0 = q_{2n+1} = 0$. To each node $i \in N$, we also associate a continuous variable $Q_i$ representing the load of the vehicle upon its departure from node $i$. Finally, we denote the vehicle capacity by $Q$. Again, if the TSPPDL is uncapacitated, one may simply set $Q = n$. Given this notation, LIFO constraints may be imposed through the following sets of constraints:

$$Q_j \geq (Q_i + q_j)x_{ij} \qquad \forall (i,j) \in A \tag{12}$$

$$Q_{n+i} = Q_i - q_i \qquad \forall i \in P \tag{13}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \qquad \forall i \in N. \tag{14}$$

We denote model (6)-(14) by (TSPPDL1). The consistency of the load variables $Q_i$ is ensured through constraints (12), while constraints (13) enforce the LIFO policy. Finally vehicle capacity is imposed through constraints (14). The following result demonstrates that constraints (7)-(14) properly define the LIFO policy.

**Proposition 2.** Constraints (10)-(13) are satisfied if and only if the loading and unloading operations satisfy the LIFO policy.

**Proof.** Consider first a solution in which loading and unloading satisfy the LIFO constraints. It is clear that in such a solution, the load of the vehicle immediately after visiting node $n + i$ is equal to $Q_i - q_i$. Indeed, if a pickup node $j$ is visited between nodes $i$ and $n + i$, then the corresponding delivery node must also be visited, and vice-versa. As a result, the load of the vehicle upon arrival at node $n + i$ is equal to its load upon departure from node $i$ and hence (13).

Now consider a solution satisfying constraints (12)-(14). In this case, the net amount delivered between each pair $i, n + i$ is equal to 0. Suppose that this solution does not satisfy the LIFO constraints. Then, one can find a node pair $i, n + i$ such that at least one pickup node $j$ is visited between $i$ and $n+i$ but not the delivery node $n+j$. Let $i', n+i'$ denote the first node pair for which this condition is satisfied in the solution and let $\tilde{P} \subset P$ represent the set of pickup nodes for which the delivery node is not visited between $i'$ and $n + i'$. Define $q(\tilde{P}) = \sum_{j \in \tilde{P}} q_j$. Because the precedence constraints are satisfied, one cannot find a delivery node $n+j$ between $i'$ and $n+i'$ if the corresponding pickup node $j$ is not also visited between $i'$ and $n + i'$. As a result, it follows that $Q_{n+i} = Q_i - q_i + q(\tilde{P})$, a contradiction because $q(\tilde{P}) > 0$ and constraints (13) are assumed to hold. □

Formulation (6)-(14) is non-linear because of constraints (12). However, these constraints can be linearized as follows:

$$Q_j \geq Q_i + q_j - W_j(1 - x_{ij}) \qquad \forall i \in N, j \in N, \tag{15}$$

where $W_j = \min\{Q, Q + q_j\}$. Moreover, Desrochers and Laporte [9] have shown that the inequalities (15) can be lifted by considering the reverse arc and imposing

$$Q_j \geq Q_i + q_j - W_j(1 - x_{ij}) + (W_j - q_i - q_j)x_{ji} \qquad \forall (i,j) \in A. \tag{16}$$

Nevertheless, this formulation is likely to be weak because of the introduction of the $W_j$ constants which typically lead to poor linear programming relaxation lower bounds.

A different formulation can be obtained by dropping the $Q_i$ variables and associating each arc $(i,j) \in A$ with a continuous variable $f_{ij}$ representing the load of the vehicle on that arc. Then, constraints (12)-(14) are replaced with the following sets of linear constraints:

$$\sum_{j \in N} f_{ij} - \sum_{j \in N} f_{ji} = q_i \qquad \forall i \in P \cup D \tag{17}$$

$$\sum_{j \in N} f_{ji} - \sum_{j \in N} f_{n+i,j} = 0 \qquad \forall i \in P \tag{18}$$

$$\max\{0, q_i, -q_j\}x_{ij} \leq f_{ij} \leq \min\{Q, Q + q_i, Q - q_j\}x_{ij} \qquad \forall (i,j) \in A. \tag{19}$$

We denote the resulting formulation, (6)-(11) and (17)-(19), by (TSPPDL2). It is in fact inspired from the one-commodity flow formulation for the vehicle routing problem initially proposed by Gavish and Graves [11].

Formulations (TSPPDL1) and (TSPPDL2) both require the introduction of additional variables in model (6)-(11). Following Gouveia [14] and Letchford and Salazar [19], it would be possible to project out these continuous variables and obtain a formulation with only the $x_{ij}$ variables. However, the Benders cuts from this projection do not seem to show a clear combinatorial structure, neither suggesting an efficient separation procedure nor strengthening constraints. Instead, we obtain this formulation as follows. We define the collection $\Omega$ of all subsets $S \subset P \cup D$ for which there is at least one request $j$ such that either $j \in S$ and $n + j \notin S$ or $n + j \in S$ and $j \notin S$. Using these definitions, the LIFO policy can be expressed in terms of the $x_{ij}$ variables as shown by the following proposition.

**Proposition 3.** The LIFO policy can be imposed through the following constraints:

$$x(i,S) + x(S) + x(S, n+i) \leq |S| \qquad \forall S \in \Omega, \forall i, n+i \notin S, i \in P. \tag{20}$$

**Proof.** Suppose that the LIFO policy is satisfied but that one of these constraints is violated. Since $x(S) \leq |S| - 1$ by constraints (9), this violation implies that $x(i,S) = x(S, n+i) = 1$ for a given set $S \in \Omega$ and node pair $\{i, n+i\}$. As a result, there is a path starting from $i$, visiting every node in $S$ and reaching $n + i$. But this is impossible because this path would violate the LIFO policy. Suppose now that all constraints are satisfied but the LIFO policy is not. This implies that between a given node pair $\{i, n+i\}$, the route visits a pickup node $j$ without visiting the delivery node $n + j$, or vice-versa. Hence, a constraint is violated for the set $S$ containing all nodes visited between $i$ and $n + i$, a contradiction. □

Using these inequalities, we finally obtain a third formulation for the uncapacitated case, denoted by (TSPPDL3), containing only $x_{ij}$ variables:

$$\text{(TSPPDL3)} \qquad \text{Minimize} \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{21}$$

subject to

$$x(\delta^+(i)) = 1 \qquad \forall i \in P \cup D \cup \{0\} \tag{22}$$

$$x(\delta^-(i)) = 1 \qquad \forall i \in P \cup D \cup \{2n+1\} \tag{23}$$

$$x(S) \leq |S| - 1 \qquad \forall S \subseteq P \cup D, |S| \geq 2 \tag{24}$$

$$x(S) \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{25}$$

$$x(i,S) + x(S) + x(S, n+i) \leq |S| \qquad \forall S \in \Omega, \forall i, n+i \notin S, i \in P \tag{26}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{27}$$

# 4 Valid Inequalities

In this section, we describe several families of valid inequalities that can be added to the three formulations introduced in the previous section in order to strengthen their linear programming relaxation. We first describe known inequalities for the TSPPD and we then introduce new inequalities that rely on the particular structure of the TSPPDL.

## 4.1 Known inequalities for the TSPPD

Since the TSPPDL is a restriction of the TSPPD, valid inequalities for the latter problem are also valid for the former. We now describe three known families of valid inequalities for the TSPPD: strengthened subtour elimination constraints and generalized order constraints.

The subtour elimination constraints (9) can be lifted in different ways by considering the precedence relations between pickup $i$ and delivery $n+j$. 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 a given subset $S \subseteq P \cup D$, respectively. Balas et al. [2] have introduced the following *predecessor* and *successor* inequalities for the precedence-constrained ATSP:

$$x(S) + \sum_{i \in S} \sum_{j \in \bar{S} \cap \pi(S)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in \bar{S} \setminus \pi(S)} x_{ij} \leq |S| - 1, \tag{28}$$

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

Cycle inequalities for the ATSP can be strengthened as explained by Grötschel and Padberg [15]. For a given ordered set $S = \{i_1, i_2, \ldots, i_k\} \subseteq N$, with $k \geq 3$, they proposed

two families of valid inequalities, called $D_k^+$ and $D_k^-$ inequalities:

$$\sum_{h=1}^{k-1} x_{i_h,i_{h+1}} + x_{i_k,i_1} + 2\sum_{h=2}^{k-1} x_{i_h,i_1} + \sum_{h=3}^{k-1}\sum_{l=2}^{h-1} x_{i_h,i_l} \leq k-1, \tag{30}$$

$$\sum_{h=1}^{k-1} x_{i_h,i_{h+1}} + x_{i_k,i_1} + 2\sum_{h=3}^{k} x_{i_1,i_h} + \sum_{h=4}^{k}\sum_{l=3}^{h-1} x_{i_h,i_l} \leq k-1. \tag{31}$$

Cordeau [6] has shown that because of precedence constraints the latter inequalities can be strengthened by considering the sets $\pi(S)$ and $\sigma(S)$ as in inequalities (28) and (29), leading to:

$$\sum_{h=1}^{k-1} x_{i_h,i_{h+1}} + x_{i_k,i_1} + 2\sum_{h=2}^{k-1} x_{i_h,i_1} + \sum_{h=3}^{k-1}\sum_{l=2}^{h-1} x_{i_h,i_l} + \sum_{n+i_p \in \bar{S}\cap\sigma(S)} x_{n+i_p,i_1} \leq k-1, \tag{32}$$

$$\sum_{h=1}^{k-1} x_{i_h,i_{h+1}} + x_{i_k,i_1} + 2\sum_{h=3}^{k} x_{i_1,i_h} + \sum_{h=4}^{k}\sum_{l=3}^{h-1} x_{i_h,i_l} + \sum_{i_p \in \bar{S}\cap\pi(S)} x_{i_1,i_p} \leq k-1. \tag{33}$$

Finally, let $U_1, \ldots, U_k \subset P \cup D$ be mutually disjoint subsets such that $i_1, \ldots, i_k \in P$ are requests for which $i_l, n+i_{l+1} \in U_l$ for $l = 1, \ldots, k$ (where $i_{k+1} = i_1$). The following *precedence cycle breaking inequalities* were introduced by Balas et al. [2] for the precedence-constrained TSP:

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

Equivalent inequalities, called *generalized order constraints* were introduced by Ruland and Rodin [27] for the undirected TSPPD.

Several other families of valid inequalities for the TSPPD have been described by Ruland and Rodin [27] and by Dumitrescu et al. [10]. However, our experiments have shown that they have a negligible impact in a branch-and-cut algorithm for the TSPPDL. In the next section we thus introduce new inequalities that take advantage of the particular structure of the problem.

## 4.2 New inequalities for the TSPPDL

We now describe three new families of inequalities for the TSPPDL. Throughout this section, we denote by $i \prec j$ the fact that node $i$ is a predecessor of node $j$ in a route.

### 4.2.1 Incompatible predecessor and successor inequalities

For all $i, j \in P$, if $x_{ij} = 1$ in a feasible integer solution, then this solution must satisfy $0 \prec i, j \prec n+j \prec n+i \prec 2n+1$. Indeed, the successor of $n+j$ is either $n+i$ or a pickup

node different from $i$. The set of possible successors to node $n+j$ if arc $(i,j)$ is used is thus $S_{n+j}(i,j) = \{n+i\} \cup (P \setminus \{i\})$. This leads to the following inequality.

**Proposition 4.** For each node pair $i, j \in P$ with $i \neq j$, the following inequality is valid for the TSPPDL:

$$x_{ij} + \sum_{l \notin S_{n+j}(i,j)} x_{n+j,l} \leq 1. \tag{35}$$

**Proof.** First note that at most one outgoing arc from node $n+j$ can be used in a solution. Two cases must be considered. If $x_{ij} = 1$, then load $j$ is placed on top of load $i$. Hence, after visiting node $n+j$ the vehicle can only visit node $n+i$ or a pickup node different from $i$. If instead the route uses an arc leaving node $n+j$ and going either to $i$ or to a delivery node different from $n+i$, then load $i$ cannot have been placed directly below load $j$ and thus $x_{ij} = 0$. □

Similarly, for all $i, j \in P$, if $x_{n+i,n+j} = 1$ is a feasible integer solution, then this solution must satisfy $0 \prec j \prec i \prec n+i, n+j \prec 2n+1$. The set of possible predecessors of node $i$ is then $P_i(n+i, n+j) = \{j\} \cup (D \setminus \{n+j\})$. This leads to the following inequality.

**Proposition 5.** For each node pair $i, j \in P$ with $i \neq j$, the following inequality is valid for the TSPPDL:

$$x_{n+i,n+j} + \sum_{l \notin P_i(n+i,n+j)} x_{li} \leq 1. \tag{36}$$

**Proof.** The proof is similar to that of Proposition 4 by interchanging pickups and deliveries. □

### 4.2.2 Hamburger inequalities

As shown in Section 2.1, the LIFO requirements lead to some interesting properties on the structure of feasible solutions. In particular consider the subset of arcs represented in Figure 3. For all $i, j \in P$, one can see that using arc $(i,j)$ implies that none of the remaining arcs can be used: $(n+i, j)$ enters node $j$, $(n+j, i)$ leads to a subtour, and $(n+i, n+j)$ results in a violation of the LIFO constraint. In fact, all arcs in Figure 3 are pairwise incompatible. As a result, the following inequality is valid for any $i, j \in P$ with $i \neq j$:

$$x_{ij} + x_{n+i,n+j} + x_{n+j,i} + x_{n+i,j} \leq 1 \qquad \forall i, j \in P, i \neq j. \tag{37}$$

When three or more requests are considered, a similar reasoning yields a new family of inequalities.

**Proposition 6.** Consider an ordered subset of requests defined by the indices $\{i_1, \ldots, i_k\}$, with $k \geq 3$, and assume that the index is circular, i.e., $i_{k+1} = i_1$, $i_{k+2} = i_2$, ... and $i_0 = i_k$, $i_{-1} = i_{k-1}$, ... The following inequality is valid for the TSPPDL:

$$\sum_{h=1}^{k} \left( x_{i_h,i_{h+1}} + x_{n+i_{h+1},i_h} + x_{n+i_h,n+i_{h+1}} \right) \leq k-1. \tag{38}$$

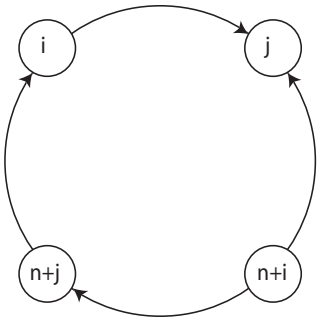


Figure 3: *Representation of valid inequality (37).*

Inequalities (38) can in fact be lifted in two different ways, as shown by the following proposition.

**Proposition 7:** Consider an ordered subset of requests defined by the indices $\{i_1, \ldots, i_k\}$, with $k \geq 3$, and assume that the index is circular, i.e., $i_{k+1} = i_1$, $i_{k+2} = i_2$, ... and $i_0 = i_k$, $i_{-1} = i_{k-1}$, ... The following inequalities are valid for the TSPPDL:

$$\sum_{h=1}^{k} \left( x_{i_h,i_{h+1}} + x_{n+i_{h+1},i_h} + x_{n+i_h,n+i_{h+1}} + \sum_{l=h+2}^{h+k-2} x_{i_l,i_h} \right) \leq k-1, \tag{39}$$

$$\sum_{h=1}^{k} \left( x_{i_h,i_{h+1}} + x_{n+i_{h+1},i_h} + x_{n+i_h,n+i_{h+1}} + \sum_{l=h+2}^{h+k-2} x_{n+i_h,n+i_l} \right) \leq k-1. \tag{40}$$

**Proof.** To prove (39) we first note that (38) can be rewritten as:

$$\sum_{h=1}^{k} \left( x_{i_{h-1},i_h} + x_{n+i_{h+1},i_h} + x_{n+i_h,n+i_{h+1}} \right) \leq k-1, \tag{41}$$

since this only implies a translation of the circular index of $x_{i_h,i_{h+1}}$ to $x_{i_{h-1},i_h}$. We now show that (41) can be lifted as follows:

$$\sum_{h=1}^{k} \left( x_{i_{h-1},i_h} + x_{n+i_{h+1},i_h} + x_{n+i_h,n+i_{h+1}} + \sum_{l=h+2}^{h+k-2} x_{i_l,i_h} \right) \leq k-1. \tag{42}$$

For a given index $h$, at most one of the arcs entering $i_h$ can belong to a feasible solution. In addition, all of these arcs are incompatible with $(n + i_h, n + i_{h+1})$. Indeed this last arc implies that in the stack, $i_h$ appears immediately on top of $i_{h+1}$. As a result, the only compatible PP arc entering $i_h$ would be $(i_{h+1}, i_h)$, which does not belong to the summation.

It is also obvious that the arcs $(n + i_h, n + i_{h+1})$ and $(n + i_{h+1}, i_h)$ are incompatible as they would violate the precedence constraint for request $h$. Hence, in any feasible solution, there can be at most one arc from each group. Assume now that exactly one arc is selected from each group. As for inequalities (38), one can check that these arcs impose a sequence of operations that violates either the precedence or LIFO constraints. Finally, (42) can be transformed into (39) by simply translating the indices of $x_{i_{h-1},i_h}$ to $x_{i_h,i_{h+1}}$. The validity of (40) is established by following a similar reasoning. □

Inequalities (39) and (40) are represented in Figures 4 and 5, respectively. We refer to these inequalities as *hamburger inequalities*.

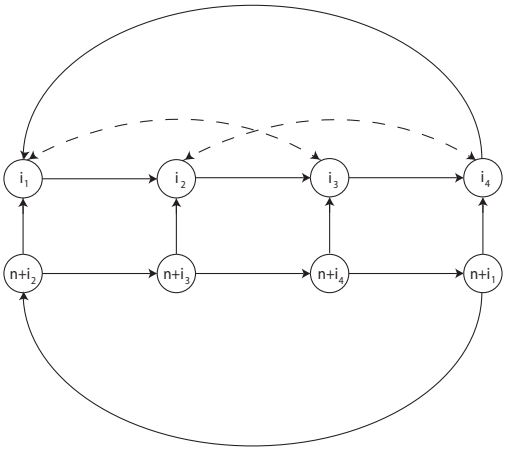


Figure 4: *Hamburger inequality (39) with four requests.*

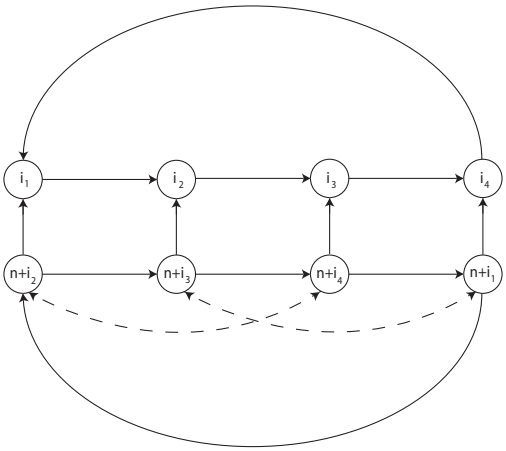


Figure 5: *Hamburger inequality (40) with four requests.*

### 4.2.3 Incompatible paths inequalities

Another family of inequalities can be derived from the classical *infeasible path* constraints. For $i, j \in P$, let $P_{ij}$ be a path starting at node $i$, finishing at node $j$, and not containing node $n+i$. Let also $P_{n+i,n+j}$ be a path from $n+i$ to $n+j$. For any path $P$, let also $A(P)$ denote the arc set of $P$. If a feasible integer solution uses all arcs in $A(P_{ij})$ then $i \prec j$. As a result, $n+j \prec n+i$, since, by the definition of $P_{ij}$, $n+i$ does not appear on $P_{ij}$ and must consequently be a successor of $n+j$. Hence, not all arcs in $A(P_{n+i,n+j})$ can be used. This observation proves the following proposition.

**Proposition 10:** Consider two requests $i, j \in P$. For any path $P_{ij} = (k_1, \ldots, k_p)$ with $k_1 = i$, $k_p = j$ and $k_h \neq n+i, 2 \leq h \leq p-1$, and any path $P_{n+i,n+j} = (l_1, \ldots, l_q)$ with $l_1 = n+i$ and $l_q = n+j$, the following inequality is valid for the TSPPDL:

$$\sum_{h=1}^{p-1} x_{k_h,k_{h+1}} + \sum_{h=1}^{q-1} x_{l_h,l_{h+1}} \leq |A(P_{ij})| + |A(P_{n+i,n+j})| - 1. \tag{43}$$

These inequalities can in fact be lifted by following the strengthening of infeasible path constraints into *tournament inequalities* introduced by Ascheuer et al. [1]. This idea leads to the following form:

$$\sum_{h=1}^{p-1} \sum_{r=h+1}^{p} x_{k_h,k_r} + \sum_{h=1}^{q-1} \sum_{r=h+1}^{q} x_{l_h,l_r} \leq |A(P_{ij})| + |A(P_{n+i,n+j})| - 1. \tag{44}$$

# 5 Branch-and-Cut Algorithm

The formulations presented in Section 3 and the valid inequalities introduced in Section 4 are used within a branch-and-cut algorithm which we now describe in detail.

## 5.1 Preprocessing and cut pool

As noted earlier, no delivery node $n+j$ can be the direct successor of a pick-up node $i$ when $j \neq i$. As a result, arcs of the form $(i, n+j)$ with $i \neq j$ can be removed from the graph.

Before starting the algorithm, we add the following set of inequalities to the models. Because there is a quadratic number of inequalities (35) and (36), these are all introduced into the cut pool. Similarly, we include all subtour elimination constraints (9) with $|S| = 2$ as well as all constraints (37) whose number is also quadratic. We also add to the model the simple predecessor and successor inequalities, obtained by setting $S$ equal to $\{i, j\}$, $\{i, n+j\}$ and $\{i, n+i, j\}$ in (28), and to $\{n+i, n+j\}$, $\{i, n+j\}$ and $\{i, n+i, n+j\}$ in (29). Finally, we include simple cases of the strengthened $D_h^+$ and $D_h^-$ inequalities: $x_{n+i,j} + x_{ji} + x_{i,n+i} + x_{n+j,n+i} \leq 2$ and $x_{i,n+i} + x_{n+i,n+j} + x_{n+j,i} + x_{i,j} \leq 2$.

## 5.2 Separation procedures

In our branch-and-cut algorithm, we use exact separation procedures for subtour elimination constraints (9), precedence constraints (10) and LIFO constraints (26). The first two groups of constraints are required to ensure feasibility in all three formulations, while the third group of constraints is necessary for formulation (TSPPDL3). For the separation of inequalities (28)-(29),(32)-(34), (39),(40) and (43), however, we use heuristics.

### 5.2.1 Exact separation procedures

Constraints (9) are separated exactly in the classical way. Given a (fractional) solution $x^*$, we create a supporting graph $G^* = (N, A^*)$, where an arc $(i, j) \in A^*$ has capacity equal to $x^*_{ij}$. We then solve a max-flow problem on $G^*$, from 0 to each pickup $i$, and from each delivery $n + i$ to $2n + 1$ $(i = 1, \ldots, n)$.

Constraints (10) are separated exactly for each $i \in P$ in the following way:

1. create the supporting graph $G^*$;
2. connect 0 to $n + i$ and $i$ to $2n + 1$ $(i = 1, \ldots, n)$ with arcs of capacity 2;
3. solve the max-flow problem from 0 to $2n + 1$ and obtain a set $S^*$ containing 0 (note that by construction $i \notin S^*$ and $n + i \in S^*$). If the max-flow value is smaller than 2, then $S^*$ defines a violated inequality (10).

In addition to generating violated inequalities of the form (10), we also consider another form of precedence constraints introduced by Balas et al. [2]:

$$x(n + i, S) + x(S) + x(S, i) \leq |S| \qquad \forall S \subset N, \forall i \in P : i \notin S, n + i \notin S. \tag{45}$$

Note that inequalities (45) are equivalent to the following inequalities:

$$x(\delta^+(S)) + x(n + i, \bar{S}) + x(\bar{S}, i) \;\; \geq \;\; 2. \tag{46}$$

Inequalities (46) are separated exactly for each $i \in P$ in the following way:

1. create the supporting graph $G^*$;
2. insert in $G^*$ a dummy node $2n + 2$, and connect $2n + 2$ to every node $k \in N$ with an arc of capacity $x_{n+i,k} + x_{k,i}$ $(k = 1, \ldots, 2n)$;
3. increase the capacity of the arcs $(2n + 1, i)$, $(0, i)$ and $(n + i, i)$ to 2;
4. solve the max-flow problem from $2n + 2$ to $i$ and obtain a set $S^*$ such that $2n + 2 \in S^*$ (note that by construction $i \notin S^*$, $n + i \notin S^*$, $0 \notin S^*$, $2n + 1 \notin S^*$). If the max-flow value is smaller than 2, then $S^*$ defines a violated inequality (46).

Finally, we show how to separate inequalities (26) which are equivalent to:

$$x(\delta^+(S)) + x(i, \bar{S}) + x(\bar{S}, n+i) \quad \geq \quad 2. \tag{47}$$

For each pair of nodes $i, j \in P$ we need to perform two searches. First, we need to check all sets $S$ such that $i \notin S$, $n+i \notin S$, $j \in S$ and $n+j \notin S$. Second, we need to check all sets $S$ such $i \notin S$, $n+i \notin S$, $j \notin S$ and $n+j \in S$. Here we describe how to perform the first search, which means that the procedure searches for a set $S$ with $j$ inside and $i, n+i, n+j$ outside. The other search is performed in an equivalent way:

1. create the supporting graph $G^*$;
2. increase the capacity of arcs $(j,k)$ by the value $x^*_{ik} + x^*_{k,n+i}$ $(k = 1, \ldots, 2n)$;
3. define the capacity of the arcs $(i, n+i)$, $(n+i, i)$, $(i, n+j)$ and $(n+j, i)$ equal to 2;
4. solve the max-flow problem from $j$ to $i$ and obtain a set $S^*$ such that $j \in S^*$ (and note that by construction $i \notin S^*$, $n+i \notin S^*$ and $n+j \notin S^*$). If the max-flow value is smaller than 2, then $S^*$ defines a violated inequality (47) with $S^* \in \Omega$.

### 5.2.2 Heuristic separation procedures

In order to separate inequalities (28) and (29) we use a tabu search heuristic, denoted TS1, derived from the one proposed by Cordeau [6]. Given a subset of customers $S$ (initialized with a randomly chosen customer), we compute all possible moves obtained by 1) deleting a customer from $S$ or 2) inserting a customer in $S$. Among the possible moves we choose the one leading to the maximum value of the left-hand side of (28) or of (29), minus $|S|$. As soon as a violated inequality has been identified, it is added to the model. When a node is inserted (resp. deleted) in $S$, its deletion (resp. insertion) is declared tabu for $\theta_1$ iterations. The algorithm is run twice, for the separation of (28) and (29), and every time is is halted when $\gamma_1$ iterations have been performed or a maximum number of cuts has been added to the model.

We have considered five intervals for $\theta_1$: $\max\{5, n/10\}$, $\max\{5, 2n/10\}$, $\max\{5, 3n/10\}$, $\max\{4, n/10\}$ and $\max\{3, n/10\}$. The performance of the heuristic is not really sensitive to the value of this parameter but we found that the first interval produced the best results on average. Similarly, $\gamma_1 = 25$ proved to be slightly better than 10, 20, 30, 40, 50 and 100.

Note that TS1 works on non-ordered subsets of customers, while inequalities (32), (33), (34), (39) and (40) are based on ordered sequences of customers. For this reason we separate these families of inequalities by means of a more elaborate tabu search algorithm (TS2).

Given an ordered sequence of customers $S$, TS2 computes all the moves obtained by 1) deleting one of the customers in $S$, 2) switching the positions of two customers in $S$, or 3) inserting a new customer in a given position in $S$. As in TS1, we also choose the move leading the maximum value of the left-hand side of the inequality being separated, minus $|S|$. We have implemented the tabu list as a set of couples $(i,j)$, where $i = -1$ means that customer

$j$ is inserted in $S$, $j = -1$ means that $i$ is removed from $S$, and $i, j \geq 0$ implies that the positions in $S$ of the two vertices $i$ and $j$ are exchanged. When a move $(i, j)$ is performed, the opposite move $(j, i)$ is forbidden for $\gamma_2$ iterations. The algorithm is halted after $\gamma_2$ iterations or after having added a maximum number of cuts to the model. We run TS2 three times, for the separation of (32), (33) and (34), respectively. We also run it a fourth time for the separation of (39) and (40). In this last case, TS2 looks for the maximization of the *common part* of the left hand sides of the two inequalities, minus $|S|$.

For TS2, we found it convenient to initialize $S$ with a random triplet of customers. Only sequences with up to $4n/10$ customers are taken into consideration during the search process. This value proved to be better than $n/10$, $2n/10$, $3n/10$ and $5n/10$. Similarly to TS1, we found it convenient to set $\theta_2 = \max\{5, n/10\}$. The maximum number of iterations was finally set to $\gamma_2 = 50$, a value preferred to 10, 20, 25, 30, 40, 60, 70 and 100.

For what concerns constraints (43), we decided to separate them by means of a series of constructive heuristics. The first simple greedy heuristic (P1) takes into consideration paths of length at most four. For each couple of pickups $i$ and $j$ ($i \in N, j \in N, j \neq i$) P1 tries to construct a path $P_{i,j}$ by considering the sequences of the forms $(i, j)$, $(i, h, j)$ and $(i, h, n + h, j)$ ($h \in N, h \neq i, h \neq j$). If a path satisfying $\sum_{(h,k)\in P_{ij}} x_{hk} > |P_{ij}| - 1$ is found, then P1 checks the corresponding path $P_{n+i,n+j}$. Also in this case the paths considered have length at most four, with the forms $(n+i, n+j)$, $(n+i, n+h, n+j)$ and $(n+i, h, n+h, n+j)$ ($h \in N, h \neq i, h \neq j$). If a violation of (43) is found the corresponding cut is added to the model; otherwise a more complex heuristic (P2) is executed.

Heuristic P2 constructs a path $P_{ij}$ by choosing a vertex $i$ and linking it to the vertex $j$ such that $x_{ij}$ has the highest value among the variables leaving $i$ (breaking ties by decreasing value of $j$). Then the path is possibly extended in the same way, by finding the highest value variable $x_{jk}$ and connecting $j$ to $k$ (forming $P_{ik}$) and so on. As soon as a pickup, say $j$, is found in the path, then a possible violation is searched by checking the corresponding path $P_{n+i,n+j}$. This is done by first considering the value of $x_{n+i,n+j}$. Then, if no violation is found, $P_{n+i,n+j}$ is extended by considering the vertex maximizing the $x$ flow between $i$ and $j$. This extension of $P_{n+i,n+j}$ is reiterated until 1) $\sum_{(h,k)\in P_{ij}} x_{hk} + \sum_{(h,k)\in P_{n+i,n+j}} x_{hk} \leq |P_{ij}| + |P_{n+i,n+j}| - 1$ or 2) a maximum path length *maxpl* is reached. For each value of $i$, the $P_{ij}$ path extension is halted whenever 1) $i$ is linked to $n + i$, 2) $\sum_{(h,k)\in P_{ij}} x_{hk} \leq |P_{ij}| - 1$ or 3) a maximum path length *maxpl* is reached.

Finally, a similar heuristic (P3) is used. It first constructs the $P_{n+i,n+j}$ path, in the same way as in P2, and then checks $P_{ij}$ for possible violations. More elaborated heuristics did not produce significant improvements in the solution quality. After a preliminary setting *maxpl* was set to $\min\{n, 20\}$.

# 6 Computational Results

Our branch-and-cut algorithm was coded in C and run on a Pentium IV 3 GHz, using CPLEX 10.0 as ILP solver. The tests were executed on the instances introduced by Carrabs

et al. [3], by considering up to 25 requests (52 nodes), leading to a total of 45 instances.

To reduce the CPU time spent by the separation procedures, at each node of the branch-and-cut tree we stop looking for violated valid inequalities after 8 violated inequalities have been added to the model. This value was chosen experimentally by performing sensitivity analyses on our set of test instances.

Table 2 presents the root-node lower bounds obtained for the three models presented in the previous sections. The names of the original TSP instances used to produce valid TSPPDL instances are given in column *Graph*. In column $n$ we give the number of requests and in column $UB$ we report the best upper bound that we obtained for each instance.

Our algorithm takes as an initial upper bound the value obtained by the VNS heuristic of Carrabs et al. [4]. These upper bounds are identical to the ones in Column $UB$, except for three cases where our algorithm could improve the solution found by the heuristic. In particular, for instances nrw1379 with $n = 17$, att532 with $n = 25$ and ts225 with $n = 25$, we obtained solutions with cost 3644 (versus 3652), 11478 (versus 11484) and 54386 (versus 54629), respectively.

The values in the remaining columns are given as percentage gaps between the lower bound obtained by a model and $UB$, computed as $100(UB - LB)/UB$. In particular we report the percentage gap obtained by each of the plain formulations TSPPDL1, TSPPDL2 and TSPPDL3. For TSPPDL3 we also report the percentage gaps obtained by adding one family of inequalities at a time to the plain formulation (e.g., column (35,36) reports the value obtained with formulation TSPPDL3 plus constraints (35) and (36)). The three plain formulations present similar percentage gaps, with TSPPDL3 obtaining a slightly better average value (11.58%). This value is consistently improved by the quadratic families of inequalities (35) and (36) which together lead to a gap of 5.28%. Smaller improvements are obtained by considering the other inequalities, with the generalized order constraints (34) being the least effective family, leading to no significant improvement.

In Table 3 we evaluate the marginal contributions of each family of inequalities on the root node lower bound for TSPPDL3. In column *Full* we give the percentage gap of the lower bound obtained by TSPPDL3 with the separation of all families. In the successive columns we remove from the full formulation one family of inequalities (e.g., column (35,36) gives the values obtained by removing constraints (35) and (36)). It can be noted that TSPPDL3 yields a good average percentage gap of 4.73% from $UB$. This gap increases to 10.01% when (35) and (36) are removed. A smaller but still significant increase is obtained when the other families of constraints are removed. It is worth noting that the majority of instances with nine requests are solved at the root node by the complete model.

In both Tables 2 and 3, one can note some cases in which adding (resp. removing) a valid inequality deteriorates (resp. improves) the lower bounds. This behavior is explained by the automatic cut generation and problem reduction routines internal to CPLEX.

Finally we report in Table 4 the complete results obtained with the full formulation TSPPDL3. Because of the poor performance of constraints (34), these were not included in the model. The algorithm was allowed to run for one hour. In column $\%gap$ we report the percentage gap between the best lower bound found and $UB$, computed as in the previous

Table 2: Root-node lower bounds for the three models.

| *Instance* | | | *TSPPDL1* | *TSPPDL2* | *TSPPDL3* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Graph* | $n$ | $UB$ | *Plain* | *Plain* | *Plain* | (35,36) | (45) | (28,29) | (32,33) | (34) | (39,40) | (44) |
| a280 | 9 | 402 | 11.94 | 10.45 | 9.20 | 0.00 | 8.21 | 8.71 | 9.20 | 9.20 | 8.96 | 8.96 |
| att532 | 9 | 4250 | 5.98 | 5.91 | 5.79 | 0.00 | 5.67 | 4.49 | 5.67 | 5.53 | 4.42 | 5.69 |
| brd14051 | 9 | 4555 | 4.08 | 3.97 | 3.14 | 0.00 | 3.16 | 2.72 | 2.94 | 3.12 | 2.59 | 3.14 |
| d15112 | 9 | 76203 | 5.66 | 5.93 | 5.84 | 0.00 | 5.72 | 5.37 | 5.70 | 5.85 | 4.68 | 5.41 |
| d18512 | 9 | 4446 | 3.46 | 3.44 | 3.24 | 0.00 | 3.24 | 2.54 | 3.35 | 3.44 | 3.04 | 3.26 |
| fnl4461 | 9 | 1866 | 1.07 | 0.75 | 0.38 | 0.00 | 0.32 | 0.00 | 0.32 | 0.38 | 0.00 | 0.00 |
| nrw1379 | 9 | 2691 | 5.09 | 2.08 | 2.56 | 0.00 | 1.60 | 1.60 | 1.60 | 2.56 | 1.52 | 1.60 |
| pr1002 | 9 | 12947 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ts225 | 9 | 21000 | 11.90 | 8.33 | 9.52 | 4.76 | 11.90 | 10.71 | 11.31 | 11.11 | 10.77 | 9.52 |
| a280 | 13 | 505 | 14.26 | 13.07 | 12.28 | 4.36 | 11.68 | 7.72 | 10.89 | 12.28 | 10.30 | 13.86 |
| att532 | 13 | 5800 | 6.72 | 6.84 | 6.83 | 2.02 | 6.83 | 5.24 | 6.93 | 6.83 | 5.86 | 6.52 |
| brd14051 | 13 | 4936 | 9.50 | 10.58 | 9.46 | 1.48 | 9.22 | 9.04 | 9.14 | 9.46 | 8.35 | 9.06 |
| d15112 | 13 | 93158 | 11.39 | 11.34 | 11.32 | 5.95 | 11.32 | 10.88 | 11.34 | 11.45 | 10.49 | 11.10 |
| d18512 | 13 | 4704 | 7.87 | 7.84 | 7.55 | 3.89 | 7.57 | 7.27 | 7.63 | 7.57 | 7.48 | 7.57 |
| fnl4461 | 13 | 2483 | 15.18 | 14.90 | 14.34 | 7.53 | 14.34 | 13.89 | 14.22 | 14.46 | 13.69 | 14.18 |
| nrw1379 | 13 | 3366 | 15.54 | 15.63 | 14.80 | 7.04 | 13.58 | 12.09 | 14.80 | 13.81 | 13.28 | 14.26 |
| pr1002 | 13 | 15566 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ts225 | 13 | 32395 | 8.46 | 8.30 | 7.45 | 3.20 | 7.45 | 7.51 | 7.32 | 7.56 | 7.37 | 7.51 |
| a280 | 17 | 647 | 11.44 | 11.90 | 10.51 | 4.02 | 11.44 | 7.26 | 9.74 | 11.44 | 10.05 | 11.44 |
| att532 | 17 | 6361 | 8.50 | 8.74 | 8.63 | 3.99 | 8.63 | 8.32 | 8.66 | 8.73 | 7.84 | 8.38 |
| brd14051 | 17 | 5196 | 11.91 | 11.91 | 11.61 | 1.50 | 11.47 | 10.74 | 11.30 | 11.59 | 9.82 | 11.61 |
| d15112 | 17 | 115554 | 15.06 | 15.38 | 16.10 | 5.16 | 15.93 | 16.12 | 16.14 | 16.19 | 15.47 | 16.21 |
| d18512 | 17 | 5186 | 13.38 | 13.25 | 12.57 | 6.63 | 12.55 | 12.15 | 12.46 | 12.53 | 12.50 | 12.57 |
| fnl4461 | 17 | 2852 | 15.60 | 15.50 | 15.04 | 6.14 | 15.04 | 15.32 | 15.15 | 15.15 | 14.31 | 15.15 |
| nrw1379 | 17 | 3644 | 15.53 | 15.26 | 15.09 | 8.95 | 14.87 | 14.16 | 15.12 | 15.12 | 13.86 | 14.38 |
| pr1002 | 17 | 17564 | 1.66 | 1.67 | 1.59 | 0.00 | 0.47 | 0.00 | 0.19 | 1.09 | 1.05 | 1.59 |
| ts225 | 17 | 36703 | 7.37 | 7.41 | 7.41 | 4.60 | 7.39 | 5.43 | 7.26 | 7.41 | 6.56 | 7.37 |
| a280 | 21 | 752 | 11.17 | 12.23 | 11.04 | 5.19 | 11.04 | 6.52 | 10.24 | 10.37 | 9.97 | 11.57 |
| att532 | 21 | 10714 | 8.57 | 8.86 | 8.89 | 4.26 | 8.89 | 8.55 | 8.81 | 8.71 | 7.22 | 8.63 |
| brd14051 | 21 | 5719 | 17.10 | 17.05 | 16.84 | 6.52 | 16.75 | 16.49 | 16.79 | 16.66 | 15.79 | 16.72 |
| d15112 | 21 | 128798 | 18.87 | 18.80 | 19.18 | 7.59 | 19.06 | 18.82 | 19.14 | 19.19 | 18.45 | 19.16 |
| d18512 | 21 | 5634 | 16.08 | 15.50 | 15.03 | 6.28 | 15.00 | 14.39 | 14.98 | 15.14 | 15.18 | 15.09 |
| fnl4461 | 21 | 3269 | 19.30 | 19.46 | 19.33 | 9.91 | 19.33 | 19.30 | 19.24 | 19.36 | 19.03 | 19.21 |
| nrw1379 | 21 | 4282 | 19.92 | 19.83 | 19.52 | 9.95 | 19.34 | 19.59 | 19.59 | 19.64 | 18.99 | 19.29 |
| pr1002 | 21 | 20173 | 3.20 | 3.18 | 2.75 | 1.47 | 2.52 | 1.16 | 2.58 | 2.70 | 2.60 | 2.68 |
| ts225 | 21 | 43082 | 13.35 | 13.25 | 12.61 | 7.17 | 12.10 | 9.12 | 11.77 | 13.13 | 10.66 | 12.61 |
| a280 | 25 | 845 | 11.24 | 12.07 | 11.83 | 4.73 | 10.65 | 7.10 | 9.94 | 11.60 | 8.28 | 11.36 |
| att532 | 25 | 11478 | 8.67 | 8.84 | 8.85 | 4.42 | 8.89 | 8.76 | 8.89 | 8.83 | 7.45 | 8.47 |
| brd14051 | 25 | 7539 | 34.69 | 34.55 | 34.35 | 22.56 | 34.33 | 34.20 | 34.35 | 34.37 | 33.44 | 34.18 |
| d15112 | 25 | 143654 | 21.93 | 22.23 | 22.08 | 8.80 | 22.03 | 21.37 | 22.00 | 22.06 | 21.57 | 22.05 |
| d18512 | 25 | 7291 | 32.94 | 32.57 | 32.12 | 21.85 | 32.09 | 31.64 | 32.12 | 32.15 | 32.26 | 32.14 |
| fnl4461 | 25 | 3860 | 24.84 | 24.84 | 24.66 | 14.17 | 24.72 | 24.48 | 24.72 | 24.64 | 24.59 | 24.56 |
| nrw1379 | 25 | 4836 | 20.35 | 20.35 | 20.37 | 10.05 | 20.29 | 20.29 | 20.37 | 20.37 | 19.77 | 20.20 |
| pr1002 | 25 | 22774 | 4.12 | 4.23 | 3.96 | 2.31 | 3.71 | 2.90 | 3.63 | 3.92 | 3.66 | 3.90 |
| ts225 | 25 | 54386 | 15.87 | 15.96 | 15.44 | 9.25 | 15.14 | 12.66 | 13.84 | 15.18 | 12.87 | 15.46 |
| *Average* | | | 12.02 | 11.87 | 11.58 | 5.28 | 11.45 | 10.59 | 11.36 | 11.60 | 10.80 | 11.50 |

Table 3: Root-node lower bounds for model 3.

| *Instance* | | | *TSPPDL3* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *Graph* | $n$ | $UB$ | *Full* | (35,36) | (45) | (28,29) | (32,33) | (34) | (39,40) | (44) |
| a280 | 9 | 402 | 1.74 | 9.45 | 1.74 | 0.00 | 1.99 | 1.49 | 0.00 | 0.00 |
| att532 | 9 | 4250 | 0.00 | 3.25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| brd14051 | 9 | 4555 | 0.00 | 1.19 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| d15112 | 9 | 76203 | 0.00 | 4.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| d18512 | 9 | 4446 | 0.00 | 2.14 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| fnl4461 | 9 | 1866 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| nrw1379 | 9 | 2691 | 0.00 | 1.45 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| pr1002 | 9 | 12947 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ts225 | 9 | 21000 | 4.76 | 9.82 | 4.76 | 4.76 | 4.76 | 4.76 | 4.76 | 4.76 |
| a280 | 13 | 505 | 0.00 | 7.72 | 0.99 | 1.98 | 0.00 | 0.00 | 0.00 | 0.59 |
| att532 | 13 | 5800 | 0.33 | 4.26 | 1.10 | 1.60 | 1.55 | 1.22 | 0.00 | 1.31 |
| brd14051 | 13 | 4936 | 1.76 | 7.41 | 1.76 | 1.99 | 1.70 | 1.54 | 1.50 | 1.76 |
| d15112 | 13 | 93158 | 5.54 | 10.17 | 5.54 | 6.06 | 5.58 | 5.49 | 5.69 | 5.78 |
| d18512 | 13 | 4704 | 3.44 | 7.25 | 3.44 | 3.87 | 3.49 | 3.51 | 3.51 | 3.44 |
| fnl4461 | 13 | 2483 | 7.29 | 13.45 | 7.29 | 7.45 | 7.65 | 7.65 | 7.73 | 7.29 |
| nrw1379 | 13 | 3366 | 6.00 | 12.33 | 6.00 | 6.12 | 5.73 | 5.56 | 5.44 | 5.88 |
| pr1002 | 13 | 15566 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ts225 | 13 | 32395 | 3.09 | 7.35 | 3.63 | 3.09 | 3.09 | 3.09 | 4.24 | 4.18 |
| a280 | 17 | 647 | 2.94 | 7.26 | 2.78 | 3.40 | 4.17 | 2.94 | 2.63 | 2.94 |
| att532 | 17 | 6361 | 3.84 | 6.84 | 3.95 | 3.84 | 2.44 | 2.70 | 3.27 | 3.84 |
| brd14051 | 17 | 5196 | 1.40 | 9.22 | 1.42 | 1.42 | 1.35 | 1.44 | 1.17 | 1.40 |
| d15112 | 17 | 115554 | 4.66 | 14.46 | 4.66 | 5.15 | 4.96 | 4.87 | 4.90 | 4.67 |
| d18512 | 17 | 5186 | 6.54 | 11.96 | 6.54 | 6.40 | 6.56 | 6.54 | 6.58 | 6.54 |
| fnl4461 | 17 | 2852 | 6.45 | 14.10 | 6.45 | 6.45 | 6.66 | 6.42 | 6.42 | 6.45 |
| nrw1379 | 17 | 3644 | 7.85 | 13.47 | 8.26 | 8.18 | 7.85 | 7.96 | 7.96 | 7.85 |
| pr1002 | 17 | 17564 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ts225 | 17 | 36703 | 3.42 | 4.56 | 3.38 | 4.58 | 3.62 | 3.41 | 3.42 | 3.44 |
| a280 | 21 | 752 | 2.66 | 7.31 | 2.93 | 3.86 | 3.06 | 3.59 | 2.93 | 2.66 |
| att532 | 21 | 10714 | 3.39 | 6.85 | 3.38 | 3.69 | 3.36 | 3.53 | 3.55 | 3.35 |
| brd14051 | 21 | 5719 | 6.15 | 15.65 | 6.22 | 6.21 | 6.17 | 6.14 | 6.12 | 6.15 |
| d15112 | 21 | 128798 | 7.09 | 17.65 | 7.06 | 7.11 | 7.30 | 6.92 | 6.90 | 6.91 |
| d18512 | 21 | 5634 | 6.27 | 14.27 | 6.32 | 6.39 | 6.35 | 6.39 | 6.35 | 6.35 |
| fnl4461 | 21 | 3269 | 9.70 | 18.75 | 9.73 | 9.70 | 9.79 | 9.67 | 9.70 | 9.54 |
| nrw1379 | 21 | 4282 | 8.97 | 18.82 | 8.94 | 8.97 | 9.22 | 8.97 | 9.41 | 8.97 |
| pr1002 | 21 | 20173 | 0.00 | 1.31 | 0.00 | 0.57 | 0.00 | 0.00 | 0.00 | 0.00 |
| ts225 | 21 | 43082 | 5.31 | 9.03 | 5.45 | 7.35 | 5.80 | 5.95 | 5.98 | 5.91 |
| a280 | 25 | 845 | 3.55 | 7.10 | 3.20 | 3.67 | 3.31 | 2.84 | 2.60 | 3.79 |
| att532 | 25 | 11478 | 4.12 | 7.07 | 4.11 | 4.15 | 4.14 | 4.14 | 4.11 | 4.19 |
| brd14051 | 25 | 7539 | 22.59 | 33.25 | 22.58 | 22.55 | 22.54 | 22.50 | 22.47 | 22.48 |
| d15112 | 25 | 143654 | 8.47 | 20.60 | 8.33 | 8.33 | 8.54 | 8.31 | 8.22 | 8.45 |
| d18512 | 25 | 7291 | 21.37 | 31.48 | 21.37 | 21.46 | 21.62 | 21.59 | 21.42 | 21.37 |
| fnl4461 | 25 | 3860 | 13.91 | 24.25 | 13.83 | 14.07 | 13.78 | 14.02 | 13.70 | 13.83 |
| nrw1379 | 25 | 4836 | 9.68 | 19.67 | 9.80 | 9.62 | 9.82 | 9.47 | 9.82 | 9.47 |
| pr1002 | 25 | 22774 | 0.16 | 2.56 | 0.87 | 1.61 | 1.44 | 1.19 | 0.16 | 1.23 |
| ts225 | 25 | 54386 | 8.59 | 11.49 | 8.75 | 8.79 | 7.95 | 8.61 | 8.13 | 8.59 |
| *Average* | | | 4.73 | 10.01 | 4.81 | 4.99 | 4.83 | 4.76 | 4.68 | 4.79 |

tables. In column *Nodes* we give the number of nodes in the enumeration tree, while in column *User* we report the number of violated inequalities added to the model (excluding those automatically added by CPLEX). In column *Secs* we give the CPU time in seconds required by the algorithm.

All instances with up to 17 requests are solved to optimality within about one minute, with the exception of instance d18512 with $n = 17$, which requires almost 13 minutes. Eight of the larger-size instances are not solved to optimality. For these instances, the percentage gap can be large, reaching a maximum of about 20%. The size of the enumeration tree exceeds only once 15000 nodes. The number of cuts added increases with the difficulty of the instance, being on average around 1500 per instance.

In summary, TSPPDL3 solves to optimality 37 out of 45 instances in less than one CPU hour, with an average percentage gap of 1.39%. Allowing larger CPU times allows the algorithm to solve all instances with up to 21 requests. In particular, for instances d15112, d18512 and nrw1379, the solutions of value 128798, 5634 and 4282 are proven to be optimal in 6135.92, 11717.19 and 11663.13 secs., respectively. No proof of optimality could however be obtained for the instances with 25 requests that were unsolved after one hour. In the same CPU time limit of one hour, TSPPDL1 and TSPPDL2 display worse computational behavior.

In Table 4 we also show the effectiveness of the separation procedures described in Section 5.2 by comparing TSPPDL3 with a reduced version of the model in which we do not include inequalities (45), (28,29), (32,33), (39–40) and (44) (remember that (34) was not included in TSPPDL3 either). This reduced version of the model is named TSPPDL3* and namely consists of (21)–(27), plus (35,36). One can observe that removing the other inequalities leads to a reduction in the number of instances solved to optimality (eight instead of ten), to a worse average gap (1.63% versus 1.39%) and to a larger average CPU time (992 seconds versus 835 seconds).

We finally compare the two variants of our algorithm with the recent branch-and-bound algorithm of Carrabs et al. [3]. This algorithm is identified as B&B in the last column of Table 4. It was run on a 2.4 GHz AMD Opteron 250 processor, with a time limit of three hours. In this time limit the branch-and-bound fails in solving one instance with $n = 17$, six instances with $n = 21$ and all instances with $n = 25$. On the instances that could be solved to optimality by both algorithms, the branch-and-cut is usually much faster.

# 7 Conclusions

We have formulated and solved a combinatorial optimization problem derived from the TSPPD by performing pickups and deliveries according to a LIFO policy. We have presented the first known ILP models for the TSPPDL. We have improved the linear relaxations of these models by applying valid inequalities from the literature and proposing new ones. We have assessed the behavior of the models and of the inequalities by solving instances from the literature. The branch-and-cut algorithm could solve to optimality all instances with up to 36 nodes, as well as a number of larger instances.

Table 4: Summary of computational results over 45 instances.

| *Instance* | | | *TSPPDL3* | | | | *TSPPDL3** | | | | *B&B [3]* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Graph* | $n$ | $UB$ | *%gap* | *Nodes* | *User* | *Secs* | *%gap* | *Nodes* | *User* | *Secs* | *Secs* |
| a280 | 9 | 402 | 0.00 | 4 | 48 | 0.89 | 0.00 | 0 | 23 | 0.30 | 0.02 |
| att532 | 9 | 4250 | 0.00 | 0 | 13 | 0.09 | 0.00 | 0 | 11 | 0.03 | 0.03 |
| brd14051 | 9 | 4555 | 0.00 | 0 | 8 | 0.03 | 0.00 | 0 | 11 | 0.03 | 1.22 |
| d15112 | 9 | 76203 | 0.00 | 0 | 35 | 0.27 | 0.00 | 0 | 18 | 0.16 | 0.10 |
| d18512 | 9 | 4446 | 0.00 | 0 | 75 | 0.31 | 0.00 | 0 | 17 | 0.13 | 0.04 |
| fnl4461 | 9 | 1866 | 0.00 | 0 | 34 | 0.14 | 0.00 | 0 | 16 | 0.06 | 0.01 |
| nrw1379 | 9 | 2691 | 0.00 | 0 | 0 | 0.03 | 0.00 | 0 | 0 | 0.02 | 0.05 |
| pr1002 | 9 | 12947 | 0.00 | 0 | 41 | 0.13 | 0.00 | 0 | 8 | 0.02 | 0.01 |
| ts225 | 9 | 21000 | 0.00 | 6 | 90 | 0.38 | 0.00 | 8 | 38 | 0.14 | 0.03 |
| a280 | 13 | 505 | 0.00 | 1 | 85 | 1.53 | 0.00 | 15 | 44 | 1.25 | 1.00 |
| att532 | 13 | 5800 | 0.00 | 7 | 62 | 1.70 | 0.00 | 12 | 47 | 0.69 | 6.95 |
| brd14051 | 13 | 4936 | 0.00 | 15 | 120 | 2.30 | 0.00 | 8 | 24 | 0.73 | 149.00 |
| d15112 | 13 | 93158 | 0.00 | 187 | 402 | 12.61 | 0.00 | 327 | 247 | 7.20 | 15.82 |
| d18512 | 13 | 4704 | 0.00 | 139 | 410 | 11.28 | 0.00 | 217 | 239 | 6.83 | 141.47 |
| fnl4461 | 13 | 2483 | 0.00 | 262 | 373 | 12.83 | 0.00 | 361 | 173 | 5.78 | 3.56 |
| nrw1379 | 13 | 3366 | 0.00 | 56 | 217 | 6.27 | 0.00 | 1792 | 576 | 25.63 | 311.32 |
| pr1002 | 13 | 15566 | 0.00 | 0 | 74 | 0.33 | 0.00 | 0 | 38 | 0.13 | 0.43 |
| ts225 | 13 | 32395 | 0.00 | 13 | 155 | 1.95 | 0.00 | 25 | 67 | 1.19 | 5.32 |
| a280 | 17 | 647 | 0.00 | 22 | 203 | 8.17 | 0.00 | 40 | 61 | 4.17 | 24.42 |
| att532 | 17 | 6361 | 0.00 | 64 | 273 | 11.20 | 0.00 | 143 | 311 | 7.98 | 1671.03 |
| brd14051 | 17 | 5196 | 0.00 | 29 | 89 | 8.36 | 0.00 | 38 | 105 | 3.95 | 4958.67 |
| d15112 | 17 | 115554 | 0.00 | 437 | 628 | 46.16 | 0.00 | 623 | 413 | 23.20 | 1935.22 |
| d18512 | 17 | 5186 | 0.00 | 5254 | 2621 | 731.30 | 0.00 | 10696 | 2555 | 984.09 | 9252.33 |
| fnl4461 | 17 | 2852 | 0.00 | 310 | 399 | 31.38 | 0.00 | 427 | 243 | 15.80 | 178.50 |
| nrw1379 | 17 | 3644 | 0.00 | 537 | 849 | 66.78 | 0.00 | 22180 | 3382 | 1678.97 | $> 10800$ |
| pr1002 | 17 | 17564 | 0.00 | 0 | 133 | 0.89 | 0.00 | 0 | 80 | 0.52 | 28.75 |
| ts225 | 17 | 36703 | 0.00 | 38 | 237 | 9.27 | 0.00 | 43 | 174 | 5.09 | 149.39 |
| a280 | 21 | 752 | 0.00 | 28 | 228 | 16.03 | 0.00 | 131 | 138 | 12.55 | 3918.54 |
| att532 | 21 | 10714 | 0.00 | 544 | 973 | 122.84 | 0.00 | 20818 | 4266 | 2069.36 | $> 10800$ |
| brd14051 | 21 | 5719 | 0.00 | 4200 | 2797 | 898.53 | 0.00 | 14725 | 3611 | 1993.53 | $> 10800$ |
| d15112 | 21 | 128798 | 1.80 | 12591 | 6106 | $> 3600$ | 2.38 | 17435 | 4358 | $> 3600$ | $> 10800$ |
| d18512 | 21 | 5634 | 2.01 | 11241 | 4550 | $> 3600$ | 0.80 | 17738 | 3838 | $> 3600$ | $> 10800$ |
| fnl4461 | 21 | 3269 | 0.00 | 16284 | 3014 | 2634.72 | 0.00 | 19180 | 1731 | 1694.34 | $> 10800$ |
| nrw1379 | 21 | 4282 | 1.63 | 10522 | 4566 | $> 3600$ | 6.45 | 14642 | 5334 | $> 3600$ | $> 10800$ |
| pr1002 | 21 | 20173 | 0.00 | 0 | 193 | 4.84 | 0.00 | 19 | 154 | 6.28 | 2435.54 |
| ts225 | 21 | 43082 | 0.00 | 298 | 676 | 56.11 | 0.00 | 849 | 673 | 56.45 | 6120.99 |
| a280 | 25 | 845 | 0.00 | 40 | 229 | 33.27 | 0.00 | 154 | 144 | 23.92 | $> 10800$ |
| att532 | 25 | 11478 | 0.00 | 2065 | 1638 | 582.91 | 1.31 | 15517 | 5140 | $> 3600$ | $> 10800$ |
| brd14051 | 25 | 7539 | 20.10 | 6697 | 8600 | $> 3600$ | 20.52 | 9083 | 7489 | $> 3600$ | $> 10800$ |
| d15112 | 25 | 143654 | 4.18 | 9188 | 4721 | $> 3600$ | 5.42 | 12208 | 4267 | $> 3600$ | $> 10800$ |
| d18512 | 25 | 7291 | 18.94 | 5173 | 7904 | $> 3600$ | 18.87 | 6149 | 7416 | $> 3600$ | $> 10800$ |
| fnl4461 | 25 | 3860 | 9.66 | 8532 | 4813 | $> 3600$ | 8.68 | 12674 | 4087 | $> 3600$ | $> 10800$ |
| nrw1379 | 25 | 4836 | 4.20 | 6409 | 5764 | $> 3600$ | 6.64 | 9263 | 5774 | $> 3600$ | $> 10800$ |
| pr1002 | 25 | 22774 | 0.00 | 16 | 344 | 26.02 | 0.00 | 138 | 339 | 27.23 | $> 10800$ |
| ts225 | 25 | 54386 | 0.00 | 10758 | 3454 | 3413.11 | 2.34 | 15363 | 5444 | $> 3600$ | $> 10800$ |
| *Average* | | 21424 | 1.39 | 2488 | 1517 | | 1.63 | 4956 | 1625 | 992.47 | |

## Acknowledgments

We thank the Italian Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), the Canadian Natural Sciences and Engineering Research Council (grants 227837-04 and 39682-05), and the Spanish Ministerio de Educación (MTM2006-14961-C05-03). Their support is gratefully acknowledged. The computational experiments were executed at the Laboratory of Operations Research of the University of Bologna (LabOR). We are grateful to the Associate Editor and to two 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*, 36:69–79, 2000.

[2] E. Balas, M. Fischetti, and W.R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995.

[3] F. Carrabs, R. Cerulli, and J.-F. Cordeau. An additive branch-and-bound algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Technical Report CIRRELT-2007-12, Centre for Research on Transportation, HEC Montréal, 2007.

[4] F. Carrabs, J.-F. Cordeau, and G. Laporte. Variable neighbourhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing*, 2007. Forthcoming.

[5] L. Cassani. Algoritmi euristici per il TSP with rear-loading. Degree thesis, DTI - Università degli Studi di Milano, 2004.

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

[7] 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, Volume 14, pages 429–466. Elsevier, Amsterdam, 2007.

[8] 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.

[9] M. Desrochers and G. Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

[10] I. Dumitrescu, S. Ropke, J.-F. Cordeau, and G. Laporte. The traveling salesman problem with pickup and delivery: Polyhedral results and a branch-and-cut algorithm. Working paper, Centre for Research on Transportation, HEC Montréal, 2007.

[11] B. Gavish and S. Graves. The travelling salesman problem and related problems. Technical Report 078-78, Operations Research Center, Massachussetts Institute of Technology, 1978.

[12] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40:342–350, 2006.

[13] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 2007. Forthcoming.

[14] L. Gouveia. A result on projection for the vehicle routing problem. *European Journal of Operational Research*, 85:610–624, 1995.

[15] 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.

[16] M. Iori, J.J. Salazar González, and D. Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41: 253–264, 2007.

[17] B. Kalantari, A.V. Hill, and S.R. Arora. An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research*, 22:377–386, 1985.

[18] S. P. Ladany and A. Mehrez. Optimal routing of a single vehicle with loading constraints. *Transportation Planning and Technology*, 8:301–306, 1984.

[19] A.N. Letchford and J.J. Salazar. Projection results for vehicle routing. *Mathematical Programming*, 105:251–274, 2006.

[20] K. Levitin. Organization of computations that enables one to use stack memory optimally. *Soviet Journal of Computer & System Science*, 24:151–159, 1986.

[21] K. Levitin and R. Abezgaouz. Optimal routing of multiple-load AGV subject to LIFO loading constraints. *Computers & Operations Research*, 30:397–410, 2003.

[22] J.D.C. Little, K.G. Murty, D.W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11:972–989, 1963.

[23] J.A. Pacheco. *Problemas de rutas con ventanas de tiempo*. PhD thesis, Departamento de Estadística e Investigación Operativa, Universidad Complutense de Madrid, 1994.

[24] J.A. Pacheco. Problemas de rutas con carga y descarga en sistemas LIFO: soluciones exactas. *Estudios de Economía Aplicada*, 3:69–86, 1995.

[25] J.A. Pacheco. Heuristico para los problemas de ruta con carga y descarga en sistemas LIFO. *SORT, Statistics and Operations Research Transactions*, 21:153–175, 1997.

[26] J.A. Pacheco. Metaheuristic based on a simulated annealing process for one vehicle pick-up and delivery problem in LIFO unloading systems. In *Proceedings of the Tenth Meeting of the European Chapter of Combinatorial Optimization (ECCO X)*, Tenerife, Spain, 1997.

[27] K.S. Ruland and E.Y. Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & Mathematics with Applications*, 33:1–13, 1997.

[28] M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

[29] Y. Volchenkov. Organization of calculations that allows the use of stack memory. *Engineering Cybernetics, Soviet Journal of Computer & System Science*, 20:109–115, 1982.

[30] H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37:347–364, 2003.