RAIRO-Oper. Res. 42 (2008) 35–67
DOI: 10.1051/ro:2008002

RAIRO Operations Research
www.rairo-ro.org

COMBINATION OF MOBILE AGENT AND EVOLUTIONARY ALGORITHM TO OPTIMIZE THE CLIENT TRANSPORT SERVICES

Hayfa Zgaya[1], Slim Hammadi[1] and Khaled Ghédira[2]

Abstract. This paper presents a migration strategy for a set of mobile agents (MAs) in order to satisfy customers' requests in a transport network, through a multimodal information system. In this context, we propose an optimization solution which operates on two levels. The first one aims to constitute a set of MAs building their routes, called Workplans. At this level, Workplans must incorporate all nodes, representing information providers in the multimodal network, in order to explore it completely. Thanks to an evolutionary approach, the second level must optimize nodes selection in order to increase the number of satisfied users. The assignment of network nodes to the required services must be followed by a Workplan update procedure in order to deduce final routes paths. Finally, simulation results are mentioned to invoke the different steps of our adopted approach.

Keywords. Mobile agents, evolutionary algorithms, multimodal information system, multimodal transport network.

Mathematics Subject Classification. 90B06, 90C29, 68M14, 68T20, 68U35.

1. Introduction

According to the exponential growth of the services and information available on large networks such as the Internet, multimodal transport network customers

Received 30 October, 2006. Accepted 12 October, 2007.

[1] LAGIS UMR 8146 École Centrale de Lille, France;
hayfa.zgaya@ec-lille.fr; slim.hammadi@ec-lille.fr

[2] SOIE, ISG-Tunis ISG, Tunis University, Tunisia; khaled.ghedira@isg.rnu.tn

require relevant, interactive and instantaneous information during their travels. A Multimodal Information System (MIS) can offer support tool in order to make available needed information. The main goal of such a system is to guide and help network customers to make good decisions when they are travelling. Information is represented by distributed data located on different nodes through a Multimodal Network (MN). These nodes represent servers which are owners of multimodal transport, corresponding to our system information providers. In a previous work [1], we proposed a MIS based on a special kind of software agent called Mobile Agent (MA). A MA [2] is a program composed of code, data and a state, able to move from a node to another through a network to perform tasks and finally return to its original node called Home [4–6]. MA technology can considerably reduce network traffic [3] as such is the case in the field of the data extraction which can require the access to huge and geographically separated data sources [7]. In fact, instead of transferring huge amount of data through the network, a MA migrates to concerned information servers and performs its code using the resources of the visited nodes to collect needed data. Finally, this agent returns to the Home node with expected results. In this paper, we present a migration strategy of MAs in order to achieve user satisfaction in term of cost and response delay. Therefore, we propose an optimization solution which operates on two levels. The first one aims to optimize the number of MAs in order to explore the whole MN respecting initial routes, which are provisional. Final routes are deduced every time a set of customers' simultaneous requests is formulated. In this case, the MIS identifies the set of nodes which propose required services. From this set and thanks to an evolutionary method, the system optimizes the selection of nodes to answer to formulated requests; this last procedure represents the second level of our solution. Selected nodes are chosen according to the total resulted cost and response delay corresponding to generated responses. A customer is satisfied if he obtains rapidly a response to his request, with a suitable cost. This paper is organized in seven parts. In the following section, we present the problem. Then, we detail our proposed solution within two steps through Sections 4 and 5 which represent respectively the first and second level of the adopted strategy. Simulation results are presented in Section 6 and the paper is concluded in last section.

2. Problem description

New distributed applications are spread on huge networks, employing heterogeneous and voluminous data sources located on geographically separated sites. Through such networks, different actions (navigation, requests, storage, update, etc) can be launched intensely and in a simultaneous way requiring the access to several heterogeneous and distant data sources. This behaviour can be illustrated by different kinds of interactions among individual entities [8]. Thus, a multi-agent approach seems to be the most appropriate to implement the complex character of this kind of system. Besides, distributed applications through wide networks are not easy to realize because of the limited aspect of bandwidth which remains

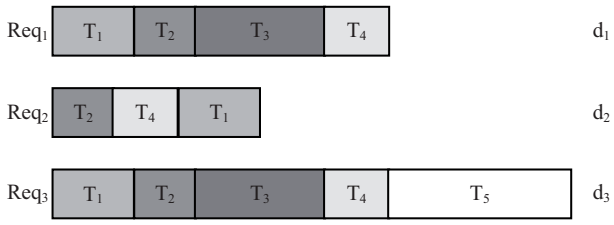

FIGURE 1. Requests decomposition.

restricted by several technical factors [9]. Therefore, permanent connexion can not be constantly available. That's why, we have to manage data information availability despite recurrent connexions in order to access and share distributed data. In this context, mobile technology [10] can complement artificial intelligence because it can reduce considerably network traffic [3]. Giving the mobility character to a software agent will allow it to migrate towards any node on the network which can receive mobile entities. Our goal is to design and implement a MIS which is able to receive and send mobile entities through huge and distributed networks, in order to optimize the management of the responses to intensive and simultaneous requests. We aim to extend our approach to new distributed applications (PDA, laptop ...) corresponding to wireless technology generation where bandwidth remains limited. In this paper, we are interested to satisfy MN customers; this is a two-steps assignment problem: firstly the assignment of network nodes to MAs to build their routes, called Workplans and then, a sub-set of these nodes are selected to assign tasks, in order to satisfy users. A task is an independent sub-request which belongs to one or several requests formulated simultaneously (Fig. 1). After that, information providers which propose services corresponding to identify tasks are recognized (Fig. 2). Finally, nodes must be assigned to tasks in order to satisfy all connected users (Fig. 3). A user is satisfied if his request was answered rapidly with a reasonable cost.

The problem is defined by:

- R requests, waiting for responses at the same instant t. The set of these requests is noted R_t,
- Each request $req_w \in R_t$ is decomposed into a set of independent tasks, noted $I_{t,w}$,
- Each request req_w has a due date $d_w$initially known (Fig. 1), an ending date D_w and a total cost C_w,
- The set of independent I tasks representing all proposed services on the MN is noted I_t,
- The realization of each task $T_i \in I_t$ requires a resource, or node, selected from a set of J available nodes, noted $S=\{S_1, \ldots, S_J\}$,
- The set of independent I' tasks$(I' \leqslant I)$composing R_t is noted $I'_t(I'_t \subseteq I_t)$,

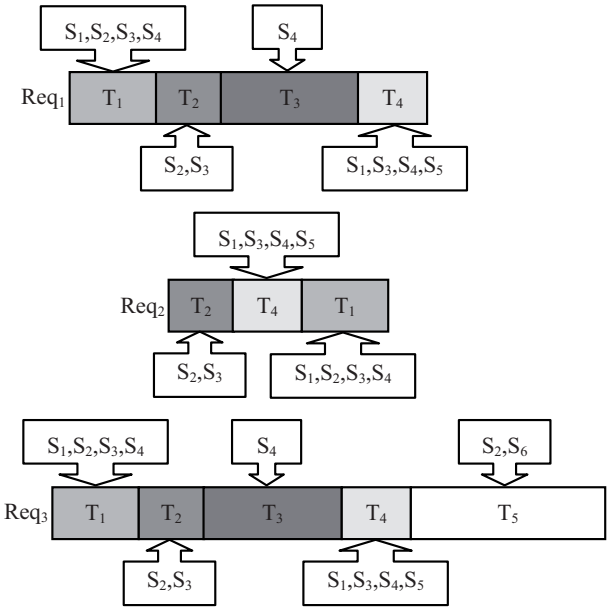

FIGURE 2. Nodes identification.

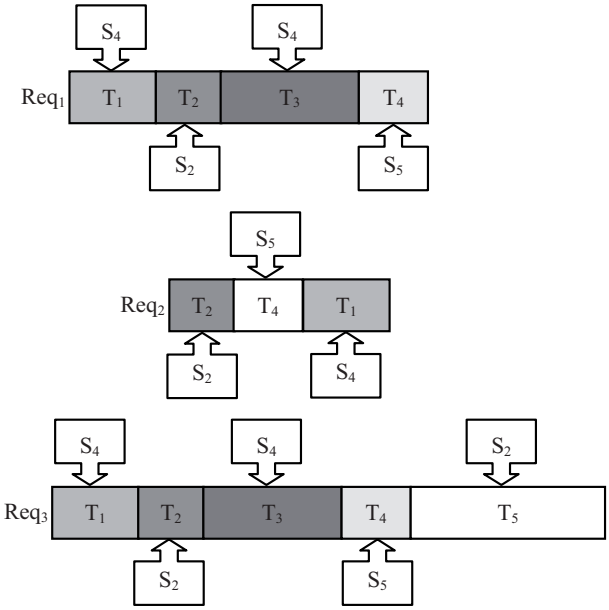

FIGURE 3. Nodes assignment.

TABLE 1. Example of available services (Processing time, cost and data size).

	S_1	S_2	S_3	S_4
T_1	(0,0,0)	(2,5,3)	(4,3,3)	(2,5,3)
T_2	(2,4,5)	(1,5,2)	(4,5,1)	(3,8,3)
T_3	(1,7,3)	(0,0,0)	(2,5,3)	(4,2,2)
T_4	(3,2,1)	(0,0,0)	(0,0,0)	(0,0,0)
T_5	(2,3,1)	(1,1,3)	(4,5,2)	(4,5,3)

- The set of J' nodes ($J' \leqslant J$) selected from S, to perform I'_t is noted S' ($S' \subseteq S$),
- There is a predefined set of processing time; for a given node S_j and a given task T_i, the processing time of T_i using S_j is defined and noted by P_{ij},
- There is a predefined set of information cost; for a given node S_j and a given task T_i, the cost of the information to collect from S_j, corresponding to the service referenced by T_i, is defined and noted by Co_{ij},
- The size of the data to collect to ensure a service is defined; for a given node S_j and a given task T_i, the data size is defined and noted by Q_{ij},
- We have partial flexibility, the realisation of each task T_i requires a node selected from a set of nodes which propose the same service performing the task T_i,with different cost, processing time and data size. Therefore, a service is described by:
  (i) A processing time P_{ij} of the task T_i on the node S_j. It corresponds to the estimated time to perform the task T_i by means of the resources of S_j,
  (ii) The cost of the service Co_{ij}corresponding to the task T_i on the node S_j,
  (iii) The data size Q_{ij} corresponding to the size of the information to collect from S_j to response to T_i.

A same task may be performed differently on several nodes, namely with different processing time, different cost and different response formats. These three characteristics (P_{ij},Co_{ij},Q_{ij}) represent successively the first, second and last term of each element of a service table. Table 1 below shows an example of different proposed services. We notice that if a provider does not offer a response to a task (partial flexibility); the correspondent term in the table above is (0,0,0). Otherwise we have $P_{ij} \neq 0$, $Co_{ij} \neq 0$ and $Q_{ij} \neq 0$.

An analogy between the problem described above and the Flexible Job Shop Problem (FJSP) can be illustrated as follows in Table 2:

Apart from the analogy expressed in Table 2 above, we manage in our problem the similarities of requests in order to avoid the same data research. Besides, we have to assign the tasks to servers (cost and delay criteria) as well as to assign remote nodes (servers) to MAs taking into account the network state (latency

TABLE 2. Analogy between FJSP and our problem.

FJSP	Our Problem
N jobs	R requests
M machines	J servers
n_j non preemptable ordered operations/job_j	n_j non preemptable ordered tasks/$request_j$
The problem: to organize the execution of N jobs on M machines	The problem: to organize the execution of R requests on S servers
The execution of each operation i of a job j ($O_{i,j}$) requires one resource or machine selected from a set of available machines	The execution of each task i (T_i) of a request j (req_i) requires one resource or server selected from a set of available servers (similarities of requests)
The assignment of the operation $O_{i,j}$ to the machine M_k entails the occupation of this machine during a processing time called $d_{i,j,k}$	The assignment of the task T_i to the server S_k entails the occupation of this server during a processing time called P_{ik}
At a given time, a machine can only execute one operation: it becomes available to other operations one the operation which is currently assigned to is completed (resource constraints).	At a given time, a server can only execute one task: it becomes available to other tasks one the task which is currently assigned to is completed (resource constraints).

criterion). Therefore our problem presents more difficulty than the FJSP which has been shown to be NP-hard.

3. Adopted approach

To resolve the problem described in last section, we propose a system based on the coordination of five kinds of software agents [8,21] (Fig. 4):

(1) *Interface Agents (IA).* These agents interact with system users, allowing them to choose appropriate form of responses to their demands so IA agents manage requests and then display results. When a MN customer access to the MIS, an agent IA deals with the formulation of his request

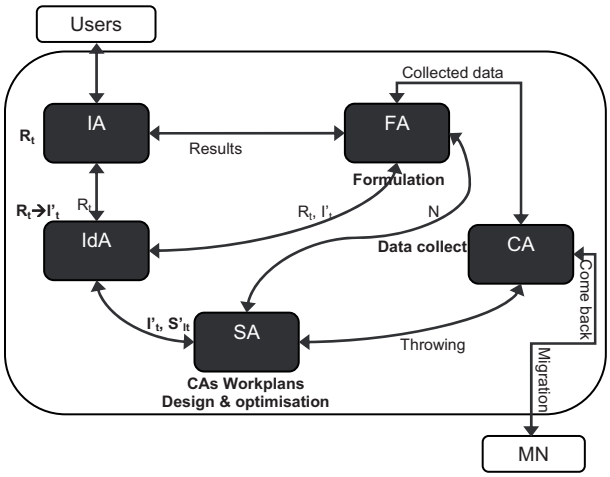

FIGURE 4. Multi-agent approach.

and then sends it to an available identifier agent. This one relates to the same platform to which several users can be simultaneously connected, thus it can receive several requests formulated at the same time. An identifier agent has to identify and to choose nodes which propose services corresponding to the requests of the users.

(2) *Identifier agents (IdA).* These agents decompose received requests into sub-requests corresponding, for example, to sub-routes or to well-known geographical zones. Sub-requests are elementary independent tasks to be performed by the available set of nodes (information providers) on the MN. Each node must login to the system registering all proposed services. A service corresponds to the response to a defined task with fixed cost, processing time and data size. Therefore, the agent IdA decomposes the set of simultaneous available requests into a set of independent tasks recognizing possible similarities, in order to avoid a redundant data research. The decomposition process occurs during the identification of the information providers. Finally, the agent IdA transmits all generated data to available scheduler agents which must optimize the choice of MN nodes, taking into account some system constraints.

(3) *Scheduler Agents (SA).* Several nodes may propose the same service with different cost and processing time and data size. The agent SA has to assign nodes to tasks minimizing total cost and processing time in order to respect due dates (data constraint). Selected set of nodes corresponds to the sequence of nodes building Workplans (routes) of the data collector agents. The agent SA has firstly to find an effective number of collector agents then he has to optimize the assignments of nodes to different tasks. This behaviour will be developed later.

(4) *Collector agents (CA).* An agent CA is a mobile software agent which can move from a node to another through a network in order to collect needed data. This special kind of agent is composed of data, code and a state [2]. Collected data should not exceed a capacity threshold in order to avoid overloading the MA. Therefore, the agent SA must take into account this aspect when assigning nodes to tasks. When they come back to the system, the agents AC must transmit collected data to available fusion agents.

(5) *Fusion Agents (FA).* These agents have to fusion correctly collected data in order to compose responses to simultaneous requests. The fusion procedure progresses according to the collected data availability. Each new answer component must be complementary to the already merged ones. Providers are already selected and tasks are supposed independent. Therefore, there is no possible conflict. A response to a request may be complete if a full answer is ready because all concerned components are available. It can be partial if at least a task composing the request was not treated, for example, because of an unavailable service. Finally, a response can be null if no component is available. If an answer is partial, the correspondent result is transmitted to the concerned user through the agent IA which deals with request reformulation, with or without the intervention of the user.

The agent IdA manages the simultaneous requests formulated by the users, decomposing them into a set of independent tasks. The decomposition process includes the identification of the requests similarities in order to formulate a set of autonomous and independent tasks which are waiting for responses at the same time t. Each task represents a service which can be proposed by different MN nodes, with different cost, processing time and data size. To response to tasks, needed data is available through the MN and their collect correspond to the CA agents job. Therefore, the SA agent must optimize the assignments of nodes to tasks, minimizing total cost and processing time, in order to respect due dates. To this assignment problem, we propose a two-level optimization solution. The first level aims to find a suitable number of CAs building their initial Workplans in order to explore the MN completely. The second level corresponds to the nodes selection increasing the number of satisfied users. These two complementary levels are detailed respectively through Sections 4 and 5.

4. The workplan problem

Latency is the needed time for a data packet to cross a network connection, from sender to receiver. It varies according network state and can affect data transmission through distributed applications. Therefore, using CA agents can decrease considerably network traffic because they do not require simultaneous connection among different nodes [3]. In this section, we consider the Workplans design problem [11, 12, 13]. In a previous work [14], we proposed a Workplan design scheme which aims to find a suitable number of CA agents minimizing

their navigation time in order to explore all MN nodes, taking into account network latency.

A. Hypothesis

In this part, we consider the following assumptions:

– Collecting data on a visited node uses processing time. We suppose that the size of the data to collect from a network node is equal to the average of the total data size on this node,

– Initially, we assume that a CA agent is not totally empty because it contains an initial quantity of data Q_0,

– We suppose that minimal latency between each pair of nodes of the network is available thanks to an existant network monitoring module,

– Information can have a multimedia aspect, so we can assume that the transmission of a quantity of data from a node to another depends on current latency.

B. Description

The MA Workplan problem can be described as follows:

AC agents are created and initially launched from an originally node (Home node). The other network nodes represent available information providers where a CA agent can move to collect data corresponding to claimed services. These ones are expressed in term of independent tasks. A same service can be proposed by different nodes with different cost, processing time and different formats. On Home node, processing time is null. We call a service response time on a node, the processing time of the correspondent task to this service on this node. Latencies are known and may affect navigation time of CA agents. Our goal is to minimize CA agent number and their navigation time in order to explore all the MN, taking into account network latency. We introduce some definitions using variables described in Table 3 below.

Definition 1. $L_{\min}(S_i, S_j)$ (Shortest latency between nodes S_i and S_j) evaluated by shortest paths linking all pairs of nodes [15]. It corresponds to the minimum length of time needed to transfer a data packet from S_i to S_j.

Definition 2. CT_i (Processing time on node S_i) corresponds to needed computing time on the node S_i to extract the data quantity Qt_i.

Definition 3. Qt_i (Data quantity on node S_i) it is the average data size to extract from S_i.

T_i represents a task corresponding to a service proposed by S_j. Therefore if nt_j is the number of all available services on the node S_j then (Sect. 2):

$$Qt_j = \frac{\sum_{i=1}^{nt_j} Q_{ij}}{nt_j} \text{ and } CT_j = \frac{\sum_{i=1}^{nt_j} P_{ij}}{nt_j}.$$

TABLE 3. Notations.

Variable	Description
m	CA agents number
$CA_1,\ldots,CA_m$	CA agents identifiers
H	Home node
Wk_i	Nodes sequence representing a CA_i agent Workplan: $(S_{i1},\ldots,S_{ip})$ with $1 \leqslant p \leqslant J$
$T(Wk_i)$	Routing time for Wk_i
$Qte_{k,u}$	The size of the data transported by CA_k from node S_u
$Tr(Qte_{k,u}, S_u, S_{u+1})$	Transmission time for $Qte_{k,u}$ from node S_u to node S_{u+1}
CT_i	Processing time on node S_i
Qt_i	Data quantity on node S_i
$L_s(S_i, S_j)$	Shortest latency between nodes S_i and S_j

Definition 4. $Qte_{k,u}$ (Data quantity transported from S_u by CA_k) corresponds to collected data quantity by CA_k during its route, until node S_u. It is computed by:

$$Qte_{k,u} = Q_0 + \sum_{r=1}^{u} Qt_r.$$

Definition 5. $\mathrm{Tr}(Qte_{k,u}, S_u, S_{u+1})$ (Transmission time) needed time for the agent CA_k to migrate from S_u to S_{u+1} transporting the data quantity $\mathrm{Qte}_{k,u}$. It is computed as follows:

$$\mathrm{Tr}(\mathrm{Qte}_{k,u}, S_u, S_{u+1}) = \frac{\mathrm{Qte}_{k,u}}{\mathrm{Qte}_{\text{paquet } R}} \times L_s(S_u, S_{u+1}).$$

$\mathrm{Qte}_{\text{paquet } R}$ is the size of a data packet on the current network R.

Definition 6. $T(Wk_k)$ (Routing time for Wk_k) needed time for the agent CA_k to visit the sequence of network nodes $(S_{k1},\ldots,S_{kp})$ with $1 \leqslant p \leqslant J$. Routing time is computed as follows:

$$T(Wk_k) = T_{\text{go}} + T_{\text{travel}} + T_{\text{return}}.$$

p	Wk_k	T_{go}	T_{return}	T_{travel}
1	(S_{k1})	$\mathrm{Tr}(Q_0, H, S_{k1})$	$Tr(Q_0 + Qt_{k1}, S_{k1}, H)$	CT_1
$1 < p \leqslant J$	$(S_{k1},\ldots,S_{kp})$		$\mathrm{Tr}(Q_0 + Qte_{k,p}, S_{kp}, H)$	X_k

With:

$$X_k = \sum_{i=1}^{p} CT_{ki} + \sum_{i=1}^{p-1} Tr(Qte_{k,i}, S_{ki}, S_{ki+1}).$$

TABLE 4. Example of available services (Processing time and data size).

	S_1	S_2	S_3	S_4
T_1	(0,0)	(2,3)	(4,3)	(2,3)
T_2	(2,5)	(1,2)	(4,1)	(3,3)
T_3	(1,3)	(0,0)	(2,3)	(4,2)
T_4	(3,1)	(0,0)	(0,0)	(0,0)
T_5	(2,1)	(1,3)	(4,2)	(4,3)

C. Proposed Workplan scheme

To propose a cost-effective Workplan MA scheme, we assume that a monitoring module exists in the system providing information about network status (latency, bandwidth, traffic, bottleneck, failure ...). Therefore, we can get latencies values between all pairs of nodes on the MN. Latencies vary according to network state and are related to the data flow. We assume that MAs are created only in the Home node and are not cloned in other nodes as was done in [11]. It is clear that sending a MA to each node gives us the best total computing time because, in this case, agents are launched simultaneously into each network node. Our goal is to keep this best total computation time to built nodes partitions, minimizing the number of MAs. A well known Travelling Salesman Problem (TSP) algorithm [17] must be applied to each partition in order to optimize each agent local routing path. To resolve the same problem, Baek proposed a cost-effective MA planning algorithm [16]. We adopt the same approach considering data transmission through the network. In current section, we aim to find a set of CA agents minimizing their navigation time in order to explore all MN nodes, taking into account network latency. Consequently, we do not care about data cost (see Sect. 2). In this case, the service table will not express information cost of a service on a node. So the new service table presented by Table 4 below, will just express processing time and data size (P_{ij},Q_{ij}).

The algorithm description

As described previously, $L_s(S_i, S_j)$ namely shortest latency between two network nodes S_i and S_j, is available. The algorithm is described as follow:

Step 1. Sort nodes in decreasing order according to their correspondent routing time $T(Wk_i = S_i) \forall 1 \leqslant i \leqslant J$. Set the threshold δ which is the routing time of the first node in the sorted list:

$$\delta = \max_{1 \leqslant i \leqslant J} (T(Wk_i = S_i)).$$

Step 2. Partition the given network into several parts by gathering nodes so that the routing time of each part does not exceed the threshold δ.

Step 3. Run a TSP algorithm to optimize each routing path.

Step 4. Allocate a CA agent for each Workplan in order to explore the correspondent network partition.

We propose a dynamic algorithm which tries to find the next node from the current node where the agent resides. In other words, this algorithm looks for the next node for a part calculating new routing time. A node is selected if the new routing time does not exceed the threshold δ. Otherwise, a Workplan is ready to be assigned to a CA and the algorithm ends if each MN node belongs to a routing path.

The algorithm above disperses all nodes into a set of CA agents in order to explore the MN totally. This step fixes the needed number m of CA agents and organizes their initial Workplans. In the next section, a sub-set S' of MN will be identified thanks to an evolutionary method, in order to optimize the computing flow management. Therefore, some nodes may not be selected, what risks to modify the number of CA agents. It is where all nodes composing a Workplan of a CA agent, are not selected. In this case, the number of CA agents is reduced. The goal is to optimize the selection of the nodes from the set of information providers proposing the required customers services. Final Workplans will be deduced from initial ones in order to collect needed data satisfying users' requests as soon as possible with reasonable costs. Let m' be the new number of CA agents. We have also $J' = |S'|$ the new number of nodes so $m' \leqslant m$, $J' \leqslant J$ and $S' \leqslant S$.

5. Data flow optimisation

Evolutionary algorithms (EA), inspired from genetic algorithms, added a new aspect to the field of artificial intelligence. These algorithms use various computational models of evolutionary processes to solve problems on a computer. EA are stochastic search methods that mimic the metaphor of natural biological evolution; they operate on a population of potential solutions applying principle of survival of the fittest, to produce successively better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the population, then breeding them together using genetic operators such as crossover and mutation [18]. Compared to traditional optimization methods such as gradient descent, EA are robust and global search technique. For this reason, the scheduling community has been quick to realize the potential of EA. In this section [19], we consider the important characteristics of EA and their relevance for solving the traditional NP-complete problem. Therefore, some points must be clarified:

- A specific genetic representation (or encoding) appropriate to the problem, to determine feasible solutions of the scheduling optimization problem,

Algorithm 1. ***Workplans Algorithm***

Step 1: Sort nodes
M:=0;
for i:=1 to J
Wk_i:=S_i;
Sort nodes by decreasing order, according to their
correspondent routing time: $T(Wk_i = S_i)$;
Let R_d=(S_{d1}, ...,S_{dn}) be the resulting sequence, so
δ=$T(Wk_1 = S_{d1})$;

Step 2 : Build Workplans
for j :=1 to J
{
$Wk_j = \emptyset$;
If $\exists$ "unassigned" S_{dk} where k is minimum
{
Select it;
m:= m+1;
Union (Wk_j, S_{dk});
Mark S_{dk}as "assigned";
}//end if
else terminate; // end step2
while (*true*)
{
Sort nodes according to $L(S_{dk}, S_{dl})$ with
$1 < l \leqslant J$ and
$S_{dl}$is "unassigned" by increasing order
Let (S_{e1}, ...,S_{eg}) be the resulting sequence;
previous_k := k;
for y:= 1 to g
{
Union (Wk_j,S_{ey});
if $T(Wk_j) \leqslant \delta$
{
mark S_{ey} as "assigned";
find k while dk=ey;
break for-loop;
}//end if
else Wk_j:= Wk_j –{S_{ey}};
} //end inner for-loop
if *previous_k*=k
break while-loop;
}// end while-loop
} // end outer for-loop

Step 3 : Run a simple TSP algorithm to optimize
each Wk_k $\forall 1 \leqslant k \leqslant m$.

- Original genetic operators that alter the composition of children during reproduction. As it was mentioned previously, a task (sub-request) must be managed by only one provider, selected from the set of nodes which propose the correspondent service (see Sect. 2). Therefore, we choose to correct generated solutions in order to respect this constraint. Consequently, each crossover or mutation operation must be followed by a correction process,
- Parents are selected randomly from current population to crossover with some probability p_c ($0 < p_c < 1$). We believe that this technique gives more chance to weak individuals to survey with minimum computation time,
- A non-elitist replacement technique is adopted to generate the new population from the previous one,
- The evaluation function estimates a possible solution according to two criteria: the cost and the delay (Sect. 5).

A. Representation of a solution

The selection of an appropriate representational scheme of a solution is fundamental to the success of EA applications. That's why we try to find an efficient coding for the chromosome respecting our problem constraints. Therefore, we propose a flexible representation of the chromosome called Flexible Tasks Assignment Representation (FeTAR). The chromosome is represented by a matrix CH*(I'×J')* where rows represent independent tasks, composing globally simultaneous requests and columns represent nodes. Each element of the matrix specifies the assignment of a node S_j to the task T_i as follow:

$$\mathrm{CH}[i,j] = \begin{cases} \textbf{1:} \text{ if } S_j \text{is assigned to } T_i \\ \textbf{*:} \text{ if } S_j \text{may be assigned to } T_i \\ \textbf{X} : \textbf{:} \text{ if } S_j \text{cannot be assigned to } T_i. \end{cases}$$

We notice that each task must be assigned, so we assume that each task must be performed at least by a node, selected from a set of nodes proposing the service which corresponds to a response to the concerned task.

Example. We consider 4 nodes and 3 requests, decomposed globally into 5 tasks as follow:
req_1 decomposed into T_1, T_2, T_3 and T_5
req_2 decomposed into T_2
req_3 decomposed into T_1, T_3 and T_4

A possible solution has a *FeTAR* scheme, specified as follows:

CH	S_1	S_2	S_3	S_4
T_1	X	1	*	*
T_2	*	*	1	*
T_3	1	X	*	*
T_4	1	X	X	X
T_5	*	*	1	*

B. Selection

Selection is one of the most important aspects of all EA. It determines which individuals in the population will have all or some of its genetic material passed on the next generations. The objective of the selection method employed in EA, is to give polynomial increasing trials to fittest individuals. We use random technique, to give chance to weak individuals to survey: parents are selected randomly from current population to crossover with some probability p_c $(0 < p_c < 1)$.

C. Genetic operators

Crossover operator

Crossover involves combining elements from two parent chromosomes into one or more child chromosomes. The role of the crossover is to generate a better solution by exchanging information contained in the current good one [18]. We propose a single crossover operator presented by the following algorithm:

Algorithm *CrossFeTAR*

C_1 (resp. C_2) representing the child 1 (resp. child 2) is given by the algorithm below:

Step 1:
Choose randomly two parents and one node:
Suppose that P_1, P_2 and S_{cj} $(1 \leqslant j \leqslant J')$ are randomly selected,

Step 2:
Tasks assignment of S_{cj} in C_1 (resp. C_2) must correspond to the same assignment of S_{cj} in P_1 (resp. P_2)

Step 3:
$k := 1$;
while $(k \leqslant J')$ and $(k \neq j)$
{
Tasks assignment of S_{ck} in C_1 (resp. C_2) must correspond to the same assignment of S_{ck} in P_2(resp. P_1);
$k := k+1$;
}

Step 4:
Correct randomly C_1 and C_2;

Crossover operator is illustrated by the example below:

Example of crossover

Step 1. Assume that P_1, P_2 and S_2 represent respectively two chromosomes and one node, randomly selected. S_2 corresponds to the second column of the matrix representing the chromosome:

Parent 1

P_1	S_1	S_2	S_3	S_4
T_1	X	1	*	*
T_2	*	*	1	*
T_3	1	X	*	*
T_4	1	X	X	X
T_5	*	*	1	*

Parent 2

P_2	S_1	S_2	S_3	S_4
T_1	X	*	*	1
T_2	*	*	1	*
T_3	*	X	1	*
T_4	1	X	X	X
T_5	1	*	*	*

Step 2. Tasks assignment of S_2 in C_1 (resp. C_2) must corresponds to the same assignment of S_2 in P_1(resp. P_2)

Child 1

C_1	S_1	S_2	S_3	S_4
T_1		1		
T_2		*		
T_3		X		
T_4		X		
T_5		*		

Child 2

C_2	S_1	S_2	S_3	S_4
T_1		*		
T_2		*		
T_3		X		
T_4		X		
T_5		*		

Step 3. Tasks assignment of S_1, S_3 and S_4 in C_1 (resp. C_2) must corresponds to the same assignment of S_1, S_3 and S_4 in P_2 (resp. P_1)

Child 1

C_1	S_1	S_2	S_3	S_4
T_1	X	*	*	1
T_2	*	1	1	*
T_3	*	X	1	*
T_4	1	*	X	X
T_5	1	*	*	*

Child 2

C_2	S_1	S_2	S_3	S_4
T_1	X	1	*	*
T_2	*	*	1	*
T_3	1	X	*	*
T_4	1	X	X	X
T_5	*	*	1	*

Step 4. C_1 is not a feasible solution contrary to C_2 because C_1 assigns task T_2 more than one time (in C_1, S_2 and S_3 are assigned simultaneously to T_2). After correction, C_1 will be randomly:

C_1	S_1	S_2	S_3	S_4
T_1	X	*	*	1
T_2	*	1	*	*
T_3	*	X	1	*
T_4	1	*	X	X
T_5	1	*	*	*

Or

C_1	S_1	S_2	S_3	S_4
T_1	X	*	*	1
T_2	*	*	1	*
T_3	*	X	1	*
T_4	1	*	X	X
T_5	1	*	*	*

Mutation operator

Mutation represents another important genetic operator. Although mutation is important, it is secondary to crossover .It introduces some extra variability into

the population and typically works with a single chromosome to create a new modified one [18]. The mutation algorithm is presented as follow:

Algorithm *MuteFeTAR*

Step 1:
Choose randomly one chromosome CH, one task $T_{ci}(1 \leqslant i \leqslant I')$ and one node S_{cj} $(1 \leqslant j \leqslant J')$;
Step 2:
if CH[i,j]= *
{
find j_1 with $1 \leqslant j_1 \leqslant$J' and
CH[i,j_1] = 1;
CH[i,j_1]:= *;
CH[i,j] := 1;
}
else
if CH[i,j]=1 and $\exists$j$_1$ with $1 \leqslant j_1 \leqslant J'$ and
CH[i,j_1] = $*$
{
CH[i,j_1] := 1;
CH[i,j] := *;
}

Mutation operator is illustrated by the example below:

Example of mutation

We use the same data of the example presented previously:

Step 1. Choose randomly one chromosome, one task and one node. We assume that the chromosome CH, the task T_3 and the node S_1 are randomly selected:

Chromosome

CH	S_1	S_2	S_3	S_4
T_1	X	1	*	*
T_2	*	*	1	*
T_3	1	X	*	*
T_4	1	X	X	X
T_5	*	*	1	*

Step 2. CH*[3,1]*=1 and $\exists j_1$ (3 or 4) with CH*[3,*j_1*]* = * so replace CH*[3,1]* with *, choose randomly j_1 (3 for example) and finally replace CH*[3,3]* with 1. New chromosome is presented as follow:

New Chromosome

CH	S_1	S_2	S_3	S_4
T_1	X	1	*	*
T_2	*	*	1	*
T_3	*	X	1	*
T_4	1	X	X	X
T_5	*	*	1	*

D. Evaluation functions

At each iteration, individuals (chromosomes) in the current population are evaluated according to the same measure of fitness. There is a number of characteristics of the evaluation function that enhance or hinder the evaluation of a program performance [18,20]. In our case, the fitness function intends to maximize the number of satisfied transport travellers, minimizing response delay and total cost. In other words, a chromosome is firstly evaluated according to the number of responses which respect due dates, namely responses minimizing correspondent ending dates and respecting correspondent due dates. Then a solution is evaluated according to its cost. Therefore, a chromosome has to express ending responses date and the information cost. To illustrate the evaluation of individuals in a population, we keep the same example previously presented: 4 nodes and 3 requests decomposed globally into 5 tasks as follow:

$$
\begin{aligned}
&Req_1 \text{ decomposed into } T_1, T_2, T_3 \text{ and } T_5 | d_1 = 10\\
&Req_2 \text{ decomposed into } T_2 | d_2 = 5\\
&Req_3 \text{ decomposed into } T_1, T_3 \text{ and } T_4 | d_3 = 10
\end{aligned}
$$

A same node (information provider) may propose different services and a same service may be proposed indifferently by several nodes. In current section, we aim to minimize processing time on MN nodes, reducing collected data cost in order to optimize the data flow management. Consequently, we do not care about collected data quantity (Sect. 2). In that follows, a service table will not express data size of a service on a node. So the new service table presented by Table 5 below, expresses the processing time and the cost (P_{ij},Co_{ij}).

TABLE 5. Example of available services (Processing time and cost).

	S_1	S_2	S_3	S_4
T_1	X	*	*	*
T_2	*	*	*	*
T_3	*	X	*	*
T_4	*	X	X	X
T_5	*	*	*	*

FeTAR design corresponding to the service table below (Tab. 5) is:

	S_1	S_2	S_3	S_4
T_1	(0,0)	(2,5)	(4,3)	(2,5)
T_2	(2,4)	(1,5)	(4,5)	(3,8)
T_3	(1,7)	(0,0)	(2,5)	(4,2)
T_4	(3,2)	(0,0)	(0,0)	(0,0)
T_5	(2,3)	(1,1)	(4,5)	(4,5)

A possible solution, instance of the FeTAR is:

	S_1	S_2	S_3	S_4
T_1	X	1	*	*
T_2	*	*	1	*
T_3	1	X	*	*
T_4	1	X	X	X
T_5	*	*	*	1

A request req_w is decomposed into $I_{t,w}$ tasks. Therefore, the total processing time $EndReq_w$ for each req_w is computed by the means of the algorithm *fitness_1* below. This time includes only the effective processing time on the MN. We assume that, the ending date D_w corresponding to the total execution time of a request req_w, includes also the average navigation time of CA agents (Sect. 3). This is expressed by:

$$\gamma = \delta - \frac{\sum_{j=1}^{J} CT_j}{J} \tag{1}$$

$$\Rightarrow \quad \forall 1 \leqslant w \leqslant R, D_w = EndReq_w + \gamma \tag{2}$$

Algorithm 2. *Fitness_1* algorithm

Step 1:

m' is the CA agents number so
$\forall k$ with $1 \leqslant k \leqslant m'$, initialize :

- The set of tasks $U_{ck}$to $\emptyset$
- Total time $EndU_{ck}$ to perform U_{ck} to 0

Step 2:

Look for the set of tasks U_{ck} performed by each CA_{ck} and their processing time $EndUk$ as follows:

```
for k:= 1 to m'
  for j := 1 to J'
    for i := 1 to I'
          if S_cj belongs to the Workplan of
CA_ck and S_cj is assigned to T_ci
          {
          U_ck:= U_ck∪{T_ci};
          EndU[ck] :=EndU[ck]+P_cicj;
          }
```

Step 3:

Compute processing time of each request require the identification of CA agents which perform tasks composing the request. Total processing time of a request is the maximum processing times of all CA agents which perform tasks composing this request. This is calculated as follow:

```
for w := 1 to R
 {
 for k := 1 to m'
   treatedAC[ck] := false;

        i := 1;
while i ⩽ I' and ∃k_1/1⩽   k_1 ⩽ m' and
        treatedAC[ck_1]=false
 {
    if T_ci   ∈ req_w
 {
    ck := 1;
 while k ⩽ m' and T_i ∉ U_k
    ck:= ck+1;//end while
  if ¬TreatedAC[ck]
        {
    EndReq[w] := max(EndReq[w],
EndU[ck]);
        TreatedAC[ck] := true;
        }//end if
      }//end if
  } //end while
 }//end outer for-loop
```

Total cost of a request req_w is *CostReq[w]* expressed by C_w, is given by the mean of the algorithm below:

Algorithm 3. *Fitness_2* algorithm

Repeat steps 1 and 2 for each request req_w ($1 \leqslant w \leqslant R$)
Step 1:
    CostReq[w] := 0
Step 2:
    for i:=1 to I'
    {
    if $T_{ci} \in req_w$ {
      find the node S_{cj} ($1 \leqslant j \leqslant J'$) assigned to T_{ci}
      in *FeTAR* instance
      CostRe[w] := *CostRe[w]* + Co_{cicj}
    }//end if
  }// end for

Knowing that using expression (1), we can deduce ending date from *fitness_1* algorithm, the new FeTAR representation of the chromosome express for each request req_w $1 \leqslant w \leqslant R$, its ending date and its cost; it is illustrated by the example below:

	S_1	S_2	S_3	S_4	w	d_w	C_w	D_w
T_1	X	1	*	*	1	10	5	6
T_2	*	*	1	*	2	5	1	1
T_3	1	X	*	*	3	10	4	2
T_4	1	X	X	X				
T_5	*	*	*	1				

Knowing that due dates of requests req_1, req_2 and req_3 are respectively 10, 5 and 10, the solution satisfy all requests at latest in 6 units of time and has an average cost equals to 2.66 units of cost.

E. Evolutionary approach

To determinate best solutions, we adopt an elitist approach [22] using an external storage to memorise the most adapted individuals during the search. The evolutionary adopted approach is shown by the diagram below (Fig. 5). During the evaluation process to crossover and mutation, best solutions are saved in external archives.

Knowing that $d_{\max} = \max(d_w)_{1 \leqslant w \leqslant R}$, we discern two archive sets:

- Main solutions archive (M) representing best solutions which respect all due dates. In other words, a chromosome CH∈M if and only if $\forall 1 \leqslant w \leqslant R$, $D_w \leqslant d_{\max}$. this archive set is decomposed into two sub-archives:
  - dominant solutions archive (d);
  - $\varepsilon-$dominant solutions archive ($\varepsilon - d$).

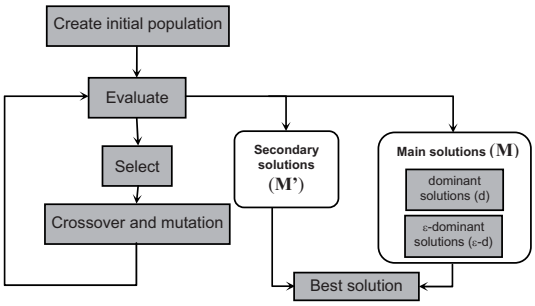

FIGURE 5. Evolutionary approach.

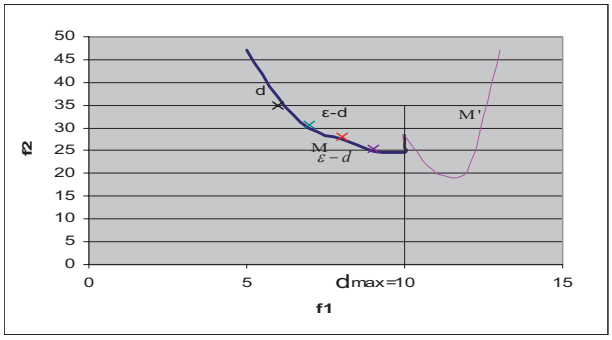

FIGURE 6. Optimal fronts.

– Secondary solutions archive (M') representing best solutions which exceed at least one due date. In other words, a chromosome CH$\in M'$ if and only if $\exists w$, $1 \leqslant w \leqslant R$ and $d_{\max} < D_w$.

We discerned two archive sets according to delay criterion satisfaction because we assume that if a response exceeds its due date, the user is not satisfied. That's why we consider that the first criterion has more priority that the second one. Consequently archive sets are firstly sorted according to delay criterion, then according to cost criterion. We notice by f1 the response delay function evaluated by fitness_1 algorithm and by f2 the cost function evaluated by fitness_2 algorithm (Sect. 5). Considering two different solutions CH and CH', if CH$\in$M and CH'$\in$M' then CH dominates CH'. Otherwise, CH dominates CH' if and only if f1(CH)=f1(CH') and f2(CH)<f2(CH').

Each archive set has a maximum number size equals to the population size. If the number of individuals in an archive exceeds this fixed size, a crowding process [27] must occur to decide which solutions must kept in the archive. The non selected solutions are deleted; and the others contribute to the next selection procedure; archive members can then transmit their characteristics to offspring

Table 6. Requests decomposition.

w	1	2	3	4
T_2		•		
T_3		•	•	
T_5	•	•	•	•
T_6	•			
T_7			•	•
T_8	•	•	•	
T_9		•		
T_{10}		•	•	
T_{11}	•	•	•	
T_{12}			•	
T_{13}	•			•
T_{14}		•		
T_{15}	•		•	
d_w	1000	1000	1000	1000

• Means that T_i composes req_w

populations. ***M*** and ***M'*** archive sets represent generated solutions having minimum f_1 and f_2 values, so if a chromosome CH of the offspring dominates any archive member *CH'*, the archive member is deleted and the offspring is accepted. Figure 6 above represents the Pareto-optimal fronts with $\varepsilon_1 = \varepsilon_2 = \varepsilon = 0.75$ [22]. We use a population size of $N = 100$ with a crossover probability $p_c = 0.8$ and mutation probability $p_m = 0.2$.

6. Simulation results

6.1. Needed data

A. Requests decomposition and due dates

We consider 4 requests decomposed globally into 13 tasks (see Tab. 6). We remind that d_w represents the due date of the request req_w ($1 \leqslant w \leqslant 4$). We have $I'_t \subseteq I_t$, the set of tasks representing services composing simultaneous requests from the set of proposed services on the whole network.

B. Shortest latencies

We consider 20 nodes representing information providers on the network. The latency range is 10 ms $\sim$ 100 ms.

TABLE 7. Available services.

	S_1	S_2	S_3	S_4	S_5
T_1	(8,2,5)	(0,0,0)	(5,3,10)	(2,7,5)	(10,2,3)
T_2	(8,3,4)	(0,0,0)	(10,3,3)	(3,7,6)	(0,0,0)
T_3	(8,5,4)	(0,0,0)	(8,3,2)	(4,3,4)	(4,10,4)
T_4	(9,5,7)	(2,5,4)	(1,9,7)	(2,7,8)	(3,4,3)
T_5	(8,10,3)	(0,0,0)	(2,6,7)	(8,8,10)	(8,2,10)
T_6	(8,6,5)	(0,0,0)	(9,9,2)	(8,2,9)	(2,2,10)
T_7	(0,0,0)	(0,0,0)	(6,7,1)	(9,3,8)	(0,0,0)
T_8	(9,7,6)	(0,0,0)	(7,8,3)	(1,7,3)	(0,0,0)
T_9	(0,0,0)	(0,0,0)	(0,0,0)	(10,7,3)	(0,0,0)
T_{10}	(0,0,0)	(0,0,0)	(0,0,0)	(9,4,5)	(10,6,7)
T_{11}	(0,0,0)	(0,0,0)	(10,6,7)	(3,10,3)	(10,9,8)
T_{12}	(7,6,4)	(0,0,0)	(1,3,2)	(9,8,2)	(2,1,10)
T_{13}	(1,7,4)	(0,0,0)	(9,6,5)	(1,6,3)	(3,4,2)
T_{14}	(0,0,0)	(0,0,0)	(0,0,0)	(2,10,8)	(0,0,0)
T_{15}	(3,7,8)	(0,0,0)	(10,7,4)	(4,10,9)	(8,8,7)

	S_6	S_7	S_8	S_9	S_{10}
T_1	(9,9,6)	(9,8,5)	(1,10,6)	(2,3,2)	(9,9,8)
T_2	(0,0,0)	(4,9,6)	(6,5,9)	(4,6,3)	(2,3,9)
T_3	(0,0,0)	(7,4,1)	(8,6,5)	(8,8,9)	(6,8,9)
T_4	(0,0,0)	(3,7,10)	(1,8,3)	(10,10,9)	(10,10,4)
T_5	(0,0,0)	(10,2,10)	(4,9,3)	(9,4,5)	(3,6,6)
T_6	(0,0,0)	(2,8,7)	(3,8,4)	(2,7,1)	(4,3,2)
T_7	(0,0,0)	(0,0,0)	(7,9,3)	(2,10,4)	(0,0,0)
T_8	(0,0,0)	(7,1,3)	(5,3,8)	(2,6,9)	(10,3,9)
T_9	(0,0,0)	(0,0,0)	(3,10,3)	(10,5,8)	(1,1,3)
T_{10}	(0,0,0)	(0,0,0)	(0,0,0)	(9,8,10)	(8,10,10)
T_{11}	(0,0,0)	(9,10,3)	(7,2,3)	(8,3,4)	(3,2,9)
T_{12}	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(9,8,8)
T_{13}	(0,0,0)	(0,0,0)	(8,7,2)	(4,5,1)	(9,5,8)
T_{14}	(0,0,0)	(10,6,1)	(0,0,0)	(0,0,0)	(2,10,5)
T_{15}	(0,0,0)	(0,0,0)	(2,8,5)	(2,9,4)	(8,1,1)

C. AVAILABLE SERVICES

We have 20 information providers on the network proposing 15 services in different way as follows in Table 7.

TABLE 7. Continued.

	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
T_1	(7,4,4)	(3,2,2)	(0,0,0)	(10,2,10)	(7,10,1)
T_2	(0,0,0)	(10,10,5)	(5,1,9)	(1,1,8)	(9,7,4)
T_3	(4,6,9)	(7,5,9)	(0,0,0)	(3,4,5)	(5,1,8)
T_4	(0,0,0)	(0,0,0)	(3,10,3)	(8,7,2)	(8,8,3)
T_5	(5,6,2)	(4,4,10)	(0,0,0)	(6,9,2)	(0,0,0)
T_6	(1,7,8)	(4,8,10)	(0,0,0)	(0,0,0)	(7,5,4)
T_7	(0,0,0)	(4,1,2)	(8,8,8)	(6,1,6)	(5,2,1)
T_8	(0,0,0)	(10,6,9)	(5,2,1)	(5,2,10)	(9,5,5)
T_9	(10,4,2)	(9,1,10)	(5,6,2)	(0,0,0)	(8,8,4)
T_{10}	(8,1,1)	(0,0,0)	(0,0,0)	(10,3,2)	(0,0,0)
T_{11}	(0,0,0)	(10,1,10)	(0,0,0)	(7,3,7)	(3,9,1)
T_{12}	(9,5,4)	(0,0,0)	(4,3,9)	(0,0,0)	(8,5,10)
T_{13}	(4,8,3)	(8,4,6)	(0,0,0)	(0,0,0)	(7,2,6)
T_{14}	(6,10,2)	(3,3,2)	(6,1,9)	(5,2,3)	(0,0,0)
T_{15}	(3,4,6)	(9,7,3)	(2,2,6)	(0,0,0)	(7,2,9)

	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}
T_1	(0,0,0)	(2,6,8)	(5,7,7)	(0,0,0)	(0,0,0)
T_2	(0,0,0)	(8,10,9)	(5,1,6)	(9,7,1)	(3,6,4)
T_3	(1,4,5)	(0,0,0)	(4,10,9)	(4,1,4)	(6,5,5)
T_4	(1,10,10)	(0,0,0)	(0,0,0)	(0,0,0)	(5,10,6)
T_5	(9,6,8)	(4,4,2)	(6,5,9)	(6,6,10)	(1,2,10)
T_6	(10,3,7)	(8,9,8)	(5,2,7)	(4,5,3)	(10,2,3)
T_7	(0,0,0)	(0,0,0)	(9,9,10)	(10,5,6)	(1,2,3)
T_8	(5,5,7)	(6,8,7)	(1,2,9)	(0,0,0)	(2,6,5)
T_9	(0,0,0)	(1,4,8)	(0,0,0)	(0,0,0)	(9,1,5)
T_{10}	(4,1,8)	(9,10,1)	(0,0,0)	(6,5,10)	(0,0,0)
T_{11}	(1,2,7)	(9,1,10)	(5,10,10)	(7,3,6)	(6,8,1)
T_{12}	(5,10,3)	(10,9,7)	(9,8,8)	(0,0,0)	(6,10,9)
T_{13}	(4,6,2)	(0,0,0)	(7,10,4)	(9,7,5)	(0,0,0)
T_{14}	(10,5,1)	(3,2,7)	(6,1,1)	(1,4,4)	(3,8,7)
T_{15}	(10,3,4)	(0,0,0)	(9,2,7)	(7,5,9)	(4,5,4)

From Table 7, we can generate for each node S_j $(1 \leqslant j \leqslant J)$, values of nt_j and CT_j corresponding the number of available services and the average of processing time on each node:

S_j	nt_j	CT_j
S_1	10	6.9
S_2	1	2
S_3	12	6.5
S_4	15	5
S_5	10	6
S_6	1	9
S_7	9	6.778
S_8	12	4.583
S_9	13	5.538
S_{10}	14	6
S_{11}	10	5.7
S_{12}	12	6.75
S_{13}	8	4.75
S_{14}	10	6.1
S_{15}	12	6.916
S_{16}	11	5.45
S_{17}	10	6
S_{18}	12	5.916
S_{19}	10	6.3
S_{20}	12	4.666

$$\text{So } \frac{\sum_{j=1}^{20} CT_j}{20} = 5.5 \text{ ms.}$$

D. Workplans

Algorithm 1 presented in Section 4 generates 9 CAs Workplans to explore totally a MN composed of 20 nodes as follows:

$\boldsymbol{Wk_1}$**=** $(\boldsymbol{S_{18}}, \boldsymbol{S_{13}}, \boldsymbol{S_{12}})$, $\boldsymbol{Wk_2}$**=** $(\boldsymbol{S_5}, \boldsymbol{S_{20}}, \boldsymbol{S_1}, \boldsymbol{S_4})$
$\boldsymbol{Wk_3}$**=** $(\boldsymbol{S_{17}}, \boldsymbol{S_3})$, $\boldsymbol{Wk_4}$**=** $(\boldsymbol{S_{16}}, \boldsymbol{S_{10}})$
$\boldsymbol{Wk_5}$**=** $(\boldsymbol{S_2}, \boldsymbol{S_8})$, $\boldsymbol{Wk_6}$**=** $(\boldsymbol{S_9}, \boldsymbol{S_{11}})$
$\boldsymbol{Wk_7}$**=** $(\boldsymbol{S_{15}}, \boldsymbol{S_{14}})$, $\boldsymbol{Wk_8}$**=** $(\boldsymbol{S_{19}}, \boldsymbol{S_7})$, $\boldsymbol{Wk_9}$**=** $(\boldsymbol{S_6})$
With $\delta = 539.8$ ms, **so** the average navigation time of CA agents is:

$$\gamma = \delta - \frac{\sum_{j=1}^{20} CT_j}{20} = 534.3 \text{ ms.}$$

According to Table 7, nodes S_2 and S_6 do not propose any service corresponding to tasks which compose simultaneous requests so J'=18. Therefore Workplans are updated as follows:

$\boldsymbol{Wk}_1 = (\boldsymbol{S}_{18}, \boldsymbol{S}_{13}, \boldsymbol{S}_{12})$, $\boldsymbol{Wk}_2 = (\boldsymbol{S}_5, \boldsymbol{S}_{20}, \boldsymbol{S}_1, \boldsymbol{S}_4)$
$\boldsymbol{Wk}_3 = (\boldsymbol{S}_{17}, \boldsymbol{S}_3)$, $\boldsymbol{Wk}_4 = (\boldsymbol{S}_{16}, \boldsymbol{S}_{10})$
$\boldsymbol{Wk}_5 = (\boldsymbol{S}_8)$, $\boldsymbol{Wk}_6 = (\boldsymbol{S}_9, \boldsymbol{S}_{11})$
$\boldsymbol{Wk}_7 = (\boldsymbol{S}_{15}, \boldsymbol{S}_{14})$, $\boldsymbol{Wk}_8 = (\boldsymbol{S}_{19}, \boldsymbol{S}_7)$, $\boldsymbol{Wk}_9 = \emptyset$

We can see here that the number of CAs changes to $m' = 8$.

6.2. Possible solution

We adjusted our program parameters as follow:

Crossover probability $p_c = 0.9$
Mutation probability $p_m = 0.2$
Population size = 100
Generations number = 5000.

The main generated solutions archive is not empty. Therefore we select a solution CH$\in M$ having the best delay response:

	S_1	S_3	S_4	S_5	S_7	S_8	S_9	S_{10}	S_{11}
T_2	*	*	*	×	*	*	*	*	×
T_3	*	*	*	*	*	*	*	*	*
T_5	*	1	*	*	*	*	*	*	*
T_6	*	*	*	*	1	*	*	*	*
T_7	×	*	*	×	×	*	1	×	×
T_8	*	*	*	×	*	*	*	*	×
T_9	×	×	*	×	×	1	*	*	*
T_{10}	×	×	*	*	×	×	*	*	*
T_{11}	×	*	*	*	*	*	*	*	×
T_{12}	*	*	*	1	×	×	×	*	*
T_{13}	*	*	1	*	×	*	*	*	*
T_{14}	×	×	*	×	*	×	×	*	*
T_{15}	*	*	*	*	×	1	*	*	*

	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}
T_2	*	*	1	*	×	*	*	*	*
T_3	*	×	*	*	*	×	1	*	*
T_5	*	×	*	×	*	*	*	*	*
T_6	*	×	×	*	*	*	*	*	*
T_7	*	*	*	*	×	×	*	*	*
T_8	*	*	*	*	*	*	1	×	*
T_9	*	*	×	*	×	*	×	×	*
T_{10}	×	×	*	×	1	*	×	*	×
T_{11}	*	×	*	1	*	*	*	*	*
T_{12}	×	*	×	*	*	*	*	×	*
T_{13}	*	×	×	*	*	×	*	*	×
T_{14}	*	*	*	×	*	1	*	*	*
T_{15}	*	*	×	*	*	×	*	*	*

6.3. Final Workplans

Nodes S_1, S_{10}, S_{11}, S_{12}, S_{13}, S_{19} and S_{20} were not selected to assign tasks. Our solution is converted into a *FeTAR* design with $I' = 13$ and $J' = 11$.

Final Workplans are:

Wk_1= (S_{18}) to perform {T_3, T_8}
Wk_2= (S_5, S_4) to perform
{T_{12}} on S_5
{T_{13}} on S_4
Wk_3= (S_{17}, S_3) to perform
{T_{14}} on S_{17}
{T_5} on S_3
Wk_4= (S_{16}) to perform {T_{10}}
Wk_5= (S_8) to perform {T_9, T_{15}}
Wk_6= (S_9) to perform {T_7}
Wk_7= (S_{15}, S_{14}) to perform
{T_{11}} on S_{15}
{T_2} on S_{14}
Wk_8= (S_7) to perform {T_6}

6.4. Fitness

fitness_1 and *fitness_2* algorithms (Sect. 5) evaluate selected solution as follow in Table 9.

We notice here that this solution satisfy all requests respecting due dates through 539.3 ms and costs 37.25 units of cost. In order to evaluate the efficiently of our proposed optimization approach, we propose to compare it to the classical client server (CS) paradigm.

TABLE 8. Generated FeTAR instance.

	S_3	S_4	S_5	S_7	S_8	S_9	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}
T_2	*	*	×	*	*	*	1	*	×	*	*
T_3	*	*	*	*	*	*	*	*	*	×	1
T_5	1	*	*	*	*	*	*	×	*	*	*
T_6	*	*	*	1	*	*	×	*	*	*	*
T_7	*	*	×	×	*	1	*	*	×	×	*
T_8	*	*	×	*	*	*	*	*	*	*	1
T_9	×	*	×	×	1	*	×	*	×	*	×
T_{10}	×	*	*	×	×	D. *	*	×	1	*	×
T_{11}	*	*	*	*	*	*	*	1	*	*	*
T_{12}	*	*	1	×	×	×	×	*	*	*	*
T_{13}	*	1	*	×	*	*	×	*	*	×	*
T_{14}	×	*	×	*	×	×	*	×	*	1	*
T_{15}	*	*	*	×	1	*	×	*	*	×	*

TABLE 9. FeTAR instance results.

	S_3	S_4	S_5	S_7	S_8	S_9	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	W	D_w	C_w
T_2	*	*	×	*	*	*	1	*	×	*	*	1	539.3	39
T_3	*	*	*	*	*	*	*	*	*	×	1	2	539.3	41
T_5	1	*	*	*	*	*	*	×	*	*	*	3	539.3	47
T_6	*	*	*	1	*	*	×	*	*	*	*	4	539.3	22
T_7	*	*	×	×	*	1	*	*	×	×	*			
T_8	*	*	×	*	*	*	*	*	*	*	1			
T_9	×	*	×	×	1	*	×	*	×	*	×			
T_{10}	×	*	*	×	×	*	*	×	1	*	×			
T_{11}	*	*	*	*	*	*	*	1	*	*	*			
T_{12}	*	*	1	×	×	×	×	*	*	*	*			
T_{13}	*	1	*	×	*	*	×	*	*	×	*			
T_{14}	×	*	×	*	×	×	*	×	*	1	*			
T_{15}	*	*	*	×	1	*	×	*	*	×	*			

7. COMPARISON

Many researchers have longer discussed the benefits of the MA paradigm and conclude that it might be efficient in some cases [28,29]. In a recent Work [30], we compared our approach with the use of the classical CS paradigm. Results are very significant. Through our system, we deduce final Workplans of CA agents according the generated FeTAR instances. Therefore, in order to response to all users' requests, the generated solution (Tab. 8) required 539.3 ms in the MA paradigm (Tab. 9) and 4076.67 ms in the CS one (Fig. 7). Besides, when we

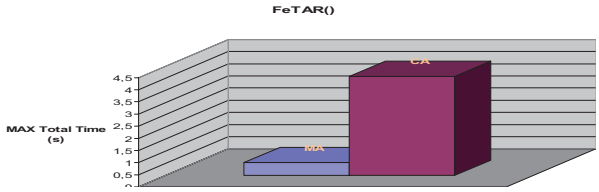

FIGURE 7. FeTAR instance result.

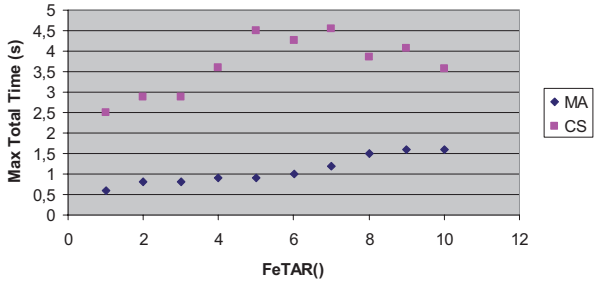

FIGURE 8. Variation of FeTAR instance.

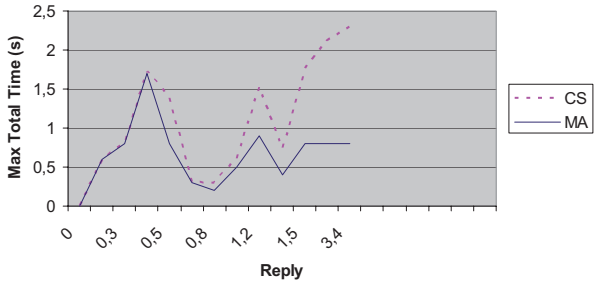

FIGURE 9. Variation of response data.

randomly generate several FeTAR instances for the same example, we observe the benefit of the use of our optimization approach, using MA paradigm instead of the CS one through our system (Fig. 8). In addition, when the quantity of response data varies, we observe that the use of MA paradigm is better (Fig. 9). The graphic is not regular because we solve the problem in a general case so we used random latencies.

8. The JADE platform

We are developing our system, with JADE platform (Java Agent Development platform) [23]. JADE is a middleware which permits a flexible implementation of multi-agents systems; it offers an efficient transport of ACL (Agent Communication Language) messages for agents communication which complies with FIPA

specifications [25]. A yellow pages services directory is available, allowing agents to advertise their services; visiting agents can then consult yellow pages looking for agents which provide the services they desire. JADE is written in java language [24], supports mobility, evolves rapidly and until there, it is the only existent multi-agent platform which tolerates web services integration [26].

9. Conclusion

In this paper, we proposed a migration strategy of MAs through a MN, optimizing the data flow management, in order to satisfy, in a better manner, customers' requests. The adopted approach decreases considerably computing time because Workplans are just deduced; they are computed when network traffic varies considerably. We aim to improve our approach according to eventual similarities, resulting from simultaneous requests sets overlap. For this purpose, we aim to detail relationships between different agents of our system, implementing their different possible kinds of cooperation.

References

[1] H. Zgaya, K. Ghédira and S. Hammadi, Proposition d'un système d'information multimodal à base d'agents mobiles destiné aux clients des réseaux de transport, in *Proc. of the International Workshop of Méthodologies et Heuristiques pour l'Optimisation des Systèmes Industriels (MHOSI'05)*, Hammamet, Tunisia 24–26 April (2005).

[2] S. Rouvrais, *Utilisation d'agents mobiles pour la construction de services distribués*. Ph.D. Thesis, University of Rennes, France (2002).

[3] W. Theilmann and K. Rothermel, Efficient Dissemination of Mobile Agents, in *Proc. of the 19*th *IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'99)*, edited by W. Sun, S. Chanson, D. Tygar and P. Dasgupta, Austin, TX, USA, 31 May–5 Juin (1999) 9–14.

[4] A. Fugetta, G.P. Picco and G. Vigna, Understanding code mobility, *IEEE Trans. Software Engineering* **24(5)** (1998) 342–361.

[5] A. Carzaniga, G.P. Picco and G. Vigna, Designing distributed applications with mobile code paradigms, in *Proc. of the 19*th *International Conference on Software Engineering (ICSE'97)*, Boston, Massachusetts, USA 17–23 May (1997). ACM Press (1997) 22–32.

[6] D. Rus, R. Gray and D. Kotz, Autonomous and adaptive agents that gather information, in *Proc. of the Thirteen National Conference on Artificial Intelligence Workshop on Intelligent Adaptive Agents (AAAI'96)*, Portland, Oregon, USA, 4–8 August (1996) 107–116.

[7] J. Yang, V. Honavar, L. Miller and J. Wong, Intelligent Mobile Agents for information retrieval and knowledge discovery from distributed data and knowledge sources, in *Proc. of the IEEE Information Technology Conference*, Syracuse, NY USA, 1–3 September (1998) 99–102.

[8] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers and R. Evans, *Software agents: A review*. Technical report, TCS-CS-1997-06, Trinity College Dublin, Ireland (1997).

[9] D. Kotz and R.S. Gray, Mobile Agents and the Future of the Internet, Department of Computer Science, Thayer School of Engineering, Dartmouth College, Hanover, New Hampshire 03755, *ACM SIGOPS Operating Systems Review* **33(3)** (1999) 7–13.

[10] G. Bernard, Technologie du code mobile: état de l'art et perspectives, in *Proc. Of the Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'99)*, Nancy, France April 26–29 (1999).

[11] K. Moizumi, *Mobile Agent Planning Problems*. Ph.D. Thesis, Thayer School of Engineering, Dartmouth College, Hanover, NH 03755 USA (1998).
[12] W. Caripe, G. Cybenko, K. Moizumi and R. Gray, Dartmouth Coll., Hanover, NH; Network awareness and mobile agent systems. *IEEE Comm. Magazine*, **36-7** (1998) 44–49.
[13] K. Moizumi and G. Cybenko, The traveling agent problem. *Mathematics of Control, Signals and Systems* (1998).
[14] H. Zgaya, S.Hammadi and K. Ghédira, Workplan Mobile Agent for the Transport Network Application, *in Proc. of 17th IMACS World Congress Scientific Computation, Applied Mathematics and Simulation (IMACS'2005)*, Paris 11–15 July, (2005).
[15] E. Horowitz and S.Sahni, *Fundamentals of Computer Algorithms*. Computer Science Press (1989).
[16] J.W. Baek, J.H. Yeo, G.T. Kim and H.Y. Yeom, Cost effective mobile agent planning for distributed information retrieval, in *Proc. of 21*st *International Conference on Distributed Computing Systems (ICDCS'01)*, Phoenix, Arizona USA, 16–19 April (2001) 65–72.
[17] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the theory of NP-Completeness*, W. H. Freeman and Co., USA (1979).
[18] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution programs*. Springer Verlag (1992).
[19] H. Zgaya, S. Hammadi and K. Ghédira, Evolutionary method to optimize Workplan mobile agent for the transport network application, in *Proc. of International Conference on Systems, Man and Cybernetics*, (SMC'2005) 10–12 October Hawaii, USA, **2**(2005) 1174–1179.
[20] L. Davics, *Handbook of genetic algorithm*. New York: Van Nostrand Reinhold (1991).
[21] M. Purvis, S. Crenefield, R. Ward, M. Nowostawski, D. Carter and G. Bush, A multi-agent system for the integration of distributed environmental information, *Environmental Modelling & Software*,Information Science Department, University of Otago, Dunedin, New Zealand, **18(6)** (2003) 565–572.
[22] E. Zitzler and L. Thiele, Multiobjective Optimization Using Evolutionary Algorithms: A Comparative Case Study, in *Lect. Notes Comput. Sci.* **1498** UK (1998) 292–301.
[23] Java Agent DEvelopment framework. `http://jade.titlab.com/doc`
[24] `www.java.sun.com`
[25] `www.fipa.org`
[26] D. Greenwood, JADE Web Service Integration Gateway (WSIG), WHITESTEIN Technologies. Jade Tutorial, in *Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2005)*, Utrecht University, the Netherlands, 25–29 July (2005).
[27] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, **6(2)** (2002) 182–197.
[28] G.P. Picco and M. Baldi, Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications, in *Proc. of 20*th *IEEE International Conference on Software Engineering (ICSE'97)*, edited by R. Kemmerer and K. Futatsugi, Kyoto, Japan, April (1998) 146–155.
[29] M. Ketel, N.S. Dogan and A. Homaifar, Distributed Sensor Networks based on Mobile Agents Paradigms, Dept. of Computer Science, North Carolina A&T State University, NC 27411, in *Proc. of the Thirty-Seventh Southeastern Symposium on System Theory* (SSST'05), Greensboro, USA, 20–22 March (2005) 411–414.
[30] H. Zgaya and S. Hammadi, Assignment and Integration of Distributed Transport Services in *Agent-Based Architecture, IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006 Main Conference Proceedings) (IAT'06)*, Hong Kong, China, 18–22 December (2006) 96–102.