REPORTS IN INFORMATICS

ISSN 0333-3590

Local search heuristics for the probabilistic dial-a-ride problem

Sin C. Ho and Dag Haugland

REPORT NO 286 **November 2004**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/ps/2004-286.ps

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Local search heuristics for the probabilistic dial-a-ride problem

Sin C. Ho* Dag Haugland

25th November 2004

Abstract

This paper describes an efficient neighborhood search procedure for the probabilistic dial-a-ride problem. This procedure requires $\mathcal{O}(n^4)$ computations compared to $\mathcal{O}(n^6)$ if computed from scratch. Two heuristics with this search procedure embedded are developed for solving the problem: a tabu search heuristic and a hybrid GRASP-tabu search heuristic. Computational results show that tabu search is superior over GRASP-tabu search on the probabilistic dial-a-ride problem, and by incorporating the search technique resulted in faster heuristics.

Keywords: probabilistic dial-a-ride problem, tabu search, GRASP

*Department of Informatics, University of Bergen, N-5020 Bergen, Norway. {sin,dag}@ii.uib.no

1 Introduction

In the *Dial-a-Ride Problem* (DARP) there are n users who specify transportation requests between given origins and destinations. Users may provide a time window on their desired departure or arrival time, or on both. A fleet of m vehicles based at a common depot is available to operate the routes. Each vehicle has a given capacity. The time each user spends in the vehicle is bounded by a threshold. The DARP consists of constructing a set of feasible minimum cost routes. A common application of the DARP arises in door-to-door transportation of the elderly and the disabled (Madsen et al. 1995, Ioachim et al. 1995, Borndörfer et al. 1997, Toth and Vigo 1997).

The DARP resembles the *Vehicle Routing Problem with Pickup and Delivery* (VRPPD) (Savelsbergh and Sol 1995), but has some extensions since it considers transportation of people rather than goods: Each user specifies time windows of possible pickup and delivery times, as well as an upper bound on the riding time. A common characteristic of DARP instances is that the vehicle capacity constraints are tight.

The DARP is a deterministic model in the sense that all requests are known with certainty when designing vehicle routes. It is defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with vertex set $\mathcal{V} = \mathcal{P} \cup \mathcal{D} \cup \{0\}$, $\mathcal{P} = \{1, \ldots, n\}$ and $\mathcal{D} = \{n+1, \ldots, 2n\}$ and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is the arc set. Subsets $\mathcal{P}$ and $\mathcal{D}$ contain vertices representing origins and destinations for the transportation requests, while 0 denotes the depot. Each vertex pair $(i, i + n)$ represents a request for transportation from origin i to destination $i + n$. With every vertex $i \in \mathcal{P}$ is associated a non-negative load q_i, and with all $i > 0$ we associate a non-negative service duration d_i, and a time window $[e_i, l_i]$. The depot is also associated with a time window $[e_0, l_0]$ representing the operating hours. For notational convenience we also define $q_{i+n} = -q_i$ for all $i \in \mathcal{P}$. Each arc (i, j) is associated with a traveling cost c_{ij} and a travel time t_{ij}. Finally, let L denote the maximum ride time of a user. The DARP consists of designing m vehicle routes on $\mathcal{G}$ such that (i) every route starts and ends at the depot; (ii) for every request i, vertices i and $i + n$ belong to the same route and vertex $i + n$ is visited later than vertex i; (iii) the load of vehicle k does not exceed at any time a preset bound Q_k; (iv) the service at vertex i begins in the interval $[e_i, l_i]$, and every vehicle leaves the depot and returns to the depot in the interval $[e_0, l_0]$; (v) the ride time of any user does not exceed L; (vi) the total routing cost of all vehicles is minimized.

In practice it cannot always be assumed that all requests are known at the time when the vehicle routes have to be constructed. In this paper we formulate the *Probabilistic Dial-a-Ride Problem* (PDARP), where operational constraints are the same as in the deterministic version of the problem, but now each user i requires service only with probability p_i. Assume, that we want to design vehicle tours to be used for a given period of time (more than one day) and that for this period, the set of user requests on a daily basis varies. In order to ensure regularity of service, and for the drivers to be acquainted with the users and with the area they are servicing, the routes are not reoptimized every day. The only adjustments of the initial decisions that are made on a daily basis, is that vertices of users that do not require service are eliminated while keeping the routes order-consistent with the original ones. The aim is to design a solution that minimizes expected travel cost. The probabilistic DARP is a generalization of two problems: the *Probabilistic Traveling Salesman Problem* (PTSP); and the *Probabilistic Pickup and Delivery Traveling Salesman Problem* (PPDTSP). The PTSP was first introduced by Jaillet (1985, 1988) where each vertex is presented with a probability. In the first stage a Hamiltonian tour which go through all the vertices are constructed and the vertices that are present are revealed. In the second stage, the tour is followed by skipping

the absent vertices. Jaillet described theoretical properties, bounds and also several heuristics for this problem. Laporte et al. (1994) describe an exact algorithm for determining an a priori TSP tour of least expected cost. This is the basis for Beraldi et al. (2003) who have studied the PPDTSP, and developed an efficient neighborhood search procedure.

Our problem is an extension of the latter problem where there are m capacitated vehicles instead of one, time windows are present, and user convenience (e.g. maximum ride time of each user) is also considered. Finding a feasible solution to the DARP is $\mathcal{NP}$-hard (Savelsbergh and Sol 1995) and this trivially generalizes to the PDARP. The purpose of this work is to device a heuristic method that with reasonable computational effort provides good, but not necessarily optimal, solutions to PDARP. To this end a fast neighborhood search procedure is essential, and we demonstrate how computational savings can be achieved when evaluating a proposed neighborhood. Two heuristics for the PDARP where this neighborhood search procedure is embedded, are developed.

The remainder of this paper is organized as follows. In Section 2 we briefly review the literature for the DARP and for stochastic vehicle routing problems. A mathematical formulation of the PDARP is given in Section 3. The computation of the expected cost of an a priori PDARP solution is described in Section 4. Section 5 presents the neighborhood evaluation procedure, while descriptions of a tabu search heuristic and a hybrid GRASP-tabu search heuristic are presented in Sections 6 and 7, respectively. This is followed by computational results in Section 8, and by the conclusion in Section 9.

2 Literature review

Much research has been done on the DARP over the past 20 years. Psaraftis (1980, 1983) developed dynamic programming algorithms for the single-vehicle case for both static and dynamic DARP. The complexity of the dynamic programming algorithm is $\mathcal{O}(n^2 3^n)$, and thus limits the size of problems that can be solved to be between 8 to 10 users. Although the method is not able to solve larger problem instances, the proposed approach could still be used as a subroutine in a multi-vehicle method given the number of requests for each vehicle is sufficiently small.

Sexton and Bodin (1985*a*, 1985*b*) treated the single-vehicle case as a subproblem in the multi-vehicle context for which the developed algorithm iterates between solving a routing problem and the associated scheduling problem. Their algorithm is based on Benders decomposition to solve the problem through the alternation between these two components. Computational experiments were carried out using actual data from Gaithersburg, Maryland, and Baltimore, Maryland, where the number of users varies from 7 to 20. The computational study also showed that this method is robust and fast.

In Healy and Moll (1995) a heuristic based on edge exchanges and the idea of sacrificing is developed. Like the former method, their strategy alternates back and forth between an optimize phase and a sacrifice phase. The first phase refers to finding a local minimum using local search with travel length as the primary objective. The next phase searches for solutions which are superior in a sacrificing sense to the current one, x. The objective in this phase is the relative factor of the travel length of x and the size of the neighborhood of x. Results are reported on randomly generated data, where the number of users varies from 10 to 100.

An early algorithm of the multi-vehicle DARP was proposed by Jaw et al. (1986) for which a sequential insertion heuristic is developed. Some service quality constraints are included in their model where an upper bound on the riding time for each user is set, and also time restrictions for when a user can be picked up and

dropped off. The algorithm chooses the users with the earliest pick up times and inserts them into the routes so as to minimize the extra cost of inserting a new user. Results are reported on simulated data of 250 users and on real data from Friedrichschafen, Germany, involving 2617 users.

Madsen et al. (1995) solved a real-life problem involving scheduling services for elderly and disabled people in Copenhagen, Denmark. The problem is characterized by time windows, multiple capacities and multiple objectives. Their proposed insertion heuristic, called REBUS, is a modified version of the one developed by Jaw et al. (1986). The algorithm has been tested on real-life cases involving 24 vehicles and 300 users. REBUS is capable of determining solutions of good quality in small amount of computing time.

Ioachim et al. (1995) developed a method based on clustering. A cluster is a temporary set of users that can feasibly be serviced by the same vehicle. Their approach is to construct a large number of clusters that satisfy all the constraints involved. Computational experiments were conducted on instances ranging from 50 to 250 users, and also on a real-life problem of 2545 users. Their study shows that their technique based on column generation is superior in terms of solution quality, to the parallel insertion approach.

Toth and Vigo (1997) studied a real-life problem involving scheduling service for handicapped people. They developed a parallel insertion heuristic which is capable of determining good solutions for large instances in a small amount of computing time where intermediate infeasible solutions are allowed. A tabu thresholding procedure is also developed to further improve the solution obtained from the parallel method. Results are reported on instances ranging from 276 to 312 requests, showing a great improvement compared with the hand-made solutions.

Borndörfer et al. (1997) also studied a real-life problem arising in Berlin. Like many others, their method is a two-phase approach where clusters of users are created first and then chaining these clusters into feasible vehicle routes. Both the clustering and the chaining approach use set partitioning to identify the best set of clusters such that each request is contained in exactly one cluster, and the best set of vehicle routes covering each cluster exactly once. The computational experiments were conducted on problem instances ranging from 859 to 1771 requests.

Cordeau and Laporte (2003*b*) developed a tabu search heuristic for the multi-vehicle static DARP. Their problem includes constraints such as time windows, vehicle capacity, route duration and the maximum ride time of any user. For each iteration the heuristic removes a request and reinserts it into another route. A feature of their method is the ability of accepting intermediate infeasible solutions. Computational experiments were conducted on randomly generated data ranging from 24 to 144 users and on real-life data involving 200 and 295 users.

Cordeau (2003) studied the same problem as described above. He proposed a branch-and-cut algorithm which uses new valid inequalities for the DARP and also known inequalities for the VRPPD and the VRP. Results are reported on randomly generated data ranging from 16 to 32 users. The study shows that the method is able to reduce the CPU time and the number of nodes in the branch-and-bound tree compared with CPLEX.

Attanasio et al. (2004) studied the dynamic DARP where the main objective is to accept as many requests as possible. Their approach is based on a tabu search heuristic developed for the static case (Cordeau and Laporte 2003*b*). Computational experiments were carried out on the same set of instances described under the static case, and the results show an advantage of applying parallel computing in solving real-time vehicle routing problems. For overviews of the DARP and VRPPD, the reader is referred to Cordeau and Laporte (2003*a*) and Desaulniers et al. (2002), respectively.

The PDARP is a stochastic vehicle routing problem and has, to our knowledge,

not been addressed explicitly. The PTSP was introduced and analyzed by Jaillet (1985, 1988), and Bertsimas (1988) explored this problem further. In the thesis of Jaillet (Jaillet 1985) properties, bounds, asymptotic results and heuristics with worst-case performance were described.

A direct extension of the PTSP is the *Probabilistic Vehicle Routing Problem* (PVRP). As in the PTSP, the absent vertices are skipped in the second stage. Bertsimas studied this problem in his thesis (Bertsimas 1988) and described properties, bounds and heuristics.

Waters (1989) compared three operating policies of scheduling vehicle routes when some customers do not require a visit in a period. He estimated the potential savings of semi-fixed routes or variable routes over fixed routes. For a practical situation, rescheduling does not make it worthwhile to utilize variable routes.

Bertsimas et al. (1990) studied the idea of a priori optimization as a strategy competitive to the strategy of reoptimization. They derived some bounds for the PTSP and PVRP. Based on the ideas of local optimality the authors developed two heuristics for the PTSP. Numerical results show that the obtained solutions are near-optimal.

Laporte et al. (1994) proposed an Integer L-shaped method for PTSP and have solved to optimality instances involving up to 50 vertices.

For reviews and surveys on stochastic vehicle routing problems, the reader is referred to Gendreau et al. (1996) and Kenyon (2000).

3 Model

The PDARP can be formulated as a stochastic integer linear program. Let $\xi = (\xi_i)$ denote a vector of Bernoulli random variables, where ξ_i is equal to 1 if i requires service (with probability p_i). As in Jaillet (1988), vertices with $p_i = 1$ are referred to as *black* vertices, while vertices with $0 < p_i < 1$ are *white* vertices. We assume there is at least one white vertex. The problem is modeled as a two-stage stochastic program with recourse. In the first stage, an a priori solution over $\mathcal{G}$ must be determined before any information on the presence of white vertices is available. In the second stage, requested vertices are visited in the same order as they appear in the a priori solution, while unrequested vertices are skipped. For a particular instance of ξ of the second stage, a number of vertices are skipped, resulting in some cost reduction $D(x, \xi)$. The negative of the expectation of this cost reduction, is referred to as the recourse function, and written $R(x) = -E_\xi D(x, \xi)$. A closed form formula for this function is given in Section 4.

For each arc $(i, j) \in \mathcal{A}$ and vehicle $k \in \mathcal{K}$ (with $|\mathcal{K}| = m$), let $x_{ijk} = 1$ if vehicle k travels directly from vertex i to vertex j, and $x_{ijk} = 0$ otherwise. For each vertex $i \in \mathcal{V}$ and each vehicle $k \in \mathcal{K}$, let B_{ik} be the time vehicle k starts to service vertex i, and Q_{ik} be the load of vehicle k after visiting vertex i. Finally, for each user i, L_{ik} denotes the ride time of user i on vehicle k.

The PDARP consists of determining an a priori solution whose expected cost is minimum, and can be stated (based on Cordeau (2003)) as

$$\min \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} c_{ij} x_{ijk} + R(x) \tag{1}$$

$$\text{subject to} \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{V}} x_{ijk} = 1 \qquad \forall i \in \mathcal{P} \tag{2}$$

$$\sum_{j\in\mathcal{V}} x_{ijk} - \sum_{j\in\mathcal{V}} x_{i+n,jk} = 0 \qquad \forall i \in \mathcal{P}, k \in \mathcal{K} \tag{3}$$

$$\sum_{j\in\mathcal{V}} x_{0jk} = 1 \qquad \forall k \in \mathcal{K} \qquad (4)$$

$$\sum_{j\in\mathcal{V}} x_{jik} - \sum_{j\in\mathcal{V}} x_{ijk} = 0 \qquad \forall i \in \mathcal{P}\cup\mathcal{D}, k \in \mathcal{K} \qquad (5)$$

$$\sum_{i\in\mathcal{V}} x_{i0k} = 1 \qquad \forall k \in \mathcal{K} \qquad (6)$$

$$B_{ik} + d_i + t_{ij} - M_{ijk}(1 - x_{ijk}) \le B_{jk} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V}, k \in \mathcal{K} \qquad (7)$$

$$Q_{ik} + q_j - W_{ijk}(1 - x_{ijk}) \le Q_{jk} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V}, k \in \mathcal{K} \qquad (8)$$

$$L_{ik} = B_{i+n,k} - (B_{ik} + d_i) \qquad \forall i \in \mathcal{P}, k \in \mathcal{K} \qquad (9)$$

$$e_i \le B_{ik} \le l_i \qquad \forall i \in \mathcal{V}, k \in \mathcal{K} \qquad (10)$$

$$t_{i,i+n} \le L_{ik} \le L \qquad \forall i \in \mathcal{P}, k \in \mathcal{K} \qquad (11)$$

$$\max\{0, q_i\} \le Q_{ik} \le \min\{Q_k, Q_k + q_i\} \qquad \forall i \in \mathcal{V}, k \in \mathcal{K} \qquad (12)$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V}, k \in \mathcal{K} \qquad (13)$$

where

$$M_{ijk} = \max\{0, l_i + d_i + t_{ij} - e_j\} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V}, k \in \mathcal{K} \qquad (14)$$

$$W_{ijk} = \min\{Q_k, Q_k + q_i\} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V}, k \in \mathcal{K} \qquad (15)$$

Constraints (2) and (3) impose that each request (i.e. the pick-up and drop-off nodes) is served exactly once and by the same vehicle. Constraints (4)-(6) ensure that vehicle k starts from the depot and terminates its route at the depot. Time and load consistencies are handled by constraints (7) and (8). The ride time of each user is defined by the equalities (9) and is bounded by constraints (11). Time windows and load constraints are imposed by (10) and (12). Finally, the nonnegativity and binary requirements are given by (13).

The validity of constraints (7) and (8) is ensured by (14) and (15).

4 Expected cost of an a priori solution

Beraldi et al. (2003) showed how to compute the expected cost of an a priori tour for one vehicle where they did not have any constraints besides the precedence ones. Although our problem is an extension of theirs, it is a more difficult and complicated problem which involves routing of several vehicles, capacity, time windows and ride time constraints. In this section we will provide the computation of the expected cost of an a priori solution of the PDARP.

Let $C(x)$ be the total cost of solution x and $EC(x)$ its expected value. We let $\tau_k = \mathcal{P}_k \cup \mathcal{D}_k \cup \{0\}$ be the set of vertices visited by vehicle k of solution x, where $\mathcal{P}_k = \{i \in \mathcal{P} : \sum_{j\in\mathcal{V}} x_{ijk} = 1\}$ and $\mathcal{D}_k = \{j \in \mathcal{D} : \sum_{i\in\mathcal{V}} x_{ijk} = 1\}$ (for notational convenience, dependence on x is suppressed). Denote the vertices in τ_k by $i_{k0} = i_{k,n_k+1} = 0$, $i_{k1}, i_{k2}, \ldots, i_{kn_k}$ where $0 \le u < v \le n_k$ means that i_{ku} is visited prior to i_{kv} by vehicle k. Given two vertices $i_{ku}, i_{kv} \in \tau_k \setminus \{0\}$, $u < v$, let $\mathcal{I}^{uv}_{k1} = \{u\}$ if $i_{kv} = i_{ku} + n$, and $\mathcal{I}^{uv}_{k1} = \{u, v\}$ otherwise, $\mathcal{I}^{uv}_{k2} = \{r \in \{u+1, \ldots, v-1\} : i_{kr} \le n\}$ and $\mathcal{I}^{uv}_{k3} = \{r \in \{u+1, \ldots, v-1\} : i_{kr} \ge n+1, i_{kr} = i_{kr'} + n, r' \in \{1, \ldots, u-1\}\}$. The expected cost of x can be computed through the probabilities of direct travel between any pair of nodes.

Given an a priori solution x, the probability $p(i_{ku}, i_{kv})$ that a given arc is traversed by vehicle k is determined as follows:

- $p(i, j) = 0$ if $\{i, j\} \not\subseteq \tau_k$;

- $p(i_{ku}, i_{kv}) = 0$ if $v < u$;
- $p(i_{ku}, i_{kv}) = 0$ if $\exists\ h \in \{u+1, \ldots, v-1\}$ such that $i_{kv} = i_{kh} + n$;
- $p(i_{ku}, i_{kv}) = 0$ if $\exists\ w \in \{u+1, \ldots, v-1\}$ such that $i_{kw} = i_{ku} + n$;
- $p(i_{ku}, i_{kv}) =$

$$\prod_{h \in \mathcal{I}_{k1}^{uv}} p_{i_h} \prod_{r \in \mathcal{I}_{k2}^{uv}} (1 - p_{i_r}) \prod_{w \in \mathcal{I}_{k3}^{uv}} (1 - p_{i_w}) \tag{16}$$

  otherwise.

Thus, the expected cost $EC(x)$ of the a priori solution x is given by

$$EC(x) = \sum_{k \in \mathcal{K}} \sum_{u=0}^{2n} \sum_{v=u+1}^{2n+1} p(i_{ku}, i_{kv}) c_{i_{ku} i_{kv}} \tag{17}$$

with $2n + 1$ being the depot.

Formula (17) is identical to the objective function stated in (1), i.e., $R(x) = EC(x) - \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} x_{ijk}$.

5 Neighborhood evaluation

Let the tours of the current solution be $\{\tau_k\}_{k \in \mathcal{K}}$, where τ_k is defined as in Section 4. A tour is infeasible if one of the constraints (2)-(13) is violated.

5.1 Definition of the neighborhood

In the case of the PPDTSP, a neighbor x' of x can be obtained through a move which eliminates a vertex from a route and reinserts it in some other position in the same route. The evaluation of such a neighborhood is described in Beraldi et al. (2003) where they successfully reduced the amount of computations from $\mathcal{O}(n^5)$ to $\mathcal{O}(n^3)$. In this section we define a similar neighborhood, where we take into account that a vertex can be moved from one tour to another, and that such a move implies that the change of tour also must apply to the companion vertex. We focus on efficient evaluation of the neighborhood in order to limit the computation time.

We say that x' is a neighbor of x if for some $k, \hat{k} \in \mathcal{K}$, x' can be obtained through a move which eliminates some i_{ks} and $i_{kw} = i_{ks} + n$ from τ_k and reinserts them in $\tau_{\hat{k}}$. This definition includes the special case where $k = \hat{k}$, i.e. the solution obtained by reinsertion of (i_{ks}, i_{kw}) in τ_k is considered to be a neighbor. Let $\mathcal{N}(x)$ be the resulting neighborhood.

Clearly, $|\mathcal{N}(x)| = \mathcal{O}(n^3)$. If $EC(x')$ is computed from scratch through (16)-(17), we arrive at $\mathcal{O}(n^3)$ operations ((17) takes $\mathcal{O}(n^2)$, and (16) takes $\mathcal{O}(n)$ for a given pair (i_{ku}, i_{kv})). A complete neighborhood evaluation that takes $\mathcal{O}(n^6)$ time is not desirable, and below we show how the calculations of the expected cost $EC(x')$, for all $x' \in \mathcal{N}(x)$, can be accomplished in $\mathcal{O}(n^4)$ time. The idea is to evaluate the neighbors of x in a specific order, such that any neighbor solution x' is evaluated by calculating the difference in expected cost from its predecessor in that order. To this end, the order must be chosen such that only minor updates are necessary when moving from one neighbor to its successor.

5.2 Evaluation order

Let $k \in \mathcal{K}, s \in \{1, \ldots, n_k - 1\}, w \in \{2, \ldots, n_k\}$, where $i_{kw} = i_{ks} + n$, be given. We now demonstrate how to evaluate all $x' \in \mathcal{N}(x)$ where i_{ks} and i_{kw} are assigned new positions.

First we evaluate all solutions where the position of i_{ks} is fixed whereas i_{kw} is moved forward. Routes encountered this way are (in the given order):

$$\begin{aligned}
\tau_k = (&i_{k0}, \ldots, i_{k,s-1}, i_{ks}, i_{k,s+1}, \ldots, i_{k,w-1}, i_{kw}, i_{k,w+1}, \ldots, i_{k,n_k}, i_{k,n_k+1}); \\
(&i_{k0}, \ldots, i_{k,s-1}, i_{ks}, i_{k,s+1}, \ldots, i_{k,w-1}, i_{k,w+1}, i_{kw}, \ldots, i_{k,n_k}, i_{k,n_k+1}); \\
&\vdots \\
(&i_{k0}, \ldots, i_{k,s-1}, i_{ks}, i_{k,s+1}, \ldots, i_{k,w-1}, i_{k,w+1}, \ldots, i_{k,n_k}, i_{kw}, i_{k,n_k+1}).
\end{aligned}$$

Starting from the last solution above, we next move i_{ks} one position forward, and fix it there while i_{kw} is moved backward. This goes on until i_{kw} succeeds i_{ks}:

$$\begin{aligned}
&(i_{k0}, \ldots, i_{k,s-1}, i_{k,s+1}, i_{ks}, \ldots, i_{k,w-1}, i_{k,w+1}, \ldots, i_{k,n_k}, i_{kw}, i_{k,n_k+1}); \\
&(i_{k0}, \ldots, i_{k,s-1}, i_{k,s+1}, i_{ks}, \ldots, i_{k,w-1}, i_{k,w+1}, \ldots, i_{kw}, i_{k,n_k}, i_{k,n_k+1}); \\
&\vdots \\
&(i_{k0}, \ldots, i_{k,s-1}, i_{k,s+1}, i_{ks}, i_{kw}, \ldots, i_{k,w-1}, i_{k,w+1}, \ldots, i_{k,n_k}, i_{k,n_k+1}).
\end{aligned}$$

Alternating between sequences of forward and backward moves of i_{kw}, and moving i_{ks} one position forward after each such pass, we finally arrive at the solution:

$$(i_{k0}, \ldots, i_{k,s-1}, i_{k,s+1}, \ldots, i_{k,w-1}, i_{k,w+1}, \ldots, i_{k,n_k}, i_{ks}, i_{kw}, i_{k,n_k+1})$$

Note that a single forward move of i_{ks} follows the forward passes of i_{kw}, whereas the backward passes require that first i_{kw} and then i_{ks} are moved one position forward before the next pass starts.

In order to evaluate the remaining solutions, let the vehicles be ordered arbitrarily except that k has to be last, and let $\hat{k}$ denote the first vehicle. The next solution to be considered is obtained by transferring i_{ks} and i_{kw} to the start of the route of vehicle $\hat{k}$. Then the above procedure is repeated for this route, eventually i_{ks} and i_{kw} reach the end of the route, and are transferred to the successor of $\hat{k}$. When all routes thus have been visited, the two nodes are finally reinserted in the start of τ_k.

After this reinsertion, we apply the process to route k again, but now we stop as soon as all solutions not encountered before are evaluated. It is easily seen that if s is odd, the stopping criterion is that i_{ks} and i_{kw} are back in positions s and w, respectively. When s is even, we stop when i_{ks} is in position s and i_{kw} is in position $s + 1$. It should be noted that in the latter case $n_k - w + 1$ solutions are evaluated twice. This extra effort could by simple means have been avoided, but doing so would reduce the running time only marginally.

The above demonstrates that a move from x to any $x' \in \mathcal{N}(x)$ can be accomplished by a sequence of *elementary moves* of either of the following types:

1. Move a pick-up node one position forward in its current route
2. Move a drop-off node one position forward in its current route
3. Remove the two last nodes from a route, where the last node is the drop-off companion of the second last, and reinsert them as the leading nodes of some route.

5.3 Probability updates

Algorithms 1-3 perform all necessary updates of the probability matrix when one elementary move of types 1-3, respectively, is applied to the current solution. Since all updates follow the same pattern, we explain only the updates needed in the case where a pick-up node is moved one position forward, and its current successor is a drop-off node. Details for all cases are given in the algorithms.

Consider the route $(i_{k0}, i_{k1}, \ldots, i_{kr}, \ldots, i_{ks}, i_{k,s+1}, \ldots, i_{kw}, \ldots, i_{kn_k}, i_{k,n_k+1})$ where $i_{kw} = i_{ks} + n$ and $i_{k,s+1} = i_{kr} + n$. Write $u = s + 1$. It is seen from (16) that by moving i_{ks} one position forward, the factor $1 - p_{i_{ku}}$ is introduced in the probabilities of arcs (i_{kh}, i_{ks}) for all $h = r + 1, \ldots, s - 1$. The probability of traveling directly from i_{ks} to i_{kh}, where $h = s + 2, \ldots, w - 1$, becomes $p(i_{ku}, i_{kh})p_{i_{ks}}/p_{i_{ku}}$. This is because the path from i_{ks} to i_{kh} after the move coincide with the path from i_{ku} to i_{kh} before the move, except that i_{ks} is visited in place of i_{ku}. Similar observations can be made for all other affected arcs, and we refer to Algorithm 1 for details.

Algorithm 1 ProbUpdatePick(k, s)

Require: $1 \leq s \leq n_k - 2$ and $1 \leq i_{ks} \leq n$.

Ensure: The current probability matrix p is overwritten with new probabilities induced by a forward move of i_{ks}.

Find $w \in \{s + 2, \ldots, n_k\}$ such that $i_{kw} = i_{ks} + n$.
if $i_{k,s+1} \geq n + 1$ **then**
  $u = s + 1$
  Find $r \in \{1, \ldots, s - 1\}$ such that $i_{ku} = i_{kr} + n$
  for all $h = r + 1, \ldots, s - 1$ **do**
    $p(i_{kh}, i_{ku}) = p(i_{kh}, i_{ks})p_{i_{ku}}/p_{i_{ks}}$
    $p(i_{kh}, i_{ks}) = p(i_{kh}, i_{ks})(1 - p_{i_{ku}})$
  for all $h = s + 2, \ldots, w - 1$ **do**
    $p(i_{ks}, i_{kh}) = p(i_{ku}, i_{kh})p_{i_{ks}}/p_{i_{ku}}$
    $p(i_{ku}, i_{kh}) = p(i_{ku}, i_{kh})(1 - p_{i_{ks}})$
  $p(i_{kr}, i_{ku}) = p(i_{kr}, i_{ks})/p_{i_{ks}}$
  $p(i_{kr}, i_{ks}) = 0$
  $p(i_{ks}, i_{kw}) = p(i_{ku}, i_{kw})/p_{i_{ku}}$
else
  $r = s + 1$
  Find $u \in \{s + 2, \ldots, n_k\}$ such that $i_{ku} = i_{kr} + n$
  for all $h = 0, \ldots, s - 1$ **do**
    $p(i_{kh}, i_{kr}) = p(i_{kh}, i_{ks})p_{i_{kr}}/p_{i_{ks}}$
    $p(i_{kh}, i_{ks}) = p(i_{kh}, i_{ks})(1 - p_{i_{kr}})$
  for all $h = s + 2, \ldots, \min\{u, w\} - 1$ **do**
    $p(i_{ks}, i_{kh}) = p(i_{kr}, i_{kh})p_{i_{ks}}/p_{i_{kr}}$
    $p(i_{kr}, i_{kh}) = p(i_{kr}, i_{kh})(1 - p_{i_{ks}})$
  if $u < w$ **then**
    $p(i_{ks}, i_{ku}) = p(i_{kr}, i_{ku})p_{i_{ks}}$
    $p(i_{kr}, i_{ku}) = p(i_{kr}, i_{ku})(1 - p_{i_{ks}})$
  else
    $p(i_{ks}, i_{kw}) = p(i_{kr}, i_{kw})/p_{i_{kr}}$
$p(i_{k,s+1}, i_{ks}) = p(i_{ks}, i_{k,s+1})$
$p(i_{ks}, i_{k,s+1}) = 0$
$p(i_{k,s+1}, i_{kw}) = 0$

Algorithm 2 ProbUpdateDrop(k, w)

Require: $2 \le w \le n_k - 1$ and $n + 1 \le i_{kw} \le 2n$.

Ensure: The current probability matrix p is overwritten with new probabilities induced by a forward move of i_{kw}.

Find $s \in \{1, \ldots, w-1\}$ such that $i_{kw} = i_{ks} + n$.
if $i_{k,w+1} \ge n + 1$ **then**
  $u = w + 1$
  Find $r \in \{1, \ldots, w-1\}$ such that $i_{ku} = i_{kr} + n$
  for all $h = \max\{s, r\} + 1, \ldots, w - 1$ **do**
    $p(i_{kh}, i_{ku}) = p(i_{kh}, i_{kw}) p_{i_{ku}} / p_{i_{kw}}$
    $p(i_{kh}, i_{kw}) = p(i_{kh}, i_{kw})(1 - p_{i_{ku}})$
  for all $h = w + 2, \ldots, n_k + 1$ **do**
    $p(i_{kw}, i_{kh}) = p(i_{ku}, i_{kh}) p_{i_{kw}} / p_{i_{ku}}$
    $p(i_{ku}, i_{kh}) = p(i_{ku}, i_{kh})(1 - p_{i_{kw}})$
  if $r < s$ **then**
    $p(i_{ks}, i_{ku}) = p(i_{ks}, i_{kw}) p_{i_{ku}}$
    $p(i_{ks}, i_{kw}) = p(i_{ks}, i_{kw})(1 - p_{i_{ku}})$
  else
    $p(i_{kr}, i_{ku}) = p(i_{kr}, i_{kw}) / p_{i_{kw}}$
    $p(i_{kr}, i_{kw}) = 0$
else
  $r = w + 1$
  Find $u \in \{r + 1, \ldots, n_k\}$ such that $i_{ku} = i_{kr} + n$
  for all $h = s + 1, \ldots, w - 1$ **do**
    $p(i_{kh}, i_{kr}) = p(i_{kh}, i_{kw}) p_{i_{kr}} / p_{i_{kw}}$
    $p(i_{kh}, i_{kw}) = p(i_{kh}, i_{kw})(1 - p_{i_{kr}})$
  for all $h = w + 2, \ldots, u - 1$ **do**
    $p(i_{kw}, i_{kh}) = p(i_{kr}, i_{kh}) p_{i_{kw}} / p_{i_{kr}}$
    $p(i_{kr}, i_{kh}) = p(i_{kr}, i_{kh})(1 - p_{i_{kw}})$
  $p(i_{ks}, i_{kr}) = p(i_{ks}, i_{kw}) p_{i_{kr}}$
  $p(i_{kw}, i_{ku}) = p(i_{kr}, i_{ku}) p_{i_{kw}}$
  $p(i_{kr}, i_{ku}) = p(i_{kr}, i_{ku})(1 - p_{i_{kw}})$
  $p(i_{ks}, i_{kw}) = p(i_{ks}, i_{kw})(1 - p_{i_{kr}})$
$p(i_{k,w+1}, i_{kw}) = p(i_{kw}, i_{k,w+1})$
$p(i_{kw}, i_{k,w+1}) = 0$

Algorithm 3 ProbUpdateTour$(k, \hat{k})$

Require: $n_k \ge 2$, $1 \le i_{k,n_k-1} \le n$ and $i_{kn_k} = i_{k,n_k-1} + n$.

Ensure: The current probability matrix p is overwritten with new probabilities induced by a removing the two last nodes of tour k and inserting them at the start of tour $\hat{k}$

$i = i_{k,n_k-1}$, $j = i_{kn_k}$
for all $h = 1, \ldots, n_k - 2$ for which $i_{kh} > n$ **do**
  $p(i_{kh}, 2n + 1) = p(i_{kh}, i) / p_i$
  $p(i_{kh}, i) = 0$
$p(0, i) = p_i$
$p(j, 2n + 1) = p(i_{\hat{k}\mu}, 2n + 1) p_i (1 - p_{i_{\hat{k}\mu} - n}) / p_{i_{\hat{k}\mu} - n}$
`/*` $i_{\hat{k}\mu}$ `denotes the first drop-off location in route` $\hat{k}$ `*/`
for all $h = 1, \ldots, n_{\hat{k}}$ for which $i_{\hat{k}h} \le n$ **do**
  $p(j, i_{\hat{k}h}) = p(0, i_{\hat{k}h}) p_j$
  $p(0, i_{\hat{k}h}) = p(0, i_{\hat{k}h})(1 - p_i)$

5.4 Running time analysis

Proposition 1. *The described neighborhood search procedure takes $\mathcal{O}(n^4)$ steps.*

Proof. Since each of the $\mathcal{O}(n)$ requests can be placed into $\mathcal{O}(n^2)$ different combinations of positions, there are $\mathcal{O}(n^3)$ solutions to be evaluated. The suggested sequence of elementary moves ensures that one new solution is visited in every move. In each move, either of Algorithms 1-3 computes new probabilities in $\mathcal{O}(n)$ time, and hence the total running time is $\mathcal{O}(n^4)$. □

6 Tabu search

Our heuristic is based on tabu search, a metaheuristic that guides the local search process beyond a local optimum. It was originally proposed by Glover (1986), and has been applied to numerous applications of continuous and combinatorial optimization problems with much success (see Glover (1997) and Glover and Laguna (1997) for surveys on tabu search applications). Tabu search starts from an initial solution x_0 and iteratively moves to a new solution found from the neighborhood of the current solution x. Since the search may be trapped in a local minimum, or it may visit earlier visited solutions, anti-cycling rules must be implemented. A basic mechanism is the *short term memory* or the *recency based memory*, and its purpose is to keep track of recent moves or solutions made in the past. The tabu list is the tool of achieving this by recording recently made moves or visited solutions. Whenever the algorithm attempts to move to a solution forbidden by the tabu list, the move is banned. This rule forces other solutions to be explored. However, this feature is not strict, as it can be overridden when some aspiration criterion is satisfied. A common criterion is that the objective function value be the best ever seen. If this is the case, it is obvious that this solution has never been encountered before.

6.1 Notation of the heuristic

A feature of our algorithm is the possibility to explore the infeasible part of the solution space. Gendreau et al. (1994) introduced this by penalizing infeasible solutions, and later several others have embedded this into their work (Cordeau et al. 1997, Cordeau et al. 2001, Cordeau and Laporte 2003*b*). An advantage of allowing infeasible solutions is that a feasible solution can be obtained from another feasible solution through a number of intermediate infeasible solutions, which may never be encountered if only feasible solutions were allowed. By allowing the search to be shifted back and forth between feasible and infeasible parts of the solution space, it will be guided into unexplored or less explored regions of the solution space. This strategy is called *strategic oscillation*. We let $\mathcal{X}$ denote the set of solutions satisfying constraints (i) and (ii). A solution $x \in \mathcal{X}$ consists of m vehicle routes starting and ending at the depot, and for every request, both the origin and destination vertices belong to the same route and the destination vertex is visited after its origin. All the other constraints may be violated.

Let $t(x) = \sum_{i=0}^{2n} \sum_{k=1}^{m} \max\{0, B_{ik} - l_i\}$, $q(x) = \sum_{k=1}^{m} \sum_{i=1}^{n_k} \max\{0, Q_{ik} - Q_k\}$ and $r(x) = \sum_{i=1}^{n} \sum_{k=1}^{m} \max\{0, L_{ik} - L\}$ denote total violation of time windows, load and ride time constraints, respectively. Each solution x is evaluated by a cost function $z(x) = EC(x) + \alpha t(x) + \beta q(x) + \gamma r(x)$, where α, β and γ are dynamically-adjusted parameters.

6.2 Initial solution

An initial solution is found by constructing routes in a greedy manner. In the current route, some i and $i+n$, which so far have not been assigned to a route, are inserted in positions such that the expected increase in travel cost is minimized. The selection of the pair $(i, i+n)$ is also made with respect to this criterion. This goes on until there are no more unrouted requests or the current route covers $\lceil n/m \rceil$ requests. In the case of the latter stopping criterion, which is added in order to have a balanced assignment of requests to routes in the initial solution, the above is repeated for some other, currently empty route. During the construction process, constraints (i) and (ii) are ensured to be satisfied, whereas, all other constraints may be violated.

6.3 Tabu moves

Given the neighborhood defined in Section 5.1, any move is identified by a tuple $(k, i, \hat{k}, s, w)$. The move in question is to remove the pickup-node i, along with its corresponding drop-off node $i+n$ from route k, and reinsert them in positions s and w in route $\hat{k}$. If $\hat{k} \neq k$, all moves where i and $i+n$ are moved back to route k, regardless of the position, are declared tabu for θ iterations, where θ is a user defined parameter. A forbidden move will still be chosen if it is the best solution encountered so far. If $\hat{k} = k$, all moves involving moving i and $i+n$ within route k, regardless of the position, are declared tabu for θ iterations.

6.4 Frequency based memory

Since the tabu status of a move expires after θ iterations, we add a memory of longer duration to the heuristic. The goal is to explore regions of the solution space that have not been visited yet, and by penalizing frequently made moves this can be achieved. The search will be directed to less explored regions. For any solution $\overline{x} \in \mathcal{N}(x)$, whenever $z(\overline{x}) \geq z(x)$, a penalty is added to $z(\overline{x})$. We only penalize non-improving moves, since improving ones have to be encouraged. The penalty $\phi(\overline{x})$ is defined to be $\lambda EC(\overline{x})\sqrt{nm}\vartheta_{ik}$. The factor ϑ_{ik} denotes the frequency of the pair (i, k), i.e. the number of times request i has been moved to route k from some other route, and λ is a user defined parameter which controls the intensity of diversification. The suggested formula can be intuitively explained as follows: The penalty should be proportional to the expected total solution cost, to the size of the problem and to the frequency of a move. The number of moves is dependent on the size of the problem, and using the factor $\sqrt{nm}$ seems to somehow compensate for the problem size. The use of such a factor was first suggested by Taillard (1993) in the context of the vehicle routing problem.

6.5 Search process

The tabu search algorithm starts off with the initial solution described in Section 6.2. In each iteration it selects a solution $\bar{x} \in \mathcal{N}(x)$ that minimizes $z(\overline{x}) + \phi(\overline{x})$ and is non-tabu or satisfies the aspiration criterion. The penalty factor α, β and γ are initially set equal to 1 and are adjusted as follows: if there are no violation of the time windows the value of α is divided by a factor of $1+\delta$, otherwise it is multiplied by this factor. Similar rules also apply to β and γ for violations of load and ride time constraints. This process terminates after η iterations and the best solution found during the search is the final solution.

7 GRASP

In this section, we describe our second heuristic which is a hybrid GRASP-tabu search heuristic. GRASP (Greedy randomized adaptive search procedures, Feo and Resende 1995) is a multi-start process, where each iteration consists of two phases: construction and local search. The construction phase builds an initial solution, and is improved by local search. The best solution from all the GRASP iterations is returned as the best overall solution. The GRASP procedure for minimizing a function f is given in Algorithm 4. The reader is referred to Pitsoulis and Resende (2002) for a tutorial and survey on GRASP applications.

Algorithm 4 GRASP

$f(x^*) = \infty$
for $i = 1$ to η **do**
  $x =$`GreedyRandomized`(ω)
  $x =$`LocalSearch`(x)
  if $f(x) < f(x^*)$ **then**
    $x^* = x$
    $f(x^*) = f(x)$
return x^*

7.1 Construction phase

Algorithm 5 shows the steps of constructing a greedy randomized solution. The input to this phase is the parameter ω which controls the amount of greediness and randomness in this phase. We assign requests to one vehicle at a time, and we start by randomly selecting a request i and insert its associated vertices i and $i+n$ sequentially in the vehicle. This solution defines the current value of the binary vector x. The partial neighborhood of x, $\mathcal{N}_k(x)$, is defined as all solutions that can be reached from x by inserting an unrouted request in route k. Every solution $\hat{x} \in \mathcal{N}_k(x)$ satisfies constraints (i) and (ii), whereas all other constraints may be violated. The probabilistic component of this phase is characterized by randomly choosing one of the best neighbor solutions $\hat{x}$, but not necessarily the best one. Let $\underline{x}$ be the best solution found in $\mathcal{N}_k(x)$ and let $\overline{x}$ be the worst solution found in $\mathcal{N}_k(x)$. The list of the best neighbor solutions are stored in the $\mathcal{RCL}$ (restricted candidate list), and these solutions are no more than $\omega(z(\overline{x}) - z(\underline{x}))$ away from the best one. In order to balance the assignment of the requests, each vehicle may have a maximum of $\lceil n/m \rceil$ requests assigned to it.

7.2 Local search

The local search phase is done by tabu search. The description of this is given in Algorithm 6. This is very much like the tabu search heuristic described in Section 6, except that frequency based memory is not incorporated in the algorithm. Since this search is to be integrated with a multi-start process, it is normally run in a relatively small number of iterations, and hence the benefit of a frequency based memory is expected to be modest.

7.3 Reactive GRASP

The parameter ω has an effect on the solution quality and diversity during the construction phase, and also on the impact of this phase on the final solution of the heuristic. As it is rather time consuming to calibrate this parameter, Prais

Algorithm 5 GreedyRandomized

Require: ω

Set $j = 0$ and $x^* = 0$.

for $k = 1, \ldots, m$ **do**

Randomly select a request to initiate vehicle k, let x be the corresponding binary vector (0 for all other routes).

Set $i = 1$ and $j = j + 1$.

while $i < \lceil n/m \rceil$ and $j < n$ **do**

Initialize $\mathcal{N}_k(x)$.

$\underline{x} = \arg\min_{t \in \mathcal{N}_k(x)} \{z(t)\}$

$\overline{x} = \arg\max_{t \in \mathcal{N}_k(x)} \{z(t)\}$

$\mathcal{RCL} = \{t \in \mathcal{N}_k(x) : z(t) \leq z(\underline{x}) + \omega(z(\overline{x}) - z(\underline{x}))\}$

Select randomly solution $\hat{x} \in \mathcal{RCL}$.

Set $x = \hat{x}$, i=i+1 and j=j+1.

$x^* = x^* + x$

return x^*

Algorithm 6 LocalSearch

Require: x

If x is feasible, set $x^* = x$ and $EC(x^*) = EC(x)$; otherwise set $EC(x^*) = \infty$.

Set $\alpha = 1$, $\beta = 1$ and $\gamma = 1$.

for $i = 1, \ldots, \pi$ **do**

Select a solution $\overline{x} \in \mathcal{N}(x)$ that minimizes $z(\overline{x})$ and is non-tabu or satisfies the aspiration criterion.

If $\overline{x}$ is feasible and $EC(\overline{x}) < EC(x^*)$, set $x^* = \overline{x}$ and $EC(x^*) = EC(\overline{x})$.

Set the reverse move tabu for θ iterations.

Compute $t(\overline{x})$, $q(\overline{x})$ and $r(\overline{x})$, and update α, β and γ.

Set $x = \overline{x}$.

return x^*

and Ribeiro (2000) introduced the Reactive GRASP for which ω is self-adjusted according to the quality of previously found solutions. This will give the heuristic an appropriate level of greediness and randomness. Algorithm 7 shows how this is done.

Algorithm 7 Reactive GRASP

$EC(x^*) = \infty$
$\rho_i = 1/r$, $\zeta_i = 0$, $\sigma_i = 0$, $i = 1, \ldots, r$
for $k = 1, \ldots, \eta$ **do**
  Randomly select $\omega = \omega_i$ from $\mathcal{W}$ using probabilities $\rho_1, \ldots, \rho_r$.
  $\sigma_i = \sigma_i + 1$
  $x =$`GreedyRandomized`(ω)
  $x =$`LocalSearch`(x)
  if $EC(x) < EC(x^*)$ **then**
    $x^* = x$
    $EC(x^*) = EC(x)$
  $\zeta_i = \zeta_i + EC(x)$
  if $\mod(k, \epsilon) == 0$ **then**
    $W_i = \zeta_i/\sigma_i$, $i = 1, \ldots, r$
    $\varrho_i = EC(x^*)/W_i$, $i = 1, \ldots, r$
    $\rho_i = \varrho_i / \sum_{j=1}^{r} \varrho_j$, $i = 1, \ldots, r$
return x^*

At each GRASP iteration ω is chosen randomly from a discrete set $\mathcal{W} = \{\omega_1, \ldots, \omega_r\}$. The probability that ω_i is chosen is ρ_i. Reactive GRASP changes the probabilities $\{\rho_1, \ldots, \rho_r\}$ every ϵ iterations to favor the values of ω that produce good solutions. Let $EC(x^*)$ be the value of the best solution found so far and let W_i be the average value of the solutions found using ω_i in the construction phase. At every ϵ iterations the values of ϱ_i (see Algorithm 7) and ρ_i, $i = 1, \ldots, r$, are updated to reflect the changes made in previous iterations.

8 Computational experiments

To our knowledge, no benchmark instances for the PDARP exist, and therefore we extended test instances provided by Cordeau and Laporte (2003*b*). The test instances contain between 24 and 144 requests, and between 3 and 13 vehicles. The vertices are clustered around a certain number of seed points in the area $[-10, 10]^2$. For each vertex, the service time d_i is equal to 10 and the load q_i is equal to 1 for all $i \in \mathcal{P}$. A part of the instances (group a) has narrow time windows and the other part (group b) has wide time windows. The vehicle capacity is set to 6, and the maximum ride time L is set to 90. All these instances are available from the Internet at `http://www.hec.ca/chairedistributique/data/darp/`.

Four groups of modified instances were created. In each group the percentage of black vertices is fixed to 0%, 25%, 50% and 75%, respectively, and the black vertices were chosen randomly (all vertices having equal probability to be chosen). Since the original test instances do not contain request probabilities, the white vertices are assigned probabilities drawn randomly from the uniform distribution on the interval $(0, 1)$.

The results from the tabu search (TS) heuristic were obtained with the following parameter settings: $\eta = 1000$, $\theta = [7.5 \log_{10} n]$ (where $[y]$ is the nearest integer to y), $\delta = 0.5$ and $\lambda = 0.015$, and for the reactive GRASP-tabu search (GRASP-TS) heuristic: $\eta = 50$, $\theta = [7.5 \log_{10} n]$, $\delta = 0.5$, $\epsilon = 20$, $\pi = 100$ and $\mathcal{W} = \{0.1, 0.2, \ldots, 1\}$. Parameters θ, δ and λ were obtained from Cordeau et al. (2001)

for a similar tabu search heuristic for the vehicle routing problem with time windows. The other parameters have not been specifically tuned for the experimentation. The heuristics were coded in C++ and all experiments were carried out on a Pentium 4, 2.53 GHz computer.

Tables 1-4 show the expected costs of the best solutions from the two heuristics for each of the twenty instances of groups 1-4, respectively. The best results of any of the two heuristics are reported in the rightmost column labeled "best", while the percentage deviation from the best is given within the parentheses of each heuristic. Since no optimal solutions are known for any of the problem instances, using the best solutions obtained by the two presented heuristics as benchmarks seems reasonable. Under the current parameter settings for TS, setting $\eta = 10$ for GRASP-TS, the two methods can be fairly compared as the computing times are approximately the same and the time needed to construct the initial solutions is insignificant. The tables show in general that TS performs very well compared to GRASP-TS for all the four groups of instances, with average deviations of less than 2% and with group 2 instances being the best with an average deviation of 0.55%. On the other hand, GRASP-TS was not able to find a feasible solution for an instance of group 2 (see Table 2), and the average deviations are between 5.36% and 8.35%. However, there are a few instances in which better results were obtained by GRASP-TS for each group of instances. We want to see how GRASP-TS performs when the number of starting points are increased, and the results of the evolution of η varying from 10 to 50 are presented in Tables 1-4. As η increased, GRASP-TS managed in average to produce better solutions (in the best cases, the gaps were reduced to be between 2.87% and 6.30%). Still, TS remains the superior one. When η reached 50 for GRASP-TS, the computing time is roughly five times as much as for TS, and although the gaps between the two heuristics are reduced, GRASP-TS is still far away from the best known results (especially for the group 2 instances).

One way of explaining the superiority of TS may be the use of the diversification strategy, frequency based memory, which implies that different parts of the solution space are explored. The concept of diversification is also utilized by GRASP-TS, since the heuristic constructs different initial solutions at each iteration. The main difference between these two methods is that TS remembers every move it has made since the first iteration, while GRASP-TS is more or less memoryless. The basic version of GRASP is totally memoryless, but in our implementation of GRASP-TS we employed the idea of reactive GRASP in which parameter ω that lead to good solutions will be favored when constructing the initial solutions. Diversification is achieved by both methods, but realized through different concepts. An observation that can be drawn from this experiment is that in order to overcome local optima and to approach near-optimal solutions, it seems to be more useful to remember its entire search history than just restarting the search without any knowledge of its past.

The computing times for the neighborhood search procedure is documented in Table 5. The second column in Table 5 refers to the number of requests, while the third column refers to the number of vehicles of a particular problem instance. The computing time for TS-O is in seconds for TS assessing neighbor solutions using formula (16), while the computing time for TS-N is in seconds for TS using the presented technique described in Section 5. As these computational times represent the times needed for searching the neighborhood in one iteration, they are also valid for the hybrid GRASP-TS since the local search procedure used is tabu search. As presented in Table 5, using our presented technique for updating the probabilities results in great savings in computing time.

Table 1: Comparison of expected routing cost for group 1 instances

Problem	TS		GRASP-TS $\eta = 10$		GRASP-TS $\eta = 20$		GRASP-TS $\eta = 30$		GRASP-TS $\eta = 40$		GRASP-TS $\eta = 50$		Best
R1a	57.89	(0.00)	60.88	(5.16)	58.40	(0.88)	58.40	(0.88)	58.40	(0.88)	58.40	(0.88)	57.89
R2a	89.57	(0.00)	94.00	(4.95)	91.01	(1.61)	91.01	(1.61)	91.01	(1.61)	91.01	(1.61)	89.57
R3a	161.94	(15.11)	149.40	(6.20)	149.40	(6.20)	149.40	(6.20)	148.24	(5.37)	140.68	(0.00)	140.68
R4a	167.63	(0.00)	177.24	(5.73)	177.24	(5.73)	174.08	(3.85)	174.08	(3.5)	174.08	(3.85)	167.63
R5a	186.79	(0.00)	202.03	(8.16)	202.03	(8.16)	202.03	(8.16)	202.03	(8.16)	202.03	(8.16)	186.79
R6a	232.37	(0.00)	248.19	(6.81)	242.18	(4.22)	242.18	(4.22)	242.18	(4.22)	242.18	(4.22)	232.37
R7a	77.60	(0.00)	80.66	(3.94)	80.66	(3.94)	80.66	(3.94)	80.64	(3.92)	80.23	(3.39)	77.60
R8a	163.75	(17.81)	142.30	(2.38)	142.30	(2.38)	142.30	(2.38)	138.99	(0.00)	138.99	(0.00)	138.99
R9a	219.81	(0.00)	292.01	(32.85)	292.01	(32.85)	292.01	(32.85)	292.01	(32.85)	260.67	(18.59)	219.81
R10a	275.06	0.00)	303.71	(10.42)	296.81	(7.91)	289.18	(5.13)	289.18	(5.13)	289.18	(5.13)	275.06
R1b	57.23	(1.72)	56.26	(0.00)	56.26	(0.00)	56.26	(0.00)	56.26	(0.00)	56.26	(0.00)	56.26
R2b	54.74	(2.07)	55.44	(3.37)	55.44	(3.37)	55.44	(3.37)	55.44	(3.37)	53.63	(0.00)	53.63
R3b	99.65	(0.00)	107.69	(8.07)	103.96	(4.33)	103.98	(4.35)	103.96	(4.33)	103.96	(4.33)	99.65
R4b	98.43	(0.00)	110.89	(12.66)	108.43	(10.16)	108.43	(10.16)	108.16	(9.89)	106.16	(7.85)	98.43
R5b	109.77	(0.00)	120.42	(9.70)	120.42	(9.70)	120.42	(9.70)	120.01	(9.33)	120.01	(9.33)	109.77
R6b	153.25	(0.00)	165.54	(8.02)	158.76	(3.60)	158.76	(3.60)	158.76	(3.60)	158.41	(3.37)	153.25
R7b	41.93	(0.00)	47.22	(12.62)	43.85	(4.58)	43.85	(4.58)	43.85	(4.58)	43.85	(4.58)	41.93
R8b	85.79	(0.00)	87.77	(2.31)	87.56	(2.06)	86.42	(0.73)	86.23	(0.51)	86.23	(0.51)	85.79
R9b	121.55	(0.00)	124.05	(2.06)	124.05	(2.06)	124.05	(2.06)	122.00	(0.37)	122.00	(0.37)	121.55
R10b	218.80	(0.00)	266.09	(21.61)	264.64	(20.95)	252.41	(15.36)	252.41	(15.36)	252.41	(15.36)	218.80
Avg.	133.68	(1.84)	144.59	(8.35)	142.77	(6.73)	141.56	(6.16)	141.19	(5.87)	139.02	(4.58)	131.27

Table 2: Comparison of expected routing cost for group 2 instances

Problem	TS		GRASP-TS $\eta = 10$		GRASP-TS $\eta = 20$		GRASP-TS $\eta = 30$		GRASP-TS $\eta = 40$		GRASP-TS $\eta = 50$		Best
R1a	83.31	(0.16)	84.28	(1.32)	83.18	(0.00)	83.18	(0.00)	83.18	(0.00)	83.18	(0.00)	83.18
R2a	150.20	(0.00)	155.53	(3.55)	155.53	(3.55)	155.53	(3.55)	155.53	(3.55)	154.04	(2.56)	150.20
R3a	226.28	(0.00)	245.54	(8.51)	245.54	(8.51)	228.61	(1.03)	228.61	(1.03)	228.61	(1.03)	226.28
R4a	258.70	(2.21)	258.20	(2.02)	258.20	(2.02)	258.20	(2.02)	254.00	(0.36)	253.10	(0.00)	253.10
R5a	258.07	(0.00)	285.49	(10.63)	280.27	(8.60)	280.27	(8.60)	280.27	(8.60)	280.27	(8.60)	258.07
R6a	309.20	(0.00)	357.33	(15.57)	357.33	(15.57)	354.58	(14.68)	354.58	(14.68)	354.58	(14.68)	309.20
R7a	105.85	(0.67)	112.88	(7.35)	112.05	(6.56)	109.00	(3.66)	109.00	(3.66)	105.15	(0.00)	10.15
R8a	226.81	(0.00)	240.71	(6.13)	240.71	(6.13)	234.31	(3.31)	234.31	(3.31)	234.31	(3.31)	226.81
R9a	338.77	(0.00)	*		462.89	(36.64)	462.89	(36.64)	432.93	(27.79)	430.44	(27.06)	338.77
R10a	389.28	(0.00)	456.61	(17.3)	441.98	(13.54)	441.98	(13.54)	441.98	(13.54)	441.98	(13.54)	389.28
R1b	82.33	(4.23)	78.99	(0.00)	78.99	(0.00)	78.99	(0.00)	78.99	(0.00)	78.99	(0.00)	78.99
R2b	86.53	(0.00)	91.06	(5.24)	91.06	(5.24)	87.65	(1.29)	87.65	(1.29)	87.65	(1.29)	86.53
R3b	190.16	(0.00)	203.21	(6.86)	201.83	(6.14)	200.07	(5.21)	200.07	(5.21)	200.07	(5.21)	190.16
R4b	186.76	(0.00)	202.90	(8.64)	200.73	(7.48)	200.73	(7.48)	199.56	(6.85)	199.56	(6.85)	186.76
R5b	200.83	(0.00)	210.24	(4.69)	210.24	(4.69)	210.24	(4.69)	210.24	(4.69)	209.92	(4.53)	200.83
R6b	237.28	(0.00)	266.13	(12.16)	266.13	(12.16)	266.13	(12.16)	251.25	(5.89)	251.25	(5.89)	237.28
R7b	83.82	(3.67)	83.97	(3.86)	80.85	(0.00)	80.85	(0.00)	80.85	(0.00)	80.85	(0.00)	80.85
R8b	164.63	(0.00)	175.36	(6.52)	173.79	(5.56)	172.70	(4.90)	171.42	(4.12)	167.14	(1.52)	164.63
R9b	210.31	(0.00)	230.84	(9.76)	230.84	(9.76)	230.84	(9.76)	230.84	(9.76)	230.84	(9.76)	210.31
R10b	318.05	(0.00)	405.90	(27.62)	398.18	(25.19)	398.17	(25.19)	398.18	(25.19)	382.26	(20.19)	318.05
Avg.	205.36	(0.55)	218.17	(8.30)	228.52	(8.87)	226.75	(7.89)	224.17	(6.98)	222.71	(6.30)	204.72

*no feasible solution was found

Table 3: Comparison of expected routing cost for group 3 instances

Problem	TS		GRASP-TS		GRASP-TS		GRASP-TS		GRASP-TS		GRASP-TS		Best
			$\eta = 10$		$\eta = 20$		$\eta = 30$		$\eta = 40$		$\eta = 50$		
R1a	106.40	(2.50)	111.53	(7.45)	109.00	(5.01)	103.80	(0.00)	103.80	(0.00)	103.80	(0.00)	103.80
R2a	219.41	(2.60)	216.73	(1.34)	216.73	(1.34)	213.86	(0.00)	213.86	(0.00)	213.86	(0.00)	213.86
R3a	335.77	(0.00)	357.67	(6.52)	357.67	(6.52)	357.67	(6.52)	357.67	(6.52)	357.67	(6.52)	335.77
R4a	343.00	(0.00)	366.02	(6.71)	366.02	(6.71)	359.69	(4.87)	359.69	(4.87)	359.69	(4.87)	343.00
R5a	400.56	(0.00)	435.20	(8.65)	435.20	(8.65)	435.20	(8.65)	428.49	(6.97)	427.92	(6.83)	400.56
R6a	487.40	(0.00)	545.22	(11.86)	545.22	(11.86)	528.19	(8.37)	528.19	(8.37)	528.19	(8.37)	487.40
R7a	144.97	(0.00)	146.08	(0.77)	146.08	(0.77)	146.08	(0.77)	146.08	(0.77)	146.08	(0.77)	144.97
R8a	335.90	(0.00)	348.43	(3.73)	347.84	(3.55)	347.84	(3.55)	339.58	(1.10)	339.58	(1.10)	335.90
R9a	457.07	(0.00)	573.86	(25.55)	573.86	(25.55)	573.86	(25.55)	573.86	(25.55)	573.86	(25.55)	457.07
R10a	557.15	(0.00)	610.14	(9.51)	610.14	(9.51)	610.14	(9.51)	610.14	(9.51)	597.02	(7.16)	557.15
R1b	114.22	(2.63)	114.44	(2.83)	112.54	(1.12)	111.61	(0.29)	111.61	(0.29)	111.29	(0.00)	111.29
R2b	155.58	(3.14)	163.82	(8.61)	156.83	(3.97)	150.84	(0.00)	150.84	(0.00)	150.84	(0.00)	150.84
R3b	336.44	(0.00)	348.84	(3.69)	348.84	(3.69)	344.67	(2.45)	338.68	(0.67)	338.68	(0.67)	336.44
R4b	310.81	(1.94)	327.71	(7.48)	304.90	(0.00)	304.90	(0.00)	304.90	(0.00)	304.90	(0.00)	304.90
R5b	273.70	(0.00)	307.73	(12.43)	307.73	(12.43)	307.73	(12.43)	299.74	(9.51)	299.74	(9.51)	273.70
R6b	396.98	(0.00)	416.12	(4.82)	416.12	(4.82)	416.12	(4.82)	416.12	(4.82)	416.12	(4.82)	396.98
R7b	136.34	(7.35)	134.16	(5.63)	127.01	(0.00)	127.01	(0.00)	127.01	(0.00)	127.01	(0.00)	127.01
R8b	300.58	(0.00)	321.13	(6.84)	321.13	(6.84)	317.72	(5.70)	317.72	(5.70)	317.72	(5.70)	300.58
R9b	299.87	(0.00)	340.46	(13.54)	340.20	(13.45)	335.87	(12.01)	335.87	(12.01)	335.87	(12.01)	299.87
R10b	513.25	(0.00)	518.28	(0.98)	518.28	(0.98)	518.28	(0.98)	518.28	(0.98)	518.28	(0.98)	513.25
Avg.	311.27	(1.01)	335.18	(7.45)	333.07	(6.34)	330.55	(5.32)	329.11	(4.88)	328.41	(4.74)	309.72

Table 4: Comparison of expected routing cost for group 4 instances

Problem	TS		GRASP-TS $\eta = 10$		GRASP-TS $\eta = 20$		GRASP-TS $\eta = 30$		GRASP-TS $\eta = 40$		GRASP-TS $\eta = 50$		Best
R1a	151.77	(1.79)	154.07	(3.33)	149.10	(0.00)	149.10	(0.00)	149.10	(0.00)	149.10	(0.00)	149.10
R2a	254.60	(0.00)	257.67	(1.21)	257.67	(1.21)	257.67	(1.21)	257.67	(1.21)	257.67	(1.21)	254.60
R3a	513.83	(6.07)	484.42	(0.00)	484.42	(0.00)	484.42	(0.00)	484.42	(0.00)	484.42	(0.00)	484.42
R4a	502.95	(0.00)	531.83	(5.74)	521.79	(3.75)	519.84	(3.36)	514.22	(2.24)	513.16	(2.03)	502.95
R5a	581.50	(0.00)	611.68	(5.19)	608.44	(4.63)	602.07	(3.54)	595.21	(2.36)	595.21	(2.36)	581.50
R6a	687.92	(0.00)	737.62	(7.22)	712.04	(3.51)	712.04	(3.51)	762.04	(10.77)	712.04	(3.51)	687.92
R7a	227.35	(0.00)	234.22	(3.02)	234.22	(3.02)	233.08	(2.52)	230.64	(1.45)	230.64	(1.45)	227.35
R8a	433.49	(0.00)	475.44	(9.68)	473.97	(9.34)	473.97	(9.34)	459.75	(6.06)	459.75	(6.06)	433.49
R9a	770.55	(5.89)	807.77	(11.00)	727.70	(0.00)	727.70	(0.00)	727.70	(0.00)	727.70	(0.00)	727.70
R10a	757.32	(0.00)	835.15	(10.28)	835.15	(10.28)	835.15	(10.28)	826.97	(9.20)	826.97	(9.20)	757.32
R1b	167.72	(13.58)	155.84	(5.53)	154.78	(4.81)	151.59	(2.65)	148.79	(0.76)	147.67	(0.00)	147.67
R2b	235.15	(0.00)	243.78	(3.67)	240.56	(2.30)	240.56	(2.30)	240.56	(2.30)	240.56	(2.30)	235.15
R3b	449.52	(0.00)	469.89	(4.53)	469.89	(4.53)	469.89	(4.53)	466.83	(3.85)	466.83	(3.85)	449.52
R4b	443.43	(0.00)	461.57	(4.09)	458.39	(3.37)	458.39	(3.37)	458.39	(3.37)	458.39	(3.37)	443.43
R5b	478.39	(0.00)	486.72	(1.74)	486.72	(1.74)	486.72	(1.74)	486.72	(1.74)	486.72	(1.74)	478.39
R6b	548.94	(0.00)	608.51	(10.85)	608.51	(10.85)	608.51	(10.85)	608.51	(10.85)	608.51	(10.85)	548.94
R7b	194.09	(2.35)	193.43	(2.00)	193.43	(2.00)	193.43	(2.00)	190.75	(0.59)	189.63	(0.00)	189.63
R8b	451.76	(4.20)	433.54	(0.00)	433.54	(0.00)	433.54	(0.00)	433.54	(0.00)	433.54	(0.00)	433.54
R9b	483.43	(0.00)	543.48	(12.42)	534.89	(10.64)	515.16	(6.56)	515.16	(6.56)	515.16	(6.56)	483.43
R10b	744.67	(0.00)	786.45	(5.61)	765.93	(2.85)	765.93	(2.85)	765.93	(2.85)	765.93	(2.85)	744.67
Avg.	453.92	(1.69)	475.65	(5.36)	467.56	(3.94)	465.94	(3.53)	466.15	(3.31)	463.48	(2.87)	448.04

Table 5: Computation times

Problem	n	m	TS-O	TS-N
R1a	24	3	2.03	0.08
R2a	48	5	11.55	0.47
R3a	72	7	36.39	1.28
R4a	96	9	64.91	2.51
R5a	120	11	106.93	4.23
R6a	144	13	197.20	7.45
R7a	36	4	5.63	0.22
R8a	72	6	50.40	1.63
R9a	108	8	168.32	4.88
R10a	144	10	378.63	10.11
R1b	24	3	1.67	0.08
R2b	48	5	12.25	0.46
R3b	72	7	36.19	1.38
R4b	96	9	64.26	2.60
R5b	120	11	107.77	4.14
R6b	144	13	204.64	7.33
R7b	36	4	5.50	0.24
R8b	72	6	49.70	1.80
R9b	108	8	169.95	4.91
R10b	144	10	379.20	10.34
Avg.			102.64	3.31

9 Conclusion

In this paper we have defined and formulated the probabilistic dial-a-ride problem, and proposed an efficient neighborhood search procedure for the problem. This technique makes it possible to update the probabilities faster than the straightforward approach if the search is conducted in a certain order. We have also developed a tabu search heuristic and a hybrid GRASP-tabu search heuristic for the PDARP. Computational results show that, with the suggested neighborhood evaluation technique embedded in the described heuristics, considerable speed-up is achieved. The experiments also show that tabu search performs better than GRASP.

Acknowledgements

This work was supported by the Research Council of Norway under grant 127533/432. This support is gratefully acknowledged.

References

Attanasio, A., Cordeau, J.-F., Ghiani, G. and Laporte, G. (2004), 'Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem', *Parallel Computing* **30**(3), 377–387.

Beraldi, P., Ghiani, G., Laporte, G. and Musmanno, R. (2003), Efficient neighborhood search for the probabilistic pickup and delivery travelling salesman

problem, Technical Report CRT-2003-38, Centre de recherche sur les transports, Université de Montréal, Canada.

Bertsimas, D. J. (1988), Probabilistic Combinatorial Optimization Problems, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

Bertsimas, D. J., Jaillet, P. and Odoni, A. R. (1990), 'A priori optimization', *Operations Research* **38**(6), 1019–1033.

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.

Cordeau, J.-F. (2003), A branch-and-cut algorithm for the dial-a-ride problem, Technical Report CRT-2003-24, Centre de recherche sur les transports, Université de Montréal, Canada.

Cordeau, J.-F., Gendreau, M. and Laporte, G. (1997), 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *Networks* **30**(2), 105–119.

Cordeau, J.-F. and Laporte, G. (2003*a*), 'The dial-a-ride problem: Variants, modeling issues and algorithms', *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* **1**, 89–101.

Cordeau, J.-F. and Laporte, G. (2003*b*), 'A tabu search heuristic for the static multi-vehicle dial-a-ride problem', *Transportation Research B* **37**(6), 579–594.

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**(8), 928–936.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M. and Soumis, F. (2002), VRP with pickup and delivery, *in* P. Toth and D. Vigo, eds, 'The Vehicle Routing Problem', SIAM Society for Industrial and Applied Mathematics, pp. 225–242.

Feo, T. A. and Resende, M. G. C. (1995), 'Greedy randomized adaptive search procedures', *Journal of Global Optimization* **6**(2), 109–133.

Gendreau, M., Hertz, A. and Laporte, G. (1994), 'A tabu search heuristic for the vehicle routing problem', *Management Science* **40**(10), 1276–1290.

Gendreau, M., Laporte, G. and Séguin, R. (1996), 'Stochastic vehicle routing', *European Journal of Operational Research* **88**(1), 3–12.

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Computers & Operations Research* **13**(5), 533–549.

Glover, F. (1997), Tabu search and adaptive memory programming - advances, applications and challenges, *in* R. S. Barr, R. V. Helgason and J. L. Kennington, eds, 'Interfaces in Computer Science and Operations Research', Kluwer, pp. 1–75.

Glover, F. and Laguna, M. (1997), *Tabu Search*, Kluwer, Boston.

Healy, P. and Moll, R. (1995), 'A new extension of local search applied to the dial-a-ride problem', *European Journal of Operational Research* **83**(1), 83–104.

Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M. M. and Villeneuve, D. (1995), 'A request clustering algorithm for door-to-door handicapped transportation', *Transportation Science* **29**(1), 63–78.

Jaillet, P. (1985), Probabilistic Traveling Salesman Problem, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

Jaillet, P. (1988), 'A priori solution of a traveling salesman problem in which a random subset of the customers are visited', *Operations Research* **36**(6), 929–936.

Jaw, J., Odoni, A. R., Psaraftis, H. N. and Wilson, N. H. M. (1986), 'A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows', *Transportation Research B* **20**(3), 243–257.

Kenyon, A. (2000), A survey on stochastic location and routing problems, Technical Report 183, Technische Universität Graz, Austria.

Laporte, G., Louveaux, F. V. and Mercure, H. (1994), 'A priori optimization of the probabilistic traveling salesman problem', *Operations Research* **42**(3), 543–549.

Madsen, O. B., Ravn, H. F. and Rygaard, J. M. (1995), 'A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities and multiple objectives', *Annals of Operations Research* **60**(1-4), 193–208.

Pitsoulis, L. S. and Resende, M. G. C. (2002), Greedy randomized adaptive search procedures, *in* P. M. Pardalos and M. G. C. Resende, eds, 'Handbook of Applied Optimization', Oxford University Press, pp. 168–183.

Prais, M. and Ribeiro, C. C. (2000), 'Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment', *INFORMS Journal on Computing* **12**(3), 164–176.

Psaraftis, H. N. (1980), 'A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem', *Transportation Science* **14**(2), 130–154.

Psaraftis, H. N. (1983), 'An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows', *Transportation Science* **17**(3), 351–357.

Savelsbergh, M. and Sol, M. (1995), 'The general pickup and delivery problem', *Transportation Science* **29**(1), 17–29.

Sexton, T. R. and Bodin, L. D. (1985*a*), 'Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling', *Transportation Science* **19**(4), 378–410.

Sexton, T. R. and Bodin, L. D. (1985*b*), 'Optimizing single vehicle many-to-many operations with desired delivery times: II. Scheduling', *Transportation Science* **19**(4), 411–435.

Taillard, É. D. (1993), 'Parallel iterative search methods for vehicle routing problems', *Networks* **23**(8), 661–673.

Toth, P. and Vigo, D. (1997), 'Heuristic algorithms for the handicapped persons transportations problem', *Transportation Science* **31**(1), 60–71.

Waters, C. D. J. (1989), 'Vehicle-scheduling problems with uncertainty and omitted customers', *Journal of the Operational Research Society* **40**(12), 1099–1108.