20

Distributed Optimisation using the Mobile Agent Paradigm through an Adaptable Ontology: Multi-operator Services Research and Composition

Hayfa Zgaya and Slim Hammadi
LAGIS UMR 8146 – Ecole Centrale De Lille
France

1. Introduction

Giving transport customers relevant, interactive and instantaneous information during their travels, represents a real challenge according to the exponential growth of services available on large distributed networks. Unfortunately, distributed applications through wide networks are not easy to realize because of the limited aspect of bandwidth that remains restricted and also because of a high incidence of network errors (bottleneck, failure, crash…). Our goal is to properly access and share distributed data located in an Extended Transport Multimodal Network (ETMN). In this context, mobile technology (Pharm & Karmouch, 1998; Theilmann & Rothermel, 1999) can complement artificial intelligence because it can reduce considerably network traffic (Carzaniga et al., 1997). Giving the mobility character to a software agent will allow him to migrate towards any node on the network that can receive mobile entities. Nodes to be visited by a Mobile Agent (MA) correspond to his route called Workplan. Many researchers have long discussed the benefits of the MA paradigm and conclude that it might be efficient in some cases (Picco & Baldi, 1997; Buse et al., 2003). In a recent work (Zgaya & Hammadi, 2006b), we demonstrated that using the MA paradigm in a Transport Multimodal Information System (TMIS) to collect needed data, is widely beneficial than using classical paradigms such as the Client Server (CS) one, if we use an optimization approach. The verification was successful thanks to a two-level optimization approach (Zgaya et al., 2005a, 2005b) that optimises, using metaheuristic, the total number of mobile entities and their different Workplans through the ETMN. However, some network errors (bottleneck, failure, crash…) can occur during the moving of MAs through the network nodes. In our work, we define a MA negotiation process in order to reassign non-attributed services, to available network nodes. Therefore we designed a flexible transport ontology that allows an easy handling of the terms and messages for negotiating. The remainder of this chapter is organized as follows: the problem complexity and the correspondent general formulation are presented in the next section. The global architecture of the Multi-Agent System (MAS) is proposed in section 3 and the optimisation approach in section 4. The proposed negotiation protocol is specified in section 5, followed by the used flexible transport ontology in section 6. Simulations are given in section 7 and finally the conclusion and prospects are addressed in last section.

Source: Multiagent Systems, Book edited by: Salman Ahmed and Mohd Noh Karsiti,
ISBN 978-3-902613-51-6, pp. 426, February 2009, I-Tech, Vienna, Austria

2. Problem formulation

The main concern of a TMIS is to satisfy users, respecting the delays of the responses (due dates) and minimizing their costs; this is a two-step optimization problem: firstly the assignment of an effective set of MAs to all existent network nodes. This assignment builds initial Workplans of the MAs in order to explore, in an optimal manner, the ETMN entirely. The second step corresponds to the best assignment of a sub-set of the ETMN nodes to identified tasks, deducing final Workplans. The selected sub-set of nodes corresponds to the possible providers to the identified tasks. A single identified task corresponds to an independent recognized sub-request which belongs to one or several requests formulated simultaneously by one or different customers through different devices (laptop, PDA…). More precisely, a single task can correspond to a transport service (sub-route, well-known geographical zone…) or to a related service (cultural event, weather forecast…). After the decomposition process, information providers (distant nodes), which propose services to the correspondent identified tasks, are recognized (fig. 1). Finally, nodes must be assigned to tasks in order to satisfy all connected users. A user is satisfied if his request was answered

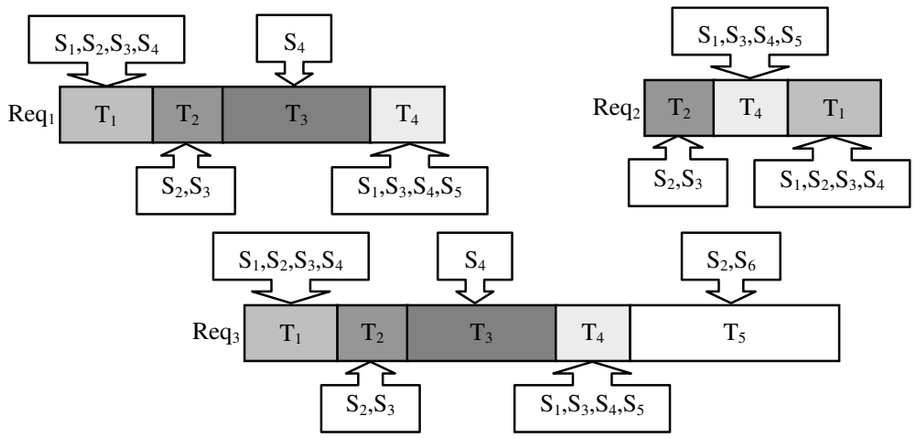

Fig. 1. Nodes identification

rapidly with a reasonable cost. This problem is called the Distributed Tasks Assignment Problem (DisTAP) and defined by:

- R requests, waiting for responses at the same instant *t*. The set of these requests is noted by R_t,
- The set of independent I tasks, representing all available services on the ETMN, is noted by $T=\{T_1,\ldots, T_I\}$,
- Each request $req_w \in R_t$ $(1 \le w \le R)$ is decomposed into a set of independent tasks, noted by $I_{t,w}=\{T_{r_i},...,T_{r_{nj}}\}$ $(1 \le n_j \le I$ and $I_{t,w} \subseteq T)$,
- The set of independent I′ tasks $(I' \le I)$, composing globally R_t, is noted by I'_t $(I'_t \subseteq T$ and $\bigcup_{w=1}^{R} I_{t,w} = I'_t)$,
- Each request req_w has a due date d_w initially known, an ending date D_w and a total cost C_w,
- The realization of each task $T_i \in T$ requires a resource, or node, selected from a set of J registered nodes in the ETMN, noted by $S=\{S_1,\ldots, S_J\}$,
- The set of J′ nodes $(J' \le J)$, selected from S to perform I'_t, is noted by 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 the resources of S_j, is defined and noted by $P_{i,j}$,

- 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 $Co_{i,j}$,
- The size of the collected data 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 $Q_{i,j}$,
- 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.

The three characteristics described above, namely ($P_{i,j}$;$Co_{i,j}$;$Q_{i,j}$), represent successively the first, second and last term of each element of what we call a service table (table 1).

	S_1	S_2	S_3	…	S_J
T_1	(0;0;0)	(0.2;5;3)	(0.4;3;3)		(0.2;5;3)
T_2	(0.2;4;5)	(0.1;5;2)	(0.4;5;1)		(0.3;8;3)
T_3	(0.1;0;3)	(0;0;0)	(0.2;0;3)		(0.4;2;2)
T_4	(0.3;2;1)	(0.3;1;1)	(0;0;0)		(0,0,0)
…					
T_I	(0.2;3;1)	(0.1;1;3)	(0.4;5;2)		(0.4;5;3)

Table 1. Example of a service table

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 $P_{i,j}\neq0$, $Co_{i,j}\neq0$ and $Q_{i,j}\neq0$. It is also possible to have $P_{i,j}\neq0$, $Q_{i,j}\neq0$ and $Co_{i,j}=0$ for a free information in the case of a promotional operation. In order to situate the complexity of our problem, an analogy was performed (table 2) between the problem described above (DisTAP) and the well-known Flexible Job Shop Problem (FJSP).

FJSP	DisTAP
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 $P_{i,k}$
At a given time, a machine can only execute one operation: it becomes available to other operations one the operation that 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 that is currently assigned to is completed (resource constraints).

Table 2. Analogy between FJSP and our problem

In DisTAP, we manage the similarities of requests in order to avoid the same data research. Besides, we have to assign the servers to tasks as well as to assign MAs to remote nodes (servers), taking into account the network state. Therefore our problem presents more difficulty than the FJSP which has been shown to be NP-hard. In addition, the distributed character of our system and the requirement to cooperate different autonomous static and mobile entities, confirm the choice of a multi-agent architecture for our system.

3. The multi-agent system

To resolve the problem described previously, we propose a system based on the coordination of five kinds of software agents (fig. 2):

- Interface Agent (IA): this agent interacts with the user of the system allowing him to choose an appropriate form of response to his demand, so this agent manages the request and then displays the correspondent result. Therefore, when a user accesses to the TMIS, an agent IA deals with the formulation of his request 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 he can receive several requests formulated at the same time,
- Identifier agent (IdA): this agent manages the decomposition of the requests that were formulated through a same short period of time ε* (ε-simultaneous requests). The decomposition process generates a set of 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 distributed nodes (information providers) through the ETMN. Initially, each node must login to the system registering all proposed services knowing that a service corresponds to the response to a defined task with fixed cost, processing time and data size. Therefore, an agent IdA decomposes the set of existing simultaneous requests into a set of independent tasks, recognizing possible similarities in order to avoid a redundant search. The decomposition process occurs during the identification of the information providers. Finally, the agent IdA transmits cyclically all generated data to available scheduler agents. These ones must optimize the selection of providers, taking into account some system constraints,
- Scheduler Agent (SA): several nodes may propose the same service with different cost, processing time and data size. The agent SA has to assign nodes to tasks, minimizing total cost and total processing time to respect due dates (data constraint). Selected set of nodes corresponds to the sequence of nodes which build the Workplans (routes) of the collector agents. An agent SA has firstly to optimize the number of collector agents before assigning nodes to tasks.
- Intelligent Collector agent (ICA): an agent ICA is a mobile software agent who can move intelligently from a node to another through a network in order to collect needed data and finally returns to his home node, noted by H. This special kind of agent is composed of data, code and a state and has an intelligent behaviour. Collected data should not exceed a capacity threshold in order to avoid the overloading, so the agent SA has to take into account this aspect when assigning nodes to tasks. When they come back to the system, the agents ICA must transmit collected data to the available fusion agents,
- Fusion Agent (FA): the agents FA have to fusion correctly collected data in order to compose responses to the simultaneous requests. The fusion procedure needs information on behalf of IdA and SA agents and progresses according to the collected data availability. Each new answer component must be complementary to the already merged ones.

* Fixed by the programmer

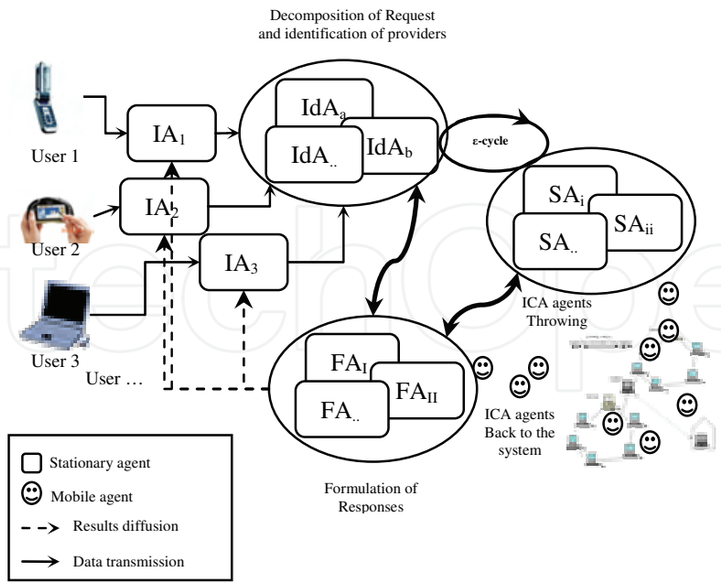

Fig. 2. System architecture

The needed data, required to satisfy the demands of the customers, are distributed through the ETMN and their collect corresponds to the jobs of ICA agents. Consequently, the SA agents have to optimize the assignments of the nodes to the identified tasks, minimizing total cost and response time. To this problem, we propose a two-level optimization solution (Zgaya et al., 2005a, 2005b) corresponding to the complex behaviour of the SA agents.

4. Scheduler Agent behaviour

The SA agent has two different basic behaviours: firstly the generation of an effective number of ICA agents in order to explore the ETMN entirely. This behaviour starts each time the network state varies considerably, in order to prepare the initial Workplans of ICA agents. Thus, we assume the existence of a network module that provides information to the system about the latest variations. The second behaviour is an ε-cyclic one, supervising the reception of the required services and their possible information providers, identified by the agents IdA. This second behaviour optimizes the assignments of nodes to the tasks in order to deduce the final Workplans of ICA agents from initial ones. The SA agents have to interact in order to share information and negotiate the different part of assignment for a global optimisation. For this problem, we just underline that we propose a solution using the formation of coalitions approach but this is not the topic of this chapter. For example, in the case of possible overlapping of the simultaneous requests, concerning SA agents have to gather, forming coalitions, in order to share the assignments about the different identified similarities. Hence, we focus here on the individual behaviour of a SA agent apart from his interaction with the other agents. We describe the two individual behaviours mentioned above in what follows.

4.1 Generation of the initial workplans

The Cost-Effective Mobile Agent Planning (CE-MAP) Algorithms, suggested by (Baek et al., 2001), are the most appropriate to our problematic. In fact, a proposed dynamic algorithm,

called BYKY2, optimizes the number of MAs minimizing the total execution time, taking into account the network state. In a previous work (Zgaya, 2005a), we adopted the same approach but we considered the transported data and then the state variation of mobile entities. The MA Workplan problem is described in what follows, assuming some hypothesis:

- Collecting data on a visited node requires a 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 an ICA 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 in the network is available tanks to an existent network monitoring module,
- Information can have a multimedia aspect, so we assume that the transmission of a quantity of data from a node to another depends on current latency.

4.1.1 Description

The MA Workplan problem can be described as follows: ICA agents are created and initially launched from an originally node (Home node). The other network nodes represent available information providers where an ICA agent can move to collect data corresponding to the 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 with different formats. We call a response time to a service on a node, the processing time of the correspondent task to this service on this node. So the response time on the Home node is null. Latencies are known and may affect navigation time of ICA agents. Our goal is to minimize the number of ICA agents and their navigation time in order to explore all the ETMN, taking into account network state. Initially, we introduce some definitions using variables described in table 3.

Variable	Description
m	Number of ICA agents
$ICA_1,\ldots,ICA_m$	Identifiers of ICA agents
H	Home node
$Wk_{i,p}$	Nodes sequence representing the Workplan of an ICA_i agent: ($S_{i_1}, ..., S_{i_p}$) with $1 \leq p \leq J$
$T(Wk_{i,p})$	Routing time for $Wk_{i,p}$
$Qte_{k,u}$	The size of the transported data by the agent ICA_k until the node S_u included
$Tr(Qte_{k,u},S_u,S_v)$	Transmission time for $Qte_{k,u}$ from node S_u to node S_v
CT_j	Processing time on node S_j for the data quantity Qt_j
Qt_j	Data quantity on node S_j
$d(S_i,S_j)$	Data transfer rate between nodes S_i and S_j

Table 3. Notations

Definition 1: CT_j (Processing time on node S_j) corresponds to needed computing time on the node S_j to extract the data quantity Qt_j.

Definition 2: Qt_j (Data quantity on node S_j) it is the average data size to extract from S_j.

T_i represents a task corresponding to a service proposed by S_j. Therefore:

$$Qt_j = \frac{\sum_{i=1}^{I} a_{i,j} Q_{i,j}}{\sum_{i=1}^{I} a_{i,j}} \quad (1)$$

$$CT_j = \frac{\sum_{i=1}^{I} a_{i,j} P_{i,j}}{\sum_{i=1}^{I} a_{i,j}} \quad (2)$$

Where $a_{i,j}$ is a Boolean value as follows: $a_{i,j}$=1 if the node S_j proposes a service for the task T_i and $a_{i,j}$=0 otherwise (according to the given service table). Moreover, we remind that $P_{i,j}$ corresponds to the processing time of a given task T_i on the node S_j (section 2).

Definition 3: $Qte_{k,u}$ (Data quantity transported until S_u by ICA_k) corresponds to the size of the collected data by the agent ICA_k during his route, until the node S_u included.

$Qte_{k,u}$ is calculated by:

$$Qte_{k,u} = Q_0 + \sum_{r=1}^{u} Qt_{k_r} \quad (3)$$

We remind that Q_0 corresponds o the initial quantity of data within an ICA agent (parag.4.1).

Definition 4: $Tr(Qte_{k,u},S_u,S_v)$ (Transmission time) needed time for the agent ICA_k to migrate from S_u to S_v transporting the data quantity $Qte_{k,u}$.

$Tr(Qte_{k,u},S_u,S_v)$ is computed like this:

$$Tr(Qte_{k,u}, S_u, S_v) = \frac{Qte_{k,u}}{d(S_u, S_v)} \quad (4)$$

Definition 5: $T(Wk_{k,p})$ (Routing time for $Wk_{k,p}$) needed time for the agent ICA_k to visit the sequence of network nodes $(S_{k_1},...,S_{k_p})$ with 1≤p≤J.

$T(Wk_{k,p})$ is computed like this:

$$T(Wk_{k,p}) = T_{go}(k, p=1) + T_{travel}(k, p) + T_{return}(k, p) \quad (5)$$

	$Wk_{k,p}$	T_{go}	T_{return}	T_{travel}
p=1	(S_{k_1})	$Tr(Q_0,H,S_{k_1})$	$Tr(Qte_{k,p},S_{k_p},H)$	CT_{k_1}
1<p≤J	$(S_{k_1},...,S_{k_p})$			$X_{k,p}$

Table 4. Routing Time

With:

$$X_{k,p} = \sum_{i=1}^{p} CT_{k_i} + \sum_{i=1}^{p-1} Tr(Qte_{k,i}, S_{k_i}, S_{k_{i+1}}) \quad (6)$$

4.1.2 Proposed workplan schemes

To propose a cost-effective Workplan MA scheme, we assume that a monitoring module exists in the system providing information about the network status (latency, bandwidth, traffic, bottleneck, failure…). Therefore, we can get data transfer rate values among all pairs of nodes through the ETMN. The goal is to find an effective set of ICA agents minimizing their navigation time, in order to explore all the ETMN nodes, taking into account network

state. It is clear that sending an ICA agent to each node gives us the best total computing time because, in this case, agents are launched simultaneously into each network node. Therefore, we keep this best total computation time to build nodes partitions, minimizing the number of ICA agents. Consequently, we just care about the data size and the processing time in the service table (section 2), ignoring the data cost. As described previously, $d(S_i,S_j)$ namely data transfer rate among two network nodes S_i and S_j, is available. We give here a brief description of the algorithm detailed in (Zgaya et al., 2005a):

The initial Workplan algorithm description
- ***Step 1***: Sort the nodes in decreasing order according to their correspondent routing time $T(Wk_i=S_i)$ $\forall 1\leq i\leq J$. Set the threshold δ which is the routing time of the first node in the sorted list: $$\delta = \max_{1\leq i\leq J}(T(Wk_i = S_i)) \quad (7)$$
- ***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.

This proposed dynamic algorithm tries to find the next node to visit from the current position where the agent resides. In other words, this algorithm looks for the next node for a part calculating, each time, the 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 an ICA agent and the algorithm ends if each available node belongs to a Workplan. The algorithm distributes all available nodes to a set of m ICA agents in order to explore the network entirely. Each built route corresponds to the initial Workplan of the correspondent ICA agent. Then, final Workplans will be deduced from initial ones thanks to our evolutionary approach described in next section. This will be done by selecting a subset S′ from S (the total number of available nodes in the ETMN) in order to optimise the management of the data flow through the network. Thus, some nodes will not be selected from S what can decrease the total number m of ICA agents. This will be happen when all the nodes composing the initial Workplan of an agent ICA are not selected. Let m' be the new number of ICA agents. We have also $J'=|S'|$ the new number of nodes so $m'\leq m$, $J'\leq J$ and $S'\subseteq S$. Thanks to the generated final Workplans, required data will be collected in an effective manner, in order to reach as soon as possible and with reasonable costs, the best schedule of the simultaneous requests.

4.2 Composition of services using an evolutionary approach

The 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 the survival principle of the fittest results, in order 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. 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, we use an evolutionary approach to resolve our assignment problem. Therefore, we use some aspects that must be clarified:

- A specific genetic representation (or encoding) appropriate to the problem, to determine feasible solutions of the scheduling optimization problem,
- Original genetic operators that alter the composition of children during the reproduction. As it was mentioned previously, a task (sub-request) must be managed by only one provider selected from the set of nodes that propose the correspondent service. 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/mutation with some probability p_c/p_m ($0 < p_c, p_m < 1$). We believe that this technique gives more chance to weak individuals to survey,
- A non-elitist replacement technique is adopted to generate the new population from the previous one,
- The evaluation functions estimate a possible solution according to two criteria: the cost and the delay.

4.2.1 The representation of an evolutionary solution

The research and the composition of distributed transport services are generated thanks to an evolutionary algorithm, managed by the active SA agents in the system. We notice that the selection of an appropriate representational scheme of a solution is fundamental to the success of EA applications. Therefore, in a previous work (Zgaya et al., 2005b), we designed an efficient coding (possible solution) for the chromosome respecting our problem constraints. Thus, 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 (the services), composing globally simultaneous requests and columns represent identified nodes (the providers). Each element of the matrix specifies the assignment of a node S_{c_j} (1≤j≤J′) to the task T_{c_i} (1≤i≤I′) as follows:

Value of CH[i,j]	Condition
1	S_{c_j} is assigned to T_{c_i}
*	S_{c_j} may be assigned to T_{c_i}
X	S_{c_j} cannot be assigned to T_{c_i}

We notice that each task must be performed by a single node, selected from the available set of nodes that propose the service corresponding to a response to the concerned task. Indeed, the assignment and the scheduling of all distributed nodes to the required services, represent the optimisation of the services composition that provide transport customers effective responses to their requests.

4.2.2 The genetic operators

(a) The crossover algorithm

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. The following algorithm specifies the crossover operator:

CrossFeTAR Algorithm
The creation of C_1 (resp. C_2) representing the child 1 (resp. child 2) is given by: - ***Step 1***: Choose randomly two parents and one node; suppose that P_1, P_2 and S_{c_j} ($1 \le j \le J'$) are randomly selected, - ***Step 2***: Tasks assignment of S_{c_j} in C_1 (resp. C_2) must correspond to the same assignment of S_{c_j} in P_1 (resp. P_2), - ***Step3***: k :=1 ; while ($k \le J'$) and ($k \ne j$) { Tasks assignment of S_{c_k} in C_1 (resp. C_2) corresponds to the same assignment of S_{c_k} in P_2 (resp. P_1); k := k + 1 ; } - ***Step 4***: if (C_1 (resp. C_2) is not a feasible solution) Correct randomly C_1 (resp. C_2);

We notice that sometimes, a generated solution resulting from a crossover process is not feasible. If it is the case, we propose a correction process that changes, in a random way, a non-feasible solution to a feasible one knowing that a feasible solution is a FeTAR instance that assigns each task composing it, only once. The algorithm ***CorrectFeTAR*** illustrates this correction process as follows:

CorrectFeTAR Algorithm
The correction of the FeTAR instance CH is given by: for (i:=1; i ≤I′; i:=i+1){ initialize to zero the vectors *IndexAssigned[]* and *IndexNotAssigned[]* of dimensions J′; k1:=0; k2:=0; for (j:=1; j ≤J′; j:=j+1){ if($CH[c_i, c_j]$=1) { k1:=k1+1; *IndexAssigned[k1]*:=j; }else if($CH[c_i, c_j]$=*) { k2:=k2+1; *IndexNotAssigned[k2]*:=j; } } if(k1=0) { Draw randomly an index p with 1≤p≤k2 ; s:=*IndexNotAssigned[p]*; $CH[c_i, c_s]$:=1; }else if(k1>1) { Draw randomly an index p with 1≤p≤k1 ; s:=*IndexAssigned[p]*; for (x:=1; x ≤J′; x:=x+1){ if($CH[c_i, c_x]$=1 et x≠s) $CH[c_i, c_x]$:=*; } }

For example, we suppose that a Crossover process generated, from two FeTAR instances parents P_1 and P_2, two new FeTAR instances childs C_1 and C_2 like so:

P_1	S_{12}	S_6	S_3	S_{24}
T_1	X	*	*	1
T_5	*	*	1	*
T_3	1	X	*	*
T_9	*	*	1	X
T_2	*	*	1	*

P_2	S_{12}	S_6	S_3	S_{24}
T_1	X	*	1	*
T_5	*	*	1	*
T_3	*	X	*	1
T_9	*	*	1	X
T_2	*	*	1	*

C_1	S_{12}	S_6	S_3	S_{24}
T_1	X	*	*	*
T_5	*	*	1	*
T_3	*	X	*	1
T_9	*	*	1	X
T_2	*	*	1	*

C_2	S_{12}	S_6	S_3	S_{24}
T_1	X	*	1	1
T_5	*	*	1	*
T_3	1	X	*	*
T_9	*	*	1	X
T_2	*	*	1	*

Both C_1 and C_2 are not feasible solutions because C_1 does not assign the task T_1 and C_2 assigns the task T_1 more than one time (T_1 is assigned twice by S_3 and S_{24}). After correction, C_1 will be randomly C_{1x}, C_{1y} or C_{1z} and C_2 will be randomly C_{2x} or C_{2y} like so:

C_{1x}	S_{12}	S_6	S_3	S_{24}
T_1	*X*	*1*	*	*
T_5	*	*	*1*	*
T_3	*	*X*	*	*1*
T_9	*	*	*1*	*X*
T_2	*	*	*1*	*

C_{1y}	S_{12}	S_6	S_3	S_{24}
T_1	*X*	*	*	*1*
T_5	*	*	*1*	*
T_3	*	*X*	*	*1*
T_9	*	*	*1*	*X*
T_2	*	*	*1*	*

C_{1z}	S_{12}	S_6	S_3	S_{24}
T_1	*X*	*	*1*	*
T_5	*	*	*1*	*
T_3	*	*X*	*	*1*
T_9	*	*	*1*	*X*
T_2	*	*	*1*	*

C_{2x}	S_{12}	S_6	S_3	S_{24}
T_1	*X*	*	*	*1*
T_5	*	*	*1*	*
T_3	*1*	*X*	*	*
T_9	*	*	*1*	*X*
T_2	*	*	*1*	*

C_{2y}	S_{12}	S_6	S_3	S_{24}
T_1	*X*	*	*1*	*
T_5	*	*	*1*	*
T_3	*1*	*X*	*	*
T_9	*	*	*1*	*X*
T_2	*	*	*1*	*

(b) **The mutation algorithm**

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. The mutation algorithm is presented as follows:

MuteFeTAR Algorithm

- ***Step 1***: Choose randomly one chromosome CH, one task T_{c_i} ($1 \le i \le I'$) and one node S_{c_j} ($1 \le j \le J'$);,
- ***Step 2***:

```
if(CH[i,j]= *){
        Find j1 with 1≤j1≤J' and CH[i,j1]=1 ;
        CH[i,j1] := * ;
        CH[i,j] := 1 ;
} else
    if(CH[i,j] = 1 and ∃ j1 / 1 ≤ j1 ≤ J' and CH[i,j1]=*){
                                        CH[i,j1] := 1 ;
                                        CH[i,j] := * ;
}
```

For example, if the chromosome C_{1x} undergoes a mutation process, muted C_{1x} may be C'_{1x} like so:

C'_{1x}	S_{12}	S_6	S_3	S_{24}
T_1	*X*	*1*	*	*
T_5	*	*	*1*	*
T_3	*	*X*	*	*1*
T_9	*	*	*1*	*X*
T_2	*1*	*	*	*

With the mutation point (T_2,S_3).

4.2.3 Evaluation functions

At each iteration, individuals (chromosomes) in the current population are evaluated according to the same measure of fitness. There are a number of characteristics of the evaluation function that enhance or hinder the evaluation of a program performance. 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 respecting due dates, then according to the average of total costs. Thus, a chromosome has to express ending responses date and the information cost (Zgaya et al., 2005b). The first evaluation function, called Fitness_1, computes the ending dates of all the requests according to the generated FeTAR solution, in order to deduce the number of satisfied users in term of response time. Then the second evaluation function, called Fitness_2, computes the total cost of each request. As we previously mentioned, a request req_w ($1 \le w \le R$) is decomposed into $I_{t,w}$ tasks and the algorithm Fitness_1 computes the total processing time D_w for each req_w. This time does not include only the effective processing time on the nodes because we have to take into account the routing time of ICA agents. For that, we assume that, the ending date D_w (EndReq[w]) corresponding to the total execution time of a request req_w, includes also some value noted by γ which is the average navigation time of ICA agents (Zgaya et al.,2008). Besides, the total cost C_w (EndReq[w]) is computed for each request req_w by the algorithm Fitness_2.

Fitness_1 Algorithm

- ***Step 1***: Initialisation (1≤k≤m)
  - Initialize to Ø each set of tasks U_k which should be performed by each ICA_k
  - Initialize to γ the total time EndU[k] to perform each set of tasks U_k.
- ***Step 2***: Compute the set of tasks U_k performed by each ICA_k and the total time to perform them as follows:

```
for (i:=1; i ≤ I'; i:=i+1) {
  Find k and j while ICA_k performs T_ci on S_cj ;
  U_k = U_k ∪ {T_ci} ;
  EndU[k] := EndU[k] + P_ci,cj ;
}
```

- ***Step 3***: Compute ending time for each request i: EndReq[w] with 1≤w≤R, by looking for each task composing this request. An ending time of a request is the maximum necessary time for all the agents ICA responsible for all the tasks composing this request, in order to carry out their Workplan. Ending time for each request is computed as follows:

```
for (w:=1; w≤R; w:=w+1){
 Initialize to false TreatedICA[k] for each ICA_k (1≤k≤m);
      EndReq[w] = 0;
      for (j :=1; j ≤I'; j:=j+1){
        if (T_cj ∈ req_w) {
          k:=1;
          while ((k ≤ m) and (T_cj ∉ U_k)) k:=k+1;
                  if (not TreatedICA[k]) {
                    EndReq[w]:=max(EndReq[w], EndU[k]);
                    TreatedICA[k] = true;
          }
        }
      }
 }
```

Fitness_2 Algorithm

The total cost for a request is the total cost of all independent tasks composing this request. It is calculated as follows:

Step 1: Initialisation (1≤w≤R)

Initialize to 0 CostRe[w] for each req_w

Step 2:

```
for (w:=1; w≤R; w:=w+1){
for each T_ci ∈ I_t,w {
Find j/ S_cj assigns T_ci in the FeTAR instance CH;
 CostRe[w]= CostRe[w] + Co_ci,cj ;
}
 }
```

The evaluation of a chromosome is illustrated by a vector, which express, for each request w (req_w), its required total time for the execution (D_w) and also its total cost (C_w). From the generated vector, we deduce the average total cost C_{av} and the maximum ending date D_{max} of all the requests managed by the chromosome.

4.2.4 Best solution

To determinate the best solutions, we adopted an elitist approach (Zgaya & Hammadi, 2008) using an external storage to memorise the most adapted individuals during the search. The evolutionary adopted approach is shown in fig. 3. During the evaluation process to crossover and mutation, best solutions are saved in external archives. Knowing that $d_{\max} = \max(d_w)_{1 \le w \le R}$, 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 ∀w (1≤w≤R), $D_w \le d_{max}$. This archive set is decomposed into two sub-archives :
  - dominant solutions archive (d)
  - ε-dominant solutions archive (ε-d)

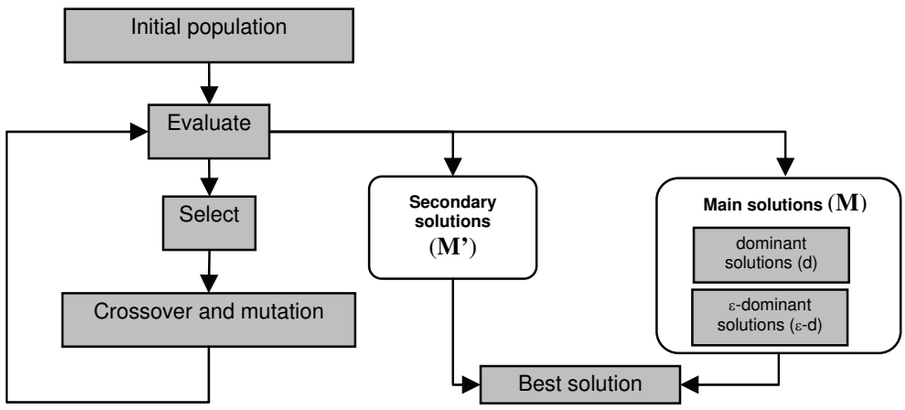

Fig. 3. Evolutionary approach

Secondary solutions archive (M′) representing best solutions which exceed at least one due date. In other words, a chromosome CH∈M′ if and only if ∃w (1≤w≤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. Considering two different solutions CH and CH′, if CH∈M and CH′∈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 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 populations. M and M′ archive sets represent generated solutions having minimum f1 and f2 values, so if a chromosome CH of the offspring dominates any archive member CH′, the archive member

is deleted and the offspring is accepted. Fig. 4 represents the Pareto-optimal fronts with $\varepsilon_1=\varepsilon_2=\varepsilon=0.75$ (Zitzler & Thiele, 1998). We use a population size of N=100 with a crossover probability $p_c=0.8$ and mutation probability $p_m=0.2$.

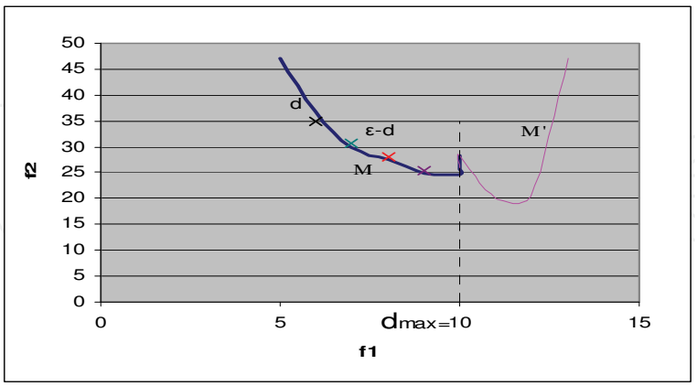

Fig. 4. Optimal fronts

4.2.5 Final workplans generation

According to the generated FeTAR instance CH, the equations (1) and (2) become:

$$Qt_{c_j} = \sum_{i=1}^{I'} (a_{c_i,c_j} \times Q_{c_i,c_j}) \quad (1')$$

$$CT_{c_j} = \sum_{i=1}^{I'} (a_{c_i,c_j} \times P_{c_i,c_j}) \quad (2')$$

With a_{c_i,c_j} a Boolean value as follows: if CH[c_i,c_j]=1 ($1 \le i \le I'$ and $1 \le j \le J'$) then a_{c_i,c_j}=1 else a_{c_i,c_j}=0. Routing time of the final Workplans are deduced using equations (1'), (2'), (3), (4), (5) and (6) according to table 4 (parag. 4.1.1).

Until this stage, the MA paradigm integrates a perfect structure without any incidence. But this is an improbable scenario because, in reality, errors may often occur (crash, bottleneck, failure…), especially in huge systems distributed through large networks. If it is the case, ICA agents have to interact with SA agents in order to negotiate the reassignments of potential cancelled services, keeping the whole robustness of the system. Consequently, we developed a negotiation protocol, described in next section.

5. The negotiation process

Some perturbations can occur through the network when ICA agents are following their correspondent final Workplans, according to the generated FeTAR instance. In this case, the ICA agents have to avoid unavailable nodes in their remained final Workplans. In addition, they have to change their itineraries in order to take into account the cancelled tasks that still need assignments because of the perturbation. Therefore, a new assignment process has to occur to find suitable new available providers. To do this, we have to benefit of active ICA agents who are still travelling through the network and to launch new ones otherwise. So ICA agents have to interact with SA agents in order to find suitable solution to the current

situation. Thus, we propose a negotiation process inspired from the well-known contract net protocol (Smith, 1980) between ICA agents representing the participants of the negotiation and SA agents who are the initiators. In our proposed solution, we allow a partial agreement of the proposed contract (a FeTAR instance) from each ICA agent, to be confirmed partially or totally by the initiator of the negotiation (SA agent). A renegotiation process is necessary while there are still tasks that need reassignment. The purpose of this solution is to allow the ICA agents to cooperate and coordinate their actions in order to find globally near-optimal robust schedules according to their priorities, preferences and constraints, which depend on their current positions in their correspondent Workplans. Through the negotiation process tours, SA agents must assure reasonable total cost and time. In what follow, we describe in detail the proposed protocol. Firstly, we present a brief description of the initiators and participants of the negociation process.

5.1 Initiators and participants

An initiator of a negotiation is a SA agent who never knows the exact position of each travelling ICA agent. However, he knows all initial Workplans schemes and the final assignments of the servers (final effective Workplans). SA agent does not need to wait for all answers to take a decision, since he can accept a subset of responses to take pressing sub-decisions; urgent actions must be taken according to the current positions of ICA agents. Consequently, SA agent can take decisions every short period of time. In that case, he must update the set of services that need to be reassigned by providers through the confirmation step. After that, SA agent has to propose a new contract according to the updated services set and to the different capabilities of the participants of the negotiation. We suppose that errors on the network are identified before that an ICA agent leaves one functioning node toward a crashed one. A participant of a negotiation is an autonomous ICA agent who never knows anything about the other participants of the same negotiation process. Obviously, he knows his own initial Workplan scheme and his final assignments of servers (final effective Workplan). In addition, each ICA agent has his own priorities, preferences and constraints that are dynamic, depending on the network state and on his current position in the already defined final Workplan. Constraints of an ICA agent express tasks that he can't perform or servers he can't visit because they might cause problems (overloading, time consuming, high latency…). Priorities express servers where the ICA agent prefers visit because they are already programmed in his remained final Workplan. Finally, preferences express servers that are already programmed in the remained initial Workplan and not in the final one. Each time an ICA agent receives a new contract, he analyzes it to make a decision (refusal or total/partial acceptance).

5.2 The protocol

A protocol defines the language used by agents to exchange information. The proposed negotiation protocol (fig. 5) is characterized by successive messages exchanges between initiators corresponding to the agents who initiate a negotiation (SA agents) and participants of the negotiation (ICA agents). We designed our protocol so that a negotiation process can occur between several initiators and participants; it can be, for example, the case of simultaneous requests overlapping. Presently, we describe a negotiation protocol between a unique initiator and several participants. Negotiation always begins with the creation of a contract by the initiator agent, proposing it to active participants. The first contract

corresponds to final Workplans that were already optimized thanks to our two-level optimization approach (Zgaya et al. 2005a, 2005b). A renegotiation means a round of modification request for a contract that "a part" has not been accepted the round before. In what follows, we show the adopted form for a communication before detailing the different exchanged messages between initiators and participants.

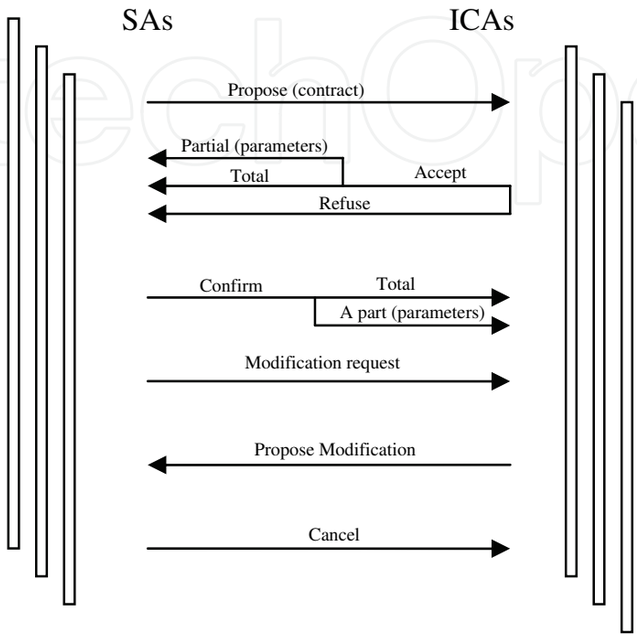

Fig. 5. The protocol

5.2.1 The agent message

We adopt the following structure for an agent message exchange:

<sender, receivers, service, perform, content, content-lang, ontology, f>

With:

- sender: the sender of the message,
- receiver: the list of receivers, they represent the recipients of the message,
- service: the "yellow-pages" service proposed by the receiver of the message,
- perform: the performative, which expresses the communicative intention,
- content: the information included in the message,
- content-lang: the content language, which represents the used syntax to express the content,
- the ontology: the vocabulary of the symbols used in the content and their meaning, used ontologies will be expressed in next section,
- f = <f1, f2, f3, f4, f5> represents some fields used to control several concurrent conversations and also to specify timeouts for receiving a reply. For the present, we don't assign this field but we just explain it for a best comprehension of message exchanges:
  - f1: reply-to *A*: the recipient of the message reply is the agent A,

- f2: conversation-id *ide*: a conversation identifier which may be fixed by the sender of the message in order to identify the ongoing sequence of communicative acts, that together form a conversation,
- f3: reply-with *exp*: identifies the reply to the current message with the expression exp,
- f4: in-reply-to *exp*: to denote that this message is a reply to an earlier action of which the reply was denoted by exp (f3) ,
- f5: reply-by *d*: time and/or date expression which indicates the latest time by which the sending agent would like to receive a reply.

5.2.2 Proposition of the contract

The contract message is a proposition of a new organization (the first contract) or reorganization of final Workplans to achieve tasks. If the execution of some services was cancelled because of some network perturbations, it is indeed the case of reorganization. This will be done by reassigning, ones more, servers to these tasks tht represent the set of the Dynamic Reassigned Tasks (DRT). The initiator sends an individual contract to each active ICA_k agent who proposes the contract-reception service. The correspondent message is:

<SA_i, ICA_k, contract-reception, propose, ∂, fipa-sl, MASOntology, f>

With:

- ∂ =∂1 if it acts of the first contract and ∂ = ∂2 otherwise,
- ∂1 ≡ Workplan (Owner : ICA_k ; Initial : $i_1,...,i_{k_i}$ Final : $f_1,...,f_{k_f}$),
- ∂2 ≡ FinalWk (Owner : ICA_k ; Final : $f_1,...,f_{k_f}$),
- $i_1,...,i_{k_i}$ represent references of nodes which belong to the initial Workplan of ICA_k,
- $f_1,...,f_{k_f}$ represent references of nodes which belong to the final Workplan of ICA_k,
- $k_i \leq k_f$.

In what follow the third field in an agent message (parag.5.2.1), corresponding to the service, will be null because the conversation will be identified thanks to the last field f to shape a conversation.

5.2.3 Response to the contract

When a participant receives the proposed contract, he studies it and answers by:

- A total acceptance if he agrees to coordinate all tasks chosen by the initiator, included in his remaining trip (remained final Workplan) and according to his current position. The correspondent message is:

<ICA_k, SA_i, Ø, accept-proposal, Ø, fipa-sl, ICANegotiationOntology, f>

- A partial acceptance if he agrees to coordinate a sub-set of the tasks selected by the initiator, included in his remaining trip (remained final Workplan) and according to his current position, the partial-accept-proposal message content expresses the references of cancelled tasks and those of non available servers (the reason of the non total-acceptance). The correspondent message is:

<ICA_k, SA_i, Ø, partial-accept-proposal, ∂, fipa-sl, ICANegotiationOntology, f>

With ∂ ≡ (tasks: $c_1,...,c_n$; nodes : $r_1,...,r_m$)

- A refusal if he does not agree with any task in the proposed contract, the refusal message content expresses the references of non available servers (the reason of the refusal). The correspondent message is:

<ICA_k, SA_i, Ø , refuse, ∂, fipa-sl, ICANegotiationOntology, f> with $\partial \equiv (r_1,...,r_m)$

The initiator does not wait for all answers because he must act rapidly, so he just waits for some answers for a very short period of time to make a decision; this feature is expressed in the last field f of an agent message, through the reply-by facet (5.2.1).

5.2.4 Confirmation

An initiator has to confirm independently the agreed part of each contract proposed to an agent ICA_k who represents an autonomous participant of the negotiation, the confirmation can be:

- Total if the initiator agrees with the total response to the previous proposed contract:

< SA_i, ICA_k, Ø, confirm, Ø, fipa-sl, Ø ICANegotiationOntology, f>

- Partial if the initiator agrees with a partial response to the previous proposed contract, the partial-confirm-proposal message content expresses the references of agreed tasks:

< SA_i, ICA_k, Ø, partial-confirm-proposal, ∂, fipa-sl, ICANegotiationOntology, f> with $\partial \equiv (g_1,...,g_p)$.

We notice here that through a confirmation, the set of tasks to reassign (the DRT table) is updated.

5.2.5 Modification request

If the DRT table is not yet empty, the initiator asks participants to propose a new distribution of the assignments of the services which are been cancelled, the request-modification message content expresses the DRT table:

<SA_i, ICA_k, Ø, request-modification, ∂, fipa-sl, ICANegotiationOntology, f>

With $\partial \equiv$ (DRT).

5.2.6 Proposition of a modification

In a previous work (Zgaya & Hammadi, 2006a), we designed a reassignment procedure strategy of servers to tasks, taking into account not only the dynamic positions of ICAs in their Workplans, but also their constraints, priorities and preferences, according to their respective current positions. Constraints of an ICA agent express tasks that he cannot perform or servers he cannot visit because they might cause problems (overloading, time consuming, high latency…). Priorities express servers where the ICA agent prefers visit because they are already programmed in his final Workplan. Finally, preferences express servers that are already programmed in the initial Workplan and not in the final one. The proposition message content expresses for each participant the new proposition of his remained Workplan according to his current state:

< ICA_k, SA_i, Ø, propose-modif, ∂, fipa-sl, ICANegotiationOntology, f>

With:

- $\partial \equiv$ FinalWk (Owner : ICA_k ; Final : $f_1,...,f_{k_f}$)
- $f_1,...,f_{k_f}$ represent references of nodes which belong to the final Workplan of the agent ICA_k.

5.2.7 Cancel

To avoid indefinite modifications tours (lack of resources, no available providers…), the initiator agent must cancel the negotiation process following a fixed period of time, illustrated by the last field of an agent message (parag. 5.2.1). Therefore he cancels the current contract creating, if it is yet possible, new ICA agents to execute the convention:

< SA_i, ICA_k, Ø, cancel, Ø, fipa-sl, ICANegotiationOntology, f>

In this section, we used MASOntology and ICANegotiationOntology which express a special vocabulary and semantic modules related to the MAS and the ICA negotiation process respectively. We present in next section the proposed ontology packages corresponding to a flexible Transport Ontology matched with the combinatorial aspect of the negotiation search space of our problem.

6. The proposition of a transport flexible ontology

We aim to define a proper vocabulary to the whole proposed multi-agent system (section 3) in order to automate the different kind of exchanges between agents. Therefore, we propose extensible ontologies packages (Fig. 6) that can adapt to all possible kind of interactions. In this chapter, we derive our different edges of ontologies from a basic one (level 0) that already defines fundamental features. Thus, in order to keep a flexible ontology aspect, we start our derivations with a Generic Ontology (level 1). This one defines the concept Element representing each constituent in any target logistic field: transport, hospital In order to perform a proper semantic checks on a given agent expression, it is necessary to classify all possible elements in the domain of discourse. Thus, we have to distinguish between predicates and terms. This classification is derived from the Agent Communication Language (ACL) defined in FIPA that requires the content of each ACLMessage to have a proper semantics according to the performative of the ACLMessage (Caire & Cabanillas, 2004). Thus, in our system, each element is identified with a distinctive reference that represents it in the global ETMN and an order number for the management. In our work, we focus on the transport field (level 2) represented by the TransportOntology where we define the Task, Server, Request, Service and ServiceTable concepts and also the "Provides" and the "Available provider" predicates. Besides, according to our system architecture, we adopted the multi-agent system methodology (level 3) so we designed the MASOntology where we define the Workplan concept, the "Performs" agent action and the "available agent", the "IsInitialOf", the "IsFinalOf" predicates. Through our proposed multi-agent approach we used a negotiation strategy (level 4) so we designed the ICANegotiationOntology that defines a special vocabulary to the negotiation of agents ICA with agents SA. This ontology is flexible for possible expansions. Initially, it contains "PartialConfirm" and "PartialAccept" agent actions that express respectively a partial confirmation or acceptation of an agent. The ICANegotiationOntology includes also "IsPriorityOf", "IsPreferenceOf" and "IsConstraintOf" predicates which express respectively the membership of a node to the priorities, preferences

or constraints of an ICA agent. Predicates and terms mentioned above are represented more in details in what follow.

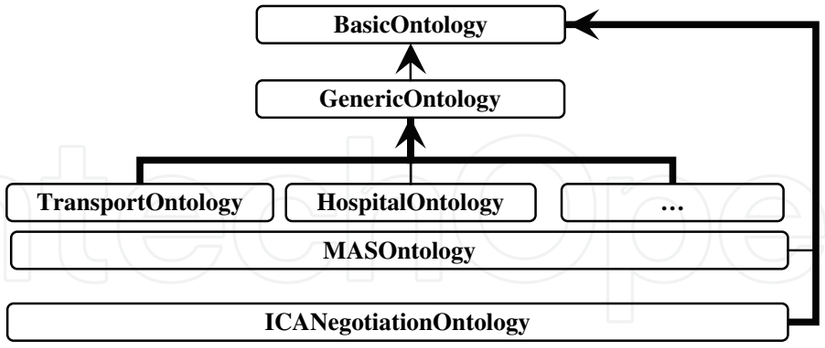

Fig. 6. Ontology Packages

6.1 Predicates

Predicates are expressions that say something about the status of the world and can be true or false. For the negotiation process, we define some useful predicates in the different proposed levels of ontologies (Fig. 7).

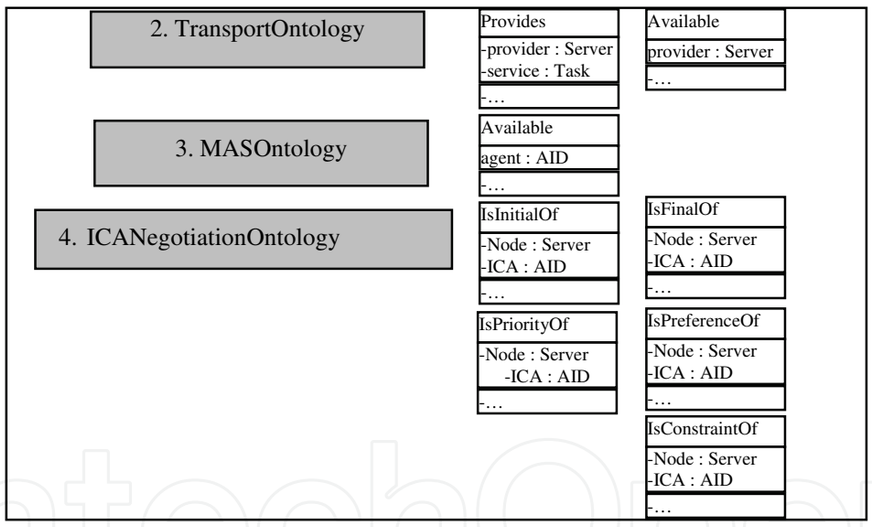

Fig. 7. Predicates

6.2 Terms

Terms are expressions identifying entities (abstract or concrete) that "exist" in the world and that agents talk and reason about. We distinguish in our design: Concepts and Agent actions (Fig. 8):

- Concepts: expressions that indicate entities with a complex structure that can be defined in terms of slots. As we previously mentioned, each element in our system is identified by a distinctive reference which represents it in the global ETMN and an order number for the management, examples:
  - (Element: ref 14 : order 2),
  - (Task: ref 2 : order 15 : providers 2 12 15 : nbProviders: 3),

- AgentActions: special concepts that indicate actions that can be performed by some agents, examples:
  - (Performs (node: ref 2 : order 5: services 2 8 6 : nbServices 3) (Task: ref 5 : order 5 : providers 2 6 : nbProviders 2)),

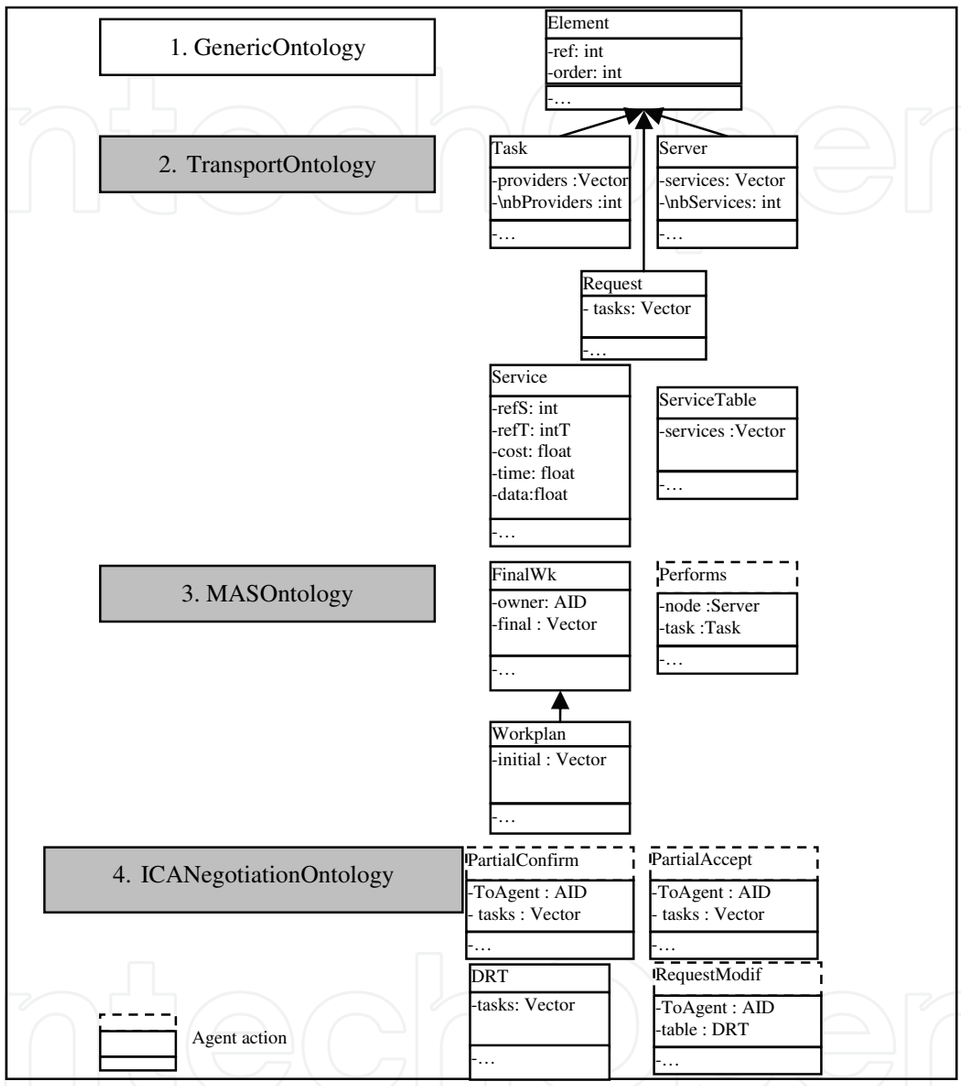

Fig. 8. Terms

7. Simulations

7.1 Global system and communication

For the implementation of our whole system (agent behaviours, communication, interactions…), we use Java Agent DEvelopment framework (JADE). It is a middleware which allows a flexible implementation of multi-agents systems and offers an efficient transport of ACL messages for agent communications complying with FIPA specifications. Jade offers the "yellow pages" service which allows agents to publish one or more services they provide so that other agents dynamically find them and successfully exploit the proposed

"yellow pages" services at a given point in time. Besides, this middleware includes a proficient support for content languages and ontologies, that's why we are implementing our semantic hierarchy of ontologies with JADE framework. Also, JADE offers a graphical tool to debug sniffs message exchange between agents. This tool is useful to debug a conversation between agents. On the left side window of the Sniffer graphic tool (fig. 9), we can see available servers containers on the network, where ICA agents can move in order to collect data according to the adopted contract model.

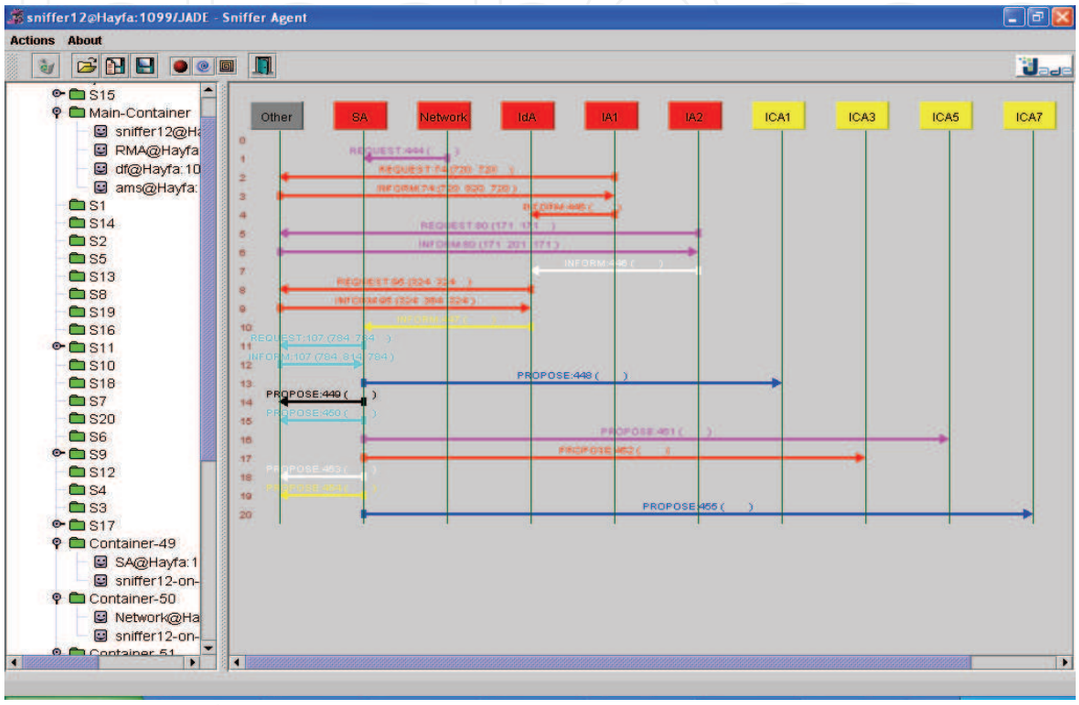

Fig. 9. Messages exchanges

7.2 Comparison of the Mobile Agent paradigm with the classical CS

Many researchers have long discussed the benefits of the MA paradigm and conclude that it might be efficient in some cases (Picco & Baldi, 1997; Buse et al., 2003). Indeed, the MA paradigm illustrates some efficient utility in several system architectures (Lu & Mori 2003; Buse et al., 2003). In a recent work (Zgaya & Hammadi, 2006b), we justified the usage of MA paradigm in our system, proposing an efficient procedure through a multi-agent transport system to optimize the management of services in the transport business domain. In order to evaluate the efficiency of our optimization approach, we propose to compare it to the classical CS paradigm (Picco & Baldi, 1997; Ketel et al., 2005); η_{CS} represents the overhead function used to send message request and $\tilde{\eta}_{CS}$ the one used for replies. Maximum response time in the CS case is:

$$T_{CS} = \sum_{j=1}^{J'} \left(\frac{\sum_{i=1}^{I'} a_{c_i c_j} (\eta_{CS} q_{c_i c_j} + \tilde{\eta}_{CS} Q_{c_i c_j})}{d(H, S_{c_j})} \right)$$

With $q_{c_i c_j}$ and $Q_{c_i c_j}$ correspond respectively to the size of the request message and the response message for the tqask T_{ci} on the server S_{cj}. The data transfer rate between nodes S_i and S_j is denoted by $d(S_i,S_j)$ and H symbolises the common home node. For a generated FeTAR instance solution CH, provided by SA agent, $a_{c_i c_j}$ is a Boolean value as follows: if $CH[c_i,c_j]=1$ ($1 \le i \le I'$ and $1 \le j \le J'$) then $a_{c_i c_j}=1$ else $a_{c_i c_j}=0$. Therefore, the overall transmission overhead for the CS case is:

$$\phi_{CS} = \sum_{j=1}^{J'} \left(\sum_{i=1}^{I'} a_{c_i c_j} (\eta_{CS} + \tilde{\eta}_{CS}) \right)$$

In the MA case, when an ICA agent moves to the node S_{cj}, he carries all the replies collected on the previous j-1 nodes. When the information on the last node collected, the ICA agent sends back to the home node all collected results. Therefore, the overall transmission overhead for an agent ICA_k is:

$$\phi_{MA_k} = \sum_{j=1}^{J'} b_{c_j c_k} \eta_{MA} + \tilde{\eta}_{MA}$$

Where η_{MA} ($\tilde{\eta}_{MA}$) represents the protocol overhead function used to send message requests (replies) in the MA paradigm and $b_{c_j c_k}$ is a Boolean value as follows: if S_{cj} belongs to the final Workplan of the agent ICA_{ck} denoted by Wk_{ck} ($1 \le j \le J'$ and $1 \le k \le m'$) then $b_{c_j c_k}=1$ else $b_{c_j c_k}=0$. The maximum response time corresponds to the maximum total travelling time of all active ICA agents: $T_{MA} = \max_{1 \le k \le m'} (T(Wk_{c_k}) + \phi_{MA c_k})$

Besides, we are interested into huge systems with important number of nodes and important request flow so ϕ_{CS} is likely to be greater than ϕ_{MA}. Therefore we do not take into account the transmission overhead in the experimental results. For example, we generated a FeTAR instance to response to 2s-simultaneous requests, decomposed globally into 8 tasks and requiring data from 19 providers. This solution required 1,4s in the MA paradigm and 14,71s in the CS one (fig. 10). Besides, when we randomly generate several FeTAR instances for the same example, we observe the benefit of the usage of our optimization approach, using MA paradigm instead of the CS one through our system (fig. 11).

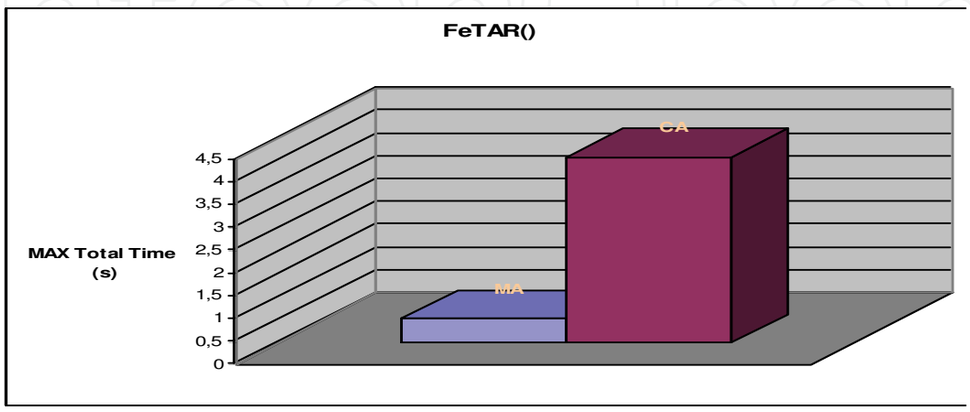

Fig. 10. The result of a FeTAR instance

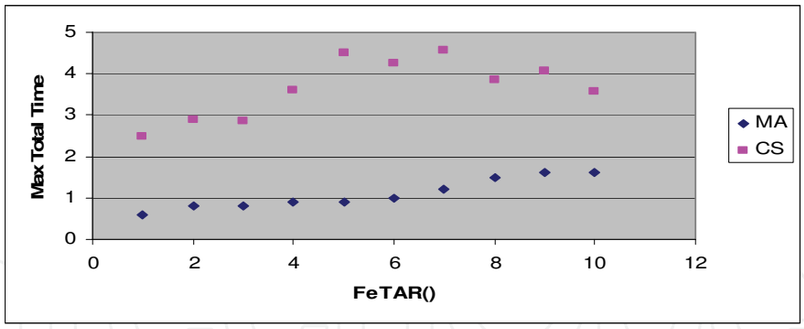

Fig. 11. Variation of FeTAR instances

7.3 Full transport application example

Through 2 seconds (ε=2s), we suppose the existence of a number of users who were connected to our system at t=11 o'clock today using different devices and then formulated a number of requests as follows; req_1: to travel at t o'clock from B to C, req_2: to travel next weekend from A to B with the minimum cost. Ask for the weather forecasting and cultural events for next weekend in B, req_3: to travel at t o'clock from A to C, req_4: to ask for a current transport perturbations from B to C, req_5: look for the best service connecting with the train X, predicted in A at 12:00 today to go to C, req_6: to travel at t o'clock from A to B, req_7: to look for a good price/quality hotel in D during next weekend and to make reservation, looking for the best route and departure time to go from B to D with car taking into account the tailback forecasting, etc. we situate our example in a ETMN composed of I=100 different services, proposed by J=20 different providers. To simplify this example, we suppose that IA agents send the ε-simultaneous requests to a single available IdA agent. This one decomposes the requests into a set of I'=64 independent tasks: $I'_t=\{T_1,T_2,T_3,T_6,T_9,T_{13},T_{16}\ldots\}$. We notice here that, we do not focus on the decomposition process, but we suppose that IdA agent decomposes R_t into independent tasks as follows: T_1= "Transport perturbations from B to C (at t o'clock)", T_2= "Weather forecasting in B (next week-end)", T_3= "To look for a good price/quality hotel in D during next weekend and to make reservation", T_6= "To look for the shortest route to go with car from B to D", T_9= "To look for the best departure time to go from B to D with car taking into account the tailback forecasting for next weekend", T_{13}= "Cultural events in B (next week-end)", T_{16}= "To travel from B to C (today, at t o'clock /today, from 12:00)", T_{19}= "To travel from A to B (today, at t o'clock / next week-end with the minimum cost/ the best service connecting with the train X at 12:00 today)", etc. We just underline the fact that there is not a direct service connecting from A to C. So the decomposition takes into account this aspect. In this paper, we do not detail the decomposition process but we point out that a task can represent several services with different constraints. For example, T_{19} represents 3 services corresponding to the same task "To travel from A to B" with different constraints: "now", "this week-end, with the minimum cost" and "The best service connecting with the train X at 12:00 today". Besides, a service is identified by a key work corresponding, for example, to an "action" and specified according to constraints which are mentioned in brackets. The full response will be composed thanks to FA agent who has to fusion services according to the user constraints, taking into account the pertinence of the information. The generated solution at t+ε is as follows:

- The generated FeTAR instance evaluated by fitness_1 and fitness_2 as follows: C_{av}=3.51 and D_{max} with d_k=6, 7 ∀k:

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}
T_1	0	×	0	0	0	0	1	0	0	0	0	0	×	0	0	×	0	0	×	×
T_2	0	×	0	0	×	×	0	0	0	1	×	0	0	0	0	×	0	0	0	0
T_3	0	×	0	0	0	×	0	0	0	0	0	0	×	0	0	0	×	0	1	0
T_6	0	×	0	1	0	×	0	0	0	0	0	0	×	×	0	0	0	0	0	0
T_9	×	×	×	0	×	×	×	0	1	0	0	0	0	×	0	×	0	×	×	0
T_{13}	0	×	0	0	1	×	×	0	0	0	0	0	×	×	0	0	×	0	0	×
T_{16}	×	×	×	×	1	×	0	×	×	×	×	×	×	×	×	×	×	×	0	×
T_{19}	1	×	×	×	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
T_{20}	×	×	0	×	0	×	×	×	×	1	×	×	×	×	0	×	×	×	×	0
T_{21}	×	×	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×	1	×	×
T_{22}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0	×	1	0
T_{25}	×	×	×	0	×	×	×	×	×	×	×	×	×	×	×	0	1	×	×	0
T_{26}	×	0	0	×	×	×	×	×	×	×	0	1	×	×	×	×	×	0	×	×
T_{28}	×	1	0	×	0	×	×	×	×	0	×	0	×	×	0	×	×	×	0	0
T_{29}	0	0	×	×	×	×	×	×	×	×	0	×	1	×	×	×	×	×	×	×
T_{30}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0	0	0	1
T_{31}	1	×	0	0	×	0	×	0	×	×	×	0	×	0	×	×	×	0	×	×
T_{32}	0	×	0	0	×	×	0	1	0	×	×	0	×	0	×	×	×	0	×	×
T_{33}	0	×	0	0	×	×	0	×	×	×	×	1	×	×	×	0	×	×	×	×
T_{34}	0	0	0	×	×	×	1	×	×	×	×	×	×	×	×	×	×	×	×	×
T_{35}	0	×	0	×	×	×	×	0	×	×	×	×	×	×	×	×	×	1	×	×
T_{36}	0	×	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0	×	×
T_{37}	×	×	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0	×	1
T_{38}	0	×	0	×	×	×	×	×	×	0	×	×	×	0	×	×	1	×	×	×
T_{39}	×	×	×	×	×	×	×	×	×	1	×	×	×	×	×	×	0	×	×	0
T_{40}	×	×	×	×	×	×	×	×	×	×	0	×	×	×	×	0	1	×	×	×
T_{41}	×	×	1	×	0	×	×	×	×	×	×	×	×	×	×	×	×	0	×	×
T_{42}	0	×	0	×	×	×	×	×	×	×	1	×	×	×	×	×	×	0	×	×
T_{44}	×	×	×	×	×	×	×	×	×	×	1	×	×	×	×	×	0	×	×	×
T_{52}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	×	0	×
T_{53}	0	×	×	×	×	×	×	×	×	×	×	×	0	×	×	1	×	×	×	×
T_{56}	×	0	0	×	×	×	×	×	×	×	1	×	×	×	×	×	×	0	×	×
T_{57}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	0	×
T_{58}	×	1	0	×	×	×	×	×	×	0	×	×	×	×	0	×	×	×	0	×
T_{59}	0	1	×	×	×	×	×	×	×	×	0	×	0	×	×	×	×	×	×	×
T_{60}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	0	0	×
T_{61}	×	×	0	×	×	×	×	×	×	×	×	×	×	×	1	×	0	×	×	×
T_{63}	0	×	0	0	0	×	0	0	0	×	×	×	×	×	×	1	×	0	×	×
T_{64}	0	0	0	0	0	×	×	×	1	×	×	×	×	×	×	×	×	×	×	×
T_{65}	0	×	×	×	×	×	×	×	×	×	×	×	×	0	×	×	×	1	×	×
T_{66}	1	×	0	×	×	×	×	×	×	×	×	×	×	×	0	×	×	×	×	×
T_{67}	×	×	0	×	×	×	×	×	×	×	×	×	1	×	×	×	×	×	0	×
T_{68}	0	×	0	×	×	0	×	1	×	×	×	×	×	×	×	×	0	×	×	×
T_{69}	×	×	×	×	×	×	×	×	0	×	1	×	×	×	×	×	×	×	×	×
T_{71}	×	×	0	×	×	×	1	×	×	×	×	×	×	0	×	×	×	×	×	×
T_{73}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	0	×	×	0	×
T_{74}	×	×	×	×	×	×	1	×	×	×	×	×	×	0	×	0	×	×	×	×
T_{75}	0	×	0	×	×	×	×	×	×	1	×	0	×	×	×	×	×	×	×	×
T_{76}	×	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	×	×
T_{77}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	×	0
T_{78}	×	0	×	×	×	×	×	×	×	×	×	1	×	×	×	×	×	×	×	×
T_{79}	1	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
T_{80}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0	×	×	1
T_{81}	0	×	0	0	×	×	×	×	×	×	×	1	×	×	×	×	×	0	×	×
T_{82}	0	×	0	×	×	×	0	0	1	×	×	×	×	0	×	×	×	×	×	×
T_{83}	×	×	×	×	0	×	0	×	×	×	×	1	×	0	×	×	×	×	×	×
T_{84}	0	1	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
T_{85}	1	×	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0	×	×
T_{86}	1	×	0	×	0	×	×	×	×	×	×	×	×	×	×	×	×	0	×	×
T_{88}	0	×	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
T_{90}	×	×	×	×	×	×	×	×	0	×	×	×	×	1	×	0	0	×	×	×
T_{95}	×	×	×	×	1	×	×	×	0	×	0	×	1	×	×	×	×	×	×	×
T_{96}	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	1	×
T_{99}	1	×	×	×	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- m=5, initial Workplans are:

Wk_1={S_{20},S_{15},S_1,S_3}, Wk_2={S_{18},S_7,S_{10},S_8}, Wk_3={S_{16},S_{14},S_5,S_{17}}, Wk_4={S_{19}, S_{12}, S_{11},S_6, S_9},Wk_5={S_2, S_{13},S_4};

- m'=5, final Workplans are :

Wk_1={S_{20}{T_{30},T_{37},T_{80}},S_{15}{T_{61},T_{73}},S_1{T_{19},T_{31},T_{66},T_{79},T_{85},T_{86},T_{99}},S_3{T_{36},T_{41},T_{88}}},Wk_2={S_{18}{T_{21},T_{35},T_{57},T_{65},T_{76},T_{77}},S_7{T_1,T_{34},T_{71},T_{74}},S_{10}{T_2,T_{20},T_{39},T_{75}},S_8{T_{32},T_{68}}},Wk_3={S_{16}{T_{53},T_{63}},S_{14}{T_{90}},S_5{T_{13},T_{16},T_{95}},S_{17}{T_{25},T_{38},T_{40},T_{52},T_{60}}},Wk_4={S_{19}{T_3,T_{22},T_{96}},S_{12}{T_{26},T_{33},T_{78},T_{81},T_{83}},S_{11}{T_{42},T_{44},T_{56},T_{69}},S_9{T_9,T_{64},T_{82}}},Wk_5={S_2{T_{28},T_{58},T_{59},T_{84}}, S_{13}{T_{29},T_{67}},S_4{T_6}};

- We assume that ICA agents are already visiting their first node of their correspondent final Workplans without incidence before the announce of the set of unavailable set of nodes. For the current example, we suppose that the set of unavailable node is :

S_1,S_3,S_7,S_{10},S_{14},S_5,S_{17},S_{12},S_9,S_{13}

- We deduce tasks to reassign:

TDR={T_{19},T_{31},T_{66},T_{79},T_{85},T_{86},T_{99},T_{36},T_{41},T_{88},T_1,T_{34},T_{71},T_{74},T_{90},T_{13},T_{16},T_{95},T_{25},T_{38},T_{40},T_{52},T_{60},T_{26},T_{33},T_{78},T_{81},T_{83},T_9,T_{64},T_{82},T_{29},T_{67}}→ 33 tasks to reassign,

The proposed negotiation process allows the reassignment of the cancelled services. F6, F7 and F9 in fig. 12 below represent three different generated FeTAR instances assignments for the same network error scenario, where the number of agreed assigned tasks was respectively 6, 7 and 9 in the priorities of ICA agents. Thanks to our proposed negotiation process, cancelled services find new available providers through an agreement between static scheduler agents and mobile collector agents of the system so the correspondent transport users are satisfied in spite of some network perturbations.

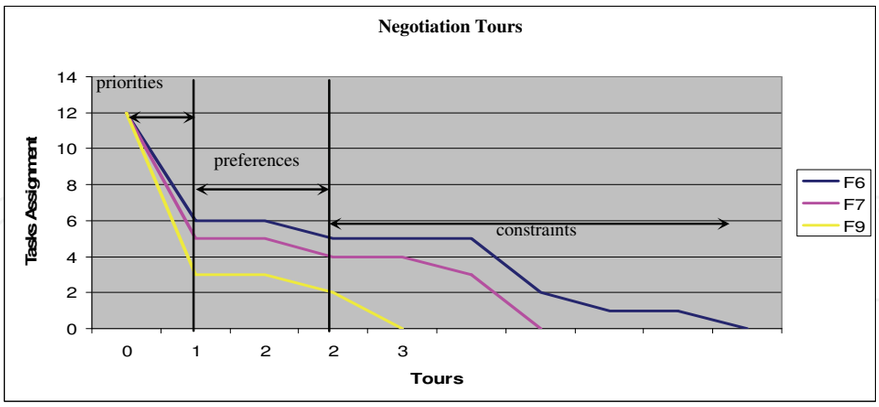

Fig. 12. Negotiation tours

8. Conclusion and prospects

In this paper, we firstly presented the proposed multi-agent architecture for the Transport Multimodal Information System. Then, we described the problem and situated the correspondent complexity. The main goal of our work is to give to transport customers the

best responses to their demands, optimising the composition of services in term of cost and total response delay. Therefore, we proposed a two-level optimization solution which optimizes the workplans of mobile entities in order to find the best management of the required information. Besides, we take into account the constraint of the possible network perturbations. In this case, mobile entities have to negotiate with the other entities of the system, in order to optimize the reassignment of the cancelled services. This procedure considers the positions of the mobile entities, their constraints, their preferences, their priorities and also the variable state of the network. However, the complex interaction between the different entities of the system exceeds the limits of the traditional negotiation procedures. That's why, the recourse to the ontologies was essential to design a special vocabulary to perform a proper semantic checks on a given agent expressions. In a future work, we aim to manage the interactions between several initiators and participants in different negotiation processes in case, for example, of simultaneous requests overlapping. Thus, the control of several concurrent conversations is indispensable. Moreover, the extension or the generalization of the proposed protocol is conceivable. Besides, until now, we have used the reference as a service index. In future works, we will offer to providers the possibility to be totally free in terms of services indication. Thus, to market its services through our system, an information provider will not be obliged to change anything locally in its structure. So, a supplier can register each proposed service with the already chosen label. In this context, the usage of a common ontology guarantees the consistency (an expression has the same meaning for all the agents) and the compatibility (a concept is designed, for the same expression, for any agent) of the information present in the system. However, it is not sure that all the agents will use a common ontology. Consequently, we will focus, in future works, on the data heterogeneity problem within our system, when each agent uses different terms with the same meaning or the same term for different meanings.

9. References

Baek, J.W.; Yeo J.H.; Kim, G.T. & Yeom, H.Y. (2001). "Cost Effective Mobile Agent Planning for Distributed Information Retrieval", *Proceedings of the* 21st International Conference on Distributed Computing Systems, pp. 65-72, School of Computer science and Engineering, Seoul national University, USA, 2001.

Buse, D.P.; Feng, J.Q. & Wu, Q.H. (2003). "Mobile Agents for Data Analysis In Industrial Automation Systems", *Proceedings of the IEEE/WIC Int'l Conf. on Intelligent Agent Technology (IAT'03)*, pp. 60-66, Halifax, Canada, October 2003.

Caire, G. & Cabanillas, D. (2004). *Jade Tutorial Applications-defined Content languages And Ontologies*, Tutorial JADE, November 2004.

Carzaniga, A.; Picco, G. P. & Vigna, G. (1997). "Designing distributed applications with mobile code paradigms", *Proceedings of the 19th Int'l Conf. on Software Engineering*, pp. 22-32, July 1997.

Ketel, M.; Dogan N.S. & Homaifar, A. (2005). "Distributed Sensor Networks based on Mobile Agents Paradigms", dept. of Computer Science, North Carolina A&T State

University, NC 27411, System Theory, SSST'2005, Greensboro, USA 2005, pp. 411-414.

Lu, X. & Mori, K. (2003). "Autonomous Information Services Integration and Allocation in Agent-Based Information Service System", *Proceedings of the IEEE/WIC Int'l Conf. on intelligent Agent Technology (IAT'03)*, pp. 290-296, Halifax, Canada, October 2003.

Pharm, V.A. & Karmouch, A. (1998). "Mobile Software Agents: an overview". *IEEE Communication Magazine*, University of Ottawa, Ontario, (July 1998) 26-37.

Picco, G.P. & Baldi, M. (1997). "Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications", *Proceedings of the 20th IEEE Int'l Conf. on Software Engineering (ICSE'97)*, pp.146-155, Kyoto, Japan, April 1998, In R. Kemmerer et K. Futatsugi.

Smith, R. G. (1980). "The Contract Net Protocol: highlevel communication and control in a distributed problem solver", *IEEE Transactions on computers*, Vol.29, No.12, (December 1980), 1104–1113.

Theilmann, W. & Rothermel, K. (1999). "Efficient Dissemination of Mobile Agents", *Proceedings of the 19th IEEE Int. Conf. on Distributed Computing Systems Workshop (ICDCSW')*, IEEE Computer Society, pp. 9-14, USA, May 1999.

Zgaya, H. & Hammadi, S. (2006a). "Dynamic Approach to Reassign Tasks when Servers Breakdown in a Multi-Modal Information System", *Proceedings of the 2006 IMACS Multiconference on Computational Engineering in Systems Applications (CESA'2006)*, pp. 985-990, Beijing, China, October 2006, Tsinghua University Press, Beijing, China.

Zgaya, H. & Hammadi, S. (2006b). "Assignment and Integration of Distributed Transport Services in Agent-Based Architecture", *Proceedings of the IEEE/WIC Int'l Conf. on Intelligent Agent Technology (IAT'06)*, HongKong, China, December 2006.

Zgaya, H. & Hammadi, S. (2008). "Combination of mobile agent and evolutionary algorithm to optimize the client transport services". *RAIRO-Operations Research Special Issue on Cooperative methods for Multiobjective Optimization RAIRO Oper. Res.*, Vol.42, No 1, (January 2008) 35-67.

Zgaya, H.; Hammadi, S. & Ghédira K. (2005a). "Workplan Mobile Agent for the Transport Network Application", *Proceedings of the 17th IMACS World Congress Scientific Computation Applied Mathematics and Simulation (IMACS'2005)*, pp. 11-15, Paris, France, July 2005.

Zgaya, H. ; Hammadi, S. & Ghédira K. (2005b). "Evolutionary method to optimize Workplan mobile agent for the transport network application", *Proceedings of the International Conference on Systems, Man and Cybernetics (IEEE SMC'2005)*, pp. 10-12, Hawaii, USA, October 2005.

Zgaya, H. ; Hammadi, S. & Ghédira, K. (2008) . "A migration strategy of mobile agents for the transport network applications". *International Journal, Mathematics and Computers in Simulation (MATCOM)*, Vol.76, No.5-6, (January 2008) 345-362.

Zitzler, E. & Thiele, L. (1998). "Multiobjective Optimization Using Evolutionary Algorithms: A Comparative Case Study". *In Lecture Notes in Computer Science*, Vol. 1498, (1998) 292-301, UK.

Multiagent Systems

Edited by Salman Ahmed and Mohd Noh Karsiti

ISBN 978-3-902613-51-6
Hard cover, 426 pages
Publisher I-Tech Education and Publishing
Published online 01, January, 2009
Published in print edition January, 2009

Multi agent systems involve a team of agents working together socially to accomplish a task. An agent can be social in many ways. One is when an agent helps others in solving complex problems. The field of multi agent systems investigates the process underlying distributed problem solving and designs some protocols and mechanisms involved in this process. This book presents an overview of some of the research issues in the field of multi agents. It is a presentation of a combination of different research issues which are pursued by researchers in the domain of multi agent systems as they are one of the best ways to understand and model human societies and behaviours. In fact, such systems are the systems of the future.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen