Recent Models and Algorithms for One-to-One Pickup and Delivery Problems

Jean-François Cordeau[1], Gilbert Laporte[2], and Stefan Ropke[2]

[1] Canada Research Chair in Logistics and Transportation, HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
`cordeau@crt.umontreal.ca`

[2] Canada Research Chair in Distribution Management, HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
`{gilbert,sropke}@crt.umontreal.ca`

Summary. In one-to-one *Pickup and Delivery Problems* (PDPs), the aim is to design a set of least cost vehicle routes starting and ending at a common depot in order to satisfy a set of pickup and delivery requests between location pairs, subject to side constraints. Each request originates at one location and is destined for one other location. These requests apply to the transportation of goods or people, in which case the problem is often called the dial-a-ride problem. In recent years, there have been several significant developments in the area of exact and heuristic algorithms for PDPs. The purpose of this chapter is to report on these developments. It contains two main sections devoted to single vehicle and multi-vehicle problems, respectively. Each section is subdivided into two parts, one on exact algorithms and one on heuristics.

Key words: Pickup and delivery; one-to-one; dial-a-ride; branch-and-cut; column generation; tabu search.

1 Introduction

In one-to-one *Pickup and Delivery Problems* (PDPs), the aim is to design a set of least cost vehicle routes starting and ending at a common *depot* in order to satisfy a set of pickup and delivery requests between location pairs, subject to side constraints. These problems are called "one-to-one" because each request originates at one location and is destined for one other location (see Hernández-Pérez and Salazar-González [35]). In some contexts, like in courier services, these requests apply to the transportation of goods, whereas in other contexts, like in dial-a-ride problems (DARPs), they apply to the transportation of people. One-to-one PDPs differ from one-to-many-to-one problems in which each customer receives a delivery originating at a common depot and sends a pickup quantity to the depot (see, e.g., Gribkovskaia and Laporte

[33]). They also differ from many-to-many problems in which a commodity may be picked up at one of many locations, and also delivered to one of many locations (see, e.g., Hernández-Pérez and Salazar-González [36]). These three problem structures are depicted in Figures 1, 2, and 3, respectively.

Formally, PDPs are defined on a directed graph $G = (V, A)$, where V is the vertex set and A is the arc set. The vertex set is partitioned into $\{P, D, \{0, 2n+1\}\}$, where $P = \{1, \ldots, n\}$ is a set of pickup vertices, $D = \{n+1, \ldots, 2n\}$ is a set of corresponding delivery vertices, and $\{0, 2n+1\}$ contains two copies of the depot, serving as the starting and ending points of m vehicle routes. The set of vehicles is denoted by $K = \{1, \ldots, m\}$, and Q_k is the capacity of vehicle k. The arc set is defined as $A = \{(i,j) : i = 0, j \in P,$ or $i, j \in P \cup D, i \neq j$ and $i \neq n+j$, or $i \in D, j = 2n+1\}$.

With each arc (i,j) are associated a travel time t_{ij} and a travel cost c_{ij}, or c_{ij}^k if one wishes to stress that the cost is vehicle-dependent. The maximum allowed duration of the route traveled by vehicle k is denoted by T_k. With each vertex $i \in V$ are associated a load q_i and a service duration d_i satisfying $q_0 = q_{2n+1} = 0$, $q_i > 0$ for $i \in P$, $q_i = -q_{i-n}$ for $i \in D$, $d_i \geq 0$ for $i \in P \cup D$, and $d_0 = d_{2n+1} = 0$. A time window $[e_i, \ell_i]$ is associated with each vertex $i \in V$, where e_i and ℓ_i are the earliest and latest time service may start at vertex i. In passenger transportation, it is common to impose a ride time limit L equal to the maximum time a passenger may spend in the vehicle. Pickup and delivery problems consist of designing m vehicle routes of least total cost, starting and ending at the depot, in order to perform all delivery requests subject to the following constraints: vertex i is visited before vertex $n+i$ (precedence), and both of these vertices are visited by the same vehicle (pairing), each vertex is visited within its time window, vehicle capacities are never exceeded, and in some contexts, ride time constraints are satisfied.

Applications of PDPs to goods transportation have been described by Shen et al. [70] in the context of courier services, and by Fisher and Rosenwein [31], Christiansen and Nygreen [12, 13] and Brønmo et al. [7] in the context of bulk product transportation by ship, a sector in rapid expansion. Solanki and Southworth [72] and Rappoport et al. [57, 58] have studied PDP applications to military airlift planning. Dial-a-ride planning systems have been described by a number of authors. Madsen et al. [48] have constructed a system capable of handling in a dynamic fashion 50,000 requests per year in the city of Copenhagen. Toth and Vigo [74, 75] have developed a parallel insertion heuristic which has been applied to a complex DARP involving taxis and minibuses in Bologna. Borndörfer et al. [6] have proposed a set partitioning based heuristic which can solve a problem containing between 1,000 and 1,500 transportation requests per day in Berlin. More recently, Rekiek, Delchambre and Saleh [60] have developed a genetic algorithm for the DARP and have tested their system on real data provided by the City of Brussels. This instance contains 164 requests and 18 vehicles. An important DARP variant is the dial-a-flight problem, faced by about 3,000 businesses offering charter flight services in the United States (Cordeau et al. [20]). Recent survey articles

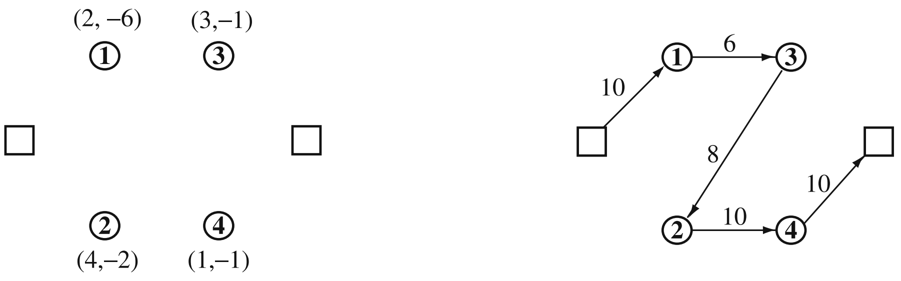

Fig. 1. One-to-many-to-one problem. Vertex label $(x, -y)$ means that the vertex supplies x units and demands y units. Arc labels indicate vehicle load.

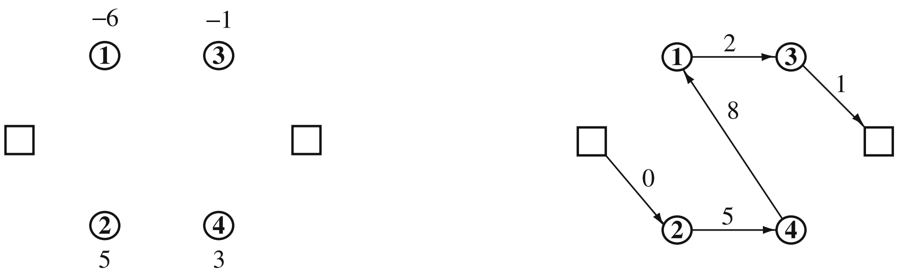

Fig. 2. Many-to-many problem. A positive vertex label x means that the vertex supplies x units; a negative vertex label $-y$ means that the vertex demands y units. Arc labels indicate vehicle load.

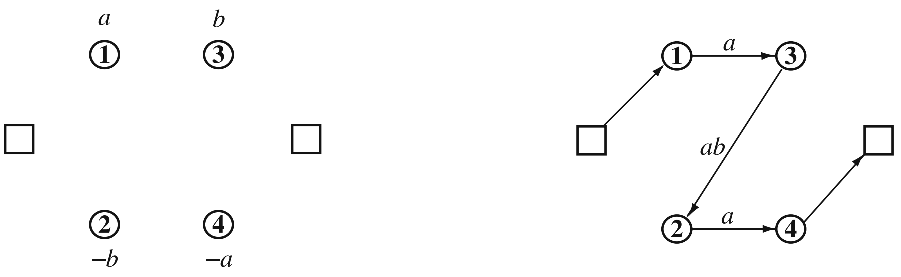

Fig. 3. One-to-one problem. Vertex label z means that the vertex supplies commodity z; vertex label $-z$ means that the vertex demands commodity z. Arc labels show the commodities carried by the vehicle.

on the PDP and its applications are those of Desaulniers et al. [23], Cordeau et al. [20], and Berbeglia et al. [3].

The early research efforts on PDPs can be traced back to the work of Wilson et al. [78, 79] who developed scheduling algorithms for the DARP. Since then several exact and heuristic algorithms have been proposed for PDPs associated with the transportation of goods or people. Significant progress has occurred in the past five years, with the development of new exact and approximate algorithms for several types of PDPs. These exact algorithms employ decomposition techniques such as branch-and-cut and branch-and-cut-and-price, while the new heuristics are based on tabu search, simulated annealing and variable neighbourhood search. While all these algorithmic techniques have now been known for some time, their massive application to PDPs is significant and has enabled researchers to break new grounds in the difficulty and size of problems that can be tackled. Our aim is to report on these new and exciting developments.

Because our focus is on recent contributions, we do not claim to provide a comprehensive and systematic coverage of the field, but rather a selective coverage of some of the most significant new algorithmic ideas. This chapter contains two main sections devoted to single vehicle and multi-vehicle problems, respectively. Each section is subdivided into two parts, one on exact algorithms and one on heuristics. Conclusions follow.

2 Single Vehicle Pickup and Delivery Problems (SVPDPs)

While most routing problems arising in practice involve several vehicles, the single vehicle case is instrumental in developing insights into the problem structure and in putting forward new algorithmic concepts. As a case in point, several exact and approximate algorithms for the *Classical Vehicle Routing Problem* (VRP) (see, e.g., Toth and Vigo [76]), are rooted in concepts that were first developed for the *Traveling Salesman Problem* (TSP), (see, e.g., Lawler et al. [42]). All known algorithmic approaches for single-vehicle PDPs stem from TSP algorithms, and may be instrumental in the development of algorithms for multi-vehicle PDPs.

In all versions of the SVPDP discussed in this section, the vehicle capacity is not binding, there are no time windows and no ride time limits. There exist meaningful applications where such constraints may be present. These can sometimes be treated as special cases of multi-vehicle PDPs.

2.1 Exact Algorithms for the SVPDP

The first algorithms developed for the SVPDP and its variants, including the *Traveling Salesman Problem with Precedence Constraints* (TSPPC), were based on branch-and-bound (Kalantari, Hill and Arora [38]), dynamic programming (Desrosiers, Dumas and Soumis [25], Bianco, Mingozzi and Ricciardelli [4]), and branch-and-cut (Balas, Fischetti and Pulleyblank [1]). In addition, Fischetti and Toth [30] have developed an additive lower bounding procedure which can be embedded within a branch-and-bound framework, and have applied this methodology to the solution of the TSPPC.

A Branch-and-cut Algorithm for the SVPDP

The most popular methodology for the solution of the SVPDP is now branch-and-cut. The two key components of this method are the generation of valid inequalities and the design of separation procedures. Our emphasis is on the modeling aspects. Recent branch-and-cut algorithms for single vehicle PDPs are rooted in the work of Ruland [65] and Ruland and Rodin [66]. These authors have considered the undirected case, i.e., when the problem is defined on a graph $G = (V, E)$ where $E = \{(i, j) = (j, i) : i, j \in V, i < j\}$ is an edge set, and the solution is a Hamiltonian cycle.

In addition to the notation already introduced, define $\bar{S} = V \backslash S$ for $S \subseteq V$, $\pi(S) = \{i \in P : n + i \in S\}$ as the set of predecessors of $S \subseteq V\backslash\{0\}$, and $\sigma(S) = \{i \in D : i - n \in S\}$ as the set of successors of $S \subseteq V\backslash\{2n + 1\}$. Let $\delta(S) = \{(i, j) : i \in S, j \notin S\}$ be the set of edges with exactly one end-vertex in $S \subseteq V$. For simplicity, we write $\delta(i)$ instead of $\delta(\{i\})$. For $S, T \in V$, let $(S : T) = \{(i, j) : i \in S, j \in T\}$ be the set of edges with one end-point in S, and one in T. The single-vehicle PDP can be formulated with binary variables x_{ij} equal to 1 if and only if edge (i, j) belongs to the cycle. For $E' \subseteq E$, let $x(E') = \sum\limits_{(i,j)\in E'} x_{ij}$; for $S \subseteq V$, let $x(S) = \sum\limits_{i,j\in S} x_{ij}$. We write $x(S : T)$ instead of $x\left((S : T)\right)$. The model proposed by Ruland is then as follows.

(SVPDP)

$$\text{Minimize} \qquad \sum_{i=0}^{2n} \sum_{j=i+1}^{2n+1} c_{ij}x_{ij} \tag{1}$$

subject to

$$x_{0,2n+1} = 1 \tag{2}$$
$$x(\delta(i)) = 2 \qquad (i \in V) \tag{3}$$
$$x(S : \bar{S}) \geq 2 \qquad (S \subset V) \tag{4}$$
$$x(U : \bar{U}) \geq 4 \qquad (U \in \mathcal{U}) \tag{5}$$
$$x_{ij} = 0 \text{ or } 1 \qquad ((i, j) \in E), \tag{6}$$

where $\mathcal{U}$ is the collection of all sets $U \subset P \cup D$ satisfying $3 \leq |U| \leq |V| - 2$, $0 \in U$, $2n + 1 \notin U$ and there exists $i \in P\backslash U$ with $n + i \in U$. In this model,

(3) are the degree constraints, (4) are the connectivity constraints, and (5) are the precedence constraints. Indeed, any feasible solution must contain a chain between each of the following four vertex pairs: 0 and i, i and $n+i$, $n+i$ and $2n+1$, $2n+1$ and 0, and each of these chains contains an edge connecting U and $\bar{U}$. This model can be reinforced through the introduction of valid inequalities. Each family of valid inequalities gives rise to another family by interchanging the roles of pickup and delivery vertices, and the roles of the two depots.

Generalized order constraints (Ruland and Rodin [66])

Let $S_1, \ldots, S_h \subset P \cup D$ be disjoint sets such that $S_i \cap \sigma(S_{i+1}) \neq \emptyset$ ($i = 1, \ldots, h$), where $S_{h+1} = S_1$. Then the inequality

$$\sum_{i=1}^{h} x(S_i) \leq \sum_{i=1}^{h} |S_i| - h - 1 \tag{7}$$

is valid for the SVPDP.

Similar constraints called precedence cycle breaking inequalities, were proposed by Balas, Fischetti and Pulleyblank [1].

Order-matching constraints (Ruland and Rodin [66], Dumitrescu et al. [29])

For $i_1, \ldots, i_h \in P$ and $H \subseteq (P \cup D) \backslash \{n+i_1, \ldots, n+i_h\}$ such that $\{i_1, \ldots, i_h\} \subseteq H$, then the inequality

$$x(H) + \sum_{j=1}^{h} x_{i_j, n+i_j} \leq |H| \tag{8}$$

is valid for the SVPDP.

Ruland and Rodin [66] proved this result for h even. Dumitrescu et al. [29] have shown that it also holds for h odd.

Generalized order matching constraints (Cordeau [14])

For $i_1, \ldots, i_h \in P, H \subseteq V \backslash \{0, 2n+1\}, T_j \subset P \cup D$ ($j = 1, \ldots, h$) such that $\{i_j, n+i_j\} \subseteq T_j, T_i \cap T_j = \emptyset$ ($i \neq j$) and $H \cap T_j = \{i_j\}$ ($j = 1, \ldots, h$), the inequality

$$x(H) + \sum_{j=1}^{h} x(T_j) \leq |H| + \sum_{j=1}^{h} |T_j| - 2h \tag{9}$$

is valid for the SVPDP.

Constraints (9), which generalize (8), were proved by Cordeau in the context of the DARP but they also apply to the SVPDP.

σ-inequalities (Balas, Fischetti and Pulleyblank [1])

For $S \subseteq V \backslash \{0\}$, the inequality

$$x\left(S \backslash \sigma(S) : \bar{S} \backslash \sigma(S)\right) \geq 1 \tag{10}$$

is valid for the SVPDP.

These inequalities were introduced by Balas, Fischetti and Pulleyblank [1] in the context of the TSPPC.

Lifted subtour elimination constraints (Dumitrescu et al. [29])

Let $S \subseteq P \cup D$ be such that there exists $i \in P$ such that $i \in S, n+i \in S$. Then the inequality

$$x(S) + \sum_{j \in P \cap S, n+j \in \bar{S}} x_{i,n+j} \leq |S| - 1 \tag{11}$$

is valid for the SVPDP.

Dumitrescu et al. [29] also prove the following generalization of (11).

Generalized lifted subtour elimination constraints (Dumitrescu et al. [29])

Let $S \subset P \cup D$ be such that there exists $i \in P \cap S$ with $n+i \in S$. Let $T_k \subset P \cup D$, $k = 1, \ldots, p$, be p sets such that there exists $i_k \in P \cap S$ and $n + i_k \in T_k$, $T_k \cap S = \{i\}$ for $k = 1, \ldots, p$, and $T_j \cap T_k = \{i\}$ for all $j, k = 1, \ldots, p$, $j \neq k$. Then the inequality

$$x(S) + \sum_{k=1}^{p} x(T_k) \leq |S| - 1 + \sum_{k=1}^{p} (|T_k| - 2) \tag{12}$$

is valid for the SVPDP.

Terminal inequalities (Dumitrescu et al. [29])

Let $S \subset V$ and $T \subset D$ be such that $0 \in S$, $2n+1 \in \bar{S}$, $S \cap T = \emptyset$ and $\pi(T) \cap S = \emptyset$. Then the inequality

$$2x(S) + x(S : T) \leq 2(|S| - 1) \tag{13}$$

is valid for the SVPDP.

Dumitrescu et al. [29] also provide a number of other more complicated valid inequalities for the SVPDP. It is worth noting that constraints (7) are not in general facet defining for the SVPDP polytope, constraints (8) are facet defining for $H = \{i_1, \ldots, i_h\}$, and precedence constraints (5) are sometimes facet defining.

The branch-and-cut algorithm of Dumitrescu et al. [29] embeds exact separation procedures for constraints (4) and (5) used by Ruland and Rodin [66], as well as exact separation procedures for constraints (7) with $h = 2$, for constraints (8) with $h = 2$ or 3, and for constraints (11). It also contains heuristic separation procedures for constraints (7) with $h \geq 3$, and for constraints (9), (10), (12) and (13). The algorithm branches on the x_{ij} variables. The algorithm uses strong branching and a best-bound node selection strategy.

The algorithm was run on an AMD Opteron 250 computer (2.4 GHz) running Linux, using CPLEX 10.0 and the Concert library. It was tested on random Euclidean instances with vertices generated in $[0, 1000]^2$, and containing between 5 and 30 requests (between 12 and 62 vertices), as well as on the Renaud, Boctor and Laporte [61] instances which were adapted from some TSPLIB instances (Reinelt [59]), and contain up to 50 requests. Results show that the lower bound at the root of the search tree after the generation of connectivity constraints (4) and precedence constraints (5) is on average over 85% of the optimal solution value. Generating valid inequalities (7) to (13) closes between 47% and 74% the residual gap, depending on the type of instances. The largest instance solved to optimality within two hours of computing time contain 30 requests (62 vertices).

A Branch-and-cut Algorithm for the SVPDP with LIFO Constraints

An interesting variant of the SVPDP arises when a last-in-first-out (LIFO) rule is imposed on the pickup and delivery operations. This means that when a load is picked up, it is placed on top of a stack and can only by unloaded when it is in that position. This problem, abbreviated as SVPDPL, was recently modeled and solved by Cordeau et al. [16]. It arises naturally in the transportation of heavy or fragile goods which are loaded linearly into a vehicle equipped with a single back door. Levitin and Abezgaouz [43] describe another application encountered in the operations of multi-load automated guided vehicles operating in a LIFO fashion. The first exact algorithms proposed for this problem used branch-and-bound (Pacheco [54]). An additive branch-and-bound algorithm combining lower bounds based on the assignment and shortest spanning r-arborescence relaxations was also recently developed by Carrabs, Cerulli and Cordeau [9].

The structure of a feasible SVPDPL solution $(i_1 = 0, i_2, \ldots, i_{2n+2} = 2n + 1)$ is such that if the solution is arranged on a line, and the origin of each request is linked to its destination by an arc, then no arcs will cross. Put differently, if vertex $n + i$ is relabeled i, then the solution consists of nested palindromes.

Cordeau et al. [16] have proposed three formulations for the SVPDPL. We only report the third one which is the most compact in terms of the decision variables, and also yields the best performance. Because the SVPDPL is naturally directed, it is defined on a graph $G = (V, A)$, where $A = \{(i, j) :$

$i, j \in V, i \neq j\}$ is the arc set. Binary variables x_{ij} take the value 1 if and only if arc (i,j) belongs to the optimal circuit. The sets $\delta^+(i) = \{(i,j) : j \in V\backslash\{i\}\}$ and $\delta^-(i) = \{(j,i) : j \in V\backslash\{i\}\}$ contain the arcs leaving and entering i, respectively. The model is the following.

(SVPDPL)

$$\text{Minimize} \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{14}$$

subject to

$$x(\delta^+(i)) = 1 \qquad (i \in V\backslash\{2n+1\}) \tag{15}$$

$$x(\delta^-(i)) = 1 \qquad (i \in V\backslash\{0\}) \tag{16}$$

$$x(S) \leq |S| - 1 \qquad (S \subseteq P \cup D, |S| \geq 2) \tag{17}$$

$$x(S) \leq |S| - 2 \qquad (S \in \mathcal{U}) \tag{18}$$

$$x(i : S) + x(S) + x(S : n+i) \leq |S| \qquad (S \in \mathcal{W}, i, n+i \notin S, i \in P) \tag{19}$$

$$x_{ij} = 0 \text{ or } 1 \qquad ((i,j) \in A)\,, \tag{20}$$

where $\mathcal{W}$ is the collection of all subsets $S \subset P \cup D$ for which at least one request $(j, n+j)$ is such that $j \in S$ and $n+j \notin S$, or $j \notin S$ and $n+j \in S$, and $\mathcal{U}$ has been defined in Section 2.1. In this model, constraints (15) and (16) are degree constraints, while connectivity, precedence and LIFO restrictions are enforced through constraints (17), (18), and (19), respectively.

Since the SVPDPL is a restriction of the SVPDP, any inequality valid for the SVPDP is also valid for the SVPDPL. In addition, Cordeau et al. [16] show that the following inequalities are valid for the SVPDPL.

Incompatible successor inequalities (Cordeau et al. [16])

Let $S_{n+j}(i,j) = \{n+i\} \cup (P\backslash\{i\})$ be the set of possible successors of vertex $n+j$ if arc (i,j) is used. Then the inequality

$$x_{ij} + \sum_{\ell \notin S_{n+j}(i,j)} x_{n+j,\ell} \leq 1 \qquad (i, j \in P, i \neq j) \tag{21}$$

is valid for the SVPDPL.

Incompatible predecessor inequalities (Cordeau et al. [16])

Similarly, let $P_i(n+i, n+j) = \{j\} \cup (D\backslash\{n+j\})$ be the set of possible predecessors of vertex i if arc $(n+i, n+j)$ is used. Then the inequality

$$x_{n+i,n+j} + \sum_{\ell \notin P_i(n+i,n+j)} x_{\ell i} \leq 1 \qquad (i, j \in P, i \neq j) \tag{22}$$

is valid for the SVPDPL.

Hamburger inequalities (Cordeau et al. [16])

The inequality

$$x_{ij} + x_{n+i,n+j} + x_{n+j,i} + x_{n+i,j} \leq 1 \qquad (i, j \in P, i \neq j) \tag{23}$$

is valid for the SVPDPL. Also, let $k \geq 3$ and consider an ordered subset of requests defined by the indices $\{i_1, \ldots, i_k\}$, where $i_{k+1} = i_1$ and $i_0 = k$. Then the inequality

$$\sum_{j=1}^{k} x_{i_j,i_{j+1}} + \sum_{j=1}^{k} x_{n+i_j,n+i_{j+1}} + \sum_{j=1}^{k} x_{n+i_j,i_{j-1}} \leq k - 1 \tag{24}$$

is valid for the SVPDPL.

Incompatible path inequalities (Cordeau et al. [16])

Let P_{ij} be the arc set of a path from i to j not containing vertex $n + i$. Similarly, let $P_{n+i,n+j}$ be the arc set of a path from $n + i$ to $n + j$. Then the inequality

$$\sum_{(h,k)\in P_{ij}} x_{hk} + \sum_{(h,k)\in P_{n+i,n+j}} x_{hk} \leq |P_{ij}| + |P_{n+i,n+j}| - 1 \tag{25}$$

is valid for the SVPDPL.

Cordeau et al. [16] have devised a branch-and-cut algorithm for the SVPDPL, incorporating these inequalities. Exact procedures are used for the separation of constraints (17), (18), and (19), while heuristics are used for the remaining valid inequalities. The algorithm uses standard CPLEX parameters. The algorithm was tested on 36 benchmark instances containing at most 25 requests (52 vertices). Twenty-nine of these instances could be solved to optimality within an hour on a Pentium IV 3 GHz, using CPLEX 9.0 as ILP solver. The percentage gap at the root was only 1.82%.

2.2 Heuristics for the SVPDP

A number of heuristics have also been proposed for several versions of the SVPDP. The most common version, involving time windows, has been solved by local search (Van der Bruggen, Lenstra and Schuur [77]), tabu search (Landrieu, Mati and Binder [40]), and genetic search (Pankratz [55]). A perturbation heuristic was also proposed by Renaud, Boctor and Laporte [61] for the SVPDP without time windows. A recent article by Cordeau and Laporte [18] surveys the single vehicle DARP literature.

A Heuristic for the SVPDPL

Carrabs, Cordeau and Laporte [10] have developed a variable neighbourhood search (VNS) heuristic for the SVPDPL. This technique, put forward by Mladenović and Hansen [51], is a local search framework in which the neighbourhood structure is allowed to vary during the search.

The search procedure applies eight operators. The first four were introduced by Cassani and Righini [11], the next three are due to Carrabs, Cordeau and Laporte [10], and the last one calls four of the seven first operators. When implementing these operators one must ensure that the LIFO property of the solution remains satisfied. In several cases, preserving LIFO feasibility requires carrying out complicated checks and handling appropriate data structures in order to maintain a low complexity. Here is a short description of these operators.

1) ***Couple-exchange:*** Select two requests $(i, n+i)$ and $(j, n+j)$. Swap the positions of i and j and of $n+i$ and $n+j$.
2) ***Block-exchange:*** A block B_i is the path $(i, \ldots, n+i)$. This procedure works like the previous one, except that it swaps blocks B_i and B_j, instead of just their extremities.
3) ***Relocate-block:*** This procedure relocates a block B_i in the best possible position.
4) ***Relocate-couple:*** This operator relocates a request $(i, n+i)$ in the best position.
5) ***Multi-relocate:*** This operator works like relocate-couple, except that it first computes the cost of relocating each request and implements the best move. However, it saves in a queue every request whose relocation produces a better tour to relocate the best request identified, and then attempts to relocate as many requests as possible to further improve the tour.
6) ***2-opt-L:*** Denote a solution by $(i_1 = 0, \ldots, i_{2n+2} = 2n+1)$. This procedure is an adaptation of the classical 2-opt operator for the TSP (Croes [21]). It substitutes two arcs (i_j, i_{j+1}) and (i_k, i_{k+1}) with two other arcs (i_j, i_k) and (i_{j+1}, i_{k+1}) and reverses the path $(i_{j+1}, \ldots, i_k)$.
7) ***Double-bridge:*** This operator is used to perturb the solution during the VNS algorithm. It works as the classical double-bridge operator (Lin and Kerninghan [45]). It replaces the arcs (i_j, i_{j+1}), (i_k, i_{k+1}), $(i_\ell, i_{\ell+1})$ and (i_h, i_{h+1}) with $(i_j, i_{\ell+1})$, (i_k, i_{h+1}), (i_ℓ, i_{j+1}) and (i_h, i_{k+1}).
8) ***Shake:*** This is another perturbation operator which randomly calls couple-exchange, block-exchange, relocate-couple, or relocate-block.

Procedures 1, 2, 3, 6, 7 were implemented to run in $O(n^2)$ time while procedures 4 and 5 require $O(n^3)$ time. In the VNS heuristic, local search is applied to a starting solution s until a local minimum s_1 has been reached, and is perturbed into another solution s_2. Local search is again applied to s_2 until a local minimum s_3 is reached. Finally, a decision criterion is applied to determine whether the search should restart from s_3 or from the incumbent s^*. The larger the cost of s_3 and the number of different arcs between s_3 and s^*, the lower is the probability of restarting from s_3.

The starting solution is obtained through one of the eight constructive procedures described by Cassani and Righini [11]. The neighbourhoods couple-exchange, block-exchange, relocate-block, 2-opt-L and multi-relocate are then applied in one of two possible orders. To perturb the solution, the double-bridge and shake operators are applied, with a tabu mechanism in the latter case.

Tests were performed on 42 instances derived from TSP instances of TSPLIB (Reinelt [59]), and containing between 12 and 350 requests. All instances were solved using the VNS heuristic and the variable neighbourhood descent (VND) heuristic of Cassani and Righini [11]. In all cases, VNS produced better solutions than VND, at the expense of an increase in computing time. In half the cases the difference in solution costs between the two algorithms was in excess of 5%. Tests were also performed to study the individual impact of each operator by successively removing each of them. The multi-relocate operator proved to be the most useful, while couple-exchange and 2-opt-L were the least useful. Comparisons with the optimal values of the Cordeau et al. [16] algorithm show that on instances with $7 \leq n \leq 25$, the VNS heuristic yields solutions whose values lie on average within 0.19% of the optimum (Carrabs [8]).

3 Multi-vehicle Pickup and Delivery Problems (MVPDPs)

Most of the research effort on PDPs is related to the multi-vehicle case. In what follows we present some of the most recent exact and approximate algorithms for MVPDPs.

3.1 Exact Algorithms for the MVPDP

The most popular exact algorithms for static MVPDPs are based on column generation (Dumas, Desrosiers and Soumis [27], Savelsbergh and Sol [68]). Within a very short time span, three new exact algorithms have been put forward for two basic variants of the MVPDP, and each improves upon its predecessors. The first two use classical branch-and-cut, while the third also embeds a pricing mechanism.

A Branch-and-cut Algorithm for the DARP

Cordeau [14] formulates the DARP on a directed graph $G = (V, A)$, using binary three-index variables x_{ij}^k equal to 1 if and only if arc (i, j) is traversed by vehicle k. For $S \subseteq P \cup D$, let $q(S) = \sum_{i \in S} q_i$. In addition, let u_i^k be the time at which vehicle k starts servicing vertex i, w_i^k the load of vehicle k upon leaving vertex i, and r_i^k the ride time of user i (corresponding to request $(i, n+i)$ on vehicle k). The model is then as follows.

(DARP)

$$\text{Minimize} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij}^k x_{ij}^k \tag{26}$$

subject to

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \qquad (i \in P) \tag{27}$$

$$\sum_{i \in V} x_{0i}^k = \sum_{i \in V} x_{i,2n+1}^k = 1 \qquad (k \in K) \tag{28}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{n+i,j}^k = 0 \qquad (i \in P, k \in K) \tag{29}$$

$$\sum_{j \in V} x_{ji}^k - \sum_{j \in V} x_{ij}^k = 0 \qquad (i \in P \cup D, k \in K) \tag{30}$$

$$u_j^k \geq (u_i^k + d_i + t_{ij}) x_{ij}^k \qquad (i, j \in V, k \in K) \tag{31}$$

$$w_j^k \geq (w_i^k + q_j) x_{ij}^k \qquad (i, j \in V, k \in K) \tag{32}$$

$$r_i^k \geq u_{n+i}^k - (u_i^k + d_i) \qquad (i \in P, k \in K) \tag{33}$$

$$u_{2n+1}^k - u_0^k \leq T_k \qquad (k \in K) \tag{34}$$

$$e_i \leq u_i^k \leq \ell_i \qquad (i \in V, k \in K) \tag{35}$$

$$t_{i,n+i} \leq r_i^k \leq L \qquad (i \in P, k \in K) \tag{36}$$

$$\max\{0, q_i\} \leq w_i^k \leq \min\{Q_k, Q_k + q_i\} \qquad (i \in V, k \in K) \tag{37}$$

$$x_{ij}^k = 0 \text{ or } 1 \qquad (i, j \in V, k \in K). \tag{38}$$

In this formulation, constraints (27) and (29) ensure that each request is served once by the same vehicle, while constraints (28) and (30) guarantee that each vehicle starts and ends its route at the depot. Constraints (31) to (33) define starts of service times, vehicle loads and user ride times, respectively, while constraints (34) to (37) ensure that these will be feasible.

The u_i^k variables can be aggregated into vehicle-independent u_i variables for $i \in P \cup D$. Constraints (31) and (32) can be linearized using standard techniques. These linearized constraints, as well as constraints (35) and (37) can be lifted as in Desrochers and Laporte [24].

Cordeau proposes a number of valid inequalities for this model. Define $x_{ij} = \sum_{k \in K} x_{ij}^k$, $x(A') = \sum_{(i,j) \in A'} x_{ij}$ for $A' \subseteq A$, and $x(S) = \sum_{i,j \in S} x_{ij}$ for $S \subseteq V$.

σ-inequalities and π-inequalities (Balas, Fischetti and Pulleyblank [1])

The standard subtour elimination constraints $x(S) \leq |S| - 1$ $(S \subseteq P \cup D)$ are of course valid for the DARP. In the directed case, precedence relationships yield the following liftings:

$$x(S) + \sum_{i \in \bar{S} \cap \sigma(S)} \sum_{j \in S} x_{ij} + \sum_{i \in \bar{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} \leq |S| - 1 \ (S \subseteq P \cup D) \quad (39)$$

and

$$x(S) + \sum_{i \in S} \sum_{j \in \bar{S} \cap \pi(S)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in \bar{S} \setminus \pi(S)} x_{ij} \leq |S| - 1 \ (S \subseteq P \cup D). \quad (40)$$

Lifted D_k^+ and D_k^- inequalities (Cordeau [14])

The following subtour elimination constraints are obtained by lifting the so-called D_k^+ and D_k^- inequalities proposed by Grötschel and Padberg [34] for the asymmetric TSP. Let $S = \{i_1, \ldots, i_h\} \subseteq P \cup D$, where $h \geq 3$. Then the inequalities

$$\sum_{j=1}^{h-1} x_{i_j i_{j+1}} + x_{i_h i_1} + 2 \sum_{j=2}^{h-1} x_{i_j i_1} + \sum_{j=3}^{h-1} \sum_{\ell=2}^{j-1} x_{i_j i_\ell} + \sum_{n+i_p \in \bar{S} \cap \sigma(S)} x_{n+i_p, i_1} \leq h - 1 \quad (41)$$

and

$$\sum_{j=1}^{h-1} x_{i_j i_{j+1}} + x_{i_h i_1} + 2 \sum_{j=3}^{h} x_{i_1 i_j} + \sum_{j=4}^{h} \sum_{\ell=3}^{j-1} x_{i_j i_\ell} + \sum_{i_p \in \bar{S} \cap \pi(S)} x_{i_1, i_p} \leq h - 1 \quad (42)$$

are valid for the DARP.

Capacity constraints (Laporte, Nobert and Desrochers [41], Cordeau [14])

The standard VRP capacity constraints

$$x(\delta(S)) \geq 2 \lceil q(S)/Q \rceil \ (S \subseteq P \cup D) \quad (43)$$

where $Q = \max_{k \in K} \{Q_k\}$, are valid for the DARP.

Lifted generalized order constraints (Cordeau [14])

Let $S_1, \ldots, S_h \subset P \cup D$ be disjoint sets and let $i_1, \ldots, i_h \in P$ be such that $0, 2n+1 \notin S_\ell$ and $i_\ell, n+i_{\ell+1} \in S_\ell$ for $\ell = 1, \ldots, h$, where $i_{h+1} = i_1$. Then the generalized order constraints (7), can be lifted as follows in the case of a directed formulation:

$$\sum_{i=1}^{h} x(S_\ell) + \sum_{\ell=2}^{h-1} x_{i_1,i_\ell} + \sum_{\ell=3}^{h} x_{i_1,i_{n+i_\ell}} \leq \sum_{\ell=1}^{h} |S_\ell| - h - 1 \tag{44}$$

and

$$\sum_{\ell=1}^{h} x(S_\ell) + \sum_{\ell=2}^{h-2} x_{n+i_1,i_\ell} + \sum_{\ell=2}^{h-1} x_{n+i_1,n+i_\ell} \leq \sum_{\ell=1}^{h} |S_\ell| - h - 1. \tag{45}$$

Infeasible path constraints (Cordeau [14])

The following inequalities make use of the maximum ride time constraints and are specific to the DARP. Assume the t_{ij} satisfy the triangle inequality. Then for any path $(i, k_1, \ldots, k_p, n+i)$ such that $t_{ik_1} + d_{k_1} + t_{k_1k_2} + d_{k_2} + \ldots + t_{k_pn+i} > L$, the inequality

$$x_{ik_1} + \sum_{h=1}^{p-1} x_{k_hk_{h+1}} + x_{k_pn+i} \leq p - 1 \tag{46}$$

is valid for the DARP.

Using the DARP model and the associated valid inequalities, Cordeau [14] has devised a branch-and-cut algorithm incorporating a preprocessing phase (time window tightening, arc elimination and variable fixing), as well as separation heuristics for subtour elimination constraints, capacity constraints, generalized order constraints and infeasible path inequalities.

The algorithm was implemented in `C++` with ILOG Concert 1.3 and CPLEX 8.1. It was run on a 2.5 GHz Pentium 4 computer. In the branch-and-cut algorithm additional aggregate variables $y_i^k = \sum_{j \in V} x_{ij}^k, i \in P, k \in K$ are added to the model. Valid inequalities are only added at the root node and whenever all y_i^k variables are integer. The algorithm first branches on y_i^k variables and only selects x_{ij}^k variables for branching when all y_i^k are integer. The algorithm was tested on 48 randomly generated instances with $16 \leq n \leq 48$ $(34 \leq |V| \leq 98)$. It is shown that the preprocessing phase played an important role in reducing the instance size and in increasing the lower bound at the root of the search tree. Valid inequalities at the root of the tree helped increase the lower bound by about 5%. Instances containing up to 30 requests could be solved optimally within four hours.

A Branch-and-cut Algorithm for the PDPTW and the DARP

More recently, Ropke, Cordeau and Laporte [62] have proposed two models and a branch-and-cut algorithm for the PDP with time windows (PDPTW) and for the DARP, where all vehicles are identical. The PDPTW is a DARP without the maximum ride time constraints. Here we describe the better of the two models. It works with a homogeneous fleet of vehicles of capacity Q and two-index variables x_{ij}. In this model, R denotes a path, $\mathcal{R}$ is the set of infeasible paths with respect to time windows and maximum ride time constraints, and $A(R)$ is the arc set of R. The model is as follows.

(PDPTW-DARP)

$$\text{Minimize} \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} \tag{47}$$

subject to

$$\sum_{i\in V} x_{ij} = 1 \qquad (j \in P \cup D) \tag{48}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad (i \in P \cup D) \tag{49}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 2 \qquad (S \in \mathcal{U}) \tag{50}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - \max\{1, \lceil |q(S)|/Q \rceil\} \qquad (S \subseteq P \cup D, |S| \geq 2) \tag{51}$$

$$\sum_{(i,j)\in A(R)} x_{ij} \leq |A(R)| - 1 \qquad (R \in \mathcal{R}) \tag{52}$$

$$x_{ij} = 0 \text{ or } 1 \qquad (i, j \in V). \tag{53}$$

In this model, precedence constraints (50) are the same as (18), constraints (51) are capacity constraints, and constraints (52) eliminate infeasible paths. An immediate strengthening of this constraint is provided by the so-called tournament constraints. Let $R = (k_1, \dots, k_r)$ be an infeasible path, then

$$\sum_{i=1}^{r-1}\sum_{j=i+1}^{r} x_{k_i k_j} \leq |A(R)| - 1 \tag{54}$$

is a valid inequality for the PDPTW. In addition, if $R' = (k_r, \dots, k_1)$ is also infeasible, then

$$\sum_{i=1}^{r-1} \left(x_{k_i k_{i+1}} + x_{k_{i+1} k_i}\right) \leq r - 1$$

is also valid. Finally, if the t_{ij} satisfy the triangle inequality and $R = (i, k_1, \ldots, k_r, n+i)$ violates the time window or ride time constraints, then (46) is also valid. All valid inequalities developed by Cordeau [14] for the DARP, except the infeasible path constraints (46), apply directly to the PDPTW. Some additional valid inequalities have also been proposed for the PDPTW.

Strengthened capacity constraints (Ropke, Cordeau and Laporte [62])

Let $S, T \subset P \cup D$ be two disjoint sets such that $q(S) > 0$. Also define $U = \pi(T) \backslash (S \cup T)$. Then the constraint

$$x(S) + x(T) + x(S:T) \leq |S| + |T| - \left\lceil \frac{q(S) + q(U)}{Q} \right\rceil \tag{55}$$

is valid for the PDPTW.

Strengthened infeasible path constraints (Ropke, Cordeau and Laporte [62])

If travel times satisfy the triangle inequality and the two paths $(j, i, n+j, k, n+i, n+k)$ and $(j, i, n+j, k, n+k, n+i)$ are infeasible, then the solution cannot contain the path $R = (i, n+j, k)$ and therefore

$$x_{i,n+j} + x_{n+j,k} \leq 1 \tag{56}$$

is valid for the PDPTW. This inequality generalizes to longer paths.

Fork inequalities (Ropke, Cordeau and Laporte [62])

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 $S, T \subset V$, then the inequality

$$\sum_{i \in S} x_{ik_1} + \sum_{h=1}^{r-1} x_{k_h k_{h+1}} + \sum_{j \in T} x_{k_r j} \leq r \tag{57}$$

is valid for the PDPTW.

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

$$\sum_{i \in S} x_{ik_1} + \sum_{h=1}^{r-1} x_{k_h k_{h+1}} + \sum_{h=1}^{r} \sum_{j \in T_h} x_{k_h j} \leq r \tag{58}$$

is valid for the PDPTW.

Similarly, let $k_j \notin S_{j+1}$ for $j = 1, \ldots, r-1$. If for any integer $h \leq r$ and any vertex pair $\{i \in S_h, j \in T\}$ the path $(i, k_h, \ldots, k_r, j)$ is infeasible, then the infork inequality

$$\sum_{h=1}^{r} \sum_{i \in S_h} x_{ik_h} + \sum_{h=1}^{r-1} x_{k_h k_{h+1}} + \sum_{j \in T} x_{k_r j} \leq r \tag{59}$$

is valid for the PDPTW.

Reachability constraints (Lysgaard [47])

Let $i \in V$, and let $A_i^- \subset A$ be the minimum arc set such that any feasible path from 0 to i uses only arcs from A_i^-; similarly, let $A_i^+ \subset A$ be the minimum arc set such that any feasible path from i to $2n+1$ uses only arcs from A_i^+. Let $T \subset V$ be such that each $i \in T$ must be visited by a different vehicle. Such a set is said to be conflicting. Define $A_T^- = \cup_{i \in T} A_i^-$ and $A_T^+ = \cup_{i \in T} A_i^+$. For any $S \subseteq P \cup D$ and any conflicting vertex set $T \subseteq S$, the inequalities

$$x(\delta^-(S) \cap A_T^-) \geq |T| \tag{60}$$

and

$$x(\delta^+(S) \cap A_T^+) \geq |T| \tag{61}$$

are valid for the PDPTW.

Ropke, Cordeau and Laporte [62] have developed a branch-and-cut algorithm for the PDPTW, using the preprocessing steps of Dumas, Desrosiers and Soumis [27] and of Cordeau [14], as well as several heuristics for the identification of violated valid inequalities. The algorithm was coded in C++ using ILOG Concert 1.3 and CPLEX 9.0, and run on an AMD Opteron 250 computer (2.4 GHz). Branching was performed on the x_{ij} variables and a best bound node selection strategy was used. The algorithm was tested on 40 PDPTW instances similar to those of Savelsbergh and Sol [68], which contain from 30 to 75 requests, and on two sets of DARP instances created by Cordeau [14], including maximum ride time constraints, which contain between 16 and 96 requests. About 75% of the 40 first instances were solved within two hours and all the Cordeau instances could also be solved within that time limit. These results clearly outperform those of Cordeau [14] who could handle instances involving at most 36 requests.

A Branch-and-cut-and-price Algorithm for the PDPTW

Ropke and Cordeau [63] have developed a branch-and-cut-and-price algorithm for the PDPTW in which all vehicles are identical and have capacity Q. Let Ω denote the set of all feasible routes r, let c_r be the cost of route r, and a_{ir} the number of times vertex $i \in P$ is visited by route r. Binary variables y_r are equal to 1 if and only if route r belongs to the optimal solution. The set partitioning formulation of the problem is then

(PDPTW)

$$\text{Minimize} \sum_{r \in \Omega} c_r y_r \tag{62}$$

$$\text{subject to} \sum_{r \in \Omega} a_{ir} y_r = 1 \qquad (i \in P) \tag{63}$$

$$y_r = 0 \text{ or } 1 \qquad (r \in \Omega). \tag{64}$$

In this formulation, constraints (63) ensure that every pickup node is served once. Since the routes of Ω satisfy pairing, precedence, capacity and time window constraints, the set partitioning constraints (63) are sufficient to ensure feasibility.

Formulation (62)–(64) is solved by a branch-and-bound mechanism in which lower bounds are computed by solving the LP relaxation by column generation. To improve the lower bounds, violated valid inequalities are introduced in the column generation master problem at each node of the enumeration tree. Branching is performed either on the outflow from the depot (i.e., on the number of vehicles used in the solution) or on the outflow from a set of vertices S when $x(\delta^+(0))$ is integer. The branch-and-bound tree is explored using a depth-first strategy.

Two pricing problems were considered to generate columns of negative reduced cost: the *Elementary Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery* (ESPPTWCPD), and the non-elementary relaxation of this problem. In the context of the PDPTW, the elementary shortest path was first used by Sol [71] while the non-elementary case was considered by Dumas, Desrosiers and Soumis [27]. Ropke and Cordeau explain how effective dominance criteria can be employed within these pricing problems, even when valid inequalities are introduced in the column generation master problem. Several existing families of valid inequalities are considered: precedence inequalities (50), infeasible path inequalities (54), (56), fork inequalities (58), (59), and reachability inequalities (60), (61). In addition, two new families of inequalities are introduced.

First, the classical rounded capacity inequalities can be strengthened by considering predecessor and successor sets $\pi(S)$ and $\sigma(S)$. This leads to the following inequalities which strengthen (43) and (51).

$$\sum_{i,j \in S} x_{ij} \leq |S| - \max\left\{1, \left\lceil \frac{q(\pi(S) \setminus S)}{Q} \right\rceil, \left\lceil \frac{-q(\sigma(S) \setminus S)}{Q} \right\rceil \right\}. \tag{65}$$

Second, when travel times satisfy the triangle inequality, 2-path inequalities introduced by Kohl et al. (1999) in the context of the *Vehicle Routing Problem with Time Windows* can also be adapted and strengthened by considering precedence relationships between vertices. If it is impossible to identify a tour serving all vertices in a vertex set S while satisfying precedence, capacity and time window constraints, then any feasible solution must use at least two arcs from the set $\delta^+(S)$. The idea can be taken further by observing that if a path serves all vertices of S by entering and leaving the set once, then the vertices $\pi(S) \setminus S$ must be served by this path before entering S, and vertices of $\sigma(S) \setminus S$ must be served after leaving S. If such a path cannot be found, then S defines a valid inequality of the form $x(\delta^+(S)) \geq 2$ even though there exists a tour through S satisfying precedence, capacity and time window constraints.

Ropke and Cordeau show that fork inequalities (58) and (59) and reachability constraints (60) and (61) are in fact implied by the set partitioning formulation when using the ESPPTWCPD as a pricing subproblem. In addition, precedence inequalities (5) are also implied by this formulation with either the ESPPTWCPD or its non-elementary relaxation.

To accelerate the solution of the pricing problems, several heuristics are used: label heuristics that limit the number of labels created by working on a reduced graph from which some arcs have been removed, a randomized construction heuristic based on a cheapest insertion criterion, and improvement heuristics based on the large neighbourhood search (LNS) paradigm (Shaw [69]). These heuristics are used sequentially until a negative reduced cost path has been identified. When the pricing heuristics fail to find new columns the separation procedures are called in order to find violated inequalities. If any are found then the pricing heuristics and the separation procedures are reapplied. The exact pricing algorithm is only called if both the pricing heuristics and the separation procedures are unsuccessful.

The branch-and-cut-and-price algorithm was tested on the instances introduced by Ropke, Cordeau and Laporte [62] and on those of Li and Lim [44]. For the first group of instances, all instances with $n \leq 75$ could be solved to optimality in just a few minutes on an Opteron 250 computer (2.4 GHz). Some larger instances with up to 175 requests were also solved to optimality within a two hour time limit. For the second group of instances, several instances with 100 requests were solved to optimality within that time limit. Computational results have shown that the two pricing problems considered perform similarly on test instances. Experiments concerning valid inequalities showed that the 2-path cuts were the most successful of the inequalities tested, and capacity inequalities were useful for instances with tight capacity constraints. Overall, this branch-and-cut-and-price algorithm outperforms the branch-and-cut algorithm of Ropke, Cordeau and Laporte [62].

3.2 Heuristics for MVPDPs

Heuristics for MVPDPs make use of insertion procedures (Jaw et al. [37], Lu and Dessouky [46]), cluster-first, route-second methods (Cullen et al. [22], Bodin and Sexton [5], Dumas, Desrosiers and Soumis [28], Desrosiers et al. [26], Toth and Vigo [74], Borndörfer et al. [6]), and tabu search (Toth and Vigo [75], Nanry and Barnes [52]). The reader is referred to Cordeau and Laporte [18] for a survey of heuristics specifically designed for the DARP. This section describes four recent heuristics for the MVPDP. The first three apply to static problems in which all data are known with certainty when solving the problem. The fourth applies to dynamic problems in which requests are gradually revealed over time, and the solution can be updated accordingly.

A Tabu Search Heuristic for the DARP

Cordeau and Laporte [17] have developed a tabu search (TS) for the DARP. It is based on the unified tabu search algorithm (UTSA) (Cordeau, Laporte and Mercier [19]) which adapts easily to a host of routing problems.

Neighbourhood search

The algorithm starts from a possibly infeasible solution s_0 and moves at each iteration t from the current solution s_t to the best solution in a subset of its neighbourhood $N(s_t)$. The algorithm uses attribute based tabu statuses (Cordeau, Gendreau and Laporte [15]). To avoid cycling, solutions possessing some attributes of recently visited solution are forbidden, or tabu, for a number of iterations, unless they improve upon the best known solution possessing one of these attributes. The algorithm also embeds a mechanism allowing the exploration of infeasible solutions, a concept introduced by Gendreau, Hertz and Laporte [32]. Denote by $c(s)$ the routing cost of solution s, and by $q(s)$, $d(s)$, $w(s)$ and $t(s)$ the violations of vehicle capacity, route duration, time window and ride time constraints, respectively. The algorithm minimizes the function $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s) + \tau t(s)$, when α, β, γ and τ are positive weights that self-adjust during the search. If a solution is feasible with respect to a given constraint, then the corresponding weight is divided by a factor $1 + \delta$, with $\delta > 0$; if the solution is infeasible, then it is multiplied by $1 + \delta$. This process produces a mix of feasible and infeasible solutions, which turns out to be particularly useful for tightly constrained instances.

Neighbourhood structure

With each solution s is associated an attribute set $B(s) = \{(i, k) : \text{request } i$ is served by vehicle $k\}$. The neighbourhood $B(s)$ of s contains all solutions obtained by removing an attribute (i, k) from $N(s)$ and replacing it with another attribute (i, k'), where $k' \neq k$. This means that vertices i and $n + i$ are removed from route k, which is then reconnected by linking the predecessor and successor of each deleted vertex, and the two vertices are then inserted in route k. The best position for i is first sought, and then $n + i$ is inserted in its best position. A tabu status is imposed on (i, k) for θ iterations.

Diversification mechanism

As suggested by Taillard [73], a frequency-based mechanism is used to diversify the search. Any solution $\bar{s} \in N(s)$ such that $f(\bar{s}) \geq f(s)$ is penalized by a term $p(\bar{s}) = \lambda c(\bar{s}) \sqrt{nm} \rho_{ik}$, where λ is a user-controlled parameter and ρ_{ik} is the number of times attribute (i, k) has been added to the solution during the search.

Forward time slacks

In order to reduce route durations, the algorithm delays as much possible vehicle departures from the depot. This can be done by computing the forward time slack F_i of each vertex i (Savelsbergh [67]) as follows. Consider a route $(i_0 = 0, \dots, i_q = 2n + 1)$, and let v_i be the waiting time at i and u_i the start of service at i. Then F_i can be computed as

$$F_i = \min_{1 \le j \le q} \left\{ \sum_{i<p\le j} v_p + (\ell_j - u_j) \right\}. \tag{66}$$

The departure time of the vehicle from the depot can then be delayed by F_0, which can be computed in $O(q)$ time. In the Cordeau and Laporte [17] algorithm, the computation of F_i is modified in order not to increase time window or ride time violations, i.e., F_i is redefined as

$$F_i = \min_{1 \le j \le q} \left\{ \sum_{i<p\le j} v_p + (\min\{\ell_j - u_j, L - r_j\})^+ \right\}, \tag{67}$$

where $(x)^+ = \max\{0, x\}$, and r_j is the ride time of the user whose destination is vertex j if $j \in D$, and $r_j = 0$ if $j \in P$.

Other features

The algorithm starts with a solution constructed by randomly assigning each request $(i, n+i)$ to a vehicle route, and by inserting i and $n+i$ at the end of the partially constructed route. Route reoptimizations are periodically performed by means of intra-route exchanges. The algorithm is run for a prefixed number of iterations.

Three versions of the algorithm were developed. Version 1 minimizes routing costs but does not minimize route durations; version 2 also minimizes route durations by computing forward time slacks; version 3 also minimizes the total ride time.

The algorithm was coded in C++ and tested on 20 randomly generated instances ($24 \le n \le 144$), and on six real-life instances provided by a Danish consultant ($n = 200$ and 295). The algorithm was run on a Pentium 4, 2 GHz for 10^4 iterations. It solved the randomly generated instances within an average of 5.16, 8.71 and 33.88 minutes, for versions 1, 2 and 3, respectively, and the Danish instances within 20.99, 34.78 and 166.12 minutes. Considering computing time and solution quality, version 2 appears to be the best option.

A Hybrid Heuristic for the PDPTW

Bent and Van Hentenryck [2] have developed a two-stage heuristic for the PDPTW. The first stage applies simulated annealing (SA) to minimize the number of routes, while the second stage minimizes the total route length through LNS (Shaw [69]).

The SA heuristic minimizes a hierarchical objective $< f_1, -f_2, f_3 >$. The function f_1 represents the number of vehicle routes in the solution; $f_2 = \sum_{k \in K} a_k^2$, where a_k is the number of requests in route k; f_3 is the total routing cost of the solution. The SA algorithm is implemented with an aspiration criterion as is commonly done in TS, and also contains a random selection mechanism that biases the search toward good moves.

The LNS mechanism uses nested neighbourhoods $N_1, \ldots, N_p$, where N_j relocates j requests from the current solution, and p is a user-controlled parameter. Because several requests are considered at once, a branch-and-bound mechanism is used to identify the best overall relocation scheme. For larger instances, the search is truncated and is only applied to a subset of the most promising relocations. The LNS mechanism only accepts improving moves.

The algorithm was run on a 1.2 GHz AMD Athlon Thunderbird IX7 processor running Linux. It was tested on benchmark PDPTW instances: 56 with 100 requests, 60 with 200 requests, and 60 with 600 requests. These instances, which are described in Li and Lim [44], are downloadable from `http://www.sintef.no/static/am/opti/projects/top/vrp/benchmarks.html`. Five runs were executed for each of the 100- and 200-request instances, and ten for the 600-request instances. The SA and LNS heuristics were each allowed to run for a preset time. On the 100-request instances, the algorithm produced two new best solutions and 54 matches; on the 200-request instances, it produced 28 new best solutions and 24 matches; on the 600-request instances, it produced 46 new best solutions and five matches.

An Adaptive Large Neighbourhood Search Heuristic for the PDPTW

The PDPTW version considered by Ropke and Pisinger [64] arises from the problem faced by a Danish food manufacturer. Each request $(i, n+i)$ can only be served by a subset K_i of the vehicles, and not all request are necessarily served. The objective is to minimize a weighted function $f = \alpha f_1 + \beta f_2 + \gamma f_3$, where f_1 is the routing cost, f_2 is the total time traveled by all vehicles, and f_3 is the number of unserved requests. It is normal to assign γ a very large value.

The heuristic proposed by the authors also uses LNS, but it differs from the Bent and Van Hentenryck [2] heuristic in several respects. Most importantly, the method uses several simple request removal and insertion procedures to explore the neighbourhood of the current solution, as opposed to the rather involved branch-and-bound process proposed by Bent and Van Hentenryck. In addition, the search mechanism of Ropke and Pisinger is embedded within an SA framework, whereas Bent and Van Hentenryck used a simple descent process.

The LNS heuristic of Ropke and Pisinger applies three removal heuristics (Shaw's [69] removal procedure, random removal, worst removal), as well as two insertion heuristics (greedy, and several types of regret-based insertions). The insertion heuristics use the true value of f to evaluate the quality of a solution, or a perturbed value $f + \varepsilon$, where ε is a randomly generated noise. During the search, the algorithm maintains a score φ_j which measures how well heuristic j has performed in the past iterations. At a given iteration, it applies a roulette wheel selection principle, i.e., it selects heuristic j with probability $\varphi_j / \sum\limits_i \varphi_i$. Because of this feature, the authors call their PDPTW heuristic an adaptive large neighbourhood search (ALNS) heuristic. The heuristic uses an SA-based acceptance rule for neighbour selection and runs for a preset number of iterations. The algorithm can easily be adapted to minimize the number of routes. It does so by iteratively deleting a route and reinserting its requests in other routes.

The algorithm was extensively tested on the 594 Li and Lim [44] instances which contain 100, 200, 400, 600, 800, and 1000 requests. Comparisons were made with results reported by Bent and Van Hentenryck (`http://www.cs.brown.edu/people/rbent/pickup-appendix.ps`). These tests showed the advantage of using several removal and insertion heuristics, they confirmed the superiority of ALNS over LNS, and they also proved the superiority of ALNS over the Bent and Van Hentenryck heuristic. The heuristic was later used to solve the *Capacitated Vehicle Routing Problem*, the *Vehicle Routing Problem with Time Windows*, and the *Multi-Depot Vehicle Routing Problem* (Pisinger and Ropke [56]).

A Double-horizon Heuristic for the Dynamic PDPTW

Mitrović-Minić, Krishnamurti and Laporte [49] have implemented a double-horizon heuristic for the dynamic PDPTW in which requests occur in real-time. The term double-horizon means that the insertion of a new request takes into account the short term effect, i.e., an immediate increase in routing cost, and the long term effect, i.e., a decrease in vehicle slack time. The algorithm combines a constructive heuristic which is applied whenever a new request occurs, and a tabu search heuristic which is applied periodically. In the constructive heuristic, the insertion cost of a new request is

$$c = [(1-\alpha_p)f_p + \alpha_p g_p] + [(1-\alpha_d)f_d + \alpha_d g_d]\,, \tag{68}$$

where α_p and α_d are user-controlled parameters, f_p and f_d are the route length increases due to the insertion of a pickup and a delivery, and g_p and g_d are the corresponding decreases in vehicle slack times. Three insertion costs were tested: c_1 (with $\alpha_p = \alpha_d = 0$), c_2 (with $0 < \alpha_p < 1$ and $0 < \alpha_d < 1$), and c_3 (with $\alpha_p = \alpha_d = 0$ if the pickup and delivery both occur within a short term horizon of length s, and $0 < \alpha_p < 1$, $0 < \alpha_d < 1$ otherwise).

The objective function minimized in the tabu search procedure is defined as

$$z = \frac{\ell}{s}\beta q_S + (1-p)\left((1-\alpha)q_L + \alpha h_L\right), \tag{69}$$

where β is a user-controlled parameter, q_S is the total length of the route portions falling within the short-term horizon, q_L is the remaining length of the routes, h_L is the average slack time over all route portions belonging to the long-term horizon, ℓ is the length of the long-term horizon, and s is the length of the short-term horizon. Again, three variants were defined: z_1 (with $\alpha = \beta = 0$, and q_L is interpreted as the total route length), z_2 (with $0 < \alpha < 1$, $\beta = 0$, q_L is the total route length and h_L is the average slack time of all routes), z_3 (with $0 < \alpha < 1$ and $0 < \beta < 1$).

The authors have also tested several waiting strategies (Mitrović-Minić and Laporte [50]). When a new request arrives, the vehicle assigned to it can drive as soon as possible, yielding a drive-first (DF) strategy, or it can wait as long as possible before moving, yielding a wait-first (WF) strategy. An intermediate strategy, called advanced dynamic waiting (ADW), works as follows. Vehicle routes are partitioned into segments, each containing locations that are reasonably close to each other, and these segments vary dynamically during the course of the algorithm. The ADW strategy applies DF as long as the vehicle remains in the same segment, and the WF strategy when it reaches the last location of the segment. The ADW strategy proved to be the best, but its superiority becomes smaller for large instances.

The double-horizon heuristic was tested with the combinations (c_1, z_1), (c_2, z_2) and (c_3, z_3) for the DF and ADW waiting strategies. Note that only (c_3, z_3) yields a true double-horizon heuristic. Computer runs were performed over three set of 30 instances containing 100, 500 and 1000 requests each. Statistical tests confirmed the superiority of ADW over DF for all (c, z) combinations, and the superiority of (c_3, z_3) over (c_1, z_1) and (c_2, z_2).

Split loads

An interesting variant of the MVPDP is the *Pickup and Delivery Problem with Split Loads* (PDPSL) recently investigated by Nowak, Ergun and White [53]. Contrary to what happens in the MVPDP, in the PDPSL customer requests can be split among several vehicles. The authors show that allowing splits can yield savings whose value is highly dependent on the load size range $[a, b]$, meaning that demands are distributed between $a\%$ and $b\%$ of the vehicle size Q. When $[a, b] = [0.41, 0.50]$ or $[0.81, 0.90]$ the savings are insignificant. However, when $[a, b] = [0.51, 0.60]$, they can reach 30%.

4 Conclusions

One-to-one Pickup and Delivery Problems arise in several contexts related to the transportation of goods and people. In the past few years several new and powerful algorithms have been developed to solve these problems. The best exact solution methodologies are based on branch-and-cut and on branch-and-cut-and-price. Their success is linked to the identification of strong valid inequalities and to the development of efficient separation procedures. New heuristics employ a variety of techniques including tabu search, simulated annealing, variable neighbourhood search, and large neighbourhood search. The success of these heuristics is dependent on the design of clever search mechanisms, some of which are of wide applicability.

Acknowledgement

This work was supported by the Canadian Natural Sciences and Engineering Research Council under grants 227837-04 and 39682-05. This support is gratefully acknowledged. Thanks are due to Bruce L. Golden and to two referees for their valuable comments.

References

1. Balas E, Fischetti M, and Pulleyblank WR (1995) The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming* 68:241–265
2. Bent R and Van Hentenryck P (2006) A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research* 33:875–893
3. Berbeglia G, Cordeau J-F, Gribkovskaia I, and Laporte G (2007) Static pickup and delivery problems: A classification scheme and survey. *TOP* 15:1–31
4. Bianco L, Mingozzi A, and Ricciardelli S (1994) Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. *INFOR* 32:19–31
5. Bodin LD and Sexton T (1986). The multi-vehicle subscriber dial-a-ride problem. *TIMS Studies in Management Science* 26:73–86
6. Borndörfer R, Grötschel M, Klostermeier F, and Küttner C (1997) *Telebus Berlin: Vehicle scheduling in a dial-a-ride system.* Technical Report SC 97–23, Konrad-Zuse-Zentrum für Informationstechnik Berlin
7. Brønmo G, Christiansen M, Fagerholt K, and Nygreen B (2007) A multi-start local search heuristic for ship scheduling – a computational study. *Computers & Operations Research* 34:900–917
8. Carrabs F (2006) *Heuristics and exact approaches for transportation problems with pickup and delivery.* Ph.D. Thesis, Università di Salermo, Italy
9. Carrabs F, Cerulli R, and Cordeau J-F (2006) An additive branch-and-bound algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Submitted for publication

10. Carrabs F, Cordeau J-F, and Laporte G (2006) Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing* forthcoming
11. Cassani L and Righini G (2004) Heuristic Algorithms for the TSP with rear-loading. Presented at the 35th Annual Conference of the Italian Operations Research Society (AIRO XXXV), Lecce, Italy
12. Christiansen M and Nygreen B (1998a) A method for solving ship routing problems with inventory constraints. *Annals of Operations Research* 81:357–378
13. Christiansen M and Nygreen B (1998b) Modelling path flows for a combined routing and inventory management problem. *Annals of Operations Research* 82:391–412
14. Cordeau J-F (2006) A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54:573–586
15. Cordeau J-F, Gendreau M, and Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* 30:105–119
16. Cordeau J-F, Iori M, Laporte G, and Salazar-González JJ (2006) A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Submitted for publication
17. Cordeau J-F and Laporte G (2003) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research B* 37:579–594
18. Cordeau J-F and Laporte G (2007) The dial-a-ride problem: Models and algorithms. *Annals of Operations Research* 153:29–46
19. Cordeau J-F, Laporte G, and Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* 52:928–936
20. Cordeau J-F, Laporte G, Potvin J-Y, and Savelsbergh MWP (2007. Transportation on demand. In: Barnhart C and Laporte G (eds.) *Transportation*, Handbooks in Operations Research and Management Science, Volume 14, pages 429–466, North-Holland, Amsterdam
21. Croes G (1958) A method for solving traveling salesman problems. *Operations Research* 6:791–812
22. Cullen FH, Jarvis JJ, and Ratliff HD (1981) Set partitioning based heuristics for interactive routing. *Networks* 11:125–143
23. Desaulniers G, Desrosiers J, Erdmann A, Solomon MM, and Soumis F (2002) VRP with pickup and delivery. In: Toth P and Vigo D (eds.) *The Vehicle Routing Problem*, pages 225–242, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia
24. Desrochers M and Laporte G (1991) Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters* 10:27–36
25. Desrosiers J, Dumas Y, and Soumis F (1986) 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
26. Desrosiers J, Dumas Y, Soumis F, Taillefer S, and Villeneuve D (1991) An algorithm for mini-clustering in handicapped transport. *Les Cahiers du GERAD*, G–91–02, HEC Montréal
27. Dumas Y, Desrosiers J, and Soumis F (1991) The pickup and delivery problem with time windows. *European Journal of Operational Research* 54:7–22
28. Dumas Y, Desrosiers J, and Soumis F (1989) Large scale multi-vehicle dial-a-ride problems. *Les Cahiers du GERAD*, G–89–30, HEC Montréal
29. Dumitrescu I, Ropke S, Cordeau J-F, and Laporte G (2006) The traveling salesman problem with pickup and deliveries: Polyhedral results and branch-and-cut algorithm. Submitted for publication
30. Fischetti M and Toth P (1989) An additive bounding procedure for combinatorial optimization problems. *Operations Research* 37:319–328
31. Fisher ML and Rosenwein MB (1989) An interactive optimization system for bulk-cargo ship scheduling. *Naval Research Logistic Quarterly* 35:27–42
32. Gendreau M, Hertz A, and Laporte G (1994) A tabu search heuristic for the vehicle routing problem. *Management Science* 40:1276–1290
33. Gribkovskaia I and Laporte G (2007) One-to-many-to-one single vehicle pickup and delivery problems. This volume
34. Grötschel M and Padberg MW (1985) Polyhedral theory. In: Lawler EL, Lenstra JK, Rinnooy Kan AHG and Shmoys DB (eds.) *The Traveling Salesman Problem*, pages 251–305, Wiley, Chichester
35. Hernández-Pérez H and Salazar-González JJ (2004) Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science* 38:245–255
36. Hernández-Pérez H and Salazar-González JJ (2007) The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks* forthcoming
37. Jaw J, Odoni AR, Psaraftis HM, and Wilson NHM (1986) A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time-windows. *Transportation Research B* 20:243–257
38. Kalantari B, Hill AV, and Arora SR (1985) An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research* 22:377–386
39. Kohl N, Desrosiers J, Madsen OBG, Solomon MM, and Soumis F (1999) 2-path cuts for the vehicle routing problem with time windows. *Transportation Science* 33:101–116
40. Landrieu A, Mati Y, and Binder Z (2001) A tabu search heuristic for the single vehicle pickup and delivery problem with time windows. *Journal of Intelligent Manufacturing* 12:497–508
41. Laporte G, Nobert Y, and Desrochers M (1985) Optimal routing under capacity and distance restrictions. *Operations Research* 33:1050–1073

42. Lawler EL, Lenstra JK, Rinnooy Kan AHG and Shmoys DB (1985) *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester
43. Levitin G and Abezgaouz R (2003) Optimal routing and multiple-load AGV subject to LIFO loading constraints. *Computers & Operations Research* 30:397–410
44. Li H and Lim A (2001) A Metaheuristic for the pickup and delivery problem with time windows. *The 13th IEEE Conference on Tools with Artificial Intelligence*, ICTAI-2001, Dallas, pages 160–170
45. Lin S and Kerninghan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* 21:498–516
46. Lu Q and Dessouky MM (2006) A new insertion-based construction heuristic for solving the pickup and delivery problem with time-windows. *European Journal of Operational Research* 175:672–687
47. Lysgaard J (2006) Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research* 175:210–233
48. Madsen OBG, Ravn HF, and Rygaard JM (1995) A heuristic algorithm for the dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research* 60:193–208
49. Mitrović-Minić S, Krishnamurti R, and Laporte G (2004) Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B* 38:669–685
50. Mitrović-Minić S and Laporte G (2004) Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B* 38:635–655
51. Mladenović N and Hansen P (1997) Variable neighborhood search. *Computers & Operations Research* 24:1097–1100
52. Nanry WP and Barnes JW (2000) Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research B* 34:107–121
53. Nowak M, Ergun O, and White CC (2006) Pickup and delivery with split loads. Submitted for publication
54. Pacheco JA (1995) Problemas de rutas con carga y descarga en sistemas LIFO: Soluciones exactas. *Estudios de Economía Aplicada* 3:69–86
55. Pankratz G (2005) A grouping genetic algorithm for the pickup and delivery problem with time windows. *Operations Research Spectrum* 27:21–41
56. Pisinger D and Ropke S (2007) A general heuristic for vehicle routing problems. *Computers & Operations Research* 34:2403–2435
57. Rappoport HK, Levy LS, Golden BL, and Toussaint K (1992) A planning heuristic for military airlift. *Interfaces* 22(3):73–87
58. Rappoport HK, Levy LS, Toussaint K, and Golden BL (1994) A transportation problem formulation for the MAC airlift planning problem. *Annals of Operations Research* 50:505–523
59. Reinelt G (1991) TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing* 3:376–384

60. Rekiek B, Delchambre A, and Saleh HA (2006) Handicapped person transportation problem: An application of the grouping genetic algorithm. *Engineering Applications of Artificial Intelligence* 19:511–520
61. Renaud J, Boctor FF, and Laporte G (2002) Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research* 29:1129–1141
62. Ropke S, Cordeau J-F, and Laporte G (2007) Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* 49:258–272
63. Ropke S and Cordeau J-F (2006) Branch-and-cut-and-price for the pickup and delivery problem with time windows. Submitted to *Transportation Science*
64. Ropke S and Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* 40:455–472
65. Ruland KS (1994) *Polyhedral solution to the pickup and delivery problem.* Ph.D. Thesis, Sever Institute, Washington University in St.Louis, MO
66. Ruland KS and Rodin EY (1997) The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers and Mathematics with Applications* 33:1–13
67. Savelsbergh MWP (1992) The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing* 4:146–154
68. Savelsbergh MWP and Sol M (1998) Drive: Dynamic routing of independent vehicles. *Operations Research* 46:474–490
69. Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, vol. 1520 of *Lecture Notes in Computer Science*, pages 417–431
70. Shen Y, Potvin J-Y, Rousseau J-M, and Roy S (1995) A computer assistant for vehicle dispatching with learning capabilities. *Annals of Operations Research* 61:189–211
71. Sol M (1994) *Column generation for pickup and delivery problems.* Ph.D. Thesis, Technische Universiteit Eindhoven
72. Solanki RS and Southworth F (1991) An execution planning algorithm for military airlift. *Interfaces* 21(4):121–131
73. Taillard ÉD (1993) Parallel iterative search methods for vehicle routing problems. *Networks* 23:661–673
74. Toth P and Vigo D (1996) Fast local search algorithms for the handicapped persons transportation problem. In: Osman IH, Kelly JP (eds) *Meta-Heuristics: Theory and Applications*, pages 677–690, Kluwer, Boston
75. Toth P and Vigo D (1997) Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science* 31:60–71
76. Toth P and Vigo D (2002) *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia

77. Van der Bruggen LJJ, Lenstra JK, and Schuur PC (1993) Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science* 27:298–311
78. Wilson NHM, Sussman J, Wong H, and Higonnet B (1971) *Scheduling algorithms for dial-a-ride systems.* Technical Report USL TR–70–13, Urban Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA
79. Wilson NHM and Weissberg H (1976) *Advanced dial-a-ride algorithms research project: Final report.* Technical Report R76–20, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA