Transportation Research Part B 56 (2013) 120–135

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

A Granular Tabu Search algorithm for the Dial-a-Ride Problem

Dominik Kirchler [a,b,c,*], Roberto Wolfler Calvo [b]

[a] LIX, Ecole Polytechnique, 91128 Palaiseau, France
[b] LIPN, Université Paris 13, 93430 Villetaneuse, France
[c] Mediamobile, 94200 Ivry-sur-Seine, France

ARTICLE INFO

Article history:
Received 1 December 2012
Received in revised form 29 July 2013
Accepted 30 July 2013
Available online xxxx

Keywords:
Granular Tabu Search
Dial-a-Ride Problem

ABSTRACT

In a Dial-a-Ride system, a fleet of vehicles without fixed routes and schedules carries people from their pick-up point to their delivery point. Pre-specified time windows must be respected, and service levels for passengers as well as operation costs should be optimized. The resulting routing and scheduling problem is $\mathcal{NP}$-hard and can be modeled by a mixed integer linear programming formulation. In this paper, we propose a Granular Tabu Search algorithm for the static Dial-a-Ride Problem with the objective of producing good solutions in a short amount of time (up to 3 min). We evaluate the algorithm on test instances from the literature. Our new algorithm performs well in comparison with a classical Tabu Search algorithm, a Genetic Algorithm, and a Variable Neighborhood Search.

1. Introduction

A Dial-a-Ride (DAR) system organizes the routing of a fleet of vehicles in order to satisfy transportation demands. Passengers may request the service by calling a central unit. They have to specify their pick-up point, their delivery point, the number of passengers, and some limitations on the service time (e.g., the earliest departure time). The routes and schedules of the vehicles vary depending on the received requests. In the *static* DAR, the passenger asks for the service in advance and the routing of the vehicles is determined before the system starts to operate; in the *dynamic* DAR, the passenger can call during the service time and the routes are updated in real time. A DAR system offers passengers the comfort and flexibility of private cars and taxis at a lower cost. It is suited to serve sparsely populated areas, weak demand periods, or passengers with specific requirements (elderly, disabled). Applications have been reported from several cities, e.g., Bologna (Toth and Vigo (1997)), Copenhagen (Madsen et al. (1995)), Milan (Wolfer Calvo and Colorni (2006)).

The resulting Dial-a-Ride Problem (DARP) is $\mathcal{NP}$-hard, as it generalizes the Pickup and Delivery Problem with Time Windows (PDPTW). Various solution strategies have been described in the literature and various types of objective functions have been studied. They include the minimization of the number of vehicles used, the mean travel or waiting time of passengers, the maximization of the number of passengers served, and the level of service provided.

Several exact algorithms have been proposed for the single-vehicle case. Among the first were Psaraftis (1983), Desrosiers et al. (1986), and Sexton and Choi (1986). Multi-vehicle cases have been studied by Dumas et al. (1991) as well as by Cordeau (2006). In the latter the author describes a Branch and Cut algorithm. To speed up running times, several heuristics for different versions of DARP have been produced. Parallel insertion algorithms have been presented by Jaw et al. (1986) and Madsen et al. (1995). A heuristic for the transportation of handicapped people with a homogeneous fleet of vehicles has been proposed by Ioachim et al. (1995). The algorithm presented by Toth and Vigo (1997) deals with a heterogeneous fleet of vehicles. Cordeau and Laporte (2003) propose a Tabu Search to solve a static DARP with a homogeneous fleet of vehicles. A DARP with a fixed number of vehicles and whose objective function maximizes service quality for passengers is discussed in

* Corresponding author at: LIX, Ecole Polytechnique, 91128 Palaiseau, France. Tel.: +33 684390387.
E-mail addresses: kirchler@lix.polytechnique.fr (D. Kirchler), wolfler@lipn.fr (R. Wolfler Calvo).

0191-2615/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.trb.2013.07.014

Wolfer Calvo and Colorni (2006). Diana and Dessouky (2004) propose a constructive heuristic for the DARP based on the assignment of passengers to vehicles according to a regret function. Moreover, Diana et al. (2006) propose a method for the fleet sizing of a transport service on-demand. Lu and Dessouky (2006) present an insertion-based constructive heuristic and Jorgensen et al. (2006) use a genetic algorithm. Recently, Parragh et al. (2010) and Parragh and Schmid (2012) use Variable Neighborhood Search and Hybrid Large Neighborhood Search to solve the DARP. Paquette et al. (2013) discuss multi-criteria optimization within a Tabu Search mechanism for the DARP.

For a broader overview of existing solution techniques and DARP-variants we refer to Cordeau and Laporte (2007) and Parragh et al. (2008). Berbeglia et al. (2010) give an overview of dynamic DARP-variants.

1.1. Our contribution

This paper addresses the static DARP with time windows and a fixed fleet of vehicles. Our objective is to maximize the number of passengers served and the quality of service, as well as to minimize overall system cost. There is a growing interest in fast methods for obtaining high quality DARP solutions, since DARPs often occur in a dynamic real-world setting. Therefore, the main contribution of this paper is the development of an efficient and fast heuristic to produce good solutions in a short amount of time (up to 3 min). We propose a new Granular Tabu Search which uses information provided by the solution of a simple and useful sub-problem to guide the local search process. This sub-problem provides distance information and clusters of close requests. The idea is that passengers who are close both spatially (in terms of the distance between pick-up and delivery points) and temporally (with respect to time windows) are probably best served by the same vehicle in order to produce good solutions. This intuition has been introduced in Wolfer Calvo and Colorni (2006) to produce good initial solutions. In this paper, we go a step further and exploit this information during the improvement phase of a Granular Tabu Search algorithm in order to limit the local search neighborhood. We compare the results of our new algorithm with the results of a Variable Neighborhood Search (VNS) algorithm presented in Parragh et al. (2010) and a Genetic Algorithm (GA) presented in Jorgensen et al. (2006). Results are based on instances from Cordeau and Laporte (2003). Our algorithm performs well and produces better results than the GA for all instances. In the long run, the VNS performs better than our Granular Tabu Search algorithm, but we are able to report better results on test instances after 60 s of optimization time.

This paper is organized as follows. The problem definition and its mixed integer linear programming formulation are given in Sections 2, 3 describes the Granular Tabu Search approach and how it has been applied to solve DARP. Computational results and conclusions close the paper.

2. The dial-a-ride problem

Let $R = \{1, \ldots, n\}$ be a set of requests. Each request i consists of two nodes, i^+ and i^-. A load q_i must be taken from i^+ to i^-. Let $N^+ = \{i^+, i \in R\}$ be the set of pick-up nodes and $N^- = \{i^-, i \in R\}$ be the set of delivery nodes ($N' = N^+ \cup N^-$). A positive amount $q_{i^+} = q_i$ is associated with the pick-up node, a negative amount $q_{i^-} = -q_i$ with the delivery node. A time window is also associated with each node, i.e., $[e_{i^+}, l_{i^+}]$ for the pick-up node and $[e_{i^-}, l_{i^-}]$ for the delivery node. The fleet of vehicles is denoted as V and all vehicles have the same capacity Q. Let $G = (N,A)$ be a directed graph with a set of nodes $N = N^+ \cup N^- \cup \{0, 2n+1\}$ and a set of arcs $A = \{(i,j): i, j \in N, i \neq j\}$. Nodes 0 and $2n+1$ represent the start and end depot, and have time windows $[e_0, l_0]$ and $[e_{2n+1}, l_{2n+1}]$, which denote the earliest departure time and the latest return time at the depots. Travel time t_{ij} is assigned to each arc $(i,j) \in A$. Service time at nodes is d_i. Ride time constraints for passengers have to be respected and may not exceed T^{ride}. Maximal route time for vehicles is bounded by T^{route}.

The problem consists in finding a set of routes starting and ending at the depots 0 and $2n+1$, respectively, such that the objective function is optimized and routing, capacity, and time window constraints are respected. We use the following variables: x_{ij}^v is 1 if vehicle v uses arc (i,j) and 0 otherwise; A_i, B_i, and D_i represent arrival, start of service, and departure time at node $i \in N$; D_0^v and A_{2n+1}^v represent departure and arrival time at the start and end depots for vehicle $v \in V$; variable y_i holds the load of the vehicle after leaving node i.

$$\min f'(s) = \min(\omega_1 \cdot c(s) + \omega_2 \cdot r(s) + \omega_3 \cdot l(s) + \omega_4 \cdot g(s) + \omega_5 \cdot e(s) + \alpha \cdot k(s)) \tag{1}$$

subject to

$$\sum_{v \in V} \sum_{j \in N} x_{ij}^v \leqslant 1 \quad \forall i \in N^+ \cup \{0\} \tag{2}$$

$$\sum_{j \in N} x_{ij}^v - \sum_{j \in N} x_{ji}^v = 0 \quad \forall v \in V, \quad \forall i \in N' \tag{3}$$

$$\sum_{j \in N} x_{i^+ j}^v - \sum_{j \in N} x_{ji^-}^v = 0 \quad \forall v \in V, \forall (i^+, i^-) \in \{(i^+, i^-) : i \in R\} \cup (0, 2n+1) \tag{4}$$

$$x_{ij}^v (y_i + q_j) \leqslant y_j \quad \forall v \in V, \quad \forall (i,j) \in A \tag{5}$$

$$q_i \leqslant y_i \leqslant Q \quad \forall i \in N^+ \tag{6}$$

$$x_{ij}^v (D_i + t_{ij}) = x_{ij}^v A_j \leqslant B_j \quad \forall v \in V, \quad \forall (i,j) \in N' \times N' \tag{7}$$

$$x^v_{0j}(D^v_0 + t_{0j}) = x^v_{0j}A_j \leqslant B_j \quad \forall v \in V, \quad \forall j \in N^+ \tag{8}$$

$$x^v_{i,2n+1}(D_i + t_{i,2n+1}) = x^v_{i,2n+1}A^v_{2n+1} \quad \forall v \in V, \quad \forall i \in N^- \tag{9}$$

$$e_i \leqslant B_i \leqslant l_i \quad \forall i \in N' \tag{10}$$

$$B_{i^-} - D_{i^+} \leqslant T^{\text{ride}} \quad \forall i \in R \tag{11}$$

$$A^v_{2n+1} - D^v_0 \leqslant T^{\text{route}} \quad \forall v \in V \tag{12}$$

$$D^v_0 \geqslant e_0, A^v_{2n+1} \leqslant l_{2n+1} \quad \forall v \in V \tag{13}$$

$$\sum_{v \in V}\sum_{j \in N^+} x^v_{0j} \leqslant m \tag{14}$$

$$x^v_{ij} \in \{0,1\} \quad \forall v \in V, \quad \forall (i,j) \in A \tag{15}$$

$$D_i \geqslant 0, A_i \geqslant 0 \quad \forall i \in N' \tag{16}$$

The objective function (1) minimizes routing cost $c(s) = \sum_{(i,j)\in A, v\in V} x^v_{ij}t_{ij}$, excess ride time $r(s) = \sum_{i\in R}(B_{i^-} - D_{i^+} - t_{i^+i^-})$, waiting time of passengers on board $l(s) = \sum_{i\in N'} w_i(y_i - q_i)$, route durations $g(s) = \sum_{v\in V}(A^v_{2n+1} - D^v_0)$, early arrival times at pickup and delivery nodes $e(s) = \sum_{i\in N'}(e_i - A_i)^+$, and finally the number of unserved requests $k(s) = n - \sum_{v\in V,(i,j)\in N^+\times N} x^v_{ij}$. The notation is summarized in Table 1. This objective function has first been introduced by Jorgensen et al. (2006).

The first three groups of constraints (2)–(4) impose that each request is served by at most one vehicle. Constraints (5) and (6) ensure the feasibility of the loads. Time constraints (7)–(10) ensure correct arrival, service, and departure time and (11) and (12) ensure that passenger ride time and bus route duration do not exceed T^{ride} and T^{route}, respectively. Finally, constraint (14) limits the number of vehicles which can be used. Note that constraints (5) and (7) can be linearized as $M(1 - x^v_{ij}) \geqslant y_i + q_j - y_j$ and $M(1 - x^v_{ij}) \geqslant D_i + t_{ij} - B_j$, respectively. The same is true for constraints (8) and (9). These equations are a generalization of the classical TSP sub-tour elimination constraints proposed by Miller et al. (1960).

3. Solution framework

We apply a *Granular Tabu Search* (GTS) to solve the DARP. It is based on the well known *Tabu Search* (TS) algorithm which is a memory-based search method introduced by Glover (1986). The TS is able to escape local optima by allowing the objective function to deteriorate and it avoids cyclic moves in the search space by keeping track of recent moves through the use of a memory structure called *tabu list*. The size, contents, and management policies of the tabu list depend on the specific problem and algorithm. To improve the effectiveness of the TS, diversification and intensification strategies are employed.

The Granular Tabu Search is a Tabu Search with a particular focus on the local search phase. It uses a reduced neighborhood (*granular neighborhood*) which ideally should not include moves which are unlikely to belong to good solutions. The size of the granular neighborhood is regulated by a granular threshold. The GTS has first been proposed by Toth and Vigo (2003). The authors apply the method to the Vehicle Routing Problem (VRP). Based on the assumption that good

Table 1
Notations.

$R = \{1, \ldots, n\}$	Set of requests
$V = \{1, \ldots, m\}$	Set or vehicles
N	Set of nodes
$\{i^+, i^-\}$	A transportation request
t_{ij}	Travel time for arc (i,j)
Q	Vehicle capacity
T^{route}	Maximum vehicle route duration
T^{ride}	Maximum passenger ride time
T^{ride}_i	Ride time of request i
q_i	Number of passengers to be picked up at node i
e_i	Beginning of time window at node i
l_i	End of time window at node i
d_i	Service time at node i
A_i	Arrival time at node i
B_i	Beginning of service at node i
$D_i = B_i + d_i$	Departure time from node i
$w_i = B_i - A_i$	Vehicle waiting time at node i
D^v_0	Departure time at start depot for vehicle v
A^v_{2n+1}	Arrival time at end depot for vehicle v
y_i	Load when leaving node i
s	A solution (routing plan)

solutions rarely contain long edges (in terms of travel time), their algorithm explores only edges with a length up to a certain threshold during the local search phase. Whenever no improving solutions can been found for a certain time during execution, the threshold is gradually increased. The authors show that, in their scenario, this approach significantly reduces computation time to find good solutions in comparison to the classical Tabu Search. See Labadie et al. (2012) for another application of a granular neighborhood in a Variable Neighborhood Search framework to solve the Team Orienting problem.

3.1. The granular neighborhood

The neighborhood $N(s)$ of a solution s includes all feasible solutions that can be obtained by applying a single simple move to the current solution s. We consider the following simple moves: move a request from a route into another one, insert an unserved request in a route, or remove a request from a route. More complex transformations can be achieved through sequences of simple moves. Since we are considering a granular neighborhood, the only moves allowed are those with *reduced cost* $\bar{c}_{ij} < T_{\text{Gran}}$, where T_{Gran} is the *granular threshold*. We construct the granular neighborhood by solving an assignment problem, which is based on the value $\overline{D}_{ij}$ and which provides the reduced costs $\bar{c}_{ij}$.

3.1.1. Average departure time $\overline{D}_{ij}$

To measure the spatial and temporal distance between two requests i and j, we introduce the average departure time $\overline{D}_{ij}$. It provides an evaluation of the feasibility and the cost of serving requests i and j by using the same vehicle and by starting at node i^+. The possible sequences of nodes for a vehicle to serve the two requests are: $\Pi^1 = (i^+,i^-,j^+,j^-)$, $\Pi^2 = (i^+,j^+,i^-,j^-)$, and $\Pi^3 = (i^+,j^+,j^-,i^-)$ (see Fig. 1).

For a generic sequence of s nodes $\Pi = (\pi_1, \ldots, \pi_s)$, the departure time at the last node of the sequence (node π_s) can be determined using the following equation (Wolfer Calvo and Colorni (2006))

$$D^{\Pi} = D_{\pi_1} + T_{\pi_1,\pi_s} + W_{\pi_1,\pi_s}, \tag{17}$$

where T_{π_1,π_s} is the total travel time along the sequence, including service time, and W_{π_1,π_s} is the total waiting time. By applying this equation to the three sequences in Fig. 1, we obtain:

$$\begin{cases} D^{\Pi_1} = e_{i^+} + d_{i^+} + t_{i^+i^-} + d_{i^-} + t_{i^-j^+} + d_{j^+} + t_{j^+j^-} + w_{j^+} + d_{j^-} \\ D^{\Pi_2} = e_{i^+} + d_{i^+} + t_{i^+j^+} + d_{j^+} + t_{j^+i^-} + d_{i^-} + t_{i^-j^-} + w_{j^+} + d_{j^-} \\ D^{\Pi_3} = e_{i^+} + d_{i^+} + t_{i^+j^+} + d_{j^+} + t_{j^+j^-} + d_{j^-} + t_{j^-i^-} + w_{j^+} + d_{i^-} \end{cases} \tag{18}$$

If for a sequence Π time or load constraints at a node are not respected, then we set $D^{\Pi} = \infty$. Note that the vehicle might have to only wait in node j^+ (w_{j^+}), since the sequence starts in i^+ and $e_{i^-} = e_{i^+} + d_{i^+} + t_{i^+i^-}$. Now we define the *average departure time* of the departure times at the last nodes of the sequences Π_1, Π_2, and Π_3 of a vehicle when serving two request i and j as:

$$\overline{D}_{ij} = \sum_{\Pi \in \{\Pi_1,\Pi_2,\Pi_3\}} \frac{D^{\Pi} \cdot k^{\Pi}}{k^{\Pi}}, \tag{19}$$

where $k^{\Pi} = 1$, if $D^{\Pi} \neq \infty$, $k^{\Pi} = 0$ otherwise. Note that if $k^{\Pi_1} + k^{\Pi_2} + k^{\Pi_3} = 0$ and thus $\overline{D}_{ij} = \infty$, it is not feasible to serve request i and request j with the same vehicle (by starting from i^+). Note also that in general $\overline{D}_{ij} \neq \overline{D}_{ji}$.

3.1.2. The assignment problem

Our goal is to construct clusters of requests which are close in respect to $\overline{D}_{ij}$. We first define an auxiliary graph $\widehat{G} = (\widehat{R}, \widehat{A})$ with a set of nodes $\widehat{R} = R \cup \{n+1, \ldots, n+m\}$ consisting of n nodes representing requests and m nodes representing the available vehicles. $\widehat{A} = \{(i,j) : i,j \in \widehat{R}, i \neq j\}$ represents the set of arcs. Arc weights $\widehat{D}_{ij}$ are assigned as follows:

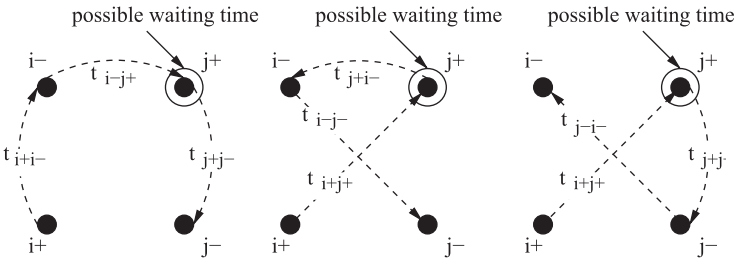


Fig. 1. Possible sequences of nodes to serve two requests i and j.

$$\widehat{D}_{ij} = \begin{cases} \overline{D}_{ij} & \forall\, i,j \in R \\ \infty & \forall\, i,j \in V \\ e_0 + t_{0j^+} + w_{j^+} + d_{j^+} & \forall\, (i,j) \in V \times R \\ l_{2n+1} - e_{i^-} - t_{i^-,2n+1} - d_{i^-} & \forall\, (i,j) \in R \times V \end{cases} \tag{20}$$

with $w_{j^+} = \max(e_{j^+} - e_0 - t_{0j^+}, 0)$.

Next we define and solve an assignment problem on $\widehat{G}$. The time window constraints (7)–(10) are relaxed. However, they are partially enforced when calculating $\overline{D}_{ij}$. Constraints (2)–(4) have to be taken into account. Constraints (11), (12), (5), and (6) are relaxed. Constraints (3), (4), and (14) are replaced by the classical assignment constraints. The problem to be solved becomes a standard assignment problem of size $n + m$:

$$\min \sum_{(i,j)\in\widehat{A}} \widehat{D}_{ij} x_{ij} \tag{21}$$

$$\sum_{j\in\widehat{R}} x_{ji} = 1 \quad \forall i \in \widehat{R} \tag{22}$$

$$\sum_{i\in\widehat{R}} x_{ji} = 1 \quad \forall j \in \widehat{R} \tag{23}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in \widehat{A} \tag{24}$$

The solution of the assignment problem is a set of clusters which contain a sequence of requests. Some of these clusters contain a vehicle (maximal one) while the rest of them, called *sub-tours*, are composed by requests only. The solution of the assignment problem can be exploited in two ways. First, it can be used to quickly construct an initial solution, as shown in Wolfer Calvo and Colorni (2006). Second, the assignment problem provides the reduced cost value for each arc (i,j) calculated as $\bar{c}_{ij} = \widehat{D}_{ij} - u_i - v_j$ where u_i and v_j are the dual variables of constraints (22) and (23). The reduced cost indicates the impact on the cost of the solution when replacing edges which are part of the current solution with edges outside the solution.

Note that several possible solutions with similar cost exist, because there are at least $n + m - 1$ arcs with reduced cost equal to 0, but which are not used in the optimal solution. In other words, the solution of the assignment problem proposes the way in which requests in the same cluster may be served by the same vehicle. Nevertheless, there are pairs of requests which in the current solution are placed in different clusters, but which could be served by the same vehicle without increasing the cost of the solution too much.

Let us consider a small example instance consisting of four requests $\{1,2,3,4\}$ which can be served by two vehicles. A possible feasible solution is the following: requests 1 and 2 are served by vehicle (A) and requests 3 and 4 are served by vehicle (B) (see Fig. 2). The graph $\widehat{G} = (\widehat{R}, \widehat{A})$ associated with this solution is reported in Fig. 3.

Each arc of this graph represents two possible moves, e.g., arc (3,2) suggests to move requests 3 to the vehicle serving request 2 or to move request 2 to the vehicle serving 3. The resulting two new solutions would be $(d,4,d)$, $(d,1,3,2,d)$ and $(d,1,d)$, $(d,3,2,4,d)$. In a similar way, by applying the moves represented by arc (2,3), the two new solutions would be $(d,4,d)$, $(d,1,2,3,d)$ and $(d,1,d)$, $(d,2,3,4,d)$. Note that the difference between considering arc (3,2) and (2,3) is the position of the pick-up nodes of requests 3 and 2.

Now let us suppose that the table in Fig. 4 represents the reduced cost matrix given by the solution of the assignment problem. We are interested in low reduced costs, as we suppose that they indicate the most promising moves. In order to limit the number of moves, we define a granular threshold T_{Gran} and ignore moves with reduced cost $\bar{c}_{ij} > T_{\text{Gran}}$. See Fig. 5.

3.2. Preprocessing

We apply graph pruning and time window tightening techniques as described in Cordeau and Laporte (2003) before the optimization procedure is started. First, time windows are tightened. In case of an outbound request, the time window at the

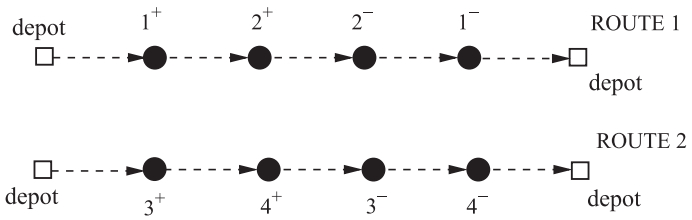


Fig. 2. The complete initial solution.

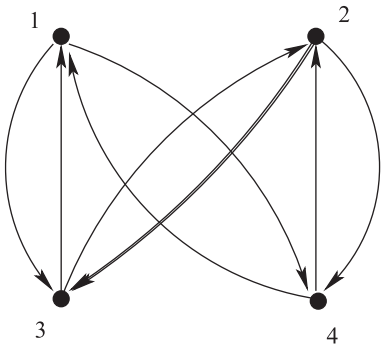


Fig. 3. The graph $\widehat{G} = (\widehat{R}, \widehat{A})$ associated with the solution $(d,1,2,d)$, $(d,3,4,d)$, without nodes representing vehicles.

	Bus A	Bus B	1	2	3	4
Bus A	\	\	0	0	0	1097
Bus B	\	\	0	0	0	1097
1	902	902	\	0	17	374
2	0	0	233	\	0	104
3	0	0	\	\	\	0
4	0	0	\	\	\	\

Arcs used in the solution

Values close to infinity

Fig. 4. Matrix of reduced costs given by the assignment solution.

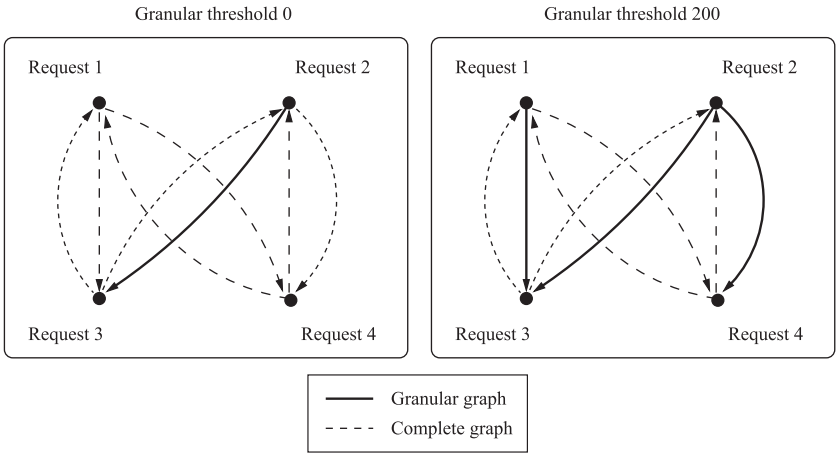


Fig. 5. Allowed moves by varying the granular threshold (bold).

pick-up node i^+ is set to $e_{i^+} = \max(0, e_{i^-} - T^{\text{trip}} - d_{i^+})$ and $l_{i^+} = \min(l_{i^-} - t_{i^+,i^-} - d_{i^+}, H)$, where H is the end of the planning horizon. In case of an inbound request, the time window at delivery node i^- is set to $e_{i^-} = e_{i^+} + d_{i^+} + t_{i^+i^-}$ and $l_{i^-} = \min(l_{i^+} + d_{i^+} + T^{\text{trip}}, H)$. Then we assign a very high value in the distance matrix to all the arcs which cannot be part of a feasible solution: arcs which connect start and end depots with delivery and pick-up nodes, respectively, arcs which connect delivery nodes with its pick-up nodes, and arcs connecting nodes which are incompatible regarding their time-windows and travel time.

3.3. Initial solution

The assignment problem produces clusters of close requests. Some clusters include a vehicle, others do not. This information can be used to construct a good feasible initial solution for the Granular Tabu Search as first discussed in Wolfer Calvo and Colorni (2006). To do this, we proceed as follows: We first set all requests which belong to clusters which have no vehicle as unserved. For each cluster which includes a vehicle, we apply a simple step-by-step procedure to produce a feasible route. Initially, the route consists only of the depot. Then, for each request of the cluster, the procedure attempts to insert the request into the route. If this fails, the request is set as unserved. Lastly, the procedure tries to insert the unserved requests into any of the feasible routes that have been produced.

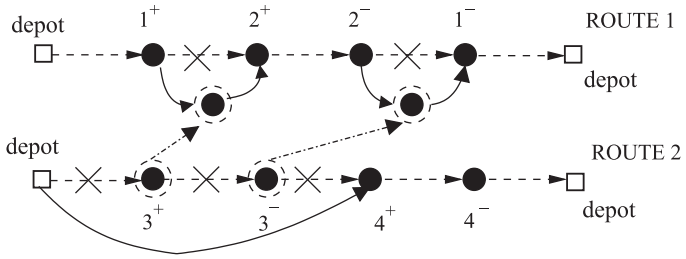
3.4. Local search

Each iteration of the Granular Tabu Search optimization process requires the evaluation of the whole granular neighborhood. The local search algorithm executes exactly one simple move and explores all the feasible positions in a route where the request can be inserted. In the worst case the complexity is $O(n)$. See Fig. 6 for an example of generated solutions when considering arc (3,2) in the reduced cost matrix.

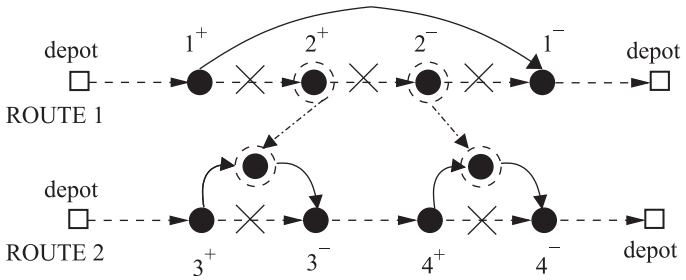
To evaluate a route, we use an adapted version of the eight-step evaluation scheme introduced by Cordeau and Laporte (2003), see Table 2. The difference lies in the fact that their algorithm also considers non-feasible solutions by penalizing constraint violations. In our Tabu Search algorithm we only consider feasible solutions. The eight-step evaluation scheme uses the forward time slack F_i for a node $i \in N$ defined by Savelsbergh (1985), adapted to the DARP:

$$F_i = \min_{i \leqslant j \leqslant q} \Big(\sum_{i < p \leqslant j} w_p + (\min(l_j - B_j, T^{\text{trip}} - P_j))^+ \Big).$$

w_p denotes the waiting time at node p, q is the last node on the route, and P_j the ride time of the passenger whose destination is $j^- \in N^-$ given that j^- is visited before i on the route; $P_j = 0$ for all other j. F_i gives the maximum amount of time by which the departure from a node i can be delayed without violating time windows and passenger ride time.



(a) Request 3 is moved into route 1



(b) Request 2 is moved into route 2

Fig. 6. Two possible solutions induced by arc (3,2).

Table 2
Eight-step evaluation scheme.

1.	Set $D_0 := e_0$
2.	Compute A_i, w_i, B_i, D_i, y_i for each node i along the route If some $B_i > l_i$ or $y_i > Q$, return false
3.	Compute F_0
4.	Set $D_0 := e_0 + \min(F_0, \sum_{0<p<q} w_p)$
5.	Update A_i, w_i, B_i and D_i
6.	Compute all T_i^{ride} If all $T_i^{\text{ride}} < T^{\text{ride}}$ return true
7.	For every node j that is an origin (a) Compute F_j (b) Set $w_j := w_j + \min(F_j, \sum_{j<p<q} w_p); B_j := A_j + w_j; D_j := B_j + d_j$ (c) Update A_i, w_i, B_i and D_i for each node i that comes after j in the route (d) Update T_i^{ride} for each request i whose destination is after j (e) If all $T_i^{\text{ride}} \leqslant T^{\text{ride}}$ of requests whose destinations lie after j, return true
8.	Return false

3.5. Tabu list and aspiration criteria

In a TS framework, the search is guided by a short term memory called tabu list. It is used to avoid cycling and to help the search process to escape local minima. At each iteration the executed move is declared *tabu* for a given number of iterations (*tabu tenure Tt*), which forbids its reversal (see Gendreau (2003)). Moves involving arcs which are tabu can be executed only if they lead to a solution whose objective function value is better than the best solution found so far: this is called *aspiration criterion*. Our algorithm declares all those edges as Tabu which are involved when moving a request. For example, Fig. 7 shows which arcs become tabu after the removal of request j from a route. Note that the number of arcs to be inserted in the tabu list depends on the position of the pickup and delivery node in the route.

3.6. Diversification and intensification

Diversification and intensification strategies serve to improve the effectiveness of the Tabu Search method (Gendreau (2003)). During intensification the search focuses on promising portions of the solutions space, while diversification moves the algorithm to other unexplored regions. Three diversification strategies have been used in this work: dynamic variation of Tabu tenure, frequency-based penalization, and the variation of the granularity threshold.

3.6.1. Dynamic variation of Tabu tenure

The *Tabu tenure Tt* can remain constant or vary according to several strategies. In our algorithm the variation is based on the objective function evolution. If the value of the objective function has improved at least once in *Nit* consecutive number of iterations, then the search process is probably exploring an interesting portion of the solutions space, thus intensification is required and the Tabu tenure value is reduced. On the other hand, if the objective function value did not improve for *Nit* iterations, the search process may have reached a local minimum, thus diversification is required and the Tabu tenure value

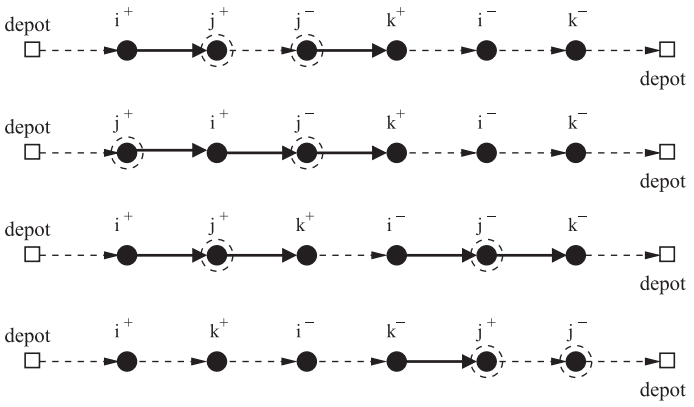


Fig. 7. The figure shows four routes. From each route, request j is removed. After their removal, the arcs in bold are declared *tabu* and are inserted into the tabu list.

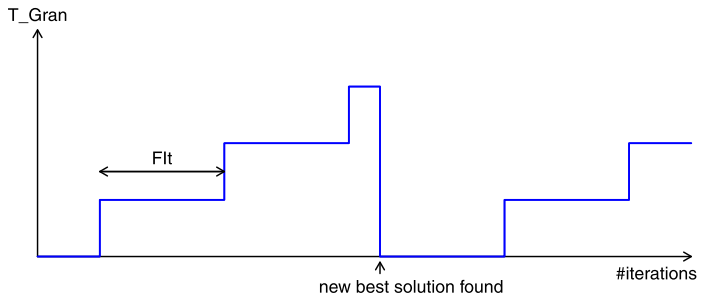


Fig. 8. Variation of granularity threshold T_{Gran}.

is increased. The function used to increase or to decrease Tt is the following: $Tt = Tt \pm \varepsilon Tt$. The maximal value and the minimal value for Tt are bounded by $Tt = Tt + \vartheta Tt$ and $Tt = Tt - \vartheta Tt$, respectively.

3.6.2. Frequency-based penalization

Frequency-based penalization uses a long-term memory for recording the number of times an arc appeared in the incumbent solution. Let s be the current solution then each solution $\bar{s} \in N(s)$ such that $f(\bar{s}) > f(s)$ is penalized by a factor $p(s) = \lambda\rho\sqrt{n \cdot m}f(\bar{s})$. ρ gives the mean value of the number of times the considered arc has been added to the current solution, λ is a parameter used to control the intensity of the diversification, and $\sqrt{n \cdot m}$ is a scaling factor required to adjust the penalties with respect to the problem size. This strategy has been proposed by Taillard (1993) and successfully applied in many other Tabu Search algorithms for vehicle routing problems, see Cordeau and Laporte (2003).

3.6.3. Variation of granularity threshold

The granularization process intensifies the search by reducing the neighborhood search space. Several diversification strategies can be defined by ways how to change dynamically granularity threshold T_{Gran} and thus the neighborhood size. We use the following strategy. At the beginning $T_{Gran} = 0$. T_{Gran} is increased, i.e., $T_{Gran} = T_{Gran} + \delta$, when at least one of the following conditions are verified: (a) all the feasible solutions in the current neighborhood are tabu or (b) the algorithm is unable to improve the best solution found so far for a fixed number of iterations (FIt). T_{Gran} is set to 0 again when the algorithm improves the best solution found so far. δ is defined as $\delta = \gamma(\max_{(ij)\in A} \widehat{\bar{c}}_{ij} - \min_{(ij)\in A} \widehat{\bar{c}}_{ij})$ (see Fig. 8).

3.7. Stopping criterion

We applied two simple stopping criteria. We set a run time limit and a maximum number of consecutive iterations (4000) during which the best global optimal solution did not improve.

4. Experimental results

The Granular Tabu Search was implemented in C++. All experiments were conducted on a Bi AMD Opteron Dual Core computer with 3.2 GHz. Following guidelines of Cordeau and Laporte (2003) and results of preliminary testing we used $\lambda = 0.01$, $Tt = 7.5\log_{10} n$, $Nit = 3$, $\epsilon = 20\%$, and $\vartheta = 70\%$. For the granular threshold we used $\gamma = 10\%$ and $FIt = 40$.

4.1. Test instances

Cordeau and Laporte (2003) produced a data set of 20 randomly generated DARP instances. They contain between 24 and 144 requests; $n/2$ requests were defined as inbound requests while the remaining $n/2$ requests were defined as outbound requests. The service time at pickup and delivery nodes was set to $d_i = 10$. At each node exactly one passenger mounts or leaves the vehicle ($q_i = 1$). Travel times t_{ij} are equal to the Euclidean distance between nodes i and j. Pickup nodes of outbound requests and arrival nodes of inbound requests were assigned a large time window [0,1440] equal to the length of the planning horizon. The data set was split into two groups. Group (a) was assigned narrow time windows whereas the time windows of group (b) were wider. Furthermore, for instances R1a–R6a and R1b–R6b the number of vehicles was set such that routes are only moderately full; instances R7a–R10a and R7b–R10b might be unfeasible with fewer vehicles. Maximum route duration for all instances was set to $T^{route} = 480$, vehicle capacity to $Q = 6$, and maximum passenger ride time to $T^{ride} = 90$.

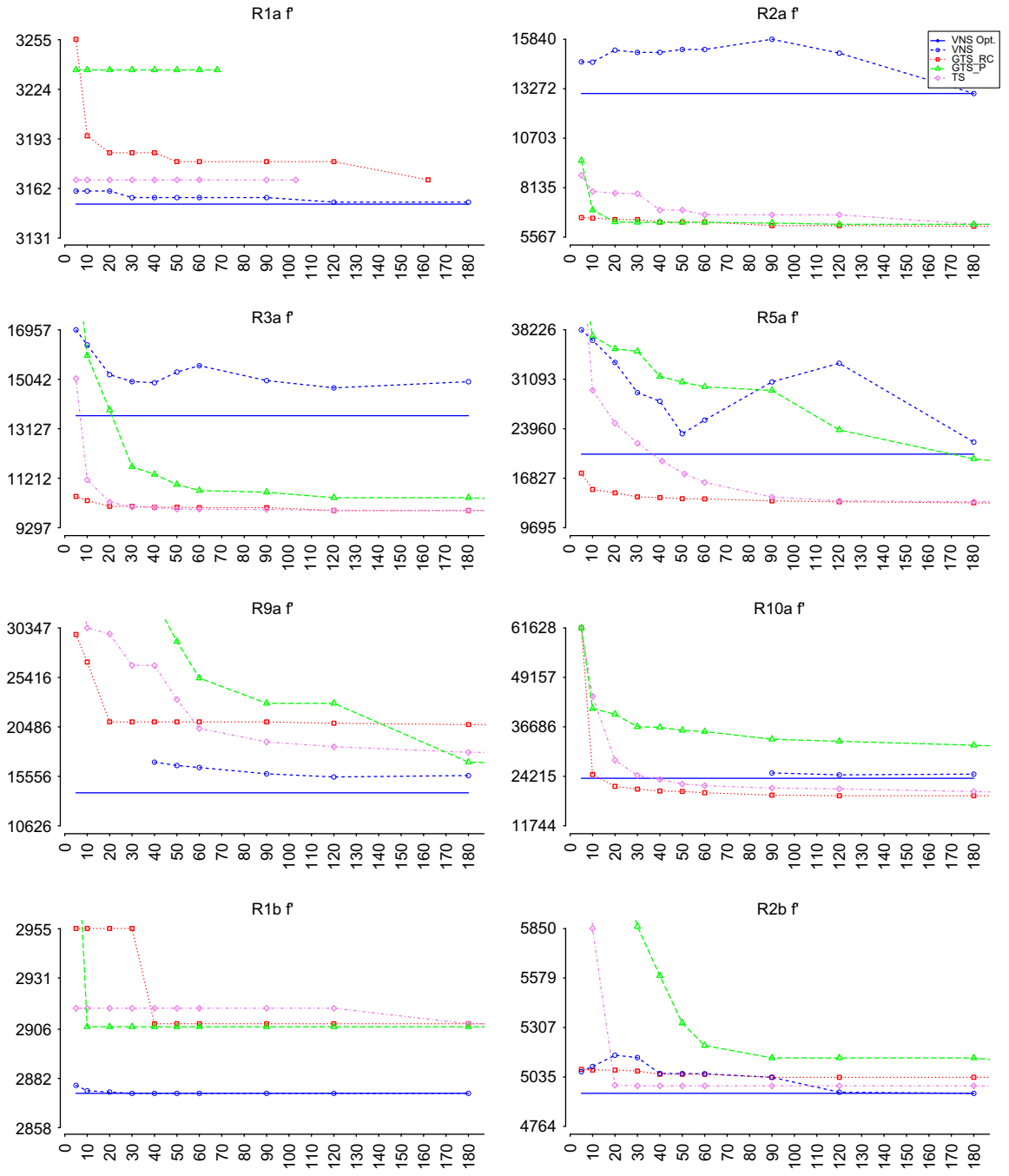


Fig. 9. Evolution of objective function f' over time (x-axis: CPU time, y-axis: value of f').

4.2. Evaluation of Granular Tabu Search

A direct comparison with the results of Cordeau and Laporte (2003) was not possible, as they only minimize routing cost. Therefore, we compared our results with the results of tests conducted on the instances of Cordeau and Laporte (2003) of a Variable Neighborhood Search algorithm (VNS) reported in Parragh et al. (2010)[1] and a Genetic Algorithm (GA) presented in

[1] We use updated results which differ slightly from the results originally published in Parragh et al. (2010). They are available from http://prolog.univie.ac.at/research/DARP.

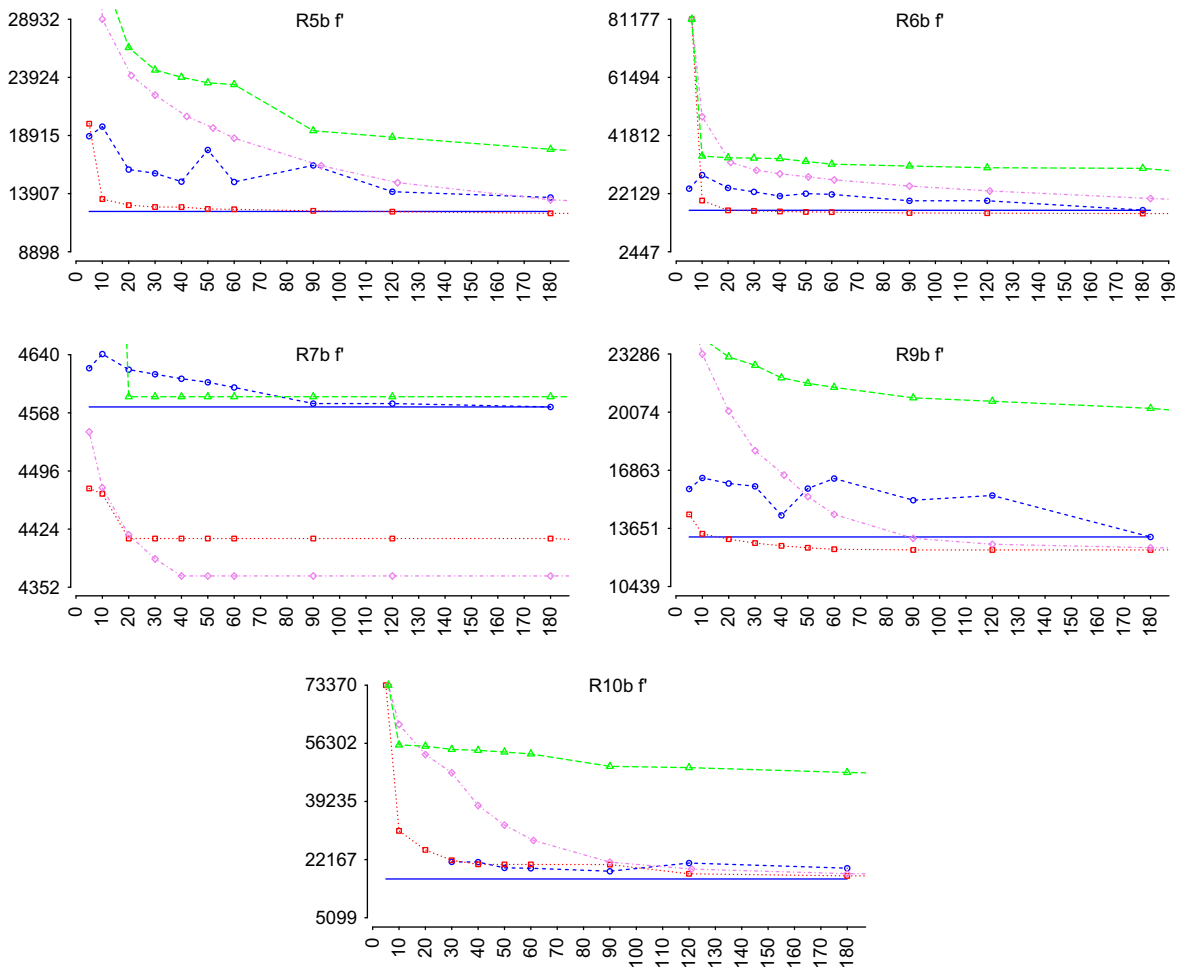


Fig. 10. Evolution of objective function f' over time (x-axis: CPU time, y-axis: value of f').

Jorgensen et al. (2006), detailed results can be found in Bergvinsdottir (2004). Note that they only provide results for 13 test instances. Of the remaining seven, two instances were used for parameter tuning and the others were excluded because detailed information about the solution (number of vehicles used, sequence of nodes served by each vehicle, etc.) is not available. Therefore, we also restrict our tests to these 13 test instances. We used the same weights for f' as proposed in Jorgensen et al. (2006) and Parragh et al. (2010): $\omega_1 = 8$, $\omega_2 = 3$, $\omega_3 = 1$, $\omega_4 = 1$, and $\omega_4 = n$. The coefficient α was set to 10,000. We applied the same CPU time for the optimization process as Parragh et al. (2010), 3–50 min depending on the test instance. The main objective of a Granular Tabu Search is to find good solutions in a short amount of time by concentrating on potential good moves (Toth and Vigo (2003)). Therefore, we recorded intermediate values of the objective function every 10 s during the optimization process. We compared these values to intermediate values of the objective function of the VNS algorithm presented in Parragh et al. (2010).[2] Note that all values reported for the VNS and the GA are *average values* over 5 runs. Our algorithm does not include a random component. We tested our Granular Tabu Search which uses the reduced costs (called GTS_RC, hereafter), as well as a second version of the Granular Tabu Search which applies the value $\overline{D}_{ij}$ which we call GTS_P. As discussed above, $\overline{D}_{ij}$ gives a measure of the temporal and spatial closeness of two requests. GTS_P treats moves involving close requests first, i.e., moves with low $\overline{D}_{ij}$. Finally, we also ran a classical Tabu Search (TS) which investigates the entire neighborhood at each step.

4.3. Comparison on $f'(s)$

In Figs. 9 and 10 we present the evolution of the values of f' during the first 3 min of the computation time. The red, green, violet, and dashed blue lines represent GTS_RC, GTS_P, TS, and VNS, respectively. The continuous blue line marks best known results for each instance which have all been calculated with the VNS. These are reported in Parragh et al. (2010) and in

[2] Results provided to us by the authors of Parragh et al. (2010), not included in their paper.

Table 3
Results for GA (Jorgensen et al. (2006)), VNS (Parragh et al. (2010)), Tabu Search (TS) and Granular Tabu Search (GTS_RC) for objective function f'. CPU indicates CPU time in minutes of the optimization process. CPU times for VNS, TS, GTS_P, and GTS_RC are the same.

	n	m	GA[c]	CPU	VNS[a]	TS[b]	GTS_P[b]	GTS_RC[b]	CPU
R1a	24	3	4694	5.57	**3152.22**	3167.40	3236.14	3167.40	2.70
R2a	48	5	19,426	11.42	14622.40	**6027.29**	6202.98	6130.38	5.16
R3a	72	6	65,306	21.58	15985.90	9965.29	**9952.00**	9962.26	6.38
R5a	120	11	213,420	58.23	25478.78	13054.80	13716.60	**13050.30**	13.93
R9a	108	8	333,283	40.78	**13912.96**	16475.90	16599.90	19449.50	33.53
R10a	144	10	740,890	65.98	25791.02	18260.20	**18259.40**	18487.50	40.27
R1b	24	3	4762	5.46	**2875.37**	2908.66	2908.66	2907.18	3.78
R2b	48	5	13,580	11.72	5003.11	**4969.28**	5037.06	5004.26	8.29
R5b	120	11	98,111	58.93	12373.00	**11747.20**	13241.90	11969.90	23.19
R6b	144	13	185,169	81.23	16486.78	**15000.50**	18438.30	15063.50	26.39
R7b	36	4	9169	8.29	4592.52	**4365.98**	4587.97	4401.78	4.49
R9b	108	8	167,709	44.66	13433.32	**12265.50**	12487.70	12274.80	30.32
R10b	144	10	474,758	66.41	16478.16	**16391.00**	18572.10	16730.80	51.81

[a] Intel Pentium D computer 3.2 GHz.
[b] Bi AMD Opteron Dual Core computer 3.2 GHz.
[c] Intel Celeron 2 GHz.

Table 4
Results for VNS (Parragh et al. (2010)), Tabu Search (TS) and Granular Tabu Search (GTS_RC) for objective function f', after 30 s, 60 s, and 180 s of CPU time.

	n	m	30 s				60 s				180 s			
			VNS[a]	TS	GTS_P	GTS_RC	VNS[a]	TS	GTS_P	GTS_RC	VNS[a]	TS	GTS_P	GTS_RC
R1a	24	3	**3156.31**	3167.40	3236.14	3184.33	**3156.31**	3167.40	3236.14	3178.78	**3153.48**	3167.40	3236.40	3167.40
R2a	48	5	15159.46	7822.93	**6333.01**	6480.51	15309.90	6733.59	**6333.01**	6346.82	13024.52	6235.93	6230.41	**6130.38**
R3a	72	7	14953.32	**10096.50**	11655.80	10125.20	15570.66	**10014.00**	10734.70	10074.70	14949.00	9965.29	10457.60	**9962.26**
R5a	120	11	29181.62	21868.40	35131.80	**14139.30**	25212.58	16218.30	30012.00	**13844.90**	22042.38	13443.10	19620.40	**13280.50**
R9a	108	8	–[b]	26620.00	37416.00	**20982.10**	**16433.16**	20327.90	25350.30	20982.10	**15635.14**	17949.30	16971.70	20713.80
R10a	144	10	–	24453.20	36708.10	**20987.00**	–	21839.10	35509.00	**20072.40**	24771.76	20376.40	32056.90	**19325.70**
R1b	24	3	**2874.74**	2916.25	2907.18	2955.14	**2874.74**	2916.25	2907.18	2908.66	**2874.74**	2908.66	2907.18	2908.66
R2b	48	5	5141.81	**4986.50**	5861.48	5067.53	5052.65	**4986.50**	5207.70	5049.97	**4945.30**	4986.50	5139.69	5033.23
R5b	120	11	15660.02	22402.80	24552.50	**12755.80**	14914.36	18686.90	23301.60	**12558.30**	13579.22	13372.20	17726.70	**12210.90**
R6b	144	13	22736.96	30036.90	34153.10	**16292.00**	21880.40	26794.60	32110.90	**15865.70**	16563.44	20482.90	30695.90	**15385.50**
R7b	36	4	4615.86	**4387.16**	4587.97	4412.31	4599.47	**4365.98**	4587.97	4412.31	4575.33	**4365.98**	4587.97	4412.31
R9b	108	8	15977.08	17946.80	22661.70	**12839.40**	16408.28	14426.50	21443.70	**12500.60**	13178.64	12585.20	20284.20	**12461.20**
R10b	144	10	**21520.88**	47677.20	54526.70	21972.10	**19625.40**	27785.60	53184.50	20746.10	19646.40	18030.60	47751.30	**17393.90**

[a] Results provided to us by the authors of Parragh et al. (2010), not included in the original paper. Average values over 5 runs by using a Xeon computer with 2.67 Ghz.
[b] A ndash (–) indicates that no feasible solution has been found yet.

Table 3 (column VNS). See Table 4 for the exact values of the objective functions after 30 s, 60 s, and 180 s. Table 3 reports the results of VNS, GA, and our Tabu Search variants after long CPU times, depending on the instance between 3 and 50 min.

4.4. Comparison on $f''(s)$

Parragh et al. (2010) only reported values for the objective function f', but they gained these values by optimizing a reduced objective function f'' which includes all factors of $f'(s)$, except excess ride time $e(s)$:

$$f''(s) = \omega_1 \cdot c(s) + \omega_2 \cdot r(s) + \omega_3 \cdot l(s) + \omega_4 \cdot g(s) + \alpha \cdot k(s). \tag{25}$$

They obtained f' at the end of the optimization process by calculating $e(s)$ of the final solution and by adding it to f''. For this reason, we ran a second test set and compared VNS with the Tabu Search variants on objective function f''. We calculated the values for $f''(s)$ of the VNS using data reported in Parragh et al. (2010). Unfortunately, this could not be done for the GA, as the necessary data was not available. Again, we present the evolution of the objective function, this time $f''(s)$ (see Figs. 11 and 12, and Table 5), and the results after long CPU times, depending on the instance, between 3 and 50 min (see Table 6). We compare the results of VNS, GTS_RC, GTS_P, and TS.

4.5. Discussion

Comparing the results of the Tabu Search variants, it can be observed that almost always GTS_RC produces a good solution in a short amount of CPU time and dominates TS. Therefore, it can be said that the choice of using the reduced cost as an

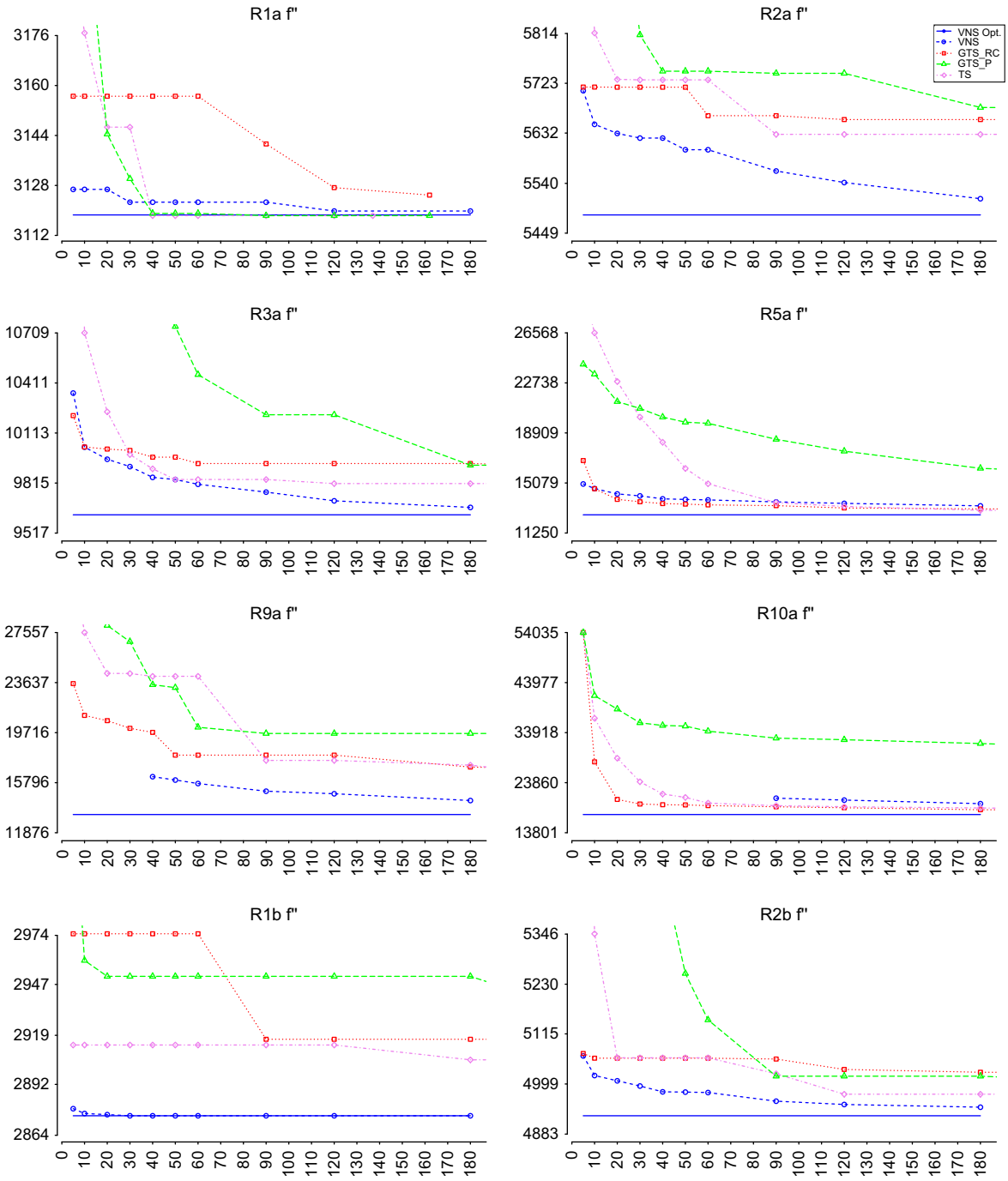


Fig. 11. Evolution of objective function f'' over time (x-axis: CPU time, y-axis: value of f'').

indicator for good moves proves to be efficient. In the long run, as expected, TS performs better, as it explores a larger search space. On the other hand, GTS_P performs poorly. See Figs. 9–12 for detailed results.

By looking at the results gained after 60 s of CPU time, we observe that GTS_RC for f' produces better results in most instances than the VNS (9 out of 13 instances, Table 4). Also when optimizing f'', GTS_RC performs strongly and provides better results than VNS for 6 out of 13 instances (Table 5).

Although not the main goal of this paper, we also provide results of the algorithms for long CPU times. We observe that when optimizing f' (Table 3) it is important to include excess ride time in the optimization process and that the results of GA

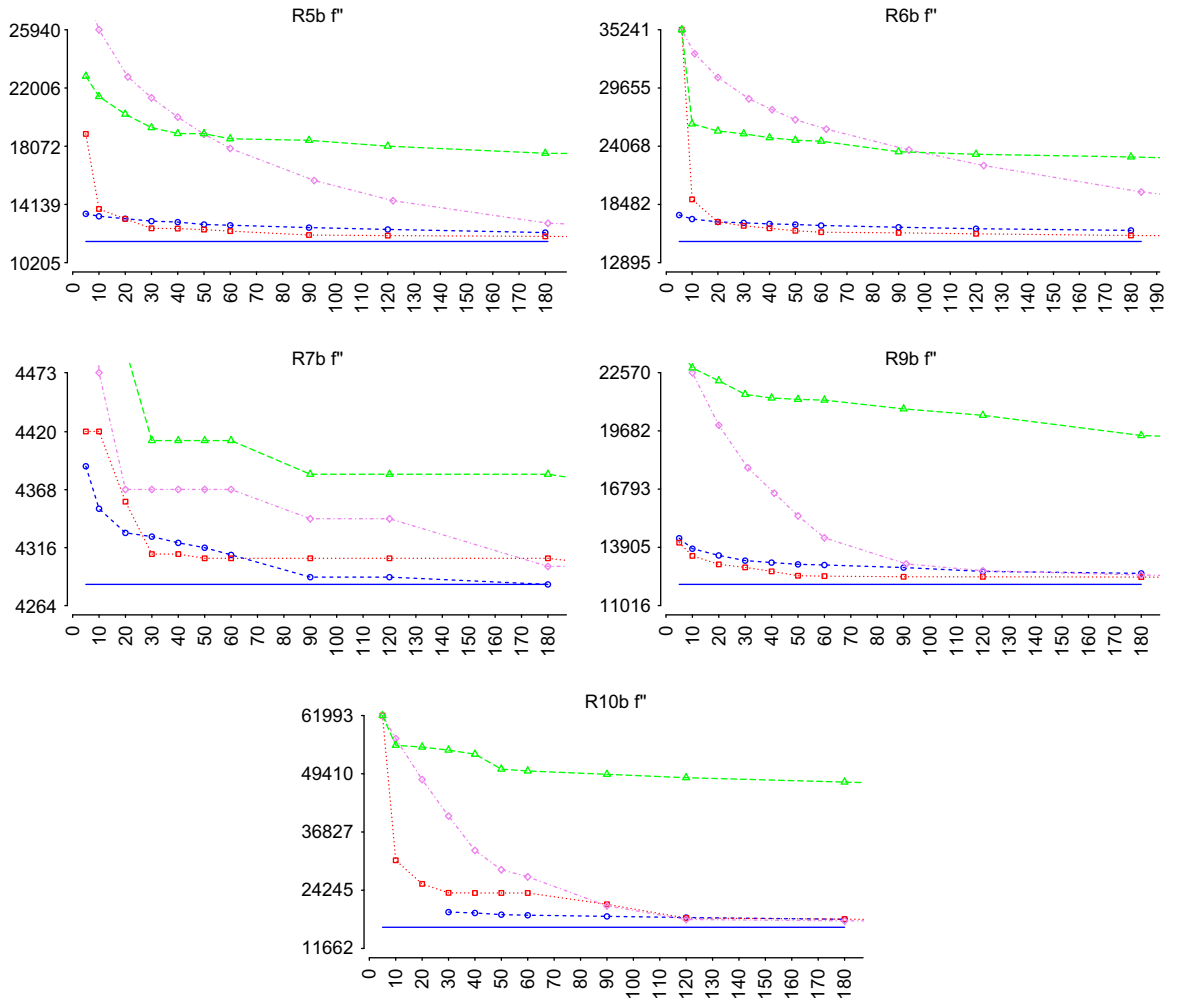


Fig. 12. Evolution of objective function f'' over time (x-axis: CPU time, y-axis: value of f'').

Table 5
Results for VNS (Parragh et al. (2010)), Tabu Search (TS) and Granular Tabu Search (GTS_RC) for objective functions f'', after 30 s, 60 s, and 180 s of CPU time.

	n	m	30s				60s				180s			
			VNS[a]	TS	GTS_P	GTS_RC	VNS[a]	TS	GTS_P	GTS_RC	VNS[a]	TS	GTS_P	GTS_RC
R1a	24	3	**3122.64**	3146.67	3130.16	3156.57	3122.64	**3118.33**	3119.04	3156.57	3119.81	**3118.33**	**3118.33**	3124.91
R2a	48	5	**5622.86**	5729.25	5811.48	5715.84	**5601.51**	5729.25	5745.08	5663.73	**5511.98**	5629.64	5678.76	5656.67
R3a	72	7	**9913.00**	9984.80	11930.20	10009.20	**9807.72**	9836.21	10460.70	9931.28	**9670.04**	9811.92	9921.22	9931.28
R5a	120	11	14092.86	20122.60	20784.20	**13642.40**	13783.50	15023.20	19646.10	**13400.90**	13328.56	**12990.10**	16200.70	13096.60
R9a	108	8	–[b]	24347.70	26847.30	**20056.50**	**15730.54**	24130.70	20138.10	17955.90	**14399.28**	17171.70	19652.10	17022.30
R10a	144	10	–	24049.10	35899.20	**19569.20**	–	19760.30	34212.20	**19251.60**	19646.14	18775.70	31734.00	**18419.00**
R1b	24	3	**2874.74**	2913.68	2951.45	2974.87	**2874.74**	2913.68	2951.45	2974.87	**2874.74**	2905.51	2951.45	2916.82
R2b	48	5	**4994.35**	5059.61	5724.77	5058.71	**4979.24**	5059.61	5147.37	5058.71	**4945.30**	4975.35	5017.05	5026.13
R5b	120	11	13011.70	21346.40	19325.70	**12523.60**	12732.84	17921.40	18574.80	**12335.30**	12238.24	12869.00	17595.30	**11980.00**
R6b	144	13	16717.76	28620.80	25259.50	**16427.80**	16462.50	25707.00	24549.40	**15817.10**	15992.08	19677.60	23036.00	**15503.20**
R7b	36	4	4325.91	4368.21	4412.07	**4310.21**	4309.52	4368.21	4412.07	**4306.44**	**4283.03**	4299.17	4381.86	4306.44
R9b	108	8	13247.00	17854.80	21491.50	**12910.20**	13030.08	14378.20	21207.50	**12476.50**	12612.86	12530.80	19452.90	**12429.90**
R10b	144	10	**19525.88**	40281.50	54522.30	23679.60	**18848.48**	27147.90	50012.40	23650.40	17968.44	**17631.40**	47617.00	17995.50

[a] Results provided to us by the authors of Parragh et al. (2010), not included in the original paper. Average values over 5 runs by using a Xeon computer with 2.67 Ghz.
[b] A ndash (–) indicates that no feasible solution has been found yet.

Table 6
Results for VNS (Parragh et al. (2010)), Tabu Search (TS) and Granular Tabu Search (GTS_RC) for objective function f'', CPU indicates CPU time in minutes of the optimization process. CPU times for VNS, TS, GTS_P, and GTS_RC are the same.

	n	m	VNS[a]	TS[b]	GTS_P[b]	GTS_RC[b]	CPU
R1a	24	3	3118.56	**3118.33**	**3118.33**	3124.91	2.70
R2a	48	5	**5546.55**	5623.06	5677.85	5656.67	5.16
R3a	72	6	**9632.47**	9775.78	9921.22	9859.95	6.38
R5a	120	11	**12642.77**	12770.50	13338.00	13002.80	13.93
R9a	108	8	**13301.65**	15419.70	19652.10	16789.50	33.53
R10a	144	10	**17459.16**	18295.50	17986.00	18069.20	40.27
R1b	24	3	**2875.36**	2905.51	2932.89	2916.82	3.78
R2b	48	5	**4929.08**	4975.35	5001.14	5020.86	8.29
R5b	120	11	**11635.79**	11731.60	14045.30	11896.10	23.19
R6b	144	13	**14927.10**	14963.80	18138.40	15127.50	26.39
R7b	36	4	4297.89	4299.17	4360.23	**4290.11**	4.49
R9b	108	8	12067.00	**12021.80**	12429.70	12243.10	30.32
R10b	144	10	**16238.27**	16641.80	17786.30	16753.30	51.81

[a] Pentium D computer 3.2 GHz.
[b] Bi AMD Opteron Dual Core computer with 3.2 GHz.

are not competitive. More in detail, for f', TS dominates GTS_RC, which dominates VNS for most instances. TS provides better solutions than VNS for 10 out of 13 instances. The GA is outperformed by all other algorithms. When optimizing f'' (Table 6), VNS performs very strongly. Still, TS and GTS_RC provide better solutions than VNS for 2 and 1 instances, respectively, and the results of TS are close to the results of VNS (which are average values over 5 runs).

5. Conclusions

In this paper, a fast algorithm for solving the static *Dial-a-Ride Problem* (DARP) has been proposed. The *Granular Tabu Search* method has been applied for the first time to solve this kind of problem. A major characteristic of the proposed algorithm is how the granularity has been produced: we reduced the original problem to an assignment problem and we exploited the reduced costs to build clusters of close requests. The computational results prove that our algorithm performs well in comparison to other solution methods and that it is able to provide good solutions in the first three minutes of CPU time of the optimization process. Directions for future research include: study of the applicability of the technique presented in this paper to other routing problems and the on-line scenario, as well as the analysis of the time-dependent case (traffic information).

Acknowledgments

The authors thank Sophie N. Parragh for running additional tests and providing them with further experimental results. They would also like to thank two anonymous reviewers for their valuable comments.

References

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. European Journal of Operational Research 202 (1), 8–15.
Bergvinsdottir, K.B., 2004. The Genetic Algorithm for Solving the Dial-A-Ride Problem. Master thesis, Technical University of Denmark.
Wolfler Calvo, R., Colorni, A., 2006. An effective and fast heuristic for the Dial-a-Ride problem 4 or 5 (1), 61–73.
Cordeau, J.-F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Operations Research 54 (3), 573–586.
Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B 37 (6), 579–594.
Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Annals of Operations Research 153 (1), 29–46.
Desrosiers, J., Dumas, Y., Soumis, F., 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences 6, 301–325.
Diana, M., Dessouky, M.M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transportation Research Part B 38 (6), 539–557.
Diana, M., Dessouky, M.M., Xia, N., 2006. A model for the fleet sizing of demand responsive transportation services with time windows. Transportation Research Part B 40 (8), 651–666.
Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal Of Operational Research 54 (1), 7–22.
Gendreau, M., 2003. An introduction to Tabu Search. Handbook of Metaheuristics 57, 37–54.
Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13 (5), 533–549.
Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M.M., Villeneuve, D., 1995. A request clustering algorithm for door-to-door handicapped transportation. Transportation Science 29 (1), 63–78.
Jaw, J.-J., Odoni, A., Psaraftis, H.N., Wilson, N.H., 1986. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B 2 (3), 243–257.
Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2006. Solving the Dial-a-Ride problem using genetic algorithms. Journal of the Operational Research Society 58 (10), 1321–1331.
Labadie, N., Mansini, R., Melechovský, J., Wolfler Calvo, R., 2012. The team orienteering problem with time windows: an LP-based granular variable neighborhood search. European Journal of Operational Research 220 (1), 15–27.

Lu, Q., Dessouky, M.M., 2006. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. European Journal Of Operational Research 175 (2), 672–687.
Madsen, O.B.G., Ravn, H.F., 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), 193–208.
Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer programming formulations and traveling salesman problems. Journal of the ACM 7 (4), 326–329.
Paquette, J., Cordeau, J.-F., Laporte, G., Pascoal, M.M.B., 2013. Combining multicriteria analysis and tabu search for dial-a-ride problems. Transportation Research Part B 52, 1–16.
Parragh, S.N., Schmid, V., 2012. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Computers & Operations Research 40 (1), 490–497.
Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008. A survey on pickup and delivery problems part II: transportation between pickup and delivery locations. Journal für Betriebswirtschaft 2 (58), 81–117.
Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 37 (6), 1129–1138.
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.W.P., 1985. Local search in routing problems with time windows. Annals of Operations Research 4 (1), 285–305.
Sexton, T.R., Choi, Y.M., 1986. Pickup and delivery of partial loads with soft time windows. American Journal of Mathematical and Management Sciences 6 (3-4), 369–398.
Taillard, E., 1993. Parallel iterative search methods for vehicle routing problems. Networks 23 (8), 661–673.
Toth, P., Vigo, D., 1997. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science 31 (1), 60–71.
Toth, P., Vigo, D., 2003. The Granular Tabu Search and its application to the vehicle-routing problem. INFORMS Journal on Computing 15 (4), 333–346.