Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 3 (2014) 442 – 451

Transportation Research Procedia

www.elsevier.com/locate/procedia

17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain

A Flexible Transport Service for Passengers

Pasquale Carotenuto[a]*, Leonardo Paradisi[b], Giovanni Storchi[c]

[a]*Istituto per le Applicazioni del Calcolo "M. Picone", National Research Council, via dei Taurini 19, Roma - 00185, Italy*
[b]*Dipartimento di Ingegneria dell'Impresa, University of Rome "Tor Vergata" via del Politecnico 1, Roma - 00133, Italy*
[c]*Dipartimento di Scienze Statistiche, University of Rome "Sapienza", piazzale Aldo Moro 5, Roma - 00185, Italy*

Abstract

The realization of innovative passengers transport services requires more and more often a greater flexibility and inexpensiveness of the service. To answer this request in many cases the physical solution is to realize a demand responsive transportation system (DRTS). A DRTS require the planning of travel paths (routing) and customers pick-up and drop-off times (scheduling) according to received requests, respecting the limited capacity of the fleet and time constraints (hard time windows) for each network's node, and the service time of the system. By the modelling point of view a DRTS can be effectively represented with a Dial-a-ride problem (DaRP). A DaRP derives from the Pick-up and Delivery Problem with Time Windows (PDPTW) and may operate according to a static or to a dynamic mode. In the static setting, all customers' requests are known beforehand and the DaRP returns the vehicles routing and the passengers pick up and drop off time scheduling. The static setting may be representative of a phase of reservation occurred the day before the execution of the service. But, if the reservation requests must be processed on-line, even during the booking process there may be a certain level ad dynamism. In fact, if the algorithm works online, it manages each and every incoming request separately, and accepts or refuses it immediately, without knowing anything about the following. The operative program is constantly updated after each received request without refusal to carry out previous accepted services. In the dynamic mode, customers' requests arrive when the service is already running and, consequently, the solution may change whilst the vehicle is already travelling. In this mode it is necessary that the schedule is updated when each new request arrives and that this is done in a short time to ensure that the potential customer will not leave the system before a possible answer. In this work, we describe a flexible people transport system capable of managing incoming transport demand in dynamic mode, using a solution architecture based on a two-stage algorithm to solve Dial-a-Ride Problem instances. In the first stage, a constructive heuristic algorithm quickly provides a feasible solution to accept the incoming demand. The algorithm in the second stage try to improve the solution evaluated at the first stage by using the time between two consecutive transportation events. The algorithm, unlike most of the works in the literature, use an objective function that optimizes the service punctuality.

Selection and peer-review under responsibility of the Scientific Committee of EWGT2014

* Corresponding author. Tel.: +39-064-927-0926; fax: +39-064-404-306.
E-mail address: carotenuto@iac.cnr.it

Selection and peer-review under responsibility of the Scientific Committee of EWGT2014
doi:10.1016/j.trpro.2014.10.025

Keywords: Dial a ride, Heuristics, Routing algorithms, Transportation planning.

1. Introduction

The realization of innovative passengers transport services requires more and more often a greater flexibility and inexpensiveness of the service. To answer this request in many cases the solution is to realize demand responsive transportation system (Ambrosino et al. 2006). A Demand Responsive Transport System (DRTS) requires the planning of travel paths (routing) and customer pick-up and drop-off times (scheduling) on the basis of received requests (Cubillos et al. 2004). In particular, it has to tackle the problem of multiple vehicles, the limited capacity of fleet vehicles and temporal constraints (time windows). The problem of working out optimal service paths and times is called a Dial-a-Ride Problem (DaRP), which derives from the well-known Vehicle Routing Problem (VRP) (Toth and Vigo, 2002; Barrie et al., 2003; Prins, 2004), with the addition of precedence constraints between pick-up and drop-off locations (Cordeau and Laporte, 2007).

Their computational complexity makes both DaRP and VRP as NP-hard problems, so attempts to develop optimal solutions have been limited to simple and small-size problems. It can be argued that heuristic procedures are more suitable for realistic networks and demand, because they allow to obtain good solutions in a limited amount of time. In the DaRP, customers formulate transportation requests from a given origin-destination pair (i.e., from a pick-up point to a delivery point). Transportation is carried out by vehicles that provide a shared service. Due to this fact, several customers may be in the same vehicle at the same time. Since the DaRP is a special Vehicle Routing Problem with Pickup and Delivery (VRPPD) (Savelsbergh and Sol, 1995), including restrictions on the time at which each point may be visited by a vehicle, the problem of finding a feasible pick-up and delivery plan is NP-hard. Clearly, the DaRP has to take into account specific constraints as we are considering people instead of goods. As a consequence, each customer specifies a possible pickup and delivery time, as well as an upper bound on the riding time (Coslovich, et al., 2006). Furthermore, in this context, customers often formulate two requests per day, specifying an outbound request from the pick-up point to a destination and an inbound request for the return trip. A DRTS may operate according to a static or to a dynamic mode. In the static setting, all the customer requests are known beforehand, and the DRTS produces, by solving a DaRP instance, the tour each vehicle has to make, respecting the pick-up and delivery time windows while minimizing the solution cost (Bergvinsdottir et al., 2004; Uchimura et al., 1999; Jaw et al., 1986; Jorgensen et al., 2007). In the dynamic mode, the customer requests arrive over time to a control station and, consequently, the solution may also change over time (Jih and Hsu, 1999; Carotenuto et al., 2006; Beaudry et al., 2010; Berbeglia et al., 2010). This work improves on previous approaches as it operates dynamically without interrupting the optimization cycle until the end of the service, providing the best solution found each time the system is modified. We address a DRTS capable of managing incoming transport demand to solve a DaRP in-stance using a solution architecture based on a two-stage algorithm. The first one is a constructive heuristic algorithm that quickly provides a feasible solution. The second algorithm is a specialized Hybrid Genetic Algorithm (HGA). Genetic Algorithms (GA) (Goldberg, 1989), are iterative stochastic algorithms in which natural evolution is used to model the search method. We obtain the so-called hybridization by including positive features of different algorithms into the GA schema.

The paper is organized as follows. In Section §2, we describe the DaRP by introducing the assumptions to be considered when a Demand Responsive Transportation System is being designed. In Section §3, we introduce the algorithms used in solving the DaRP instances. In particular, in Section §4, we describe the heuristic algorithm to compute an initial feasible DaRP plan. Section §5 describes the Genetic Algorithm implementation, in Section §6 some results related to a real case study are reported and finally conclusions are given.

2. DaRP definition

In DaRP (Bodin and al. 1983), each transportation request is dynamically formulated and specifies a single origin and a single destination as well as the number of passengers, a time of pick-up and delivery, the related time windows defined as maximum deviation by the time agreed and the maximum time that each passenger can remain on board the vehicle. The capacity of all vehicles is limited. For brevity, we do not provide the model formulation

here, but the complete mathematical structure is presented in Savelsbergh and Sol, 1995 and Cordeau and Laporte, 2007. We would like to point out that the DaRP has to notify customers as soon as possible the acceptance or denial of their requests (Horn, 2002, Coslovich, et al., 2006, Quadrifoglio et al., 2007). Moreover, according to the dynamic mode, the DaRP has to be capable of easily inserting new customer requests into one of the already initiated tours without violating any previously accepted customer requests. The following notations are used throughout the paper: Let N be the set of N_i ($i=1,...,n$) customer requests, M be the set of m_k vehicles ($k=1,...,m$) operating the tours, and $G(V,A)$ be a symmetric graph representing the road network. The depot of the vehicles and the vehicle stops are represented as nodes in V. The cost c_{hj} denotes the distance between nodes h and j belonging to set V. The cost matrix $\{c_{hj}\}$ satisfies the triangle inequality.

Regarding the maximum time that each passenger can remain on board the vehicle (Maximum Ride Time), there is an additional parameter to consider. To set an upper limit to the ride time within which the user has to reach its destination, the DRTS system manager can define a ϑ constant parameter of time (or a ϑ parameter as function of τ_{hj}), so given the pick-up time, the delivery time can be calculated as follows: $t_i^j = t_i^h + \tau_{hj} + \vartheta$ (or as $t_i^j = t_i^h + \tau_{hj} + \vartheta(\tau_{hj})$).

Let us introduce τ_{hj} as the minimum time required to travel from the pick-up point (h) to the destination point (j), both defined on graph G. If customer i specifies the pick-up time (the delivery time), the DRTS controls the delivery time (the pick-up time), that must be as follows:

$$t_i^j \geq t_i^h + \tau_{hj} \quad (t_i^h \geq t_i^j - \tau_{hj}) \tag{1}$$

When both the pick-up and delivery times are given, the DRTS assumes as valid the pick-up time and then controls the delivery time feasibility. Consequently, τ_{hj} represents the minimum ride time to make the service. Each ride request must be carried out within two time windows, one for pick-up and the other for delivery. In order to fix the time windows, the DRTS assumes that the time windows are centred in the preferred delivery time and not centred in the preferred pick-up time. In fact, we can assume that the customers normally, don't arrive to the stop before the time agreed, while probably they could accept to arrive earlier than their preferred delivery time. The goal is to construct a set of routes in order to satisfy the set N of customer transportation requests by using at most m vehicles. Let t_i^h and t_i^j be the preferred pick up time (on node $h \in V$), and delivery time (on node $j \in V$), of customer i, respectively. T_k represents the tour associated to vehicle k, where tour $T_k=\{v_{1k},..., v_{rk}\}$ is a set indicating the ordered list of r_k nodes in V visited by the vehicle k, for all $k = 1,..., m$.

Therefore, a solution for the DaRP is a set of routes T_k satisfying the pick-up and delivery time windows. In particular, let $s_{i,k}^h$ and $s_{i,k}^j$ be the pick-up and delivery times (at nodes $h, j \in V$), respectively, assigned to customer i whose request has been inserted into tour T_k.

We aim to identify the DaRP plan (i.e. a solution S) minimising cost z, representing customer delays, defined as follows:

$$z = \sum_{i=1,...,n} | s_{i,k}^h - t_i^h | + | s_{i,k}^j - t_i^j |, \quad \text{for all } k=1,..., m, \text{ and } h,j \in V \tag{2}$$

We emphasize the following additional statement and hypothesis. Each customer request has to also specify the number of people requiring the transportation service. The vehicle capacity is a bound of the number of customers that can be on board the vehicle at any given time. The following sections aim to identify the algorithms adopted for the DaRP solution and describe how they interact.

3. The two stage framework

This Section defines the general algorithmic architecture implemented in solving the DaRP. We assume that the problem is characterized by hard time window constraints where each customer i specifies either the pick-up time, or the delivery time, or both pick-up and delivery times. As a result, the DRTS has to pre-process the customer requests in order to verify the pick-up and delivery time feasibility and to define the time windows, thus the DaRP instance.

Then the DRTS produces a feasible requests instance for the DaRP, and to solve this requests instance we implement a two-level algorithm. In the first level an 'Insertion' heuristic algorithm is implemented, that produces, in a fast way, a set of feasible solution S for the requests instance. The second algorithm is an Hybrid Genetic Algorithm (HGA). Therefore, the best solution S constitutes the input for the HGA algorithm that produces and evaluates new improved solutions. Figure 1 represents the DaRP general schema

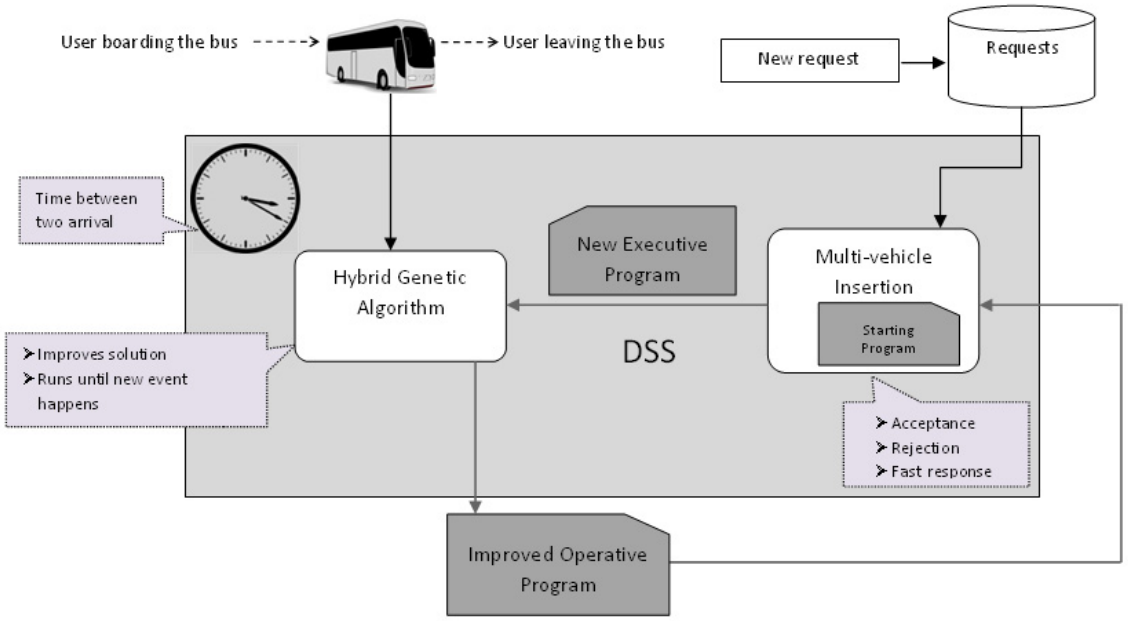

Fig. 1. The framework.

We emphasize two critical aspects. The first one concerns the 'Insertion' algorithm. In fact, one of the main requirements of a DaRP is its capability to quickly notify customers of the acceptance or denial of their requests. In this setting, the computational time of the algorithm used to solve the DaRP instance could be relevant for the DRTS consideration. This is the main reason why we have decided to implement a fast heuristic algorithm. Obviously, this DRTS capability will also be exploited in the dynamic mode, where customer requests arrive over time and it is necessary to provide notification as soon as possible.

The second critical aspect deals with the HGA having two main goals. First of all, it has to improve the solutions given by the 'Insertion' algorithm by allowing it to obtain a more profitable solution from the customer's point of view. That is, once a set of customer requests has been accepted, then the solution can be improved by using HGA. Therefore, in the dynamic mode, once the 'Insertion' algorithm accepts a new request, it has to produce a new solution, even if the tours have already been initiated.

4. Insertion algorithm

The algorithm called 'Insertion' is a constructive heuristic considering one customer request at a time. Initially, the tour T_k associated to each vehicle is empty, for each $k = 1,...,m$, and at each iteration the algorithm enlarges the tours by selecting one of the customer transportation requests. Therefore, the 'Insertion' algorithm finds a solution in *n* iterations, one for each customer requests.

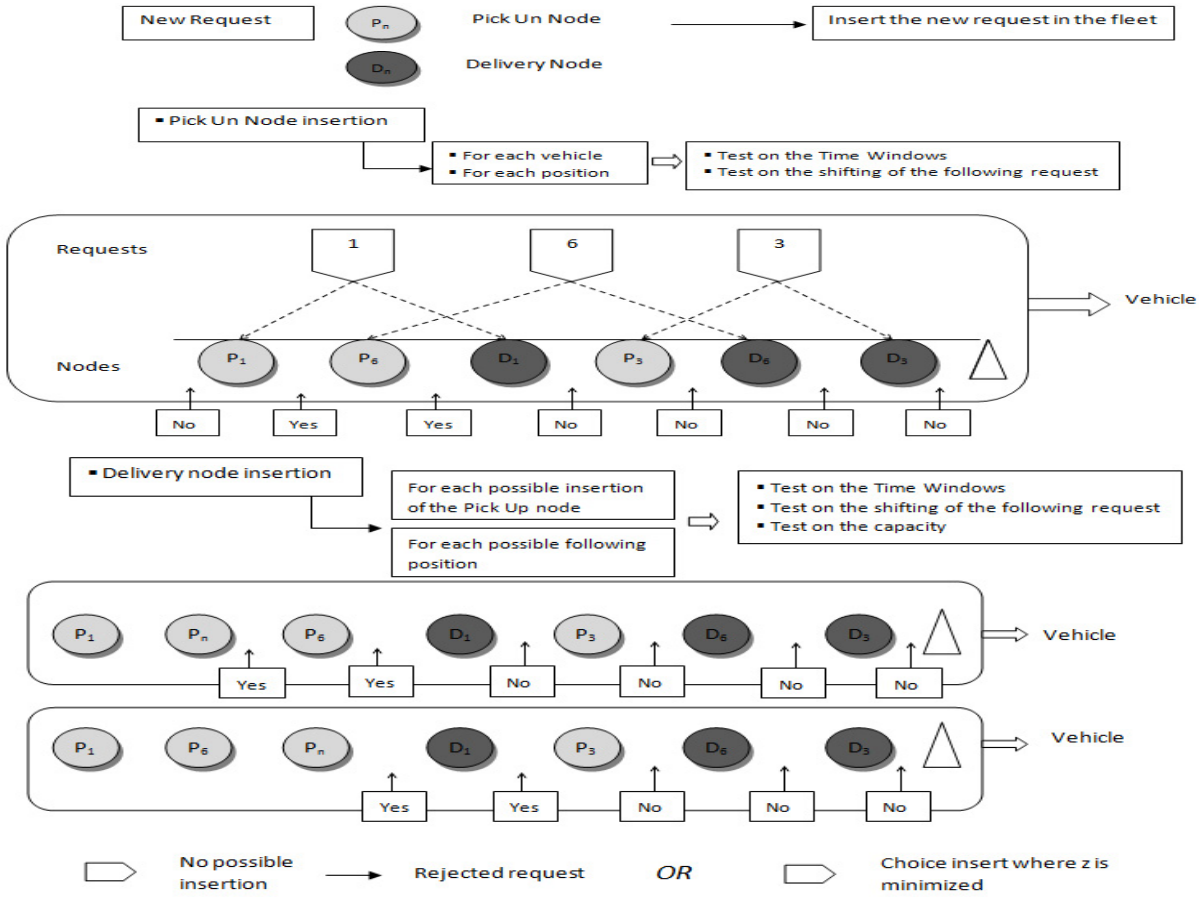

Fig. 2. The insertion heuristic scheme.

To insert a new customer request, selected by using a FIFO structure, the algorithm evaluates the request on the basis of previous solution, and if feasible a new solution S is given. The algorithm ends when all the requests have been evaluated. Clearly, the set of tours identifies an operative plan. In the following, we define the steps of the 'Insertion' algorithm (Nanry and Barnes, 2000):

- STEP 1: Initially, each tour $T_k = \varnothing$.
- STEP 2: For customer request N_i, the algorithm first examines the pick-up node $h \in V$. It analyses one tour T_k ($k=1,\ldots, m$) at a time and finds all the possible insertions of the pick-up node respecting the pick-up time window. Then, for all the tours whose pick up time window has been respected, the algorithm also tries to insert the delivery node $j \in V$, respecting the delivery time window. Clearly, the insertion of customer request N_i can be accepted if the previously inserted customer requests are still feasible, and if the vehicle capacity is not violated.
- STEP 3: Among the possible insertions providing a feasible pair (s_{ik}^{h} , s_{ik}^{j}), the algorithm selects the best one, that is the one minimising the value of z (see(1)). If a feasible insertion does not exist, the customer request is rejected, and the algorithm goes to step 4.

STEP 4: If all the customer requests have been evaluated, then the 'Insertion' algorithm stops. Otherwise, it goes to step 2 by analysing the next customer request N_{i+1}.

The 'Insertion' algorithm allows a rapid computation of the DaRP plan. Figure 2 shows the scheme of the insertion algorithm

5. Genetic schema

Our proposed HGA adopts the general structure of a classic genetic algorithm (Goldberg, 1989), but the chromosome is connected to a further string corresponding to the visiting sequence, and the fitness function values are provided by a modified 'Insertion' algorithm. Next, we briefly describe the main HGA steps:

- Initial Population. The initial population is made with the best feasible solutions previously found by the 'Insertion' algorithm. The feasible solutions found by the "Insertion" heuristics are more than one. We use all the solutions found to build the initial population, but these solutions are insufficient for achieving the entire initial population, therefore, we also use randomly generated solutions.
- Solution Encoding. We consider exactly n genes each one corresponding to an element of the set N of customer requests. Gene i specifies the vehicle k that serves customer request i. Clearly, the set of n genes constitutes the "chromosome". On the other hand, this solution encoding does not identify a single solution, because the value of the solution depends on the sequence in which the customers are served by the vehicles (this occurs during the modified insertion heuristics execution). Due to this fact, and to speed up individual evaluations in terms of the fitness function value, the chromosome contains a further string "sequence" corresponding to the visit sequence for each vehicle (Genes). The following table refers to a simple chromosome composed of four genes, where customers 1 and 3 are served by vehicle 1, while customers 2 and 4 are served by vehicle 2 and 3, respectively.

Customers	1	2	3	4
Genes	1	2	1	3
Sequence	1	1	2	1

- There may be different string sequences for the same genes string. In fact we could also consider the chromosome having the same genes string but the sequence values 2, 1, 1, 1. In this case the vehicle 1 serves at first the customer request 3 and then the customer request 1. The two chromosomes represent two different solutions and different fitness function values.
- *Fitness*. The HGA uses the value of the objective function z as defined in (1), to evaluate the fitness function of the solutions. The value of z function is given by a specialized version of the 'Insertion' algorithm.
- Notice that, since the HGA implements the roulette wheel selection, the fitness function (*ff*) is the function to maximize, while our objective function (z) is to minimize. For this reason we consider the following relationship (Goldberg, 1989) to compute the *ff*:

$$ff = M - z \tag{3}$$

where M is the maximum delay reachable to the stops by the m vehicles and z the value of the objective function (*fo*). In such way, maximizing the *ff* we select with high probability, individuals with lower z *(lower fo)*.

- *Reproduction*. It is based on the classical roulette wheel selection (Goldberg 1989), where a chromosome x of the old population will be selected according to its fitness value as follows

$$pselect = ff_x / \sum_x ff_x \tag{4}$$

- *Crossover and mutation.* As described in Goldberg (1989) but slightly modified, the HGA selected two chromosomes in the current population, with a given p_c probability value, execute a single-point crossover jointly with a modified version of the 'Insertion' algorithm, then with a given p_m probability value apply the mutation operator to the newly generated child.

The single-point crossover recombines a new individual using a modified version of the 'Insertion' algorithm. Working together, they make a 'placement to comb' for each specific vehicle and in the chromosomes, of the genetic

material from the second part of the second chromosome in the first part of the first chromosome. The semi-ordered sequences can intersect or in some cases juxtapose. The 'Insertion' algorithm is modified in the STEP 2 as follows:

- STEP 2 MODIFIED (HGA): For customer request N_i the algorithm examines first the pick-up node $h \in V$. It analyses only the tour T_k specified by the gene N_i and finds all the possible insertions for the pick-up node respecting the pick-up time window. Then, the algorithm tries to insert in this tour also the delivery node $j \in V$, respecting its time window. Clearly, the insertion of customer request N_i can be accepted if the previously inserted customer requests are still feasible, and if the vehicle capacity is not violated. After the crossover execution, the sequence information could be partial, so that mutation and crossover could result in a lack of elements in the sequence.
  For instance, consider two individuals as follows:

Customers request	1	2	3	4
Genes	1	2	1	3
Sequence	1	1	2	1

Customers request	1	2	3	4
Genes	1	1	1	1
Sequence	1	2	3	4

Fixing the single-point crossover equal to gene 2, the HGA considers the following new chromosome:

Customers request	1	2	3	4
Genes	1	2	1	1
Sequence	1	1	3	4

Customers request	1	2	3	4
Genes	1	1	1	3
Sequence	1	2	2	1

At this point, HGA executes the modified 'Insertion' algorithm in order to build the visit sequence for each vehicle. Therefore, by analysing the chromosome on the left, the algorithm constructs the tour for vehicle 1 and inserts customer request 1, and then customer 2. Then, customer 3 is inserted into the best available position in the current tour, while customer 4 is inserted according to its order in the arrangement S, so that, it has to be inserted in the best position after customer 3. Analogously, for the child on the right, the 'Insertion' algorithm considers all the customers having the same priority value, and inserts customers 1, 2, and 3 into the tour related to vehicle 1, while customer 4 is inserted into the tour of vehicle 3. After the crossover, the HGA has to apply the mutation operator and then the next chromosome could be valued. The mutation operator moves a request from one vehicle to another, to correctly insert the request in the new vehicle we still use the modified 'insertion' heuristic. When a single solution that violates the constraints is generated, it is assigned a low value to *ff* in order to have low probability of being selected.

- *Stopping Criterion*: A given number of generations or, in an operative situation, an event that imposes a change in the operative plan (i.e. a new request to evaluate, a user takes the vehicle, a user leaves the vehicle).

The hybrid genetic algorithm continuously obtains improved solutions and provides the best current solution to every system request.

6. Preliminary test

To validate the algorithm, a series of preliminary tests have been run using random instances and a test network with horizontal and vertical arcs with different travel times: 2 minutes for horizontal arcs and 1 minute for vertical arcs (Taniguchi, *et al.*, 2001). Every test has been done on 20 random instances considering the average value of the fitness function. To build the set of request, a procedure randomly generates a pair of pick-up and delivery nodes and the pick-up time requested by the user, then the delivery time is established by adding to the pick-up time the minimum time necessary to travel between the two nodes, plus a random value between 0 and 5 minutes.

6.1. Test on the evolution of the genetic algorithm

The first tests have shown that the maximum value of the fitness function has a tendency to rise quickly and then to stabilize, while the average value gradually rise, bringing it close to the maximum.

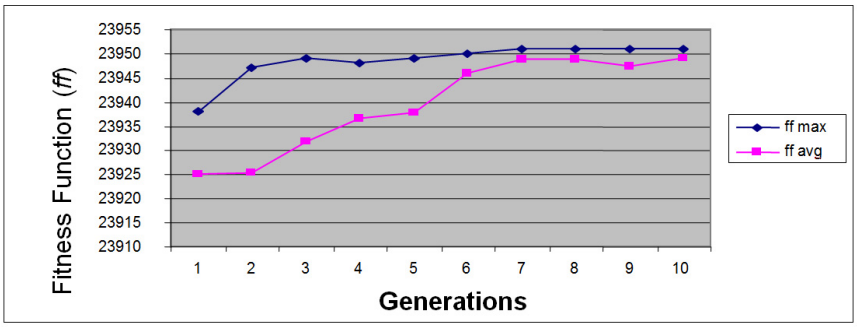

Fig. 3. The trend of the fitness function of the best individual and the average value of the generation.

The instance used for this preliminary test has a number of requests equal to 25 and a fleet of 3 vehicles. The data showed in figure 3 are related to 10 generations based on a population of 20 elements. The diagram presents the trend of the best individual fitness function and the average value of the individuals generation by generation. In the diagram, the maximum value of the fitness function quickly rises up until it settle itself in the last generations. Also the average value of the population show an upward positive trend with some small oscillation.

6.2. Test on the evolution of the genetic algorithm

The genetic algorithm has been modified to evaluate the effectiveness of elitism (Goldberg 1989). Eight scenarios have been considered by varying the ratio between the number of requests/number of vehicles whilst leaving the network of tests unchanged. The genetic algorithm that uses elitism always obtains the best values. The elitism has been realized maintaining in memory the best individuals created in the various generations through a Best Solutions Container (BSC). Its size is set equal to half of the population number. When the BSC is full and a new solution with a better fitness than the last element stored is generated, the lowest element is removed from the BSC, and the better one just found is properly inserted. The size of the BSC remains constant. When a new population will be created, individuals from the old population and individuals from the BSC, with given probabilities, are selected to act as parents for new individuals.

In the version without elitism instead, every new created population is “daughter” of the individuals in the preceding generation. The best individual in absolute is memorized because it represents the solution to the problem, but it cannot be used for the crossover until the following generation is created. In the HGA version with the elitism, the program needs two new input parameters: the size of the BSC; the probability to choose from the BSC. The size of the BSC is set equal to half of the population for all the resolved instances. The population number is set equal to 20. The *pBSC* probability regulates the frequency with which an individual of the BSC can be selected to be a parent. To perform this test a value of $pBSC$=0.8 was chosen.

Table 1: Comparison of the results using the elitism or not

n	m	n/m	Trial order Increasing n/m	fo (BSC)	fo (NOBSC)	fo(BSC)/n	fo(NOBSC)/n
25	3	8,33	2	62,25	62,85	2,49	2,51
50	3	16,67	5	121,80	144,45	2,44	2,89
75	3	25,00	8	207,67	289,00	2,77	3,85
75	4	18,75	7	185,94	243,85	2,48	3,25
75	5	15,00	4	182,60	224,60	2,43	2,99
75	10	7,50	1	174,90	196,00	2,33	2,61
100	5	20,00	7	270,50	352,17	2,71	3,52
100	10	10,00	3	234,35	277,45	2,34	2,77

This is a very high probability to choose from the BSC. In this way the system works in an condition significantly distant from the condition of the simple version of the hybrid genetic algorithm. The test carried out has the purpose to establish whether it is better to adopt a genetic algorithm that uses the BSC or not. To obtain this we have considered 8 scenarios with different characteristic and compared the value of the objective function, maintaining unchanged the number of generations. By comparing the results of the objective function reached by the 2 procedures showed in the table 1, we can easily see how the HGA that uses the elitism, *fo* (*BSC*), has provided the best results for all the 8 different scenarios considered.

6.3. Tuning of the parameters

The following parameters of the algorithm have been assessed: the probability of mutation, the probability of crossover and the probability of pick out by the BSC (De Jong, 1975, Goldberg, 1989). The test on the probability of mutation has been done, using empirical evidence and considering the fol-lowing values: 0.10, 0.03 and 0.00 (no mutation). The tests make us to prefer the value 0.03. In the tests, the worst result was obtained with the value of mutation of 0.0, this means that the mutation operator is necessary to better explore the solution space. The tests on the probability of crossover were performed considering the following values: 0.5, 0.7 and 0.9. In this case, it appears preferable the value 0.9. To test the probability of choosing a solution from the BSC, were considered the following values: 0.2, 0.5, 0.8. With regards to BSC probability, if the number of requests/number of vehicles is low, it is better to utilize 0.8 as a probability value, whilst if the number of requests/number of vehicles is high, it is better to utilize 0.5.

6.4. Test on the graph

We have also analyzed the variation of the percentage of rejected requests by varying the number of vehicles and the number of requests while keeping the network characteristics unchanged. The results show that if the number of vehicles in the network is kept constant, the number of accepted requests increases until it reaches a point of break-up. In addition, we have alternatively varied the number of nodes and the length of the arcs. The results show that by increasing the area covered by the service (the network) the number of request refusals grows but the number of request refusals doesn't depend on the number of nodes in the network. This is an advantage for users because it increases their access to the service whilst the service manager is not subject to additional costs.

6.5. Test on the heuristic

We have also evaluated the performance of the genetic algorithm compared to the heuristic one. A number of 12 scenarios have been considered, and the value of the objective function provided by the heuristic has been compared with the best solution provided by the HGA. The results show that the genetic algorithm does indeed improve the solutions found by the heuristic algorithm. Furthermore, it is able to find some solutions even when the heuristic

algorithm fails. If only the heuristic algorithm should be used, there would be a risk of refusing otherwise accepted requests.

7. Conclusion

This paper proposes a Hybrid Genetic Algorithm (HGA) for the DaRP and described its implementation. The DaRP is part of a DRTS whose goal is to support a public Service Provider in the management of transport activities. In particular, for the DRTS we approached the problem using a solution architecture based on a two-stage algorithm. After setting the parameters of the algorithm, (population size, crossover rate and mutation rate), we have implemented some computational experiments and obtained some interesting preliminary results. In future we plan to apply this method to a larger case and experiment other ways to improve this algorithm. For example we could take in account the use of the path relinking method (Ho and Gendreau, 2006; Rahimi-Vahed *et al.* 2012), applied successfully in similar context.

References

Ambrosino G., J.D. Nelson and M. Romanazzo (2004). *Demand Responsive Transport Services: Towards the Flexible Mobility Agency*. Published by ENEA, Rome.

Barrie M. B. and M.A. Ayechew, (2003). A genetic algorithm for the vehicle routing problem. *C&OR*, **30**, 787-800.

Beaudry, A., G. Laporte, T. Melo and S. Nickel, (2010). Dynamic transportation of patients to hospitals. *OR Spectrum*, 32, 77-107.

Berbeglia, G., J.-F. Cordeau, G. Laporte, (2010). Dynamic pickup and delivery problems. *EJOR*, 202, 8-15.

Bergvinsdottir K. B., J. Larsen and R. M. Jørgensen (2004). *Solving the Dial-a-Ride Problem using Genetic algorithms*, IMM-Technical report-2004-20, Informatics and Mathematical Modelling, Technical University of Denmark.

Bodin, Lawrence, Bruce Golden, Arjang Assad, and Michael Ball. (1983). Routing and Scheduling of Vehicles and Crews: The State of the Art, *Computers & Operations Research,* **10**(2), 63–210.

Carotenuto P., C. Cis And Storchi G. (2006). "Hybrid genetic algorithm to approach the DaRP in a demand responsive passenger service", *"Information Control Problems In Manufacturing 2006" Proceedings of the 12th IFAC International Symposium*, Elsevier Science, ISBN: 978-0-08-044654-7, v.**3**, 349-354,.

Coslovich L., R. Pesenti and W. Ukovich (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem, *EJOR,* **175**, 1605-1615.

Cordeau J.-F. and G. Laporte, (2007). The Dial-a-Ride Problem: Models and Algorithms, *Annals of Operations Research*, **153**, 1, 29-46.

Cubillos C., F. Guidi-Polanco and C. Demartini, (2004). Multi-Agent Infrastructure for Distributed Planning of Demand-Responsive Passenger Transportation Service, *Proceeding IEEE International Conference on Systems, Men and Cybernetics,* 2013-2017.

De Jong K.A., (1975). An analysis on the behaviour of a class of genetic adaptive systems. (Doctoral dissertation, University of Michigan). *Dissertation Abstract International* **36**(10), 5140B. (University microfilms n.76-9381).

Goldberg D.E. (1989). *Genetic Algorithms in Search Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

Ho S. C. and M. Gendreau, (2006). Path relinking for the vehicle routing problem. *Journal of Heuristics*, **12**, 55–72.

Horn M. E.T. (2002). Fleet scheduling and dispatching for demand-responsive passenger services, *Transp. Res. part C,* **10**, 35-63.

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

Jih W. J. and Y. Hsu (1999). Dynamic Vehicle Routing using Hybrid Genetic Algorithms. Proceedings of the 1999 IEEE Conference on *Robotics and Automation*, 453-458.

Jorgensen RM, J. Larsen and KB. Bergvinsdottir (2007). Solving the dial-a-ride problem using genetic algorithms. *JORS*, 58:1321-31.

Nanry W. and J.W. Barnes (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research, part B,* **34**, 107-121.

Prins C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *C&OR*, **31**, 1985-2002.

Quadrifoglio L., M.M. Dessouki and K. Palmer (2007), An insertion heuristic for scheduling Mobility Allowance Shuttle Transit (MAST) services, *Journal of Scheduling,* **10,** 25-40.

Rahimi-Vahed A., T. G. Crainic, M. Gendreau, W. Rei (2013). A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Journal of Heuristics,* **19**, 3, 497-524.

Savelsbergh M.W.P. and M. Sol (1995). The General Pickup and Delivery Problem. *Transportation Research* **29**, 17-29.

Taniguchi E., R. G. Thompson, T. Yamada and R. van Duin (2001). *City Logistics: network modelling and intelligent transport system.* Pergamon, Amsterdam.

Toth P. and D. Vigo (2002). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Pennsylvania.

Uchimura K., T. Saitoh and H. Takahashi (1999). The dial-a-ride problem in a public transit system. *Electronics and Communication in Japan*, part III 82, n. 7, 30-38.