Asia-Pacific Journal of Operational Research
Vol. 30, No. 1 (2013) 1250046 (17 pages)

DOI: 10.1142/S0217595912500467

A HYBRID GREEDY RANDOMIZED ADAPTIVE SEARCH HEURISTIC TO SOLVE THE DIAL-A-RIDE PROBLEM

FRANCESCA GUERRIERO
Dipartimento di Ingegneria Meccanica, Energetica e Gestionale, Università della Calabria, Via P. Bucci 41C, 87030 Rende (Cosenza), Italy
guerriero@deis.unical.it

MARIA ELENA BRUNI
Dipartimento di Ingegneria Meccanica, Energetica e Gestionale, Università della Calabria, Via P. Bucci 41C, 87030 Rende (Cosenza), Italy
mebruni@deis.unical.it

FRANCESCA GRECO
Dipartimento di Ingegneria Meccanica, Energetica e Gestionale, Università della Calabria, Via P. Bucci 41C, 87030 Rende (Cosenza), Italy
f.greco@netps.it

Published 6 December 2012

This paper presents a hybrid metaheuristic for solving the static dial-a-ride problem with heterogeneous vehicles and fixed costs. The hybridization combines a reactive greedy randomized adaptive search, used as outer scheme, with a tabu search heuristic in the local search phase. The algorithm is evaluated on well-known instances taken from the literature and on a set of randomly generated test problems, varying in the number of customers. Extensive computational results show the effectiveness of the hybrid approach in terms of trade-off between solution quality and computational time.

Keywords: Dial-a-ride; metaheuristics; tabu search; greedy randomized adaptive search.

1. Introduction

The dial-a-ride problem (DARP) consists of determining the best routing schedule for a fleet of vehicles, which minimizes overall transportation costs, while maintaining a prescribed level of customer service. Service level constraints may involve customers ride times, deviations from desired departure or arrival times, route durations and other application-dependent operational constraints. The challenge is to balance transportation costs and high level of service, which are in general conflicting factors. The DARP arises in a variety of practical fields, ranging from public transport provided by local government to specialized transportation systems of children, disabled or elderly people (see Borndörfer *et al.*, 1997). Depending on the

specific context, dial-a-ride services may operate according to either a static or a dynamic mode. In the first case, all transportation requests are known beforehand, while in the second case vehicle routes are adjusted in real-time to meet incoming requests gradually revealed throughout the day. Another interesting variant, that we study in this paper, is constituted by the presence of a heterogeneous fleet of vehicles which includes standard vehicles and special vehicles with limited capacity. In the foregoing, we shall concentrate on this class of DARP problems, referring the reader to the exhaustive reviews of Cordeau and Laporte (2003b, 2007), Berbeglia *et al.* (2007), and Bräysy and Gendreau (2005) for a detailed review of homogeneous DARP models and methods.

The heterogeneous version of the DARP has been investigated mostly with reference to health care applications. Toth and Vigo (1997), for instance, devised a parallel insertion heuristic and a tabu thresholding algorithm for the handicapped person transportation problem, considering two modes of transportation (seated and in a wheelchair) and several types of vehicles. Melachrinoudis *et al.* (2007) developed a multiple objective DARP for client transportation considering the conflicting goals of minimizing transportation costs and customer inconvenience. Rekiek *et al.* (2006) applied a grouping genetic algorithm to find near optimal routes for a real life handicapped persons transportation problem in Belgium. Parragh *et al.* (2012) introduced patient transportation specific constraints into a set-partitioning formulation of the problem whose relaxation is solved by a column generation algorithm integrated within a tailored Variable Neighborhood Search heuristic into a collaborative framework. Parragh (2011) extended all valid inequalities proposed for the standard DARP to the heterogeneous case, embedding them into a branch and-cut algorithm tested on problems with a number of requests up to 48.

In a dynamic environment, Beaudry *et al.* (2009) analyzed and solved with a two-phase heuristic a patient transportation problem arising in large hospitals, whereas Hanne *et al.* (2009) developed a computer based planning system supporting all phases of the transportation flow of a large German hospital.

Despite this flourishing literature, a few contributions consider the heterogeneous multi-vehicle static DARP with fixed costs (HDARPFC), representing the problem addressed in this paper. To the best of our knowledge, the only works dealing with the HDARPFC are the papers of Xiang *et al.* (2006) and Wong and Bell (2006). In particular, Xiang *et al.* (2006) considered the minimization of a combination of vehicle fixed costs and other variables variable costs, under several operating constraints. The heuristic proposed, based on local search strategies along with diversification and intensification strategies, has been shown to be effective for instances with up to 2,000 requests.

Wong and Bell (2006) considered a version of the DARP with multidimensional capacity constraints. The authors proposed a parallel insertion procedure followed by an optional post-optimization phase evaluated on randomly generated instances.

In this paper, we present a hybrid metaheuristic approach (Festa and Resende, 2009c) for the solution of HDARPFC. The proposed approach combines two well

known metaheuristics, the greedy randomized adaptive search (GRASP) (Festa and Resende, 2009a, 2009b), which acts at a strategic level as a diversification mechanism and the tabu search (Glover and Laguna, 1997) used as a tactical tool to efficiently explore the solution space.

The interest on the hybridization of these two metaheuristics lies on the one hand, on the success of tabu search in heuristically solving the DARP (Aldaihania and Dessouky, 2003; Cordeau and Laporte, 2003a; Nanry and Barnes, 2000; Toth and Vigo, 1997), and on the other hand on the strong diversification mechanism provided by the GRASP.

The remainder of the paper is organized as follows. In Sec. 2, the problem under investigation is formally defined, while in Sec. 3 a detailed description of the metaheuristic is provided. In Sec. 4, the computational results obtained on various data sets are presented and final conclusions are drawn in Sec. 5.

2. The Dial-a-Ride Problem

The HDARPFC involves scheduling a heterogeneous fleet of $(m + e)$ vehicles, partitioned into two subsets, based at a single depot. The ordinary m vehicles have capacity of Q, whereas the e extra vehicles have capacity one and higher costs. Each of the m vehicles has associated a routing cost $c_k,\ k = 1, \ldots, m$ for each unit distance between different locations. If extra vehicles are utilized, besides routing costs $\hat{c}_k,\ k = m + 1, \ldots, e$ also fixed costs $Y_k,\ k = m + 1, \ldots, e$ must be payed.

A set of n customer requests, indexed by subscript i, is known in advance. Each request specifies a pick-up location i and a delivery location $n + i$. Pick-up and drop off operations entail a service duration time. Typically, the same customer provides to the system both an outbound request from home to a destination and an inbound request for the return trip. We define EPT_i and LPT_i as the earliest and latest pickup times for request i, and EDT_i and LDT_i as the earliest and latest delivery times for request i. Each customer i is able to specify a desired minimum pick-up time for inbound requests EPT_i or a desired maximum delivery time for outbound requests LDT_{i+n}.

The problem is to find a set of routes starting and ending at the depot, such that all the customer requests and the following set of constraints are satisfied.

- Every route starts and ends at the depot.
- For every request i, the same vehicle visits the delivery location $n + i$ later than the pick-up location i.
- The load of any vehicle k does not exceed its capacity.
- The total duration of route k is bounded above by maximum route duration value T_k.
- The service is delivered within the user specified time windows.
- The ride time of any user does not exceed a threshold value L.

The objective function consists in minimizing the total transportation cost and the fixed cost of extra vehicles used in the solution. Given a solution s, its cost is

calculated as $Z(s) = c_{\text{fix}}(s) + c_{\text{rout}}(s)$, where c_{fix} is the fixed cost related to the possible use of extra vehicles and c_{rout} is the total routing cost. We notice that, when the use of extra vehicles is not allowed, the problem described is exactly the one considered in Cordeau and Laporte (2003a).

3. The Hybrid Reactive GRASP

The GRASP, first introduced by Feo and Bard (1989), is an iterative optimization procedure based on a randomized sampling of the search space. A typical iteration consists of two main phases: A *construction* phase, based on a randomized heuristic, and a *local search* phase, that improves the solution found. Before presenting the overall scheme of the hybrid reactive GRASP (HRGRASP), we present the main ingredients of the search procedure.

3.1. *The greedy randomized construction phase*

During the constructive phase, the algorithm uses a randomized greedy heuristic to iteratively construct an initial solution to the problem, one element at a time, according to the description that follows.

A trivial nonfeasible solution is initially built by creating routes containing only the depot. Requests are, then, inserted into routes, as long as the solution is maintained feasible. At each iteration of the construction phase, one request is selected from a list of candidate requests (called CL in the sequel) in which only feasible insertions are retained, that is $\text{CL} = \{(i,k) : \text{request } i \text{ can be feasibly assigned to vehicle } k, \forall\, k = 1, \ldots, m\}$. Requests in the CL are prioritized according to their insertion cost $\phi_{(i,k)}$ determined as follows. The best insertion position is determined for the critical vertex (according to Cordeau and Laporte (2003a)). Then, holding the critical vertex in its best position, the best insertion position is determined for the noncritical vertex and the insertion cost calculated. Only the best elements in the CL are considered and stored in a restricted list of candidates called RCL. Let us denote with $\phi_{\min} = \min_{(i,k)\in \text{CL}} \phi_{(i,k)}$ and $\phi_{\max} = \max_{(i,k)\in \text{CL}} \phi_{(i,k)}$, respectively, the minimum and the maximum value attained by the candidates in the CL. The RCL can be then defined as follows:

$$\text{RCL} = \{(i,k) \in \text{CL} \,|\, \phi_{(i,k)} \leq \phi_{\min} + \alpha(\phi_{\max} - \phi_{\min})\}$$

for a given value of $\alpha \in [0,1]$. The choice of the pair (i,k) to be inserted in the partial solution is random among the requests included in RCL. Once the element (i,k) is drawn from the RCL, the corresponding insertion is executed and the request i is included in the set S of scheduled requests. Since not all requests will be feasibly scheduled, the remaining unassigned requests are assigned to randomly selected vehicles, by inserting the associated vertices i and $i+n$ sequentially at the end of the partially constructed routes of extra vehicles. A formal description of the construction phase is reported in Algorithm 1.

3.2. *The local search phase*

Once a solution has been determined by invoking Algorithm 1, a local search phase is performed in an attempt to improve it. Given its outstanding success in the solution of DARPs, we have embedded the TS as local search into our framework.

Algorithm 1. Randomized construction phase

Require: a value of α

Initialization

Build m trivial routes containing only the depot

Set $S = \{\}, \text{CL} = \{\}$

While $(|S| < n)$ do

Construction of the CL:

- For each request $i \notin S$ do
  - For each route $k = 1, \ldots m$ do
  - If i can be feasibly inserted in k store the pair (i, k) in the CL
  - End for
- End for

If CL $= \{\}$ then break

Evaluation of the greedy function:

- For each pair $(i, k) \in$ CL evaluate the impact of inserting vertices i and $i + n$ (corresponding to pick-up and delivery locations) in route k determining the best insertion position first for the critical vertex and then for the noncritical vertex. Store in $\phi_{(i,k)}$ the cost of such insertion

Construction of the RCL:

- Order the elements on the CL for increasing insertion costs
- Let $\phi_{\min} = \min_{(i,k)\in \text{CL}} \ \phi_{(i,k)}$ $\phi_{\max} = \max_{(i,k)\in \text{CL}} \phi_{(i,k)}$
- RCL $= \{(i, k) \in \text{CL} \,|\, \phi_{(i,k)} \leq \phi_{\min} + \alpha(\phi_{\max} - \phi_{\min})$ for a given value of α

Partial route construction

- Remove randomly one pair (i, k) from the RCL
- Perform the best insertion for i in the route k; $S := S \cup \{i\}$

End while

Insertion of the remaining requests into extra vehicle routes:

While $|S| < n$ do

- Create a new extra vehicle route containing only the depot
- Insert randomly chosen unassigned requests to the current extra vehicle route by inserting the associated vertices sequentially at the end of the partially constructed route until no feasible assignment is possible

End while

Return the solution $s(\alpha)$

The state-of-the-art TS heuristic of Cordeau and Laporte (2003a), initially designed for the standard DARP, has been adapted and applied to HDARPFC. Generally speaking, the TS systematically explores the solution space by moving at each iteration from the current solution s to the best one amongst its neighbors. The neighborhood $N(s)$ of a solution s, is composed of all solutions that can be obtained from s by removing a request i from the route k and reinserting it in another route $k' \neq k$. Also infeasible solutions are allowed, but not preferred. In fact, solutions are evaluated using an evaluation function $f(s)$ in which, besides the solution cost, load violations $q(s)$, duration violations $d(s)$, time window violations $w(s)$, and ride time violations $t(s)$ are penalized with penalty factors α, β, γ, and τ, respectively. These positive parameters are dynamically adjusted at each iteration by a factor $1+\delta$, with $\delta > 0$. To avoid cycling, solutions possessing some attributes of recently explored solutions are temporarily declared tabu (i.e., forbidden) and the number of iterations an attribute remains tabu is denoted with θ. A diversification phase is implemented by holding information about the number of times $\rho_{i,k}$ that an attribute has been added to the solution during the search. Based on these values and on a parameter λ used to control the intensity of the diversification, a penalty $p(s) = \lambda c(s)\sqrt{nm\rho_{i,k}}$ is added to $f(s)$. We notice that the solution cost and the problem size are also accounted in $p(s)$ through the scaling factor $c(s)\sqrt{nm}$. Every 10 iterations, the values of δ, θ, and λ are updated.

During the search, an intensification mechanism is provided every κ iterations or whenever a new incumbent is found performing intra-route exchange, i.e., a request is removed from its current position and reinserted in the best possible position within the same route. The whole process is repeated for η iterations.

We adapted this state-of-the-art method in the following way. First, a new evaluation function that accommodates the characteristics of HDARPFC has been devised. Besides routing costs and constraint violations, also the total fixed costs for the use of extra vehicles has to be considered. Henceforth $f(s) = c_{\text{fix}}(s) + c_{\text{rout}}(s) + \alpha q(s) + \beta d(s) + \gamma w(s) + \tau t(s)$. Second the penalty function has been modified as follows: $p(s) = \lambda c(s)\sqrt{n(m+e)\rho_{i,k}}$. An overall description of the TS is reported below.

Algorithm 2. Tabu search

Require: an initial solution s

For $t = 1, \ldots, \eta$ do

- Identify the neighborhood set $N(s)$
- Identify the tabu set $T(s)$ forbidding for the next θ iterations the attributes of recently explored solutions
- Choose the best $s \in \{N(s) - T(s)\}$ with respect to the evaluation function $f(s) + p(s) = c_{\text{fix}}(s) + c_{\text{rout}}(s) + \alpha q(s) + \beta d(s) + \gamma w(s) + \tau t(s) + \lambda c(s)\sqrt{n(m+e)\rho_{i,k}}$
- If s improves the previous best known solution then set $s_{\text{tabu}} := s$
- If $t \mod \kappa = 0$ or a new best incumbent has been found then perform intra-route optimization

- Update parameters α, β, γ, and τ
- If $t \mod 10 = 0$ then update parameters δ, λ, and θ

End for

Return s_{tabu}

3.3. *Reactive GRASP algorithm*

Given that the main ingredients of the heuristic have been specified, now we are ready to present a formal description of the hybrid randomized GRASP (HRGRASP). The whole process is summarized in Algorithm 3, in which $N_{\max}$ is the number of the HRGRASP iterations, g the iteration counter, $Z(s)$ the cost of solution s and $s_{\min}$ and $Z_{\min}$ the best solution found and its associated cost.

In the spirit of a reactive mechanism (Feo and Resende, 1995; Prais and Ribeiro, 2000; Resende and Ribeiro, 2003), we let the algorithm find the best value of α in a small discrete set of L allowed values $\Psi = \{\alpha_1, \ldots, \alpha_L\}$. Each value of α_l, $l = 1, \ldots, L$ in the set Ψ can be selected with a probability $p(\alpha_l)$. Usually starting from the values $p(\alpha_l) = 1/L$ for $l = 1, \ldots, L$, the probabilities are updated every N_a iterations proportionally to the value of $Z_{\min}/Av_l$, where Av_l the average value of the solutions obtained with α_l.

We remark that the value of the parameter α controls the size of RCL. In fact, a value of $\alpha = 0$ corresponds to a pure greedy construction phase (and leads to a deterministic algorithm), whereas if the value of α is too high, well-diversified but low-quality solutions are obtained.

Algorithm 3. Hybrid Reactive GRASP

Set $Z_{\min} := \infty$ and select a value for L

Generate randomly the values α_l $l = 1, \ldots, L$

Set $Av_l = 0$, $p(\alpha_l) = 1/L$, $l = 1, \ldots, L$

For $g = 1, \ldots, N_{\max}$ do

- Pick randomly $\alpha^* \in \{\alpha_1, \ldots, \alpha_L\}$ according to probabilities $p(\alpha_l)$, $l = 1 \ldots, L$
- Invoke Algorithm 1 with $\alpha = \alpha^*$ to compute the solution $s(\alpha^*)$
- Improve $s(\alpha^*)$ using Algorithm 2: let s_{tabu} the best solution found
- If $Z(s_{\text{tabu}}) < Z_{\min}$ then $S_{\min} := s_{\text{tabu}}; Z_{\min} := Z(s_{\text{tabu}})$
- If $g \mod N_a = 0$ then update Av_l and $p(\alpha_l) = \frac{Z_{\min}/Av_l}{(\sum_{l=1,\ldots,L} Z_{\min}/Av_l)}$

End for

Return $Z_{\min}$

Although original, the use of the GRASP in the context of vehicle routing problem is not completely new. In fact, Kontoravdis and Bard (1995) proposed a two-phase GRASP for the vehicle routing problem with time windows in which the construction procedure first initializes a number of routes by selecting seed

customers and afterwards, for every unrouted customer, finds their best feasible insertion location in each route on the basis of specific penalty value and a greedy criterion. Then a local search, in which each route is considered for elimination and a 2-Opt exchange procedure are applied to the best feasible solution found.

To the best of our knowledge, this is the first contribution in which a hybrid reactive GRASP is used in the solution of the HDARPFC.

4. Test Instances

The efficiency of HRGRASP has been tested on three data sets. The first two are based on existing data sets from the literature, enriched with the introduction of heterogeneous vehicles and the associated fixed costs, while the latter one has been randomly generated by the authors. In the remainder of the section, the characteristics of the two test beds will be described.

4.1. *Test instances from the literature*

The first test bed used in the computational experiments is based on an existing data set proposed by Cordeau and Laporte (2003a). It consists of two subsets of problems: a set of 20 randomly generated instances, containing between 24 and 144 requests and a set of six real life instances from a door-to-door transport service in Denmark. The first data set contains instances with service time in each location equal to 10, maximum route duration equal to 480, vehicle capacity equal to six, maximum ride time equal to 90 and planning horizon equal to 1,440. Two groups of customer-specific time windows are considered: the first one has narrow time windows, while the second one has wide time windows. As far as the real life data set is concerned, two basic instances were considered (D1 and D2). Instance D1 contains 200 requests and D2 (a superset of D1) contains 295 requests. For each instance, three different time window widths were considered.

In order to account for the presence of a heterogeneous vehicle fleet, the two data sets have been perturbed in such a way that the number of standard available vehicles is 30% lower than the number of vehicles in the original instance and extra vehicles have been added instead. The e extra vehicle have capacity 1, a routing cost $\hat{c}_k$ doubled in comparison to the standard vehicles routing cost c_k and a fixed cost equal to 500. We shall highlight these perturbed test problems with *.

4.2. *Randomly generated test problems*

The second test bed, available from the authors upon request, has been randomly generated with the following settings.

- For every instance, the positions of the depot, pick-up sites and delivery sites are uniformly generated at random over the service area in a 20 km square Euclidean plane.

Table 1. Characteristic of the random test problems.

Test problem	n	m	e
R1	500	35	5
R2	600	38	5
R3	700	42	5
R4	800	50	5
R5	1,000	70	5

- The travel time and the travel cost between any two coordinates is taken to be equal to the direct distance between them divided by the average vehicle speed, which is assumed to be 20 km/h here.
- The fixed costs of extra vehicles is assumed to be 500.
- The service time for every request is taken equal to 3 minutes.
- Vehicle capacity is equal to eight for the m ordinary vehicles, whereas for the e extra vehicles is equal to one as common for taxis.
- The maximum ride time is equal to 120 minutes in all instances, the maximum route duration is set to 720 minutes.
- The planning horizon, T, has been set in these experiments to 1,440, that is, the number of minutes in one day.
- The randomly generated instances contain between 500 and 1,000 requests assumed to be uniformly distributed over the service area.
- Two groups of customer-specific time windows are constructed in the data sets. The first one has wide time windows, while the second one has narrow time windows. Wide time window widths have been set to 30 minutes for the instances with up to 800 requests, whereas for the largest instance involving 1,000 requests the time window width has been set to 60 minutes. This choice has been motivated by the high computational time required for the solution of the problem. Narrow time window widths have been considered by randomly drawing a number in the range [10,30) and [10,60) for the instances with up to 800 requests and for the largest instance, respectively. Time windows have been constructed by choosing an uniform random number ur in the interval [60,480] and then considering the interval [ur,ur+time window width].

Table 1 reports the characteristics of the random test problem considered. In the following, we shall refer to them by using the notation test.time windows structure. Thus, problem R1.I is problem R1 with time windows drawn in the first interval.

5. Computational Results

The objective of this section is to assess whether the crafted combination of different metaheuristics can produce higher solution quality than individual metaheuristic itself. Therefore, we have analyzed the performance of our HRGRASP with the TS

embedded in the local search phase, with respect to the performance of the TS applied alone.

The results have been obtained by executing, on a personal computer with 2G RAM, each HRGRASP and TS configuration with five different random seeds, and therefore, 100 runs for each test problem and each algorithm. The algorithms were coded using Visual C++ language.

As far as HRGRASP parameters are concerned, they were configured as follows: $N_a = 15$, $\Psi = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Several HRGRASP configurations have been tested by considering two possible values for $N_{\max}$, namely $N_{\max} = 30$ and $N_{\max} = 50$ and by varying the number of local search tabu iterations within each HRGRASP iteration in the set $[100, 200, 300, 400, 500, 600, 700, 800, 900, 1{,}000]$.

The number of iterations η for the TS, when applied alone, has been considered equal to the total number of iterations performed by HRGRASP (i.e., for the HRGRASP configuration 30-100, $\eta = 3{,}000$). The frequency of intra-route reoptimizations κ has been set to 10 iterations, δ, λ, θ have been set to 0.5, 0.015 and $7.5 \log_{10}(n/2)$ and their values were updated every 10 tabu iterations, according to uniform distributions in the intervals $[0, 0.5]$, $[0, 0.015]$, and $[0, 7.5 \log_{10}(n/2)]$, respectively.

Table 2 summarizes the results obtained on the 20 randomly generated instances of Cordeau and Laporte (2003a) modified to account for fixed costs and heterogeneity. The first two columns show the average cost and the best cost identified, over all the experiments, for both HRGRASP and TS. The percentage deviation from the cost of the best solution identified with the same algorithm (i.e., the relative percentage deviation) and the percentage deviation from the cost of the absolute best solution identified during all the experiments (i.e., the absolute percentage deviation) are reported in columns three and four. The last column reports the CPU time in minutes (averaged over the five different seeds) spent to find the best solution. All entries marked in bold correspond to the best values identified in each instance.

Several comments are possible on the basis of these data. HRGRASP outperforms TS algorithm in terms of average solution cost in 15 out of 20 problem instances. For these test problems, the cost reduction is on average around 23% with peaks up to about 60% for the test problem R9a (the average cost identified by the HRGRASP is 1061.75 against 2583.25 of the TS).

When considering the best cost identified over the runs performed, the average gap is equal to 36%. For the test problem R9a the best HRGRASP cost is on average 62.4% lower than the TS best cost.

We further observe that the average CPU time of the HRGRASP over the whole modified test bed is slightly less (3.95%) than the TS CPU time. We also notice that HRGRASP becomes increasingly superior in solution quality to TS, as time windows constraints become more restrictive, namely for problems R1a*–R10a* with narrow time windows.

Table 2. Computational results for the modified test bed of Cordeau and Laporte (2003a).

Test	Avg. cost		Best cost		Relative deviation		Absolute deviation		CPU time	
	HRGRASP	TS	HRGRASP	TS	HRGRASP	TS	HRGRASP	TS	HRGRASP	TS
R1a*	**193.50**	**193.50**	**193.50**	**193.50**	0.00	0.00	0.00	0.00	3.58	3.56
R2a*	**495.23**	852.94	**467.12**	849.12	6.02	0.45	6.02	82.60	11.72	15.04
R3a*	**921.47**	1231.71	**872.19**	1182.60	5.65	4.15	5.65	41.22	19.17	27.75
R4a*	635.29	**627.18**	630.85	**619.41**	0.70	1.25	2.56	1.25	67.65	69.29
R5a*	**1194.30**	1248.01	**1078.18**	1235.75	10.77	0.99	10.77	15.75	64.22	75.19
R6a*	877.44	**844.89**	871.88	**840.36**	0.64	0.54	4.41	0.54	138.05	123.56
R7a*	**464.70**	942.19	**441.62**	922.68	5.23	2.11	5.23	113.35	4.46	5.50
R8a*	**711.34**	1166.60	**637.52**	1138.85	11.58	2.44	11.58	82.99	20.99	26.91
R9a*	**1061.75**	2583.25	**959.61**	2552.37	10.64	1.21	10.64	169.20	42.11	45.39
R10a*	**1379.55**	2662.03	**1344.06**	2647.22	2.64	0.56	2.64	98.06	67.70	76.84
R1b*	**170.44**	170.57	**170.42**	170.48	0.01	0.05	0.01	0.09	4.34	4.05
R2b*	**323.62**	323.71	322.33	**318.23**	0.40	1.72	1.69	1.72	20.17	19.22
R3b*	**870.69**	1127.76	**822.73**	1121.51	5.83	0.56	5.83	37.08	23.07	32.30
R4b*	625.09	**577.29**	614.76	**576.71**	1.68	0.10	8.39	0.10	59.20	51.41
R5b*	624.25	**598.56**	619.41	**593.96**	0.78	0.77	5.10	0.77	120.46	108.04
R6b*	783.32	**763.87**	776.60	**756.30**	0.87	1.00	3.57	1.00	149.64	114.59
R7b*	**382.50**	827.88	**324.38**	823.74	17.92	0.50	17.92	155.22	6.66	6.60
R8b*	**791.07**	1030.89	**741.96**	1028.64	6.62	0.22	6.62	38.94	26.40	29.24
R9b*	**1105.69**	1817.37	**1051.78**	1801.91	5.13	0.86	5.13	72.79	39.93	48.32
R10b*	**1306.94**	2587.53	**1221.06**	2575.67	7.03	0.46	7.03	111.91	69.12	83.78

Note: Relative deviation: $100(\text{cost} - \text{bestcost}_{\text{HRGRASP}})/\text{bestcost}_{\text{HRGRASP}}$.
Absolute deviation: $100[\text{cost} - \min(\text{bestcost}_{\text{HRGRASP}}, \text{bestcost}_{\text{TS}})]/\min(\text{bestcost}_{\text{HRGRASP}}, \text{bestcost}_{\text{TS}})$.

In column **Relative deviation** is reported the deviation of the solution values of the five random runs from the best value out of these runs. This variability of the solutions arises, in part, from the choice of the random seeds, and in part from the randomness involved in the construction of an initial solution for both algorithms. This information is important to assess how robust the heuristics are in terms of the solution consistency.

From the observations of the results we notice that the relative deviation of the TS is consistently smaller than that of HRGRASP. This implies that TS average costs fluctuate only slightly around the best solution identified by the same algorithm. The higher variability of solution costs of HRGRASP clearly depends on its multi-start nature which provides a randomized way of strategically sampling the solution space. The column **Absolute deviation** reports the deviation of the solution values of the five random runs from the best values (obtained either by TS or by HRGRASP). We observe that even if HRGRASP might fails to obtain the best result, the average cost little deviates form the best cost achieved (the maximum deviation is in fact 17.92). Looking at the figures concerning the TS, we may notice that it is characterized by a wider variability (on average 51.23) with high peaks. Summarizing these results we can conclude that the strong random diversification mechanism of HRGRASP can lead to dramatic improvements in the objective function at a cost of a higher solution variability. We observe also that when HRGRASP improves, it achieves a remarkable gain in the objective function, within a slightly lower computational time.

In the second set of experiments, we have compared the HRGRASP and the TS on the real life instances of Cordeau and Laporte. The results, reported in Table 3 have been collected considering one HRGRASP configuration (30–1,000). The number of TS iterations has been accordingly set to $3 * 10^4$. Only the best solution value **Cost** and the time **CPU** expressed in minutes are shown in Table 3. We can notice that in all but one instances, HRGRASP outperforms TS with an average improvement in the objective function of (3.05%). For the instance D1c*, HRGRASP deteriorates the solution of the TS by 0.2%. We remark in this case that the time spent by HRGRASP to find the solution is 7.17% less than the TS CPU time.

Table 3. Results for the modified real life instances of Cordeau and Laporte (2003a).

Test	TS		HRGRASP	
	CPU	Cost	CPU	Cost
D1a*	103.4	12141.79	74.7	**12071.32**
D1b*	155.26	4881.48	151.05	**4828.27**
D1c*	192.53	**3817.1**	178.72	3824.70
D2a*	110.52	17134.2	155.07	**16903.32**
D2b*	160.97	9970.68	197.5	**9968.2**
D2c*	373.93	7112.24	380.3	**7111.3**

Table 4. Results for the random test problems.

Test	TS		HRGRASP	
	CPU	Cost	CPU	Cost
R1.I	373.97	9050.51	332.05	**8921.05**
R1.II	359.40	10558.52	271.27	**10449.49**
R2.I	511.06	11788.15	463.67	**11013.97**
R2.II	429.08	16414.30	347.40	**15735.94**
R3.I	844.38	13410.00	758.20	**12304.26**
R3.II	773.97	15318.11	633.97	**14667.79**
R4.I	882.87	14038.49	838.57	**13356.45**
R4.II	836.75	**15684.57**	671.02	15948.62
R5.I	1380.99	13641.82	1018.87	**13512.88**
R5.II	1486.97	**14625.63**	973.02	15177.94

Finally, in order to test the HRGRASP sensitivity on instances size and time windows widths, we have carried out the last set of experiments on the randomly generated test problems described in Sec. 4.2. The results are presented in Table 4.

The analysis of results shows that the hybrid algorithm works very well on all the test problems but two, i.e., for the largest instances tested with narrow time windows, namely R4.II and R5.II. Indeed, the cost of the solution found by TS is very close to the cost of the solution found by the HRGRASP. Actually, the largest gap, achieved on instance R4.II, is approximately 3.7%. We further observe that in these cases, the speed up achieved by the hybrid method is of 24.7% for the test problem R4.II and 52.8% for the most difficult instance. On average, the improvements in the objective function achieved by the HRGRASP is up to 3.14% with faster computation (17.79% on average).

As a final remark, the computing time of both the algorithms increases with number of requests to be serviced. Concerning the HRGRASP running times, they remain reasonable for tactical problems that need to be solved every day.

6. Additional Results on Homogeneous DARPs

An other set of experiments has been carried out with the aim of comparing the results of our HRGRASP with the best results obtained by the TS developed by Cordeau and Laporte (2003a) on standard DARP. Tables 5 and 6 give the results obtained. As a general observation, HRGRASP seems to be less effective than before. Nonetheless, some improvements can be detected. From Table 5, we might observe that the HRGRASP outperforms the TS on eight out of the twenty random Cordeau and Laporte instances (namely R1a, R2a, R3a, R7a, and R8a for narrow time windows and R1b, R2b, and R7b for wide time windows), finding one new best result for problems R2a (301.73 versus 302.08[1]). For the other instances the worsening in the objective function is on average very low (1.04%).

[1]As reported in Cordeau and Laporte (2003a).

Table 5. Results for the test bed of Cordeau and Laporte (2003a).

Test	Avg. cost		Best cost		Relative deviation		Absolute deviation		CPU time	
	HRGRASP	TS	HRGRASP	TS	HRGRASP	TS	GRASP	TS	HRGRASP	TS
R1a	**190.02**	190.41	**190.02**	**190.02**	0.00	0.20	0.00	0.20	0.63	2.62
R2a	**301.73**	302.43	**301.34**	301.48	0.13	0.32	0.13	0.36	12.52	21.56
R3a	**536.83**	539.20	**535.11**	536.51	0.32	0.50	0.32	0.76	28.84	29.35
R4a	584.11	**575.50**	579.90	**570.55**	0.73	0.87	2.38	0.87	56.98	38.65
R5a	649.54	**641.17**	647.26	**38.58**	0.35	0.41	1.72	0.41	79.37	66.26
R6a	823.89	**815.06**	818.38	**804.70**	0.67	1.29	2.38	1.29	101.00	163.89
R7a	**291.71**	292.40	**291.71**	**291.71**	0.00	0.23	0.00	0.23	4.44	7.88
R8a	**495.40**	495.41	**494.33**	494.47	0.22	0.19	0.22	0.22	44.12	70.20
R9a	693.06	**675.25**	687.94	**670.39**	0.74	0.72	3.38	0.72	73.11	117.34
R10a	895.37	**870.76**	892.65	**858.53**	0.30	1.42	4.29	1.42	215.74	239.55
R1b	**164.46**	165.22	**164.46**	**164.46**	0.00	0.46	0.00	0.46	3.71	3.13
R2b	**297.92**	298.34	**297.04**	298.30	0.30	0.01	0.30	0.44	17.52	11.11
R3b	494.14	**493.00**	492.71	**491.41**	0.29	0.32	0.56	0.32	30.76	18.72
R4b	549.41	**540.27**	547.02	**538.30**	0.44	0.37	2.06	0.37	57.95	32.57
R5b	598.84	**589.13**	594.92	**586.58**	0.66	0.43	2.09	0.43	84.02	70.52
R6b	772.30	**753.75**	768.66	**750.32**	0.47	0.46	2.93	0.46	130.49	92.29
R7b	**248.47**	**248.47**	**248.46**	248.47	0.00	0.00	0.00	0.00	5.83	4.53
R8b	468.54	**467.45**	467.10	**465.98**	0.31	0.32	0.55	0.32	41.54	21.69
R9b	620.65	**607.08**	619.23	**605.17**	0.23	0.31	2.56	0.31	92.92	56.26
R10b	844.68	**820.81**	843.29	**810.69**	0.17	1.25	4.19	1.25	155.41	120.65

Table 6. Results for the real life instances of Cordeau and Laporte (2003a).

Test	TS		HRGRASP	
	CPU	Cost	CPU	Cost
D1a	96.59	**3909.82**	81.86	3959.73
D1b	100.6	**3401.9**	96.06	3430.95
D1c	104.94	**3273.28**	97.37	3344.07
D2a	117.26	6882.52	157.69	**6826.81**
D2b	124.96	**5980.27**	147.33	6010.41
D2c	111.98	5453.78	174.85	**5437.55**

Regarding the real life instances, we notice from Table 6, that the HRGRASP cost is better than the TS cost for only two instances, namely D2a and D2. However, for these instances two new best results are achieved, with an average improvement around 2.4%. We also notice, as a misleading result, that in this case the HRGRASP is slower than the TS.

7. Conclusions

We have described a hybrid algorithm for the heuristic solution of the HDARPFC. We have implemented a reactive GRASP framework with an embedded TS as local search. Computational results, collected on a variety of test problems, have shown that the use of hybridization can be beneficial in solving HDARPFC, providing a good compromise between solution quality and computational time. As additional results, the application of the proposed method on standard homogeneous DARPs led to the discovery of new best known results for some test problems.

References

Aldaihania, M and M Dessouky (2003). Hybrid scheduling methods for paratransit operations. *Computers and Industrial Engineering*, 45, 75–96.

Beaudry, A, G Laporte, T Melo and S Nickel (2009). Dynamic transportation of patients to hospitals. *OR Spectrum*, 32, 77–107.

Berbeglia, G, JF Cordeau, I Gribkovskaia and G Laporte (2007). Static pickup and delivery problems: A classification scheme and survey. *Top*, 15, 1–31.

Borndörfer, R, M Grötschel, F Klostermeier and C Küttner (1997). Telebus Berlin: Vehicle scheduling in a dial-a-ride system. Computer-Aided Transit Scheduling. In *Lecture Notes in Economics and Mathematical Systems*, Vol. 471, pp. 391–422. Berlin: Springer-Verlag.

Bräysy, O and M Gendreau (2005). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science*, 39(1), 119–139.

Cordeau, JF and G Laporte (2003a). A Tabu search heuristic for the static multi–vehicle dial-a-ride problem. *Transportation Research Part B*, 37, 579–594.

Cordeau, JF and G Laporte (2003b). The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms, *4OR*, 1, 89–101.

Cordeau, JF and G Laporte (2007). The dial–a–ride problem: Models and algorithms. *Annals of Operations Research*, 153, 29–46.

Feo, TA and J Bard (1989). Flight scheduling and maintenance base planning. *Management Science*, 35(12), 1415–1432.

Feo, A and MGC Resende (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6, 109–133.

Festa, P and MGC Resende (2009a). An annotated bibliography of GRASP-Part I: Algorithms. *International Transactions in Operational Research*, 16(1), 1–24.

Festa, P and MGC Resende (2009b). An annotated bibliography of GRASP-Part II: Applications. *International Transactions in Operational Research*, 16(2), 131–172.

Festa, P and MGC Resende (2009c). Hybrid GRASP heuristics. *Foundations of Computational Intelligence Volume 3- Studies in Computational Intelligence*, 203, 75–100.

Glover, F and M Laguna (1997). *Tabu Search*. Norwell, MA: Kluwer.

Hanne T, T Melo and S Nickel (2009). Bringing robustness to patient flow management through optimized patient transports in hospitals. *Interfaces*, 39(3), 241–255.

Kontoravdis, GA and JF Bard (1995). A GRASP for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 7, 10–23.

Melachrinoudis, E, AB Ilhan and H Min (2007). A dial-a-ride problem for client transportation in a healthcare organization. *Computers and Operations Research*, 34, 742–759.

Nanry, WP and JW Barnes (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research B*, 34, 107–121.

Parragh, SN (2011). Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C*, 19(5), 912–930.

Parragh, SN, JF Cordeau, KF Doerner and RF Hartl (2012). Models and algorithms for the heterogeneous dial-a-ride problem with driver related constraints. *OR Spectrum*, 34(3), 593–633.

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

Rekiek, B, A Delchambre and HA Salehb (2006). Handicapped person transportation: An application of the grouping genetic algorithm. *Engineering Applications of Artificial Intelligence* 19(5), 511–520.

Resende, MGC and CC Ribeiro (2003). Greedy randomized adaptive search procedures. In *Handbook of Metaheuristics*, F Glover and G Kochenberger (eds.), pp. 219–249. Dordrecht: Kluwer Academic Publishers.

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

Wong, KI and MGH Bell (2006). Solution of the dial-a-ride problem with multi-dimensional capacity constraints. *International Transactions in Operational Research*, 13, 195–208.

Xiang, Z, C Chu and H Chen (2006). A fast heuristic for solving a large–scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, 174, 1117–1139.

Francesca Guerriero is Associate Professor of Operations Research, at the Faculty of Engineering, University of Calabria (UNICAL), Italy. She earned a PhD in Engineering of Systems and Informatics, from the same University. Her main research interests lie in field of network optimization. Other area of interests include logistics, revenue management and combinatorial optimization. She has published extensively in these fields, in a variety of journals, including Operations Research, Computational Optimization and Applications, Optimization Methods

and Software, Journal of Optimization Theory and Applications, European Journal of Operational Research, Journal of the Operational Research Society and Parallel Computing. She is actually the director of the LOGICA Laboratory, University of Calabria.

Maria Elena Bruni is Assistant Professor of Operations Research at the Department of Electronics, Informatics and Systems of University of Calabria, Italy, since 2006. She obtained a PhD in Operations Research, from the same University in 2005. Her areas of expertise are the design of algorithms for nonlinear integer stochastic programming, stochastic programming under probabilistic constraints, modeling of health care systems. She is co-author of papers accepted in refereed journals, author of a chapter in a book of global optimization and reviewer for international journals.

Francesca Greco received the graduate degree in Management Engineering from the University of Calabria (Italy) in 2005. After that she worked as software analyst in an Italian software company for nearly three years. Currently, she works as software engineer in a consulting company focused on management and optimization of business operations. As a senior software engineer, she is responsible for the development of management software products produced by the company.