An Integer L-Shaped Algorithm for the Dial-a-Ride Problem with Stochastic Customer Delays

Géraldine Heilporn[a,b], Jean-François Cordeau[a], Gilbert Laporte[b]

[a] *Canada Research Chair in Logistics and Transportation and CIRRELT, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7*
[b] *Canada Research Chair in Distribution Management and CIRRELT, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7*
Corresponding author: gilbert@crt.umontreal.ca

Abstract

This paper considers a single-vehicle Dial-a-Ride problem in which customers may experience stochastic delays at their pickup locations. If a customer is absent when the vehicle serves the pickup location, the request is fulfilled by an alternative service (e.g., a taxi) whose cost is added to the total cost of the tour. In this case, the vehicle skips the corresponding delivery location, which yields a reduction in the total tour cost. The aim of the problem is to determine an a priori Hamiltonian tour minimizing the expected cost of the solution. This problem is solved by means of an integer L-shaped algorithm. Computational experiments show that the algorithm yields optimal solutions for small and medium size instances within reasonable CPU times. It is also shown that the actual cost of an optimal solution obtained with this algorithm can be significantly smaller than that of an optimal solution obtained with a deterministic formulation.

Key words: dial-a-ride problem, stochastic programming, integer L-shaped algorithm.

1. Introduction

The single-vehicle *Dial-a-Ride Problem* (DARP) consists in satisfying at minimum cost a set of customer transportation requests, each defined by a specific origin-destination pair, while respecting side constraints related to operational considerations and to quality of service. These typically include load and capacity constraints, total duration and ride time constraints, as

well as time windows. Customer requests can either be outbound — a desired arrival time at destination is specified —, or inbound — a desired departure time from the origin is specified. Time windows of given widths are then constructed around these desired times.

Our aim is to propose an exact algorithm for a stochastic version of the DARP called the single-vehicle *DARP with stochastic customer delays* (S-DARP). In this problem, customers arrive at their origin with a stochastic delay. Such delays are frequently encountered when customers must be picked up at hospitals or other healthcare facilities. Indeed, because waiting times and the duration of medical appointments are often unpredictable, customers cannot guarantee at which time they will become available to be picked up for their inbound request. We assume that if a customer is absent when the vehicle serves the pickup location, the vehicle moves immediately to the next location. In this case, the "missed" customer request is fulfilled by an alternative service such as a taxi whose cost must be added to the total cost of the tour. Furthermore, the vehicle will skip the corresponding delivery node, yielding a reduction in the tour cost. The single-vehicle S-DARP consists in determining an a priori Hamiltonian tour that minimizes the expected cost of the tour actually followed by the vehicle.

There exists a rich literature on the DARP. The single-vehicle case was introduced by Psaraftis [19, 20], who solved it by dynamic programming. Desrosiers et al. [10] formulated the problem as an integer program, and solved instances with up to 40 requests, also by dynamic programming. More recently, Cordeau [5] presented some valid inequalities and a branch-and-cut algorithm for the multi-vehicle DARP. The author solved instances involving up to 32 requests. Ropke et al. [22] later presented stronger formulations and new valid inequalities for the DARP and the *Pickup and Delivery Problem with Time Windows* (PDPTW), which can be viewed as a DARP without ride time constraints. They solved instances with up to 96 requests using a branch-and-cut algorithm. Ropke and Cordeau [21] then proposed a branch-and-cut-and-price method for the PDPTW. This algorithm uses some of the inequalities introduced by Ropke et al. [22] within a column generation framework and it could solve some tightly constrained instances with up to 500 requests. Very recently, Bartolini [1] also formulated the PDPTW as a set partitioning problem with additional cuts. He proposed an exact algorithm for the problem, using both relaxations of the formulation and a branch-and-cut-and-price algorithm. His method provided better results than that of Ropke and Cordeau [21] in terms of lower bound quality and computing

time. For recent reviews of the DARP, see Cordeau and Laporte [6] and Cordeau et al. [7].

The S-DARP is related to the *Probabilistic Traveling Salesman Problem* (PTSP), in which vertices are present with given probabilities. The PTSP is solved in two stages. In the first stage, an a priori Hamiltonian tour must be determined before any information on the present vertices is known. The set of present vertices is then revealed. In the second stage solution, the vehicle follows its planned tour but skips the absent vertices. The PTSP consists in determining an a priori Hamiltonian tour that minimizes the expected length of the tour actually followed by the vehicle in the second stage. The PTSP was introduced by Jaillet [14, 15] who presented several combinatorial and asymptotic results, among which an efficient method to compute the expected length of the second stage tour. Laporte et al. [17] proposed an exact integer L-shaped algorithm for this problem and solved instances with up to 50 vertices. The latter algorithm, which we will adapt to our problem, is based on the L-shaped method for continuous programs (Van Slyke and Wets [24]) and on Benders decomposition [2]. The integer L-shaped algorithm was put forward by Laporte and Louveaux [16] for stochastic integer programs with integer recourse. It applies branch-and-cut to an initial relaxed model, which is then iteratively tightened by appending lower bounding functionals and optimality cuts to the current problem. The optimality cuts require the knowledge of an integer feasible solution and are thus only imposed at nodes of the branch-and-cut tree corresponding to integer solutions. In contrast, the lower bounding functionals can be imposed at any node of the tree.

The integer L-shaped algorithm was also applied by Gendreau et al. [11] to the *Vehicle Routing Problem* (VRP) with stochastic customers and demands. In this problem, each vertex has a given probability of being present and has a stochastic demand. The authors have solved instances involving up to 70 vertices. Hjorring and Holt [13] presented improved optimality cuts and lower bounding functionals for the related single vehicle problem with stochastic demands only (i.e., all vertices are present), and solved instances with up to 90 vertices. Finally, Laporte et al. [18] derived better optimality cuts and lower bounding functionals for a stochastic capacitated VRP with Poisson or normal demands. These authors have solved instances involving up to 100 vertices. For more details about the stochastic VRP, we refer the interested reader to the surveys of Gendreau et al. [12] and of Cordeau et al. [8].

In related work, Campbell and Thomas [3, 4] studied a PTSP in which customers should be visited before a known deadline. In the first paper [3], the authors presented two recourse models and a chance constrained model for the problem, and discussed several special cases. Whereas the recourse models penalize deadline violations in the objective function, the chance constrained model restricts the probability that a deadline violation occurs. The authors also compared through computational experiments the solution values obtained using stochastic or deterministic formulations. In a follow-up paper, Campbell and Thomas [4] proposed approximation methods to quickly compute deadline violations. These methods provide good quality solutions and yield significant reductions in computing time with respect to an exact computation of the deadline violations. They can thus be incorporated within local search algorithms.

The purpose of this paper is to introduce an integer L-shaped algorithm for the single-vehicle S-DARP. The remainder of the paper is organized as follows. Section 2 provides a formal description of the S-DARP, together with a mixed-integer linear programming formulation. Section 3 describes our integer L-shaped algorithm. This section also includes details about the computation of the delay cost associated to a feasible Hamiltonian tour, and provides the optimality cuts appended to the stochastic model. Note that no lower bounding functionals are generated because no strong constraints of this type could be identified. Computational results are presented in Section 4, followed by the conclusion in Section 5.

2. Formal problem description

Consider a set of n customer requests $r_1, \ldots, r_n$, where each r_i is composed of a pickup node i and a delivery node $n+i$. Let $G = (N, A)$ be the corresponding directed graph, with $N = P \cup D \cup \{0, 2n+1\}$, $P = \{1, \ldots, n\}$ the set of pickup nodes, $D = \{n+1, \ldots, 2n\}$ the set of delivery nodes, $\{0, 2n+1\}$ the depot nodes, and $A = \big\{(i,j) : i \in N \setminus \{2n+1\}, j \in N \setminus \{0, i, i-n\}\big\}$ the set of arcs.

To each node $i \in N$ corresponds a load q_i such that $q_0 = q_{2n+1} = 0$, $q_i > 0$ and $q_{n+i} = -q_i$ $(i = 1, \ldots, n)$, and the vehicle capacity is given by Q. Furthermore, a service duration d_i as well as a time window $[e_i, l_i]$ are provided for each node $i \in N$. In a deterministic context, the latter corresponds to an interval in which the vehicle must begin service at node i, which implies that a customer making a request r_i is supposed to be available

at the pickup node at time e_i at the latest. With each arc $(i, j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij}, which satisfy the triangle inequality. We assume that $e_i \geq e_0 + d_0 + t_{0i}$ for all $i \in N \setminus \{0\}$ since the vehicle starts from the depot node 0. Also, a maximal ride time R is imposed on the duration of any customer trip, while an upper bound T is imposed on the total tour duration.

In the S-DARP, customers can be delayed, i.e., the customer is present at node $i \in P$ at time $e_i + \xi_i$, where ξ_i is a nonnegative random variable. The taxi cost corresponding to a "missed" customer request r_i is denoted by b_i. We assume that $b_i \geq \max_{k,l \in N}\{c_{k,n+i} + c_{n+i,l} - c_{kl}\}$ for all $i \in P$, i.e., the cost of fulfilling a customer request by taxi is always larger than the reduction in the tour cost obtained by skipping the corresponding delivery node. Furthermore, in the DARP, it is common to assume that if the vehicle arrives at node $j \in N$ earlier than e_j, then waiting occurs before the beginning of service at this node (Cordeau et al. [7]). In the S-DARP, we consider that the vehicle should avoid arriving at node $j \in N$ earlier than e_j to maximize the probability of picking up a delayed customer. Hence, if the vehicle travels on arc (i, j), any waiting necessary before the beginning of service at node j will be replaced by a postponement of the beginning of service at node i, up to time l_i.

To model the S-DARP, we define binary flow variables x_{ij} equal to 1 if and only if the vehicle travels on arc $(i, j) \in A$. Let y_i, $i \in N$, be variables equal to the beginning of service at the nodes of G, and let u_i be the vehicle load upon leaving node $i \in P \cup D$. The model is then:

$$\text{S-DARP: minimize} \sum_{(i,j) \in A} c_{ij} x_{ij} + \Theta(x, y) \tag{1}$$

subject to:

$$\sum_{j \in P} x_{0j} = 1 \tag{2}$$

$$\sum_{j \in D} x_{j,2n+1} = 1 \tag{3}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad i \in P \cup D \tag{4}$$

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

$$y_j \geq y_i + d_i + t_{ij} - M_{ij}(1 - x_{ij}) \qquad (i,j) \in A \quad (6)$$

$$y_i + d_i + t_{i,n+i} \leq y_{n+i} \qquad i \in P \quad (7)$$

$$y_{n+i} - y_i - d_i \leq R \qquad i \in P \quad (8)$$

$$u_j - u_i \geq q_j - Q_i(1 - x_{ij}) + (Q_i - q_i - q_j)x_{ji} \qquad i,j \in P \cup D \quad (9)$$

$$\max\{0, q_i\} \leq u_i \leq \min\{Q, Q + q_i\} \qquad i \in N \quad (10)$$

$$y_{2n+1} - y_0 \leq T \qquad (11)$$

$$e_i \leq y_i \leq l_i \qquad i \in N \quad (12)$$

$$y_i \geq e_i + \sum_{j \in N} \max\left\{0, \min\{e_j - e_i - d_i - t_{ij}, l_i - e_i\}\right\} x_{ij} \qquad i \in N \quad (13)$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A, \quad (14)$$

with $M_{ij} = \max\{0, l_i + d_i + t_{ij} - e_j\}$, $Q_i = Q$ for $i \in P$ and $Q_i = Q - 1$ for $i \in D$. The objective function (1) minimizes the expected cost of the tour actually followed by the vehicle. In addition to the cost of the a priori tour, a function $\Theta(x, y)$ measures the expected cost of delay caused by absent customers (also called *delay cost* in the following). Constraint (2) (resp. (3)) imposes that the first node after (resp. before) the depot is a pickup (resp. delivery) node. Constraints (4) and (5) mean that all pickup and delivery nodes are visited. Constraints (6) and (7) ensure that the beginning of service variables are consistent, and constraints (8) enforce a maximal ride time R for each customer. Constraints (9) guarantee the consistency of load variables. Indeed, they consist of the linearization of constraints $u_j - u_i \geq q_j x_{ij}$, lifted by using the reverse arc (j, i), as in Desrochers and Laporte [9]. Constraints (10) define lower and upper bounds on the vehicle load. Constraints (11) impose a total maximal duration T for the tour, while (12) are the time window constraints. Finally, constraints (13) ensure that the vehicle avoids arriving at a node earlier than the beginning of the corresponding time window, if possible. Indeed, if the vehicle travels on an arc (i, j) such that $e_i + d_i + t_{ij} \leq e_j$, the earliest time e_i is postponed by $\min\{e_j - e_i - d_i - t_{ij}, l_i - e_i\}$. The last term $l_i - e_i$ ensures that the deadline l_i is not exceeded.

3. The Integer *L*-Shaped Method for the S-DARP

In our implementation of the integer L-shaped method for the S-DARP, we solve a relaxed version of model S-DARP, which is iteratively tightened

by means of optimality cuts. Hence the objective function (1) is replaced with

$$\text{minimize} \sum_{i \in N} \sum_{j \in N \setminus \{i\}} c_{ij} x_{ij} + \theta, \tag{15}$$

where θ is a lower bound on the delay cost $\Theta(x, y)$, while integrality constraints (14) are also relaxed. A lower bounding constraint $\theta \geq L$ is also included in the formulation, where L is a global lower bound for the delay cost. Since we have assumed that the cost of fulfilling a request by taxi is larger than any corresponding reduction in the tour cost, we can clearly set $L = 0$.

Our implementation of the integer L-shaped method for the S-DARP can be summarized as follows:

Step 1 (Initialization) Set a solution counter $r = 0$ and the best objective function value $z^* = \infty$. The first subproblem in the search tree is the relaxed problem defined above.

Step 2 (Subproblem selection) Choose a subproblem in the search tree (according to a best-bound rule). If none exists, the best solution has been found: stop.

Step 3 (Subproblem solution) Solve the current subproblem and let z be its optimal value. If $z > z^*$, fathom the corresponding node of the search tree and go to Step 2.

Step 4 (Integrality test) If the current solution is not integer, create two subproblems by branching on a fractional variable x_{ij}, add these to the search tree and go to Step 2.

Step 5 (Delay cost lower bounding) Set $r = r + 1$. The current solution (x^r, y^r, θ^r) is feasible. Compute a lower bound $\underline{\theta}$ on the delay cost $\Theta(x^r, y^r)$. Compute the corresponding lower bound on the objective function value $\underline{z}^r := c^T x^r + \underline{\theta}$. If $\underline{\theta} > \lambda \theta^r$ $(\lambda \in \mathbb{R}_0^+)$ or $\underline{z}^r > z^*$, generate optimality cuts and go to Step 2.

Step 6 (Delay cost computation) Compute the delay cost $\Theta(x^r, y^r)$. If $\Theta(x^r, y^r) > \theta^r$, generate optimality cuts and go to Step 2.

Step 7 (Best solution test) Compute the objective function value $z^r := c^T x^r + \Theta(x^r, y^r)$ associated with the current feasible solution (x^r, y^r, θ^r). If $z^r \leq z^*$, set $z^* = z^r$ and save x^r as the new best solution. Fathom the node of the search tree and go to Step 2.

The main difficulty of this method consists in computing the delay cost $\Theta(x^r, y^r)$ associated with a feasible solution (x^r, y^r, θ^r) in Step 6. In the

following section, we define the delay cost more precisely and provide details on its computation.

3.1. Computing the delay cost associated with a feasible Hamiltonian tour

The probability that the vehicle picks up a customer at a node depends on the customer delay relative to the beginning of service at this node. In order to ensure that the vehicle picks up as many customers as possible, the beginning of service at any pickup node has to be scheduled as late as possible in order to maximize the related probabilities. Consider that a feasible solution (x^r, y^r, θ^r) to the stochastic relaxed model is known, and let the vector $(s_0 = 0, s_1, \ldots, s_{2n}, s_{2n+1} = 2n+1)$ describe the corresponding Hamiltonian tour. The waiting times before the beginning of service at nodes are defined as

$$w^r_{s_i} = \max\{0, y^r_{s_i} - t_{s_{i-1}s_i} - d_{s_{i-1}} - y^r_{s_{i-1}}\} \qquad i = 1, \ldots, 2n+1. \tag{16}$$

Similarly to Savelsbergh [23], we define forward time slack variables z_{s_i} ($s_i \in N$) such that

$$z_{s_{2n+1}} = l_{2n+1} - y^r_{2n+1} \tag{17}$$

$$z_{s_i} = \min\{l_{s_i} - y^r_{s_i}, z_{s_{i+1}} + w^r_{s_{i+1}}\} \qquad i = 0, \ldots, 2n. \tag{18}$$

These correspond to the largest postponements that can be imposed on the beginning of service at the nodes so that the Hamiltonian tour remains feasible. In order to satisfy constraint (11) on the maximal tour duration, these variables are iteratively adjusted. Indeed, assume that $y^r_0 + z_0 + T > y^r_{2n+1} + z_{2n+1}$, i.e., the maximal postponements of beginning of service at nodes 0 and $2n+1$ yield a tour duration larger than T. In this case, we set $z_{2n+1} = y^r_0 + z_0 + T - y^r_{2n+1}$, and the variables z_{s_i} ($i = 0, \ldots, 2n$) are recomputed using (18). Then we can set the updated beginning of service $y^R_0 = y^r_0 + z_0$.

Next, the beginning of service at nodes are updated in order to maximize the probabilities that the vehicle picks up customers. In order to satisfy constraints (8) on the maximal customer ride times, the first pickup node s_j of the tour is postponed:

$$y^R_{s_j} = y^r_{s_j} + z_{s_j}, \tag{19}$$

where $y^R_{s_j}$ is the updated beginning of service at node s_j, and the forward time slack variable at the corresponding delivery node is updated as follows:

$$z_{n+s_j} = \min\{z_{n+s_j}, R + y^R_{s_j} + d_{s_j} - y^r_{n+s_j}\} \tag{20}$$

to ensure a maximal ride time R for this customer request. Let k be the index of the delivery node, i.e., $n + s_j = s_k$. Now let j denote the index of the second pickup node of the tour. Forward time slack variables for nodes s_i such that $j \leq i < k$ are recomputed using (18). Then, the beginning of service at the second pickup node can be postponed following (19), and this process is iterated until the last pickup node of the tour has been reached. Note that the beginning of service at delivery nodes, as well as the beginning of service at the depot node $2n + 1$, do not have to be scheduled as late as possible. The latter are set according to the postponements at pickup nodes, i.e.,

$$y^R_{s_i} = \max\{y^r_{s_i}, y^R_{s_{i-1}} + d_{s_{i-1}} + t_{s_{i-1}s_i}\} \qquad s_i \in D \cup \{2n+1\}. \tag{21}$$

Now consider a customer request r_j, $j \in \{1, \ldots, n\}$, and let $(s_0, \ldots, s_k = j, \ldots, s_l = n + j, \ldots, s_{2n+1})$ be a feasible Hamiltonian tour. With probability $1 - p_j$, an additional taxi cost b_j must be included in the delay cost whereas the vehicle skips the delivery node $n + j \in D$, yielding a reduction in the tour cost and thus in the delay cost. In this case, the beginning of service at each pickup node $s_i \in P$ such that $k < i < l$ is postponed by at most $g^{(r_j)} = t_{s_{l-1}s_l} + d_{s_l} + t_{s_l s_{l+1}} - t_{s_{l-1}s_{l+1}}$ in order to increase the corresponding probabilities p_{s_i}. The beginning of service at delivery nodes $s_i \in D$ such that $k < i < l$ are postponed accordingly. However, note that the full postponement $g^{(r_j)}$ is only achieved at nodes such that the remaining tour remains feasible. We thus define the following updated waiting times:

$$w^{R(r_j)}_{s_{l+1}} = \max\{0, y^R_{s_{l+1}} - t_{s_{l-1}s_{l+1}} - d_{s_{l-1}} - y^R_{s_{l-1}}\} \tag{22}$$

$$w^{R(r_j)}_{s_i} = \max\{0, y^R_{s_i} - t_{s_{i-1}s_i} - d_{s_{i-1}} - y^R_{s_{i-1}}\} \quad i = k+2, \ldots, l-1. \tag{23}$$

Hence new forward time slack variables $z^{(r_j)}_{s_i}$ are defined for the nodes

$s_i \in N$ such that $k < i < l$:

$$z_{s_{l-1}}^{(r_j)} = \min\{l_{s_{l-1}} - y_{s_{l-1}}^R, z_{s_{l+1}} - y_{s_{l+1}}^R + y_{s_{l+1}}^r + w_{s_{l+1}}^{R(r_j)}\} \qquad s_{l-1} \in P \quad (24)$$

$$z_{s_{l-1}}^{(r_j)} = \min\{l_{s_{l-1}} - y_{s_{l-1}}^R, z_{s_{l+1}} - y_{s_{l+1}}^R + y_{s_{l+1}}^r + w_{s_{l+1}}^{R(r_j)}, R + y_{s_{l-1-n}}^R + d_{s_{l-1-n}} - y_{s_{l-1}}^R\} \qquad s_{l-1} \in D \quad (25)$$

$$z_{s_i}^{(r_j)} = \min\{l_{s_i} - y_{s_i}^R, z_{s_{i+1}}^{(r_j)} + w_{s_{i+1}}^{R(r_j)}\} \qquad s_i \in P, i = k+1, \ldots, l-2 \quad (26)$$

$$z_{s_i}^{(r_j)} = \min\{l_{s_i} - y_{s_i}^R, z_{s_{i+1}}^{(r_j)} + w_{s_{i+1}}^{R(r_j)}, R + y_{s_i-n}^R + d_{s_i-n} - y_{s_i}^R\} \qquad s_i \in D, i = k+1, \ldots, l-2. \quad (27)$$

These correspond to the largest feasible postponements of beginning of service at nodes when the customer request r_j is fulfilled by taxi. In this case, updated beginning of service at nodes $y_{s_i}^{R(r_j)}$ are set as follows:

$$y_{s_i}^{R(r_j)} = y_{s_i}^R + \min\{g^{(r_j)}, z_{s_i}^{(r_j)}\} \qquad k < i < l, s_i \in P \quad (28)$$

$$y_{s_i}^{R(r_j)} = \max\{y_{s_i}^R, y_{s_{i-1}}^{R(r_j)} + d_{s_{i-1}} + t_{s_{i-1}s_i}\} \qquad k < i < l, s_i \in N \setminus P, \quad (29)$$

with $y_{s_k}^{R(r_j)} = y_{s_k}^R$. Note that the computation of the forward time slack variables implies that, even if the beginning of service is further postponed at a pickup node j, a maximal ride time of R still holds with respect to y_j^R.

If several delivery nodes are skipped by the vehicle, the services times at nodes must be updated accordingly. Let r_{j_1} and r_{j_2} be two customer requests whose corresponding delivery nodes $n + j_1, n + j_2 \in D$ are skipped by the vehicle with probability $(1 - p_{j_1})$ and $(1 - p_{j_2})$, respectively. Let $(s_0, \ldots, s_h = j_1, \ldots, s_k = j_2, \ldots, s_l = n + j_1, \ldots, s_m = n + j_2, \ldots, s_{2n+1})$ be the feasible tour. The updated waiting times are then defined as:

$$w_{s_{l+1}}^{R(r_{j_1}, r_{j_2})} = \max\{0, y_{s_{l+1}}^{R(r_{j_2})} - t_{s_{l-1}s_{l+1}} - d_{s_{l-1}} - y_{s_{l-1}}^{R(r_{j_2})}\} \qquad m > l+1 \quad (30)$$

$$w_{s_{m+1}}^{R(r_{j_1}, r_{j_2})} = \max\{0, y_{s_{m+1}}^{R} - t_{s_{l-1}s_{l+1}} - d_{s_{l-1}} - y_{s_{l-1}}^{R(r_{j_2})}\} \qquad m = l+1 \quad (31)$$

$$w_{s_i}^{R(r_{j_1}, r_{j_2})} = \max\{0, y_{s_i}^{R(r_{j_2})} - t_{s_{i-1}s_i} - d_{s_{i-1}} - y_{s_{i-1}}^{R(r_{j_2})}\} \qquad i = k+2, \ldots, l-1. \quad (32)$$

In what concerns the forward time slack variables, several particular cases must be considered for the node s_{l-1}, which depend on the relative positions

of the nodes s_l and s_m:

$$z_{s_{l-1}}^{(r_{j_1},r_{j_2})} = \min\{l_{s_{l-1}} - y_{s_{l-1}}^{R(r_{j_2})}, z_{s_{l+1}}^{(r_{j_2})} - y_{s_{l+1}}^{R(r_{j_2})} + y_{s_{l+1}}^{R} + w_{s_{l+1}}^{R(r_{j_1},r_{j_2})}\}$$
$$s_{l-1} \in P, m > l+1 \quad (33)$$

$$z_{s_{l-1}}^{(r_{j_1},r_{j_2})} = \min\{l_{s_{l-1}} - y_{s_{l-1}}^{R(r_{j_2})}, z_{s_{l+1}}^{(r_{j_2})} - y_{s_{l+1}}^{R(r_{j_2})} + y_{s_{l+1}}^{R} + w_{s_{l+1}}^{R(r_{j_1},r_{j_2})},$$
$$R - y_{s_{l-1}}^{R(r_{j_2})} + y_{s_{l-1}-n}^{R} + d_{s_{l-1}-n}\}$$
$$s_{l-1} \in D, m > l+1 \quad (34)$$

$$z_{s_{l-1}}^{(r_{j_1},r_{j_2})} = \min\{l_{s_{l-1}} - y_{s_{l-1}}^{R(r_{j_2})}, z_{s_{m+1}} - y_{s_{m+1}}^{R} + y_{s_{m+1}}^{r} + w_{s_{m+1}}^{R(r_{j_1},r_{j_2})}\}$$
$$s_{l-1} \in P, m = l+1 \quad (35)$$

$$z_{s_{l-1}}^{(r_{j_1},r_{j_2})} = \min\{l_{s_{l-1}} - y_{s_{l-1}}^{R(r_{j_2})}, z_{s_{m+1}} - y_{s_{m+1}}^{R} + y_{s_{m+1}}^{r} + w_{s_{m+1}}^{R(r_{j_1},r_{j_2})},$$
$$R - y_{s_{l-1}}^{R(r_{j_2})} + y_{s_{l-1}-n}^{R} + d_{s_{l-1}-n}\}$$
$$s_{l-1} \in D, m = l+1. \quad (36)$$

The remaining forward time slack variables are defined as follows:

$$z_{s_i}^{(r_{j_1},r_{j_2})} = \min\{l_{s_i} - y_{s_i}^{R(r_{j_2})}, z_{s_{i+1}}^{(r_{j_1},r_{j_2})} + w_{s_{i+1}}^{R(r_{j_1},r_{j_2})}\}$$
$$s_i \in P, i = k+1, \ldots, l-2 \quad (37)$$

$$z_{s_i}^{(r_{j_1},r_{j_2})} = \min\{l_{s_i} - y_{s_i}^{R(r_{j_2})}, z_{s_{i+1}}^{(r_{j_1},r_{j_2})} + w_{s_{i+1}}^{R(r_{j_1},r_{j_2})},$$
$$R - y_{s_i}^{R(r_{j_2})} + y_{s_i-n}^{R} + d_{s_i-n}\}$$
$$s_i \in D, i = k+1, \ldots, l-2. \quad (38)$$

For the corresponding beginning of service at nodes, we obtain

$$y_{s_i}^{R(r_{j_1},r_{j_2})} = y_{s_i}^{R(r_{j_2})} + \min\left\{g^{(r_{j_1},r_{j_2})}, z_{s_i}^{(r_{j_1},r_{j_2})}\right\}$$
$$k < i < l, s_i \in P \quad (39)$$

$$y_{s_i}^{R(r_{j_1},r_{j_2})} = \max\{y_{s_i}^{R(r_{j_2})}, y_{s_{i-1}}^{R(r_{j_1},r_{j_2})} + d_{s_{i-1}} + t_{s_{i-1}s_i}\}$$
$$k < i < l, s_i \in D, \quad (40)$$

where

$$g^{(r_{j_1},r_{j_2})} = t_{s_{l-1}s_l} + d_{s_l} + t_{s_l s_{l+1}} - t_{s_{l-1}s_{l+1}} \qquad m > l+1 \quad (41)$$

$$g^{(r_{j_1},r_{j_2})} = t_{s_{l-1}s_l} + d_{s_l} + t_{s_l s_{l+2}} - t_{s_{l-1}s_{l+2}} \qquad m = l+1. \quad (42)$$

Indeed, if the vehicle skips node $n + j_2 \in D$, the beginning of service at pickup nodes $s_i \in P$ such that $k < i < m$ can be postponed by at most $g^{(r_{j_2})}$. Since the vehicle skips both nodes $n + j_1, n + j_2 \in D$ with probability $(1 - p_{j_1})(1 - p_{j_2})$, the beginning of service at pickup nodes $s_i \in P$ such that $k < i < l$ can be further postponed by at most $g^{(r_{j_1}, r_{j_2})}$. This quantity results from the elimination of node $n + j_1$ from the tour, considering that node $n + j_2$ has already been skipped. As before, the beginning of service at delivery nodes are postponed according to the previous modifications. Also, the maximal ride time of R for customer request r_j is still ensured with respect to the updated beginning of service y_j^R at the corresponding pickup node.

The beginning of service at nodes can be updated similarly when more than two delivery nodes are skipped by the vehicle. In order to compute all possible updated beginnings of service at nodes, we use the following pseudo-codes. We denote by $\pi(k)$ the index of node $k \in N$ in the current feasible tour. The purpose of Algorithm 1 is to compute all possible updated beginnings of service at nodes. The updated beginning of service at nodes $y_i^R, i \in N$ are first provided. For each pickup node $i \in P$ which is not immediately before the corresponding delivery node $n + i \in D$ in the Hamiltonian tour, Algorithm 1 also computes the updated beginning of service at nodes $y_{\pi(l)}^{R(r_i)}, \pi(i) < \pi(l) < \pi(n + i)$, i.e., at the nodes lying between i and $n + i$ in the Hamiltonian tour. Next, for each pickup node $j \in P$ such that the corresponding delivery node $n + j \in D$ (i) lies between i and $n + i$ and (ii) is not the immediate successor of j in the Hamiltonian tour, Algorithm 1 selects the maximal index I between $\pi(i)$ and $\pi(j)$, and then calls a function 'Revision$(j, I, y^{R(r_i)})$'. In this function, whose description is provided in Algorithm 2, the updated beginning of service at nodes $y_{\pi(l)}^{R(r_j, r_i)}, I < \pi(l) < \pi(n + j)$ are computed. Then, as above, for each pickup node $k \in P$ such that $n+k \in D$ (i) lies between I and $n+j$ and (ii) is not the immediate successor of k in the Hamiltonian tour, the maximal index J between I and $\pi(k)$ is selected and the recursive function Revision$(k, J, y^{R(r_j, r_i)})$ is called again. The latter will compute the updated beginning of service at nodes $y_{\pi(l)}^{R(r_k, r_j, r_i)}, J < \pi(l) < \pi(n + k)$, and so on until all possible updated beginnings of service at nodes are provided.

Algorithm 1 Computation of all updated beginnings of service at nodes (represented by a vector y)

compute y^R
for all $i \in P$ s.t. $\pi(n+i) > \pi(i)+1$ **do**
  compute $y^{R(r_i)}$ from y^R
  for $j \in P$ s.t. $\pi(i) < \pi(n+j) < \pi(n+i)$ and $\pi(n+j) > \pi(j)+1$ **do**
    $I \leftarrow \max\{\pi(i), \pi(j)\}$
    Revision$(j, I, y^{R(r_i)})$
      {this function computes the updated beginning}
      {of service at nodes of the form $y^{R(r_j,r_i)}$, from $y^{R(r_i)}$}
  end for
end for
return y

Algorithm 2 Revision (j, I, y)

compute $y^{R(r_j,\dots)}$ from y {for instance, compute $y^{R(r_j,r_i)}$ from $y^{R(r_i)}$}
for all $k \in P$ s.t. $I < \pi(n+k) < \pi(n+j)$ and $\pi(n+k) > \pi(k)+1$ **do**
  $J \leftarrow \max\{I, \pi(k)\}$
  Revision$(k, J, y^{R(r_j,\dots)})$
    {this function computes the updated beginning}
    {of service at nodes of the form $y^{R(r_k,r_j,\dots)}$, from $y^{R(r_j,\dots)}$}
end for

The delay cost associated with a feasible Hamiltonian tour of the form $(s_0, s_1, \ldots, s_{2n}, s_{2n+1})$ can now be defined as follows:

$$\begin{aligned}\Theta(x^r, y^r) &= E_\xi \Theta(x^r, y^r, \xi) \\ &= \sum_{s_i \in D} b_{s_i - n}(1 - v_{s_i}) + \sum_{i=0}^{2n} \sum_{j=i+1}^{2n+1} c_{s_i s_j} v_{s_i} v_{s_j} \prod_{k=i+1}^{j-1} (1 - v_{s_k}) - \sum_{i=0}^{2n} c_{s_i s_{i+1}}, \quad (43)\end{aligned}$$

where $\prod_{k=i+1}^{j-1}(1 - v_{s_k}) = 1$ for $i = j - 1$, and v_{s_i} is the probability that the vehicle actually visits node $s_i \in N$. This probability is defined as

$$v_{s_i} = \begin{cases} 1 & \text{if } s_i \in P \cup \{0, 2n+1\} \\ p_{s_i - n} & \text{if } s_i \in D, \end{cases} \quad (44)$$

where p_{s_i} is the probability that the vehicle picks up the customer at node $s_i \in P$. To obtain the latest p_{s_i}, we need to aggregate the probabilities that the vehicle picks up the customer at node $s_i \in P$, over all possible values for the beginning of service variable y_{s_i}. Define $R(s_i)$ as the set of customer requests r_j such that node s_i appears between the nodes j and $n + j$ on the tour, i.e., $R(s_i) = \{r_j : \pi(j) < i < \pi(n + j)\}$. The set $Y(s_i)$ of all possible values for y_{s_i} can be described as $Y(s_i) = \{y_{s_i}^{R(S)} : S \subseteq R(s_i)\}$. We obtain:

$$\begin{aligned} p_{s_i} &= \sum_{y_{s_i}^{R(S)} \in Y(s_i)} P(\xi_{s_i} \leq y_{s_i}^{R(S)} - e_{s_i}) P(y_{s_i} = y_{s_i}^{R(S)}) \quad (45) \\ &= \sum_{y_{s_i}^{R(S)} \in Y(s_i)} p_{s_i}\big|_{y_{s_i}^{R(S)}} P(y_{s_i} = y_{s_i}^{R(S)}). \quad (46) \end{aligned}$$

with

$$P\big(y_{s_i} = y_{s_i}^{R(S)}\big) = \prod_{r_k \in S} (1 - p_k) \prod_{r_k \in R(s_i) \setminus S} p_k. \quad (47)$$

Indeed, the probability that the beginning of service at node $s_i \in P$ equals $y_{s_i}^{R(S)}$ depends on what happened before this node on the Hamiltonian tour. More precisely, it depends on the probabilities p_k that the vehicle picks up the customer at node k, where $r_k \in R(s_i)$.

Note that, because calculating all possible updated beginnings of service at nodes of N is computationally expensive, the same is true for the computation of the exact delay cost $\Theta(x^r, y^r)$ using (43) to (47). This explains why,

in Step 5 of the integer L-shaped algorithm, a lower bound $\underline{\theta}$ for $\Theta(x^r, y^r)$ is first computed. In the following section, we provide a valid setting for $\underline{\theta}$ and focus on the optimality cuts appended to the stochastic model in Steps 5 and 6 of the integer L-shaped algorithm.

3.2. Optimality cuts

Every time a feasible solution (x^r, y^r, θ^r) of the stochastic model is found in Step 5 of the integer L-shaped algorithm, a lower bound $\underline{\theta}$ for $\Theta(x^r, y^r)$ is computed. Similarly to Hjorring and Holt [13], we define a partial route as a sequence $(s_0 = 0, s_1, s_2, \ldots, s_p)$ such that $s_p \in P$ and $x_{s_i s_{i+1}} = 1$ for $i = 0, \ldots, p-1$. The next proposition provides a valid setting for $\underline{\theta}$.

Proposition 1 *Assume that $(s_0 = 0, s_1, s_2, \ldots, s_p)$ is a partial route, where $s_p \in P$ and $V \subseteq N$ is the corresponding node set. Then a lower bound for the delay cost associated to the partial route can be computed as:*

$$\underline{\theta} = \sum_{s_i \in D \cap V} b_{s_i - n}(1 - \bar{v}_{s_i}) + \sum_{i=0}^{p-1} \sum_{j=i+1}^{p} c_{s_i s_j} \underline{v}_{s_i} \underline{v}_{s_j} \prod_{k=i+1}^{j-1} (1 - \bar{v}_{s_k}) - \sum_{i=0}^{p-1} c_{s_i s_{i+1}} \tag{48}$$

where $\prod_{k=i+1}^{l}(1 - \bar{v}_{s_k}) = 1$ for $i = l$, while $\underline{v}_{s_i}$ and $\bar{v}_{s_i}$ $(s_i \in V)$ are computed as follows:

$$\underline{v}_{s_i} = \bar{v}_{s_i} = 1 \qquad s_i \in P \tag{49}$$

$$\underline{v}_{s_i} = \underline{p}_{s_i - n} = P(\xi_{s_i - n} \leq \underline{y}_{s_i - n} - e_{s_i - n}) \qquad s_i \in D \tag{50}$$

$$\bar{v}_{s_i} = \bar{p}_{s_i - n} = P(\xi_{s_i - n} \leq \bar{y}_{s_i - n} - e_{s_i - n}) \qquad s_i \in D, \tag{51}$$

with

$$\underline{y}_{s_0} = e_0 \tag{52}$$

$$\underline{y}_{s_i} = \max\{e_{s_i}, y_{s_{i-1}} + d_{s_{i-1}} + t_{s_{i-1} s_i}\} \qquad 1 \leq i \leq p \tag{53}$$

$$\bar{y}_{s_p} = \min\left\{l_{s_p}, l_{n+s_p} - t_{s_p, n+s_p} - d_{s_p}, \min_{i \in P \setminus V}\{l_{s_i} - t_{s_p, s_i} - d_{s_p}\}\right\} \tag{54}$$

$$\bar{y}_{s_i} = \min\left\{l_{s_i}, l_{n+s_i} - t_{s_i, n+s_i} - d_{s_i}, \bar{y}_{s_k} - t_{s_i, s_k} - d_{s_i}\right\} \qquad s_i \in P \tag{55}$$

$$\bar{y}_{s_i} = \min\left\{l_{s_i}, \bar{y}_{s_k} - t_{s_i, s_k} - d_{s_i}\right\} \qquad s_i \in D, \tag{56}$$

where s_k is the next pickup node appearing after s_i in the partial route. Further, for any pickup node s_i such that the corresponding delivery node $n + s_i$ appears before s_k in the partial route, we can replace l_{n+s_i} by $\bar{y}_{n+s_i}$ in (55).

PROOF The lower bound $\underline{\theta}$ can be decomposed into several parts: a reduction in the tour cost according to the sequence of nodes that are actually visited by the vehicle, and an additional taxi cost for customer requests whose corresponding delivery node belongs to the partial route.

Since s_1 and s_p are pickup nodes, these are actually visited by the vehicle. Hence the reduction in the tour cost for the partial route is equal to the total cost of the successive arcs that are actually traversed by the vehicle, minus the cost of all arcs belonging to the sequence.

The probability p_{s_i} that the vehicle picks up the customer at node $s_i \in P$ is $p_{s_i} = P(\xi_{s_i} \leq y_{s_i} - e_{s_i})$, where y_{s_i} is the beginning of service at node s_i. By constraints (6) and (12), the lower bounds (52) and (53) can easily be deduced. Next, the beginning of service at node s_i can be postponed depending on the delivery nodes that the vehicle will skip on its tour. One can easily check that it cannot be scheduled after $\bar{y}_{s_i}$, and we obtain $1 - p_{s_i} = P(\xi_{s_i} > y_{s_i} - e_{s_i}) \geq P(\xi_{s_i} > \bar{y}_{s_i} - e_{s_i})$. The result follows. □

Now let $S \subseteq A$ be the arc set corresponding to a partial route. As in Hjorring and Holt [13], a *general optimality cut* for this partial route is:

$$\theta \geq \underline{\theta} \left(\sum_{(i,j) \in S} x_{ij} - |S| + 1 \right), \tag{57}$$

where $\underline{\theta}$ is a lower bound on the delay cost associated to the partial route. The number of general optimality cuts associated with any feasible Hamiltonian tour x^r is in $O(n)$. In order to avoid appending all these to the stochastic model in Step 5, this step is executed as follows:

Step 5.1 (Initialization) Let $(s_0 = 0, s_1, \ldots, s_{2n}, s_{2n+1} = 2n + 1)$ represent the feasible Hamiltonian tour x^r. Set a boolean $b = 0$, an arc counter $k = 1$, $S = \{(s_0, s_1)\}$ and $\underline{\theta} = 0$.

Step 5.2 (Partial route construction) While $s_{k+1} \in D$, set $S = S \cup \{(s_k, s_{k+1})\}$ and $k = k + 1$. If $s_{k+1} = 2n + 1$: stop (go to Step 6 of the algorithm).

Step 5.3 (Delay cost lower bounding) Set $S = S \cup \{(s_k, s_{k+1})\}$, $k = k + 1$ and update $\underline{\theta}$. If $b = 0$ and $\underline{\theta} \geq \lambda \theta^r$ $(\lambda \in \mathbb{R}_0^+)$, append the corresponding optimality cut to the model and set $b = 1$.

Step 5.4 (Optimality cut test) If $c^T x^r + \underline{\theta} \geq z^*$, append the corresponding optimality cut to the stochastic model and stop (go to Step 2 of the algorithm). Otherwise go to Step 5.2.

With the above decomposition of Step 5, at most two general optimality cuts are appended to the stochastic model from a given feasible solution (x^r, y^r, θ^r). Next, if we obtain $s_{k+1} = 2n + 1$ in Step 5.2, this means that the partial route corresponds to the feasible Hamiltonian tour x^r. In this case, the exact delay cost $\Theta(x^r, y^r)$ is computed in Step 6 and compared to the current value θ^r. If $\Theta(x^r, y^r) > \theta^r$, then the *specific optimality cut*

$$\theta \geq \Theta(x^r, y^r) \left(\sum_{(i,j) \in A : x_{ij}^r = 1} x_{ij} - 2n \right) \tag{58}$$

is included in the stochastic model. However, we should note that this cut is only active when $x = x^r$, which means that numerous optimality cuts could be required during the algorithm.

4. Computational results

The integer L-shaped algorithm for the single-vehicle S-DARP was incorporated within the branch-and-cut algorithm of Cordeau [7] and tested on several instances. The algorithm was programmed in C++ and implemented with ILOG CPLEX 10.1 and the Concert Library. All tests were run on an AMD Opteron 285 computer (2.6 GHz) running Linux.

The algorithm was applied to four sets of randomly generated instances involving from 12 to 26 customer requests. As in Cordeau [5], the node positions are randomly chosen in a square $[-10, 10]^2$ according to a continuous uniform distribution, and the depot is located at the center of the square. Routing costs and travel times are both equal to the Euclidean distance between the nodes. All instances include half inbound requests and half outbound requests. For an inbound request, an earliest time e_i at the pickup node is randomly generated in $[0, T - 60]$, where T is the maximal tour duration. For an outbound request, a deadline l_{n+i} at the delivery node is randomly generated in $[60, T]$. The corresponding deadline l_i at the pickup node and earliest time e_{n+i} at the delivery node are then set according to a prespecified time window width. The latter is equal to 15 for half of the inbound and outbound requests; the time window width for the other half of the requests is equal to 30 in instances C1 and CL1, and to 60 in instances D1 and DL1. Furthermore, instances C1 and D1 are generated with $R = 30$, $Q = 3$, $q_i = 1$ and $d_i = 3$. Instances CL1 and DL1 are generated with $R = 45$, $Q = 6$ and $d_i = q_i$, where q_i is randomly chosen according to a

uniform distribution on $\{1, \ldots, Q\}$. Finally, the maximum tour duration is set to $T = 720$ for instances with up to 18 customer requests, and to $T = 840$ otherwise.

In what concerns the taxi costs, we consider that the cost of a taxi from node i to node $n + i$ is equal to twice the Euclidean distance between these nodes plus a fixed cost of 25, i.e., $b_i = 2d_{i,n+i} + 25$. The fixed cost can be interpreted as an administrative cost related to calling a taxi and updating data in the computer system. Furthermore, we assume that each stochastic delay variable ξ_i ($i \in P$) follows a semi-triangular distribution on the interval $[0, l_i - e_i]$, whose density function $f(x)$ is given by

$$f(x) = \frac{-2x}{(l_i - e_i)^2} + \frac{2}{l_i - e_i} \qquad x \in [0, l_i - e_i]. \tag{59}$$

It follows that

$$P(\xi_i \leq x) = \int_0^x \frac{-2t}{(l_i - e_i)^2} + \frac{2}{l_i - e_i} dt = \frac{-x^2}{(l_i - e_i)^2} + \frac{2x}{l_i - e_i}. \tag{60}$$

In Tables 1 to 4, we compare several possible choices in terms of optimality cuts for instances C1, D1 and CL1, DL1, respectively. The columns 'CPU' and 'Cuts' provide the CPU times (in seconds) and the number of optimality cuts appended to model S-DARP. The first six columns provide the results obtained when adding general optimality cuts during the algorithm. The notation 'O1' means that a cut is appended to the model when the current feasible solution is such that $c^T x^r + \underline{\theta} \geq z^*$, i.e., a lower bound on the current objective function value is larger that the best current objective function value. The notation 'O2 (1.1)' (resp. 'O2 (2.5)') means that a cut is appended to the model when $\underline{\theta} \geq \lambda \theta^r$ with $\lambda = 1.1$ (resp. $\lambda = 2.5$), i.e., a lower bound on the current delay cost exceeds the current θ^r value by 10% (resp. 150%). The last two columns provide the results obtained when appending only specific cuts during the algorithm. We have also imposed a time limit of two hours on the solution of any instance, after which the solution process was aborted.

From Tables 1 to 4, we observe that appending general optimality cuts to S-DARP allows us to solve more instances than with specific optimality cuts. Also, appending general cuts yields smaller CPU times and fewer cuts than with specific cuts. However, the differences are not always important. The largest instances solved to optimality within two hours involve 20 customer requests for instances C1, 14 customer requests for instances D1, 26

	General cuts						Specific cuts	
	O1 + O2 (1.1)		O1 + O2 (2.5)		O1			
Inst.	Cuts	CPU	Cuts	CPU	Cuts	CPU	Cuts	CPU
C1-12	161	6	161	6	160	6	166	6
	101	3	101	3	101	3	100	3
	26	3	26	3	34	2	47	3
C1-14	4154	328	4146	328	3958	350	6395	560
	66	4	66	4	66	4	73	4
	1260	82	1296	80	1250	85	1833	122
C1-16	10054	3466	10052	3795	10135	4154	10368	3900
	3370	260	3366	272	3669	396	11574	2240
	21977	7200	21834	7200	20945	7200	22045	7200
C1-18	754	65	754	65	751	65	759	64
	15549	7200	15209	7200	15175	7200	15310	7200
	18050	7200	18241	7200	17985	7200	19751	7200
C1-20	16810	7200	17476	7200	16076	7200	17861	7200
	5659	1327	5505	1365	5252	1415	5387	1615
	18859	7200	18043	7200	16338	7200	15824	7200

Table 1: Appending general or specific optimality cuts to (S-DARP) for instances C1

	General cuts						Specific cuts	
	C1 + C2 (1.1)		C1 + C2 (2.5)		C1			
Inst.	Cuts	CPU	Cuts	CPU	Cuts	CPU	Cuts	CPU
D1-12	310	20	310	21	310	21	350	23
	232	14	232	14	284	19	363	21
	14744	7200	14513	4002	13478	7091	17247	7200
D1-14	13995	7200	12883	7200	13508	7200	16052	7200
	5499	608	5499	609	5499	601	5495	546
	780	43	780	44	780	44	780	42

Table 2: Appending general or specific optimality cuts to (S-DARP) for instances D1

customer requests for instances CL1, and 22 customer requests for instances DL1. For most instances, the best strategy in terms of optimality cuts consists in appending both types ('O1' and 'O2') of general cuts. Furthermore,

	General cuts						Specific cuts	
	C1 + C2 (1.1)		C1 + C2 (2.5)		C1			
Inst.	Cuts	CPU	Cuts	CPU	Cuts	CPU	Cuts	CPU
CL1-12	190	5	189	5	186	6	186	5
	591	23	591	25	591	23	622	24
	33	1	33	1	33	1	33	1
CL1-14	42	2	42	2	40	3	114	8
	21530	7200	20350	7200	21932	7200	23400	7200
	932	52	928	49	944	50	1306	63
CL1-16	69	3	68	3	66	4	66	4
	21374	7200	18652	7200	19187	7200	19840	7200
	1533	115	1533	104	1532	102	1841	120
CL1-18	116	7	114	7	115	7	148	8
	2273	202	2273	200	2130	205	2402	222
	14011	7200	14206	7200	14167	7200	16217	7200
CL1-20	4987	677	4987	641	4983	650	4983	627
	18197	7200	19152	7200	17818	7200	18866	7200
	137	14	137	14	137	14	137	14
CL1-22	10581	2215	10567	2174	10564	2300	10564	2112
	17122	7200	17292	7200	16287	7200	17492	7200
	14555	7200	14916	7200	14859	7200	15919	7200
CL1-24	17879	7200	19278	7200	17799	7200	19488	7200
	11911	7200	12075	7200	9311	7200	10045	7200
	5041	1215	4825	1129	4709	1164	4709	1083
CL1-26	2541	572	2290	572	2107	559	2138	547
	5731	7200	6273	7200	4837	7200	5212	7200
	14982	7200	15430	7200	15213	6913	18309	7200

Table 3: Appending general or specific optimality cuts to (S-DARP) for instances CL1

the second type 'O2' of general cuts should not be used too often, and the parameter $\lambda = 2.5$ is preferred to $\lambda = 1.1$. However, note that appending only the first type 'O1' of general cuts provides better CPU times for several large instances.

In Tables 5 to 8, we compare the results obtained by solving the S-DARP with the integer L-shaped algorithm to the results obtained by first solving the corresponding deterministic model, and then computing the expected

	General cuts						Specific cuts	
	C1 + C2 (1.1)		C1 + C2 (2.5)		C1			
Inst.	Cuts	CPU	Cuts	CPU	Cuts	CPU	Cuts	CPU
DL1-12	306	8	303	8	316	9	364	10
	2549	182	2549	182	2525	157	4236	292
	286	31	286	31	286	31	287	30
DL1-14	103	3	98	3	94	3	97	3
	1523	128	1522	130	1516	130	1607	132
	147	8	147	8	147	8	147	7
DL1-16	367	34	363	34	351	35	418	35
	7431	1486	7845	1698	6950	1294	10533	2327
	16913	7200	16639	7200	11763	7200	14429	7200
DL1-18	882	64	882	64	882	64	882	61
	21976	7200	21722	7200	20688	7200	22396	7200
	750	148	741	150	607	153	1418	221
DL1-20	15253	7200	15280	7200	15089	7200	16604	7200
	12973	7200	13059	7200	13166	7200	14050	7200
	366	27	364	27	291	26	295	26
DL1-22	2774	359	2774	356	2774	358	2773	344
	13080	7200	12945	7200	13235	7200	14311	7200
	14589	7200	12636	7200	13342	7200	14539	7200

Table 4: Appending general or specific optimality cuts to (S-DARP) for instances DL1

delay cost associated with the optimal solution. Both types of general optimality cuts (with parameter $\lambda = 2.5$) are appended during the algorithm, i.e., cuts are added whenever $c^T x^r + \underline{\theta} \geq z^*$ or $\underline{\theta} \geq 2.5\theta^r$. To compare the stochastic and deterministic models in terms of optimal solution values, we have also computed the delay cost associated to the deterministic optimal solution. Columns 'F.Cost' and 'D.Cost' denote the fixed and delay costs associated with the optimal tour. For the stochastic model, column 'IGap' provides the percent gap between the first integer feasible solution and the optimal solution. For the deterministic model, column 'Gap' provides the percent gap between the optimal solutions of the linear relaxation and of the integer problem, respectively. Columns 'CPU' and 'Nodes' provide the CPU times (in seconds) and the number of nodes in the branch-and-cut tree. As above, a time limit of two hours was imposed, after which the solution

process was aborted. In this case, the reported results are those corresponding to the best integer feasible solution found by the algorithm. Finally, the column '% Red' provides the percent cost reduction achieved by solving the stochastic model optimally, compared with solving a deterministic model and then computing the associated delay cost.

	Stochastic					Deterministic				
Inst.	F.Cost	D.Cost	IGap	CPU	Nodes	F.Cost	D.Cost	Gap	CPU	% Red
C1-12	142.84	3.24	22.42	6	428	125.86	63.30	0.00	<1	22.76
	142.54	5.41	31.91	3	209	139.84	11.18	0.00	<1	2.02
	133.74	0.34	34.63	3	184	121.84	58.67	0.00	<1	25.71
C1-14	153.54	12.05	30.85	328	14233	150.14	54.41	7.35	3	19.04
	146.00	5.10	9.39	4	212	142.12	23.17	0.00	<1	8.57
	135.87	6.24	36.36	80	4283	123.93	61.50	0.00	<1	23.35
C1-16	159.70	44.96	11.84	3795	34475	141.12	90.13	8.69	7	11.49
	173.40	21.06	23.07	272	9488	157.84	80.80	0.00	<1	18.5
	161.39	32.94	15.05	7200	75737	146.07	89.84	3.55	3	17.62
C1-18	208.93	26.57	19.15	65	2056	204.26	40.66	2.31	3	3.84
	191.45	27.66	28.58	7200	48320	176.38	107.33	5.91	4	22.76
	185.34	18.99	49.72	7200	70061	171.61	72.70	0.00	1	16.35
C1-20	195.27	26.02	19.60	7200	45831	181.90	125.56	1.64	4	28.02
	194.36	54.00	24.43	1365	17174	188.99	120.06	3.73	5	19.63
	182.49	42.69	18.12	7200	61950	164.97	101.02	16.55	433	15.33

Table 5: Comparison of stochastic and deterministic optimal solutions for instances C1

Comparing Tables 5, 6, 7 and 8, we conclude that the wider time windows of instances D1 and DL1 make the problem more difficult to solve. Indeed, we observe larger CPU times for these instances, both for the stochastic and for the deterministic models. These tables also show that instances C1 and D1 are more difficult to solve than instances CL1 and DL1. Hence the S-DARP is easier to solve for larger vehicles.

As expected, the CPU times are larger for the S-DARP than for the corresponding deterministic model. For the former problem, we also observe very large gaps between the first integer feasible solutions and the optimal solutions, as well as a large number of nodes in the branch-and-cut tree. This can be explained by the fact that the integer L-shaped algorithm starts

	Stochastic					Deterministic				
Inst.	F.Cost	D.Cost	IGap	CPU	Nodes	F.Cost	D.Cost	Gap	CPU	% Red
D1-12	117.35	0.96	35.93	21	1564	109.80	36.84	5.53	1	19.31
	150.64	8.67	44.62	14	1158	142.38	82.68	0.00	<1	29.20
	115.32	23.16	29.64	4002	85047	84.86	94.67	0.00	<1	22.86
D1-14	135.20	5.77	31.85	7200	130725	119.58	68.53	10.09	62	25.05
	141.64	13.95	5.32	609	19044	140.24	20.80	5.12	2	3.37
	129.12	16.77	20.42	44	2602	128.86	17.17	0.00	<1	0.09

Table 6: Comparison of stochastic and deterministic optimal solutions for instances D1

without any information on the delay cost. Yet, we can solve several small to medium size instances. Furthermore, from the last columns '% Red' of Tables 5 to 8, we conclude that using a stochastic model yields a significant reduction of the optimal solution values (i.e., fixed costs plus delay costs).

Since solving the deterministic model does not encourage Hamiltonian tours with late beginnings of service at the nodes, we questioned the fairness of the above comparison. In order to better assess the value of a stochastic model, we have compared the corresponding optimal solution values with those of restricted or penalized deterministic models, both constructed to encourage Hamiltonian tours with late beginnings of service at the nodes. In the restricted deterministic model, the earliest times e_i of inbound requests (i.e., those for which a desired departure time is specified by the customer) were increased by $E(\xi_i) = (l_i - e_i)/3$. In the penalized deterministic model, the term $\sum_{i \in P}(l_i - y_i)/(l_i - e_i)$ was appended to the objective function. We then compared the corresponding optimal solution values with those of the stochastic and deterministic models presented earlier. We first observed that several instances became infeasible when reducing the time window width of inbound requests. In addition, we observed that the modified models do not necessarily decrease the gap with respect to the optimal solution values of the stochastic model. Indeed, modifying the deterministic model implies changes in the optimal tour, which could force the vehicle to serve some customers earlier than what is desirable. Instead, the stochastic model allows the identification of good tradeoffs, i.e., it encourages the vehicle to serve some customers early and thus to follow a tour in which several other customers are served sufficiently late.

	Stochastic					Deterministic				
Inst.	F.Cost	D.Cost	IGap	CPU	Nodes	F.Cost	D.Cost	Gap	CPU	% Red
CL1-12	123.12	13.44	18.43	5	385	112.00	49.73	0.00	<1	15.55
	111.31	6.66	13.66	25	1914	103.02	31.07	0.00	<1	12.01
	128.39	5.58	0.31	1	84	125.46	8.92	0.00	<1	0.30
CL1-14	156.53	3.36	13.94	2	111	147.10	35.09	0.00	<1	12.23
	130.47	8.62	21.30	7200	68370	119.36	59.26	3.15	1	22.12
	131.01	15.01	1.25	49	2911	122.32	25.53	0.00	<1	1.23
CL1-16	184.23	26.81	27.40	3	133	174.66	80.72	0.00	<1	17.35
	153.61	18.60	14.24	7200	59626	146.94	69.20	8.17	10	20.31
	154.02	33.39	28.47	104	4464	148.40	82.07	0.00	<1	18.67
CL1-18	183.10	35.24	17.95	7	238	182.17	75.35	0.00	<1	15.20
	177.36	18.84	33.55	200	6817	177.34	32.37	0.00	<1	6.43
	170.17	42.06	26.07	7200	69633	156.06	100.29	0.83	1	17.20
CL1-20	206.04	9.23	16.89	641	13203	195.16	45.75	0.00	<1	10.63
	191.27	0.00	60.83	7200	46160	155.62	62.04	5.09	6	12.11
	210.37	20.39	0.48	14	450	205.01	26.85	0.00	<1	0.46
CL1-22	233.57	31.55	26.07	2174	22132	225.59	80.74	6.87	7	13.44
	212.21	8.07	17.23	7200	46421	175.00	81.38	0.23	1	14.07
	203.82	12.12	26.25	7200	57918	193.74	57.27	2.40	2	13.96
CL1-24	217.06	46.76	1.30	7200	48587	209.80	89.24	0.00	1	11.77
	212.71	57.06	20.11	7200	36330	185.44	132.03	3.94	10	15.02
	256.60	75.21	30.54	1129	13192	247.77	125.20	3.58	9	11.03
CL1-26	214.75	70.89	10.36	572	7453	206.37	108.88	6.25	20	9.38
	251.20	47.90	30.61	7200	56776	238.39	154.94	9.42	340	23.95
	276.76	35.19	26.51	7200	45867	260.37	176.48	3.11	10	28.58

Table 7: Comparison of stochastic and deterministic optimal solutions for instances CL1

5. Conclusion

This paper was concerned with a single-vehicle Dial-a-Ride Problem with stochastic customer delays, a problem often arising when customers need to be picked up after a medical appointment. The aim is then to determine an a priori Hamiltonian tour minimizing the expected cost of the tour followed by the vehicle. Since customer delays can yield important modifications of the objective function, we have decomposed the actual cost of a tour into two parts. The first one corresponds to the deterministic tour cost, whereas

	Stochastic					Deterministic				
Inst.	F.Cost	D.Cost	IGap	CPU	Nodes	F.Cost	D.Cost	Gap	CPU	% Red
DL1-12	116.31	9.49	19.48	8	640	105.45	52.41	0.00	<1	20.30
	101.54	3.30	34.55	182	11058	93.14	47.92	0.00	<1	25.67
	113.01	12.67	45.91	31	2561	106.78	31.03	2.62	2	8.79
DL1-14	152.58	16.79	12.75	3	176	145.84	39.35	0.00	<1	8.53
	161.06	6.42	26.90	130	7364	159.68	9.75	3.61	1	1.14
	140.98	9.60	51.04	8	418	136.15	87.61	0.00	<1	32.69
DL1-16	163.94	17.68	36.22	34	1473	154.37	53.43	5.66	2	12.59
	141.31	0.00	52.98	1698	35343	125.75	82.58	0.00	<1	32.16
	148.56	7.77	16.32	7200	55011	137.09	44.75	5.46	7	14.02
DL1-18	177.15	7.21	17.18	64	2122	168.13	42.06	0.00	<1	12.28
	180.41	17.97	36.96	7200	42428	166.74	78.30	2.94	1	19.03
	168.97	22.56	1.59	150	5047	167.93	26.64	4.28	3	1.55
DL1-20	200.05	5.36	14.47	7200	76469	196.04	27.72	3.74	2	8.19
	172.31	30.44	12.97	7200	40324	168.36	69.77	8.35	12	14.85
	200.34	14.84	14.08	27	850	185.55	59.93	0.00	<1	12.33
DL1-22	219.97	18.2	13.43	356	7512	209.11	61.05	0.18	1	11.83
	215.67	43.04	14.50	7200	55415	187.19	160.39	4.05	13	25.56
	207.67	32.46	4.96	7200	45647	206.29	38.16	6.58	19	1.76

Table 8: Comparison of stochastic and deterministic optimal solutions for instances DL1

the second one is a cost associated with stochastic customer delays. We have described an integer L-shaped algorithm for the problem. Computational results have shown that this algorithm provides optimal solutions for small to medium size instances within reasonable computing times. We have also observed that solving the problem as a stochastic program instead of a deterministic program can yield reductions of up to 33% in the expected solution cost.

Acknowledgments

This work was partly funded by the Canadian Natural Sciences and Engineering Research Council under grants 227837-09 and 39682-10. This support is gratefully acknowledged.

References

[1] E. Bartolini. *Algorithms for network design and routing problems*. PhD thesis, Università di Bologna, 2009.

[2] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[3] A.M. Campbell and B.W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42:1–21, 2008.

[4] A.M. Campbell and B.W. Thomas. Runtime reduction techniques for the probabilistic traveling salesman problem with deadlines. *Computers & Operations Research*, 36:1231–1248, 2009.

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

[6] J-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153:29–46, 2007.

[7] J-F. Cordeau, G. Laporte, J.-Y. Potvin, and M.W.P. Savelsbergh. Transportation on demand. In C. Barnhart and G. Laporte, editors, *Handbook in Operations Research & Management Science*, volume 14, pages 429–466. Elsevier Amsterdam, 2007.

[8] J-F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Handbook in Operations Research & Management Science*, volume 14, pages 367–428. Elsevier Amsterdam, 2007.

[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] J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical Management Science*, 6:301–325, 1986.

[11] M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29:143–155, 1995.

[12] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88:3–12, 1996.

[13] C. Hjorring and J.Holt. New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research*, 86:569–584, 1999.

[14] P. Jaillet. *Probabilistic traveling salesman problems*. PhD thesis, Massachusetts Institute of Technology, 1985.

[15] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36:929–936, 1988.

[16] G. Laporte and F.V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[17] G. Laporte, F.V. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42:543–549, 1994.

[18] G. Laporte, F.V. Louveaux, and L. Van hamme. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50:415–423, 2002.

[19] H.N. Psaraftis. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14:130–154, 1980.

[20] H.N. Psaraftis. An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17:351–357, 1983.

[21] S. Ropke and J-F. Cordeau. Branch-and-cut-and-price for the pickup and delivery problem with time windows. *Transportation Science*, 43:267–286, 2009.

[22] S. Ropke, J-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49:258–272, 2007.

[23] M.W.P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4:146–154, 1992.

[24] R.M. Van Slyke and R. J-B. Wets. *L*-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17:638–663, 1969.