Population Metaheuristics to solve the Professional Staff Transportation Problem

Rachida Abounacer [†,††], Ghizlane Bencheikh[†,††], Jaouad Boukachour[††], Btissam Dkhissi[†] and Ahmed Elhilali Alaoui[†],

[†] Modelling and Scientific Computing Laboratory, Department of Mathematics, Sciences and Techniques Faculty of Fez. B. P. 2202 Route of Imouzzer Fez Morocco
[††]CERENE Laboratory, ISEL, Quai Frissard B.P. 1137 76063 Le Havre Cedex France

Summary
The Professional Staff Transportation Problem (PSTP) consists to build the vehicle routing for transporting the staff of one or several companies, in order to minimize the total cost of transport, and taking into account the level of service offered to users. In this paper, we care about the quality of service given by the professional transporter in addition to the transportation cost. The first section presents an overview of recent works on Dial-a-Ride Problem (DARP) which is a generalization of our problem. The second section is dedicated to the mathematical modeling of the (PSTP) by introducing a measure of performance corresponding to the level of service provided to users. In the third section, we present two evolutionary metaheuristics to solve the problem, namely: Ant Colony Algorithm (ACO) and Genetic Algorithm (GA). The last section of this work is devoted to experimental results.

Key words:
Staff transportation, vehicle routing problem with time window, Dial-a-Ride problem, Ant Colony Algorithm, Genetic Algorithm.

1. Introduction

The professional staff transportation still attracts large and small companies for several reasons, mainly because it is a pledge of comfort, safety and punctuality. Thus, this type of transportation must respect logistics punctuality and comfort that the carrier must ensure with all means necessary to achieve it. We can define the staff transport as a system responding to a number of companies' requests (or demands) via a fleet of vehicles under constraints of feasibility and quality. Each request is characterized by the number of users to transport; the origin, the destination and the destination latest time (working time). The problem is to build the routing of the available vehicles reconciling two objectives:

1) Minimizing the cost of transport
2) Providing a good quality of service.

At first view, it appears some similarity between the PSTP and the Travelling Salesman Problem (TSP) [3, 11], which is to find a Hamiltonian cycle minimal of a Hamiltonian graph with a minimum total cost. However, the PSTP is much more complex; indeed, additional constraints must be taken account, such as, the precedence between the origin and the destination, the working times which are not the same for all requests and the capacity of vehicle.

In this paper, we consider the static case where all transportation requests are known in advance. We define an upper limit on the number of vehicles available and assume that the requests cannot be rejected, and we consider that all vehicles have identical capacity.

We formulate the PSTP as combinatorial optimization problem, where we introduce the constraints on three categories: routing constraints, capacity constraints and scheduling constraints. The objective is to cooperate the transportation cost and the service quality expressed, in our formulation, in term of:

- The waiting time of users before their working time.
- The ride time of the users.

We use two evolutionary approaches, Ant Colony Algorithm and Genetic Algorithm, to solve the problem. The Ant Colony Algorithm is applied in two steps (1) Repartition of requests into vehicles and (2) construction of routing for each vehicle. In the first step, we assign to each vehicle a subset of request in order to minimize the distances between origins, distances between destinations, the difference between the working times and respecting the capacities of vehicles. To apply the Ant Colony Algorithm, we consider a complete graph whose vertices represent all requests of the problem. In the second step, we apply for each vehicle an Ant Colony Algorithm in order to build a minimal routing for each vehicle by assigning to each request a time of service.

Unlike the first approach, we apply the Genetic algorithm to solve the problem in one step. Each individual represents for all needed vehicles, the routings, the arrival times, the departure times and the transported loads in each location.

Manuscript received July 5, 2009
Manuscript revised July 20, 2009

2. Literature review

The problem studied in this paper belongs to the general class of Dial-a-Ride problems (DARP). The DARP, officially classified NP-Hard [2], is a particular case of the Vehicle Routing Problem with Pickup and Delivery (VRPPD) [14, 15, 19, 23] arising in context where passengers are transported, either in groups or individually, from specified origins to specified destinations. The DARP distinguishes itself from the basic VRPPD by its focus on maximizing the level of service. This objective is often controlled by imposing a limit on the ride time of each user (i.e, the time spent by a user in a vehicle), on the waiting time and on deviations from the desired times for pickup and delivery. Maximizing service is weighted against minimizing the total cost of transportation [8].

In practice, dial a ride service can be operated according to one of two cases, static or dynamic. Static case is when all requests are known in advance. Static versions of DARP are described in [21]. In the dynamic case [6, 9, 16, 29], a request can arrive in a real time during the planning.

The DARP is classified an NP-hard, the proof is based on related NP-hard travelling salesman problem with time windows, into which the DARP can be classified [2].
The related works are most closely to our work in terms of problem definition and/or method of resolution. In 2003, Cordeau and Laporte [7] present a tabu search heuristic for the static multi-vehicle Dial-a-Ride problem with a specific case, where the users impose a time window of a pre-specified width on the arrival time of their outbound trip and, similarly, a window on the departure time of their inbound trip. In addition to this there is also an upper limit of the ride time of any user as well as constraints regarding vehicle capacity and route duration.

In 2004 A. Attanasio et al. [1] describe and compare a number of implementations of tabu search heuristic previously developed for the static Dial-a-Ride problem. Indeed, the authors use the static tabu search of Cordeau & Laporte [7] to find a solution to the static problem of the requests known at the start of the planning horizon. The experiments made indicate that parallel computing can be beneficial to solve real-time DARP.

Other metaheuristics have been applied to the DARP. In 1998 Baugh et al. [2] use simulated annealing to solve the DARP. A cluster-first, route-second approach is used. Simulated annealing is used for the clustering, and a greedy algorithm is used for the routing after that the clusters are made.

In 2007 K. B. Bergvinsdottir et al. [4] have applied a Genetic algorithm based on the "Cluster-first route-second " approach, in which the clustering is the most important phase. Customers are first clustered into feasible groups to be served by the same vehicle (cluster first) without regard to any preset ordering and then efficient routes are designed for each cluster (route second). In this approach, customers are first organized into clusters and then, the routes are developed for each individual cluster. The contribution of this paper is the demonstration that genetic algorithm can be effectively be implemented in a "Cluster-first route-second" approach. Their algorithm was tested on the same data used by Cordeau & Laporte [7], the results are comparable.

Many authors have also used different forms of insertion heuristics to solve the DARP, this approach is to construct a feasible solution, i.e., a set of feasible routes, by a set of feasible routes, inserting customer (not yet served) into a partially constructed feasible solution, the insertion decision is made based on the additional increase of the objective function. The insertion with the least incremental cost will be chosen

In 1995, Madsen et al. [20] use an algorithm based on an insertion heuristic to solve the DARP with multiple capacities and multiple objectives. The algorithm was developed to solve a real-life problem of scheduling transportation for elderly and disabled in Copenhagen, Denmark.

In 2004 Diana and Dessouky [10] presented a parallel regret insertion heuristic to solve large instances of the DARP. Data sets of 500 and 1000 requests have been tested.

Finally, Toth and Vigo in 1997 [31] developed a parallel insertion heuristic to be able to find good solutions for large instances within quite small computational times.

3. Mathematical formulation

Here, we propose a mathematical model of the problem, where we limit the work to the sense home→ workplace, the other sense will be similar.

3.1 Parameters

The following notation is used in the formulation:

- Nd : number of requests
- a_0 : the depot
- $R = \{r_1, r_2, \ldots, r_{Nd}\}$ set of requests

Each request is characterized by:

- i : Origin or domicile
- $i + Nd$: Destination
- q_i : Number of users to serve in the location i
- e_i : Working time
- t_{ij} : period of direct transport between the location i and j
- Nv : number of vehicles
- Q_k : capacity of vehicle k

The decision variables are:

- q_i^k : number of places occupied in the vehicle k at the location i
- b_i^k : number of users taken by the vehicle k at the location i
- d_i^k : departure time of the vehicle k at the location i
- h_i^k : arrival time of the vehicle k at the location i
- $x_{ij}^k = \begin{cases} 1 & \text{if the vehicle k passes through } a_i \textit{ before } a_j \\ 0 & \text{Otherwise} \end{cases}$

3.2 Objective functions

We consider two different objectives:

- Minimize the cost of transport
- Maximize the level of service for users

The relative cost of transport can be expressed by:

$$f_1: \sum_{i=0}^{2Nd} \sum_{j=0}^{2Nd} \sum_{k=1}^{Nv} t_{ij} . x_{ij}^k \quad (1)$$

Ideally, with one or more vehicles per request, each user could be transported directly from its origin station to its work place. As the number of vehicles available is limited, each user is imposing an additional route, because the vehicle must also pick up or deliver other people in his trip. For each request, the *additional travel time* is the difference between the ride time and duration of the shortest (direct) route between the origin and the destination. Moreover, it is possible that some requests arrive at their destinations too early, i.e. before the working time. The time that request will be at its destination until the beginning of work is called *waiting time*.

The feature quality proposed is expressed in terms of additional travel time and the waiting time of all requests; these two functions are respectively expressed by the following functions:

$$f_2: \sum_{i=1}^{Nd} \sum_{k=1}^{Nv} (h^k{}_{i+Nd} - d^k{}_i - t_{i,i+Nd}) \quad (2)$$

$$f_3: \sum_{i=1}^{Nd} \sum_{k=1}^{Nv} (e_i - h^k{}_{i+Nd}) \quad (3)$$

The overall aim is therefore to achieve a compromise between these three objectives which can be formulated as a multi-objective function:

$$Min\ (f_1, f_2, f_3)$$

3.2 Constraints

We distinguish for our problem, three types of constraints: the routing constraints, capacity constraints and schedule constraints.

3.2.1 Routing constraints

Constraints (4) and (5) ensure that each vehicle tour begins and ends at the depot.

$$\sum_{j=1}^{2Nd} x_{0j}^k \le 1 , \forall \quad k = 1, \ldots . Nv \quad (4)$$

$$\sum_{i=1}^{2Nd} x_{i0}^k \le 1 , \forall \quad k = 1, \ldots . Nv \quad (5)$$

When a vehicle k arrives to a location a_i, it has to pull through, which respect the law of Kirtchoff.

$$\sum_{j=1}^{2Nd} x_{ji}^k = \sum_{j=1}^{2Nd} x_{ij}^k , \forall k = 1, \ldots . Nv ; \forall i = 1, \ldots, Nd \quad (6)$$

Constraints (7) (resp. (8)) ensures that on a given location a_i, a vehicle k could arrive (resp. leave) from (resp. to) a single origin (resp. destination).

$$x_{ij}^k . \sum_{\substack{j'=1 \\ j' \neq j}}^{2Nd} x_{ij'}^k = 0 , \forall k = 1, \ldots . Nv ; \forall i, j = 1, \ldots, 2Nd \quad (7)$$

$$x_{ij}^k . \sum_{\substack{i'=1 \\ i' \neq i}}^{2Nd} x_{i'j}^k = 0 , \forall k = 1, \ldots . Nv ; \forall i, j = 1, \ldots, 2Nd \quad (8)$$

$$\sum_{j=1}^{2Nd} x_{ij}^k = \sum_{j=1}^{2Nd} x_{j,Nd+i}^k$$
$$\forall k = 1, \ldots . Nv ; \forall i = 1, \ldots, Nd \quad (9)$$

Constraint (9) states that each request i should be served by a vehicle k without stop.

$$x_{ij}^k \in \{0,1\}$$
$$\forall k = 1, \ldots . Nv ; \forall i, j = 1, \ldots, 2Nd$$

3.2.2 Capacity constraints

The capacity of each vehicle k must not be exceeded throughout its travel

$$q_i^k + b_j^k x_{ij}^k \le Q_k , \forall k = 1, \ldots . Nv ; \forall i, j = 1, \ldots, 2Nd \quad (10)$$

Constraint (11) updates the load of vehicles after the visit of a location *i*

$$q_j^k = \left(q_i^k + b_j^k\right)x_{ij}^k \,, \forall k=1,\ldots Nv, \forall i,j=1,..,2Nd \quad (11)$$

We must ensure that if a vehicle *k* does not pass through a location *j*, then the number of users served by *k* in this location is zero.

$$\left(x_{ij}^k - 1\right).b_j^k = 0 \,, \forall k=1,\ldots Nv; \forall i,j=1,\ldots,2Nd \quad (12)$$

$$q_i = \sum_{k=1}^{Nv} b_i^k \quad , \quad \forall \quad i = 1, \ldots . Nd \quad (13)$$

Constraint (13) ensures that the load in a location *i* is equal to the sum of those served by all vehicles in this location.

$$q_0^k = 0 \,, \forall \quad k = 1, \ldots . Nv \quad (14)$$

$$q_i^k \geq 0 \,, \forall k = 1, \ldots . Nv \,; \forall i = 1, \ldots, Nd \quad (15)$$

Constraints (14) and (15) respectively ensure that each vehicle leaves the depot empty and that his load is always positive.

Constraint (16) keeps, for each vehicle, the number of users getting in at the origin and getting out at the destination of each request.

$$b_{i+Nd}^k = -b_i^k \,, \forall k = 1, \ldots . Nv \,; \forall i = 1, \ldots, Nd \quad (16)$$

$$b_i^k \geq 0 \,, \forall k = 1, \ldots . Nv \,; \forall i = 1, \ldots, Nd \quad (17)$$

3.2.3 Schedule constraints

Each user must arrive at its destination before his working time:

$$h_{i+Nd}^k \leq e_i \,, \forall k=1,\ldots Nv; \forall i=1,\ldots,Nd \quad (18)$$

Constraint (19) updates the departure time of a vehicle *k* at a location a_i.

$$d_i^k = h_i^k + T \,, \forall k=1,\ldots Nv; \forall i=1,\ldots,Nd \quad (19)$$

Where *T* is the period of service made by users to get in or get out the vehicle. It can depend on the nature of the vehicle or on the number of users served in the location.

The arrival time of a vehicle *k* to a location *j* must be equal or greater than its departure time from a location *i* plus the direct time between *i* and *j*.

$$h_j^k \geq \left(d_i^k + t_{ij}\right)x_{ij}^k \,, \forall k=1,\ldots Nv; \forall i=1,\ldots,2Nd \quad (20)$$

Constraint (21) respects the precedence between origin and destination of each request.

$$h_{i+Nd}^k \geq h_i^k \quad , \quad \forall k=1,\ldots Nv; \forall i=1,\ldots,Nd \quad (21)$$

$$h_i^k \geq 0 \,, \forall k = 1, \ldots . Nv \,; \forall i = 1, \ldots, 2Nd \quad (22)$$

$$d_i^k \geq 0 \,, \forall k = 1, \ldots . Nv \,; \forall i = 1, \ldots, 2Nd \quad (23)$$

We note that our model is non-linear due to the constraints (7), (8), (10), (11) and (12).

4. Ant Colony Algorithm for the Professional Staff Transportation Problem

Ant Colony Algorithm was first proposed by Dorigo and Gambardella [12] as a muti-agent approach for difficult combinatorial optimization problems such as travelling salesman problem (TSP) [26] and the quadratic assignment problem (QAP) [17]. ACO has been applied to other problems such as graph coloring [24], job shop scheduling [32] and vehicle routing [27].

The results obtained by Ant Colony Optimization are comparable to those with other general purpose heuristic algorithms [5]. A convergence proof for a generalized Ant System Algorithm is provided in [18]. This algorithm is inspired by the behaviour of real ants. Ants, when searching for food, mark the traversed paths with a pheromone quantity, which depends on the quality of the food source. Other ants observe these pheromone trails and are attracted to follow them, thus reinforcing the paths. Gradually, paths leading to rich food sources will be used more frequently.

To apply the Ant Colony Optimization to solve the professional staff transportation problem, we proceed in two steps, the first step is to distribute requests on vehicles within their capacities and the second step is to build the optimal routing for each vehicle.

4.1 Distribution of requests on vehicles

This step aims to distribute requests on vehicles, in order to assign to each vehicle a subset of requests, respecting its capacity and minimizing the distances between origins,

distances between destinations and the difference between the working times.

For this, we consider the complete graph (A, U, V) where:

- A= $\{r_0, r_1, r_2, \ldots, r_{Nd}\}$ is a set of vertices, where r_0 is a fictitious request corresponding to the depot and characterized by:
  - $a_0 = a_n$ where their coordinates are (0,0)
  - $q_0 = 0$
  - $e_0 = 0$
- $U = \{\{r_i, r_j\} / r_i \in A\}$ is a set of edges, at each one is associated a value

$$v(i,j) \in V \text{ where } v(i,j) = (t_{ij} \times t_{i+Nd\ j+Nd}, |e_i - e_j|)$$

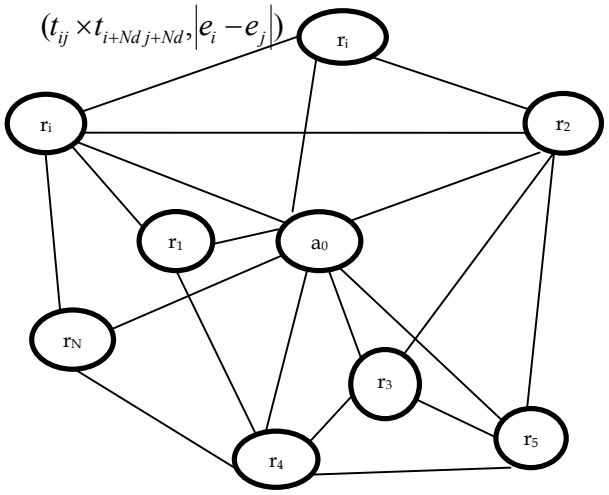

Fig. 1 the graph structure for the problem of distribution of requests on vehicles.

We apply the ACO to build an optimal path contained all vertices in order to minimizing the total cost depending on the couples ($t_{ij} \times t_{(i+Nd,\ j+Nd)}$, $|\ e_i - e_j\ |$). The obtained tour will then be divided according to vehicles' capacities.

In some cases, a request may be assigned to two or more vehicles where each vehicle serves a set of users. This case can happen if the current vehicle can not bear the totality of users, and then the remainder of users is affected to following vehicles.

In the ACO, we define the trace's initial intensity τ_{01} which can be set to a small and positive arbitrary value. The heuristic information (visibility) η_{ij} is defined as follows:

$$\eta_{ij} = \frac{1}{t_{ij} t_{(i+Nd)(j+Nd)} + 1} \times \frac{1}{|e_i - e_j| + 1} \qquad (24)$$

An ant k located in vertex i, selects the vertex j, $j \in J_i^k$ (set of feasible vertices) to move to, according to a probabilistic decision rule:

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij})^{\alpha_1} (\eta_{ij})^{\beta_1}}{\sum_{l \in J^k_i} (\tau_{il})^{\alpha_1} (\eta_{il})^{\beta_1}} & \text{if } j \in J_i^k \\ 0 & \text{otherwise} \end{cases} \qquad (25)$$

The parameters α_1 and β_1 define the relative importance of the trace $\tau_{ij}(t)$ with respect of the visibility $\eta_{ij}(t)$.

$\tau_{ij}(t+1)$ is the trace intensity (pheromone in the case of real ants) associated to the edge (i, j) in the iteration $t+1$. This quantity satisfies the following equation:

$$\tau_{ij}(t+1) = \rho_1 \tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad (26)$$

Where:

- ρ_1 is a coefficient of evaporation (it must be fixed to value < 1 to avoid an unlimited accumulation of trace)
- $\Delta\tau_{ij}(t)$ is the additional quantity of trace left on the edge (i, j) by the colony at the end of the iteration t.

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{M} \frac{m_{ij}^k(t)}{M} \qquad (27)$$

Where M is the total number of ants and:

$$m_{ij}^k(t) = \begin{cases} 1 & \text{if the ant k takes the edge (i, j)} \\ 0 & \text{otherwise} \end{cases} \qquad (28)$$

After a fixed number of iterations, we obtained the best path which contains all requests. This optimal path is then divided on sub-path according to vehicles' capacities. Thus, each vehicle is assigned to a set of requests near in term of working time, origins' distance and destinations' distance.

This first step of the resolution is illustrated by the following instance where we consider 10 requests. Obtaining a minimal Hamiltonian path which contains all requests, we distribute these requests on 4 vehicles with respect to their fixed capacity limit of 12 places. As shown in the following table, a request may be assigned to two vehicles.

Table 1: Procedure of distribution of requests on vehicles

Vehicles	V1		V2			V3				V4		
Requests	r_5	r_1	r_7	r_4	r_2	r_2	r_9	r_3	r_6	r_6	r_{10}	r_8
Charges	6	5	3	5	7	7	3	4	7	7	4	3

Ant Colony Algorithm for distribution of requests on vehicles

1- Initialize the matrix of pheromone by the value τ_0
2- Repeat
For each ant k = 1, ..., M do
Initialize the candidate list by the available requests
While (the candidate list is not empty)
Select a request according to decision rule (25)
End
End For
Update the pheromone matrix according to (26)
Until maximum number of iterations is reached

3- / Distribution of requests on vehicles */*
a – Initialize the vehicle v ← 0
b – Initialize the charge of the vehicle q_v ← 0
c – Initialize the request r by the first request of the path given by the best ant
d – If ($q_v + q_r <$ Vehicle Capacity)
Affect the request r to the vehicle v
Set the charge in the request r by the load q_r
Pass to the next request of the path
Go to step 3 - d
Else
Affect the request r to the vehicle v
Set the charge in the request r by the load "Capacity – ($q_v + q_r$)"
Update the load at the request r
Pass to the next vehicle v ← v + 1
Go to step 3 – b
End If

4.2 Construction of vehicles routing

This step aims to assign, in parallel, for the available vehicles routings with a minimal cost, at each request a time of service.
Based on the results of the first step, we consider for each vehicle k a graph $G_k = (A_k, U_k, V_k)$ where:

- $A_k = \left\{ a_0 \ , \ \underset{r_i \in R_k}{\mathrm{U}} \{a_i, a_{i+Nd}\} \right\}$, R_k is the set of origins and destinations of all requests affected to the vehicle *k*.
- $U_k = \left\{ \{a_i, a_j\} / \ a_i \in A_k \right\}$ is the set of edges, at each one is associated a value $v(i,j) \in V_k \ where \ v(i,j) = (t_{ij}, e_j)$

In this second step, we apply the ACO for each vehicle; this consists to build, in parallel, routings which minimize the total cost of transport, the waiting time and the ride time of each user. These objectives are controlled by both heuristic and pheromone information.
We express the heuristic information by the term:

$$\eta_{ij} = \frac{1}{t_{ij} t_{(i+Nd)(j+Nd)} + 1} \times \frac{1}{e_j + 1} \tag{29}$$

This expression promotes the near location in term of distance and the earliest working time.

In our Algorithm, we use a global pheromone update, according to the following formula:

$$\tau_{ij}(t+1) = \rho_2 \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{30}$$

Where

$$\Delta\tau_{ij} = \begin{cases} \dfrac{1}{\sum_{i \in best} |e_i - h_i| + 1} \times \dfrac{1}{\sum_{i \in best} (h_{i+Nd} - d_i - t_{i,i+Nd}) + 1} & \text{if } (i,j) \in \text{best} \\ \dfrac{1}{\sum_{i \in bad} |e_i - h_i| + 1} \times \dfrac{1}{\sum_{i \in bad} (h_{i+Nd} - d_i - t_{i,i+Nd}) + 1} & \text{if } (i,j) \notin \text{best} \end{cases} \tag{31}$$

best and *bad* are, respectively, the routings of the best and the bad ants.

This expression promotes the routing with a minimal total waiting time and ride time.

Our algorithm proceeds as follows:

Ant Colony Algorithm for each vehicle k

Initialization
Placement of ants on the depot
Initialization of the pheromone on all edges of the graph
For *each iteration t = 1,..., t_{max}*
For *each ant k = 1,..., M*

Initialize the list of candidates by all the origins of the graph
Initialize the list taboo by depot
While *the candidate's list is not empty*
Choose a vertex j according to the probabilistic transition rule:

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij})^{\alpha_2}(\eta_{ij})^{\beta_2}}{\sum_{l \in J^k_i}(\tau_{il})^{\alpha_2}(\eta_{il})^{\beta_2}} & \text{if } j \in J_i^k \\ 0 & \text{else} \end{cases}$$

Insert j in the list taboo.
Update the times of arrival and departure from a visited location after the first destination chosen.
if $j \leq Nd$
then Replace j by j + Nd in the candidate's list
End if
End while
End for
End for
Update pheromone according to the formula (30)

5. Genetic Algorithm for the Professional Staff Transportation Problem

Genetic Algorithm has been developed by John Holland in 1975 [25] inspired by the Darwin's theory of evolution. The main is to maintain a population which is a set of solutions (called in this context individuals), through a fixed number of iterations. To each individual is associated a numerical value called fitness depending on the objective function of this solution. At each iteration, a number of individuals are selected according to their fitness in order to create new ones by applying two operators: Crossover and Mutation. At the end of the iteration, a phase of replacement is applied to select the individuals which pass to the next iteration.

The Genetic algorithm is organized as follows:
1. Coding the solutions
2. Generate the initial population
3. Repeat
   a. Selection
   b. Crossover
   c. Mutation
   d. Replacement
   Until a fixed number of iterations

In our GA, an individual representation considers both the assignments of requests to the available vehicles and construction of the vehicles' routings.

5.1 Coding

The coding is one of the most important steps of the GA, in this paper we represent an individual by a set of the required vehicles where each vehicle consists on four lists:
- The trajectory
- The arrival time at each location
- The departure time at each location
- The load corresponding of the number of people transported at each location

Example of an individual, where the number of requests is 4,
We note that:
$i_1, \ldots, i_4$ designate origin locations
$i_5, \ldots, i_8$ designate destination locations

Table2: Example of an individual

Vehicles	V1				V2			
Trajectory	i_4	i_1	i_5	i_8	i_3	i_2	i_7	i_6
Arrival time	490	500	510	514	513	520	528	541
Departure time	493	504	512	516	515	521	531	543
Load	3	4	- 4	- 3	2	4	- 2	- 1

This individual consists on *2* vehicles where the first one pick up *3* employees of the request *4* at its origin location i_4 on *490 min* (*8h10min*), the vehicle leaves the location at *493min* to the location i_1 which is the origin of request *1*, the vehicle arrives at *500min* and pick up *4* employees. The next station is the destination of the request *1*, the number *-4* shows that the employees whose get up the vehicle at the origin i_1 are the same whose get down at the destination.

5.2 Initial Population

The initial population is generated by a local heuristic, which construct the routing of available vehicles beginning by the first one by inserting the request's origin and destination in its trajectory. If the capacity of the vehicle is achieve, we pass to the second one, until all requests have been served. This local heuristic is based on the following steps:
1. *Create a candidate list which contains all the available requests with their load*
2. *Initialize the number of vehicle at 1*
3. *While the candidate list is not empty*
   - *Select randomly a request r_i from the candidate list*
   - *Select the position "pos" of the origin randomly from {1, ... , Ncompt} where Ncompt is the number of locations affected to the current vehicle and the*

position of the destination "Pos_des" randomly from {pos + 1, ... , Ncomp}.

- *If all the available employees have been transported*
  Then delete the request r_i from the candidate list
  Else
  Update the charge of the request r_i in the candidate list
- *Update the arrival and departure times*
- *If the vehicle capacity is achieve*
  Then increase the number of vehicles by 1 and pass to next vehicle and go to step 3
  Else
  Go to step 3

5.3 Crossover

The crossover aims to create new individuals (called children) from those (parents) selected according to their fitness. In this paper, we apply two different crossover operators: one-point (1X) [13] and Uniform [13]. The one point crossover consists on dividing the parent's chromosomes according to one point, when the uniform crossover consists on creating the children's chromosomes according to a binary vector which decides if the gene corresponded must be taken from the first or the second parent.

In both crossover operators, we are interested just by the first and second lists of the individuals, corresponding respectively to the vehicles and their trajectories. From these, we create (for each parent) a list containing the requests in the order of their apparition in the parents' vehicles trajectory beginning by the first vehicle into the last one. As shown below:

Let be the following parents:

P1:

V1	V2	V3	V4	V5
3 8 18 13	10 20 2 12	4 10 1 11 20 14	6 9 1 19 11 16 7 17	9 19 8 5 15 18

P2:

V1	V2	V3	V4	V5
3 2 5 12 15 13	9 6 3 16 13 19	7 1 8 9 19 11 17 18	4 8 18 14	8 10 18 20

The request lists corresponding to *P1* and *P2* are as follows:

P1 : 3 8 10 2 4 1 6 9 7 5
P2 : 3 2 5 9 6 7 1 8 4 10

The second step is to apply the crossover operator:
In one-point crossover, the two lists are divided according to a crossover point:

P1 : 3 8 10 2 4 1 6 9 7 5
P2 : 3 2 5 9 6 7 1 8 4 10

The first (resp. second) child's list contains the first part of the first parent (resp. second) and completed by the missing requests in the same order of the second (resp. first) parent's list

Child1 : 3 8 10 2 4 5 9 6 7 1
Child2 : 3 2 5 9 6 8 10 4 1 7

In uniform crossover, the children's lists are constructed according to a binary vector which is randomly generated and characterized by a size equal to the parents' one, this vector decides if the gene must be taken from the first or the second parent as shown below:

P1 : 3 8 10 2 4 1 6 9 7 5
P2 : 3 2 5 9 6 7 1 8 4 10

Binary list : 1 0 0 1 0 1 1 0 0 0

The number *0* in a position *i*, indicates that the request in this position of the first parent must be inserted in the first child list (if this request is not already inserted) and the request in the same position of the second parent must be inserted in the second child list. For the number *1* we permute the parent's roles. Finally, we complete the child list by missing requests.
The result candidate lists are as follows:

Child1 : 3 8 10 9 4 7 1 5
Child 2 : 3 2 5 6 1 8 4 10

Requests in red are those taken from parent 2.

To complete the lists, we insert requests *2* and *6* into the first list and requests *7* and *9* into the second list. Finally, we have the following children *1* and *2*:

Child 1: 3 8 10 9 4 7 1 5 2 6
Child 2: 3 2 5 6 1 8 4 10 7 9

5.4 Mutation

The mutation operator ensures the diversity of individuals in the GA. The objective is to create a new individual by modification one or more components of an individual even this modification causes a deterioration of the objective function. We applied two mutation operators in our work. The first one consists in permuting the position

of two requests by exchange their origins and destinations in the trajectory of a given vehicle:

V1	V2	V3	V4	V5
3 8 18 13	10 20 2 12	4 10 1 11 20 14	6 9 1 19 11 16 7 17	9 19 8 5 15 18

V1	V2	V3	V4	V5
3 8 18 13	10 20 2 12	4 10 1 11 20 14	9 6 1 16 11 19 7 17	9 19 8 5 15 18

The second mutation operator changes the assignment order of the requests to vehicles. We construct a request list by adding the first request of each vehicle, then the second one, and so on:

Example:

V1	V2	V3	V4	V5
3 8 18 13	10 20 2 12	4 10 1 11 20 14	6 9 1 19 11 16 7 17	9 19 8 5 15 18

The request list is:

3 10 4 6 9 8 2 1 5 7

The new individual is created by inserting the request in the list order.

5.5 Fitness

For each individual, we associate a fitness function corresponding to a weighting of the three objective functions: transportation cost, the total waiting time and total ride time. We represent the fitness function as follow:

$$\alpha_1 \sum_{i=0}^{2Nd} \sum_{j=0}^{2Nd} \sum_{k=1}^{Nv} t_{ij}.x_{ij}^k + \alpha_2 \sum_{i=1}^{Nd} \sum_{k=1}^{Nv} (h^k{}_{i+Nd} - d^k{}_i - t_{i,i+Nd}) + \alpha_3 \sum_{i=1}^{Nd} \sum_{k=1}^{Nv} (e_i - h^k{}_{i+Nd}) \quad (32)$$

Where α_1, α_2 and α_3 represent the importance of each cost

6. Experiment results

In this experiment, we used a 1.6 GHz Intel Pentium M processor, 1 Go RAM.

6.1 Generation of Instances

We generated randomly a set of instances of 10 to 500 requests, each request is characterized by:

- A number between 1 and 11 users to be served.
- A working time e_i between 06h00 and 08h00.
- A homogenous fleet of vehicles, each one includes a fixed capacity.
- An origin's and a destination's coordinates taken in the square [-20 Km, 20 Km], where the depot is localized in the origin (0, 0).
- An Euclidean distance between two locations a_i and a_j.
- An average speed of vehicles fixed at 60 km/h.
- A service period T (the period made by a user to get in or get out the vehicle) is fixed at 0.15 minutes.

6.1 Setting parameters

To maximize the performance of an algorithm it is essential to optimize the choice of its parameters. In this work, we used the experimental design method whose goal is generally, to define for each parameter, its effect on the objective function, by several tests [28]. For this, we use the following terminology [24]:

- **Response**: function to optimize *(y)*,
- **Factors**: parameters $(x_1, x_2 \ldots x_n)$,
- **Levels**: the possible values of a factor in experiments.
- **Effect** of a factor x_i on a response y, is the variation of y with respect to the levels of a factor.
- **Interaction**: two factors x_1 and x_2 are in interaction, if the effect of x_1 depends on the level of x_2.

Among the different experimental designs that exist, we choose to use the factorial design at two levels 2^n which consists to define two levels for each factor (higher level « +1 » and lower level « -1 »). In these designs, any combination of the two levels is present during the experiment, and the goal is to define the effects of each factor on the response *(y)*. An example of a factorial model 2^2 *(n=2)* is given by the equation:

$$y = a_0 + a_1x_1 + a_2x_2 + a_{12}x_1x_2$$

With: a_0, a_1 and a_2 the coefficients of the model. They are called the effects of factors.

Example:

For an instance of 10 requests, we use Minitab software tool [22] to fix different parameters of our GA. The influential factors results, in our study, are defined as follows:

- x_1 Populations size *(Np)*
- x_2 Mutations probability *(Pm)*
- x_3 Type of crossover *(Cr)*
- x_4 Type of mutation *(mut)*
- x_5 Crossovers probability *(Pc)*
- x_6 Maximal Number of generations *(Ngmax)*

We define below a 2-level (-1, +1) of values x_1, x_2, x_3, x_4, x_5 and x_6.

Table3: Levels of the GA parameters

Levels	Factors					
	x_1	x_2	x_3	x_4	x_5	x_6
	Pc	Pm	Crossover	Mutation	Np	Ng
-1	0.6	0.1	1X	Permutation	50	500
+1	0.9	0.4	Uniform	Exchange	100	1000

The figure 2 below shows that the best results are obtained using the following values of different parameters of the algorithm:

- Pc=0.6
- Pm=0.4
- Crossover: one-point (1X)
- Mutation: PERMUTATION
- Np=100
- Ng=1000

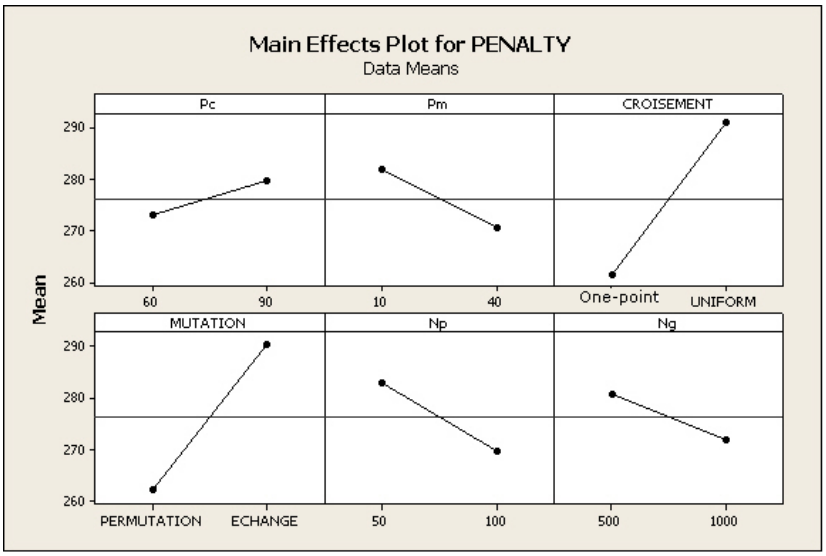

Fig. 2 graphic of main effects

We precede similarly for the others instances and we obtained the results represented in the following table:

Table 4: The values of the GA parameters

Benchmarks	Ngmax	Np	Pc	Pm	Cr	Mut
1	100	10	0.6	0.4	1X	Permutation
2	200	10	0.7	0.3	1X	Permutation
3	500	100	0.9	0.4	1X	Permutation
4	1000	100	0.9	0.4	1X	Permutation
5	1000	100	0.5	0.4	1X	Permutation
6	1000	100	0.9	0.4	1X	Permutation
7	1000	100	0.9	0.5	1X	Permutation
8	1000	100	0.6	0.4	1X	Permutation
9	1000	100	0.9	0.4	1X	Permutation
10	1000	100	0.9	0.4	1X	Permutation

The table bellow shows the best values of different parameters of the ACO:

Table 5: The values of the ACO parameters

Bk.	Number of iteration	Nb. ants	α_1	β_1	α_2	β_2	τ_{01}	τ_{02}	ρ_1	ρ_2
1	100	5	1	1	3	3	0.5	1	0.2	0.8
2	200	10	3	0.5	1	2.5	0.3	0.3	0.9	0.5
3	500	10	1	0.5	1.5	1	0.5	0.3	0.5	0.6
4	500	20	0.8	3	4	4	1	0.8	0.4	0.9
5	500	20	1	1.5	4	1	1	1	0.7	0.5
6	1000	30	1	3	4	1	0.5	0.5	0.8	0.9
7	1000	30	1	4	3	3	1	0.3	0.5	0.9
8	1000	30	1	1	3	0.5	1	1	0.5	0.9
9	1000	30	1	4	3	0.5	0.3	1	0.9	0.9
10	1000	30	1	4	3	3	0.3	0.3	0.9	0.5

6.3 Computational results

Firstly, we present detailed results obtained for the first instance which contained 10 requests.
The following table shows the best routing found for each vehicle:

Table 6: Best vehicles routings

	Vehicle	Routing
ACO	V1	(0, 6, 10, 20, 5, 16, 15, 0)
	V2	(0, 6, 2, 4, 16, 14,12, 0)
	V3	(0, 3, 7, 13, 2, 17, 12, 0)
	V4	(0, 3, 9, 8, 19, 13, 18, 0)
	V5	(0, 8, 1, 11, 18, 0)
GA	V1	(0, 1, 11, 9, 19, 0)
	V2	(0, 6, 16, 2, 12, 0)
	V3	(0, 7, 3, 17, 2, 13, 8,18, 0)
	V4	(0, 3, 4, 14, 13, 0)
	V5	(0, 10, 5, 15, 20, 0)

Table 7 shows the assignments of requests to the required vehicles, the number of users of each request *i* served by the vehicle *k*, the ride time *(RT)*, the waiting time *(WT)* and the transportation cost of all vehicles. We can see that the transportation costs are close for the GA and ACO.

Table 7: Results obtained by both the ACO and the GA for the instance of 10 requests

	Vehicles	Requests	b_i^k	*RT (min)*	*WT (min)*	*Transport cost (Km)*
ACO	V1	6	3	8.58	23.27	74.63
		10	10	0.59	17.43	
		5	2	3.70	0.29	
	V2	6	3	8.58	23.27	
		2	1	13.78	13.44	
		4	10	4.93	6.22	
	V3	3	7	5.11	19.06	
		7	1	10.63	2.48	
		2	7	2.75	1.04	
	V4	3	2	9.93	0.29	
		9	8	6.37	2.24	
		8	5	5.64	39.24	
	V5	8	5	9.77	105.69	
		1	7	2.32	1.04	
GA	V1	1	7	2.32	1.04	79.71
		9	8	4.50	62.99	
	V2	6	7	4.26	1.04	
		2	8	2.52	12.19	
	V3	7	1	2.79	0.15	
		3	4	4.63	1.09	
		8	10	3.64	31.98	
	V4	3	5	14.75	15.96	
		4	10	4.02	1.50	
	V5	10	10	7.32	6.94	
		5	2	2.30	0.29	

The following graph gives a comparison between the waiting time for each request given by ACO and GA.

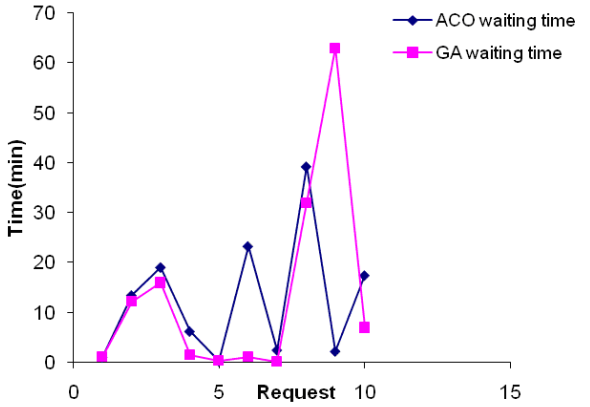

Fig. 3 Waiting time given by ACO compared with waiting time given by GA

As we can see from the graph, GA results are better than those given by ACO for 90% of the requests. For 10%, ACO results are better than those given by GA.

The following figure shows a comparison between the ride time of a request calculated by ACO and GA and the direct ride time corresponding to the time which takes a vehicle to transport a request from the origin to the destination without stop.

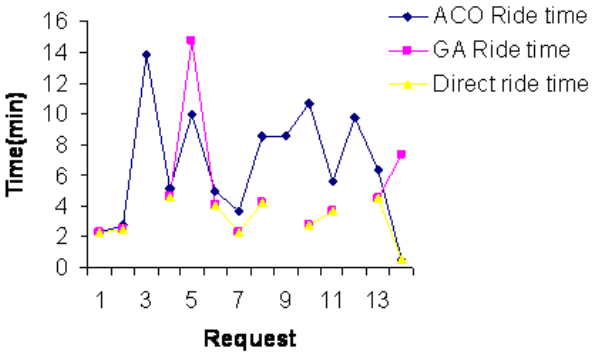

Fig. 4 Comparison of the Ride time given by both ACO and GA with the direct ride time

For 80% of requests, the requests are directly transported from the origin to the destination by applying the GA, since for ACO, there are just 40% of requests that are directly transported. But, we can see from the following table, that the transport cost calculated by ACO is better than transport cost calculated by GA.

In the second part of this section, we evaluate results obtained for the 10 different instances of 10 to 500 requests.

The following table resumes results obtained for all instances involving up to 500 requests,

Table 8: Results obtained for the 10 instances

Bk	Nd	Nb of users	Nb. of vehicles		Total Waiting time		Total Ride time		Transport cost (Km)		Times(s)	
			ACO	GA	ACO	GA	ACO	GA	ACO	GA	ACO	GA
1	10	72	5	5	255	135.17	92.98	53.05	74.63	79.71	0.05	0.01
2	30	180	9	9	750.00	589.85	222.11	173.71	416.16	645.39	1.24	0.11
3	60	363	19	19	849.26	1274.75	465.55	438.89	830.92	1167.73	10.95	19.08
4	100	535	18	18	2083.12	3664.65	1234.90	1535.13	940.54	1890.16	67.13	31.89
5	150	770	26	26	2261.82	5730.61	2104.24	2897.17	1428.93	2894.39	180.56	27.86
6	200	1045	35	35	2627.13	7455.21	2519.99	4074.95	1844.01	3610.57	475.65	82.90
7	250	1363	46	46	2951.23	18952.35	3178.47	4773.67	2450.65	5021.03	119.56	108.76
8	300	1713	58	58	3449.48	7455.21	3830.61	4074.95	3034.47	3610.57	192.32	104.08
9	400	2204	74	74	4190.93	16445.79	4507.53	5907.23	3747.60	8082.64	430.95	306.83
10	500	2725	91	91	4370.73	19340.02	5572.96	8738.37	4553.88	9992.58	1584.26	470.54

We observe that ACO and GA provide the same number of vehicles for all the instances. For the first one involving 10 requests, the results given by GA are better that those given by ACO, but, when we increase the number of requests, the ACO provides much better results than GA as shown in the following figures.

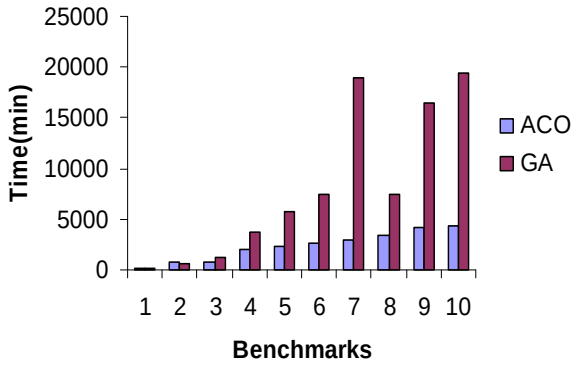

Fig. 5 Comparison between the Waiting time given by ACO and GA

The fact that results given by ACO is much better than those given by GA owes to the first step of ACO. In this step, we construct a global routing containing all requests such as the distance between origins, destination and difference between work times of requests, is minimal. Therefore, requests which are close are served by the same vehicle.

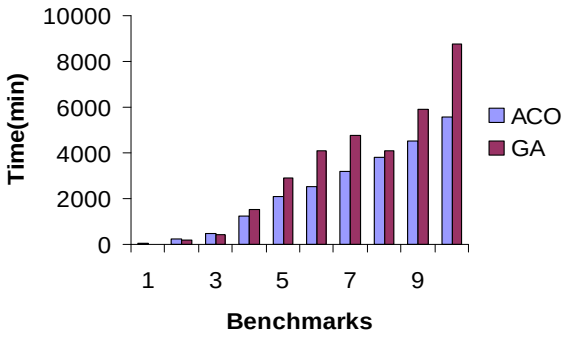

Fig. 6 Ride time

The constructive aspect of the ACO, has a large effect on results because, each ant during its path construction, selects stations which optimize the objective function. Thus, ACO is more guided than GA.

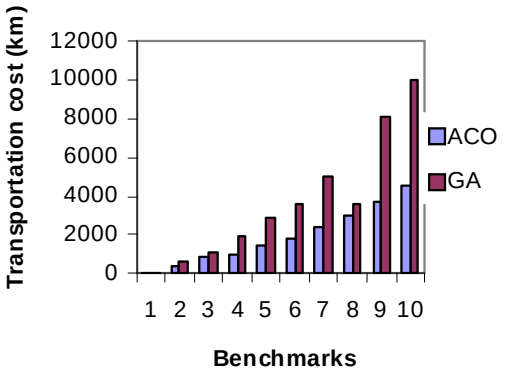

Fig. 7 Transport cost

However, if we compare the two algorithms in term of CPU time, we observe that ACO is very expensive than GA. Unlike GA which inserts station randomly in the trajectory, each ant makes several tests before selecting the next station.

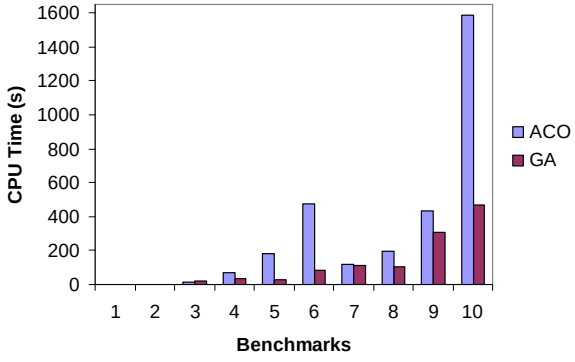

Fig. 8 CPU Time

7. Conclusion

In this paper, we have proposed a mathematical formulation for the PSTP and adapted two population heuristics, namely, ACO and GA to resolve it. Our model takes into account the optimization of service's quality adding to the cost transportation. So, we aim to minimize the ride time and the waiting time in the working place for each user.

To evaluate our approaches and a comparison between them, we have tested them on several sizes of random generated data. The two approaches seem to be consistent with different generated instances.

The experiments presented in this paper, shows clearly that the constructive aspect of the ACO has an important role in the generation of efficient solutions in term of the solutions quality comparing with GA. And that, through the insertion decision of the current request to the partial solution is made according to the objective function. i.e, the insertion with the best cost will be chosen.

References

[1] Attanasio, A., J.-F. Cordeau, G. Ghiani, G. Laporte, Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem, Parallel Computing , 30 377–387, 2004.

[2] Baugh J.W.Jr., Kakivaya, D.K.R., Stone, J.R., Intractability of the dial-a-ride problem and a multi objective solution using simulated annealing, Engineering Optimization, 30(2):91-124, 1998.

[3] M. Bellmore, G.L. Nemhauser, The traveling salesman problem: a survey, Operations Research 16, 538–558, 1968.

[4] Bergvinsdottir, K.B., J. Larsen, R. Jørgensen, Solving the dial-a-ride problem using genetic algorithms. Journal of the Operational Research Society, forthcoming, vol. 58, pp. 1321-1331, 2007.

[5] Bonabeau, Dorigo, Theraulaz, From nature to artificial swarm intelligence. New York: Oxford University Press, 1999.

[6] Colorni, A., Righini, G., Modelling and optimizing dynamic dial-a-ride problems, International transactions in operational research, 8, 155–166, 1999.

[7] Cordeau, J.-F., G. Laporte, A tabu search heuristic for the static multi-vehicle dial-a-ride problem, Transportation Research Part, 37 579–594, 2003.

[8] Cordeau, J.-F., Laporte, G., The dial-a-ride problem (DARP): Variants, modelling issues and algorithms. 4OR: A Quarterly journal of operations research 1 89-101, 2003.

[9] Coslovich, L., Pesenti, R., Ukovich, W, A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. European Journal of Operational Research, 175, Issue 3, 16, 1605-1615, 2006.

[10] Diana, M., Dessouky, M., A new regret insertion heuristic for solving large scale dial-a-ride problems with time windows, Transportation Research Part B, 38, 539–557, 2004.

[11] Dorigo M, Gambardella L M., Ant colonies for the traveling salesman problem, Bio Systems, 43(2): 73-81, 1997.

[12] Dorigo, M., Gambardella, L.M, Ant Colony System: A cooperative learning approach to the travelling salesman problem, IEEE Transaction on Evolutionary Computation. I(1), 53-66, 1997.

[13] J. Dréo, A. Pétrowski, P. Siarry and E. Taillard : Métaheuristiques pour l'optimisation difficile, Eryolles 2003.

[14] Dumas, Y. , Desrosiers, J., Soumis, F., The pickup and delivery problem with time windows, European Journal of Operational Research 54, 7-22, 1991.

[15] Ferman, A., Tang, M., Diéguez Galvão, R., A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, Computers & Operations Research, 33, pp 595-619, 2006.
[16] Feuerstein, E., Stougie, L., On-line single-server dial-a-ride problems, Theoretical Computer Science, 268, 91–105, 2001.
[17] Gambardella, L.M, Taillard, E.D., Dorigo, M., Ant colonies for the quadratic assignment problem, Journal of the Operational Research Society, Volume 5, pp. 167-176(10), 1999.
[18] Gutjahr, W.J, A graph –based Ant System and its convergence, Future Generation Computing System 16, 873-888, 2000.
[19] Kalantari, B., Hill, A.V., Arora, S.R., An algorithm for the traveling salesman problem with pickup and delivery customers, European Journal of Operational Research 22, 377-386, 1985.
[20] Madsen, O., Ravn, H., Rygaard, J. M, A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives, Annals of Operations Research, 60, 193–208, 1995.
[21] Melachrinoudis, E., Ilhan, A., Min, H, A dial-a-ride problem for client transportation in a health-care organization, Computers and Operations Research, Vol. 34, Issue 3, 742-759, 2007.
[22] Minitab, "Introduction à Minitab version 14", Septembre 2003
[23] Nagy, G., Salhi, S., Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries, European Journal of Operational Research Volume 162, Pages 126-141, 2005.
[24] Pillet M. (Ed), Le Plans d'expériences par la méthode Taguchi, Les Editions d'organisation, Paris, 1997.
[25] Reeves, C.R., Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill International Limited UK, 151 - 188, 1995.
[26] Shang, G., Lei, Z., Fengting, Z, Fengting, Z., Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rul, Computer Society Washington, DC, USA. Pages 693-698, ISBN:0-7695-2875-9, IEEE, 2007.
[27] Stutzle, T., Dorigo, M., Evolutionary Algorithms in Engineering and computer science: Recent Advances in Genetic Programming. Genetic Programming and industrial Applications, Chapter ACO Algorithms for the travelling Salesman problem. Tohn Wiley and Sons, 1999.
[28] Tagushi G., System of Experimental Design, Kraus International Publication, 1987.
[29] Teodorovich, D. Radivojevic, G., A fuzzy logic approach to dynamic dial-a- ride problem, Fuzzy sets and systems, 116, 23–33, 2000.
[30] Thang N. Bui, ThanhVu H. Nguyen , Chirag M. Patel , Kim-Anh T. Phan, An ant-based algorithm for coloring graphs, Elsevier Science Publishers B. V. Amsterdam, The Netherlands Volume 156 , Pages 190-200, 2008.
[31] Toth, P., Vigo, D., Heuristic algorithms for the handicapped person's transportation problem, Transportation Science, 31, 60–71.
[32] Zhang, J., Hu, X., Tan, X., Zhong, J.H., Huang, Q., Implementation of an Ant Colony Optimization technique for job shop scheduling problem, Transactions of the Institute of Measurement and Control, Vol. 28, No. 1, 93-108, 2006.

Rachida Abounacer is a PhD student of the Laboratory of Modeling and Scientific Calcul and CERENE laboratory, she is a member of Operational Research and Computer group at the Faculty of Sciences and Techniques of Fez, Morocco. She works on scheduling problems and metaheuristics.

Ghizlane Bencheikh is a PhD student of the Laboratory of Modeling and Scientific Calcul and CERENE laboratory, she is a member of Operational Research and Computer group at the Faculty of Sciences and Techniques of Fez, Morocco. She works on scheduling problems and metaheuristics.

Jaouad Boukachour is an Associate Professor of Computer Sciences at Le Havre University, France. His research interests include: Scheduling Problems, Operational Research, and Supply Chain Management. He has supervised a number of PhD researchers in areas such as logistics and scheduling aircraft landings. Currently, he is supervising six PhD students working on traceability, modelling road traffic, job shop scheduling, scheduling aircraft landings and vehicle routing. He has published more than 30 referred research papers. Within the French CPER 2006 (State-Region Project Contract), he was responsible for Modelling Optimisation and Simulation of physical and information flows in an industrial logistics project. Currently, he heads two projects about tracking container shipments, funded by French National Research Agency (ANR) and CPER 2008.

Btissam Dkhissi is a PhD student of the Laboratory of Modeling and Scientific Calcul at the Faculty of Sciences and Techniques of Fez, Morocco, she is a member of Operational Research and Computer group. She works on staff scheduling, timetabling problems, vehicle routing and metaheuristics methods.

Ahmed Elhilali Alaoui is a PhD of Operational Research at the Faculty of Sciences and Techniques of Fez, Morocco. His research interests include: Scheduling Problems and Operational Research. He is responsible for the operational research and computer group, and he is supervising 10 PhD students working on job shop scheduling, scheduling aircraft landings, vehicle routing and optimization algorithms. He is member of the La Société Marocaine de Recherche Opérationnelle (SOMARO).