ETFA'2006 - 11th IEEE International Conference on Emerging Technologies and Factory Automation
Prague, Czech Republic, 20-22 September 2006

A Clone-Pair Approach for the Determination of the Itinerary of Imprecise Mobile Agents with Firm Deadlines

Luciana Rech, Carlos Montez and Rômulo de Oliveira
Department of Automation and Systems Engineering
Programa de Pós Graduação em Engenharia Elétrica
Federal University of Santa Catarina
P.O.Box 476, CEP 88040-900.
Florianópolis, SC, Brazil.
lrech, montez, romulo@das.ufsc.br

***Abstract* - In some applications, as part of a mobile agent mission, there may be the necessity of meeting a firm end-to-end deadline, although with some itinerary flexibility due to optional resources. In this paper it is considered an execution model where agents do not previously know their itineraries. The objective is to propose and to evaluate heuristics to be used by pairs of imprecise mobile agents that must accomplish a mission within a firm deadline.**

I. INTRODUCTION

A mobile agent is a self-contained software element that is responsible for the execution of a task, but it is not limited to the system where its execution begins [1, 2]. The mobile agent is capable of migrating autonomously through nodes of a network, by interrupting its execution in a place, being transferred to another place, and resuming its execution there. Mobile agents reduce the network load. They execute asynchronously and autonomously, and they can adapt dynamically.

A possible scenario is the use of mobile agents with timing constraints in the production line of a factory. The mobile agent's mission is to supervise the operation of a production line. The agent function is to send reports describing the current situation of the system, and to diagnose failures. The agent has the function of gathering data in order to form an image of the problem and to decide the next visit. In this case, the mobile agent reaches the fault diagnosis by traveling through several nodes and by collecting data for the decision making. Examples of that type of environment can be found in applications of factory supervision (mobile agents in the production line [3]). This scenario is not different from the usual approach of having an engineer or technician with a laptop or measuring device walking around the factory floor, trying to figure out what is the source of a detected problem.

In this work we consider the association of real-time requirements to mobile agents. We consider mobile agents that have the necessity of meeting timing requirements in a distributed system. An imprecise mobile agent is a mobile agent capable of reducing the quality of its result in order to meet the mission deadline.

In [4] it was defined a real-time imprecise mobile agent model together with some simple heuristics to guide the mobile agents in the definition of their itinerary. In this article, differently of [4], we assume that the itinerary possibilities are ignored before the mobile agent's departure, and we will compare the performance of the heuristics according to different quality metrics.

The goal of this paper is to describe an approach based on pairs of mobile agent clones that are imprecise but have real-time constraints. We describe and analyze algorithms for the dynamic determination of their itinerary.

Section 2 presents other work found in the literature related to real-time mobile agents. Section 3 has an overview of the execution model. Section 4 contains the heuristics used for definition of the itinerary. Section 5 presents the approach based on pairs of mobile agent clones. It is also presented a comparison of the performance of some pairs of heuristics. Section 6 has the conclusions.

II. RELATED WORK

Some work in the literature present studies on mobile agent technology and real-time requirements. In [5] a routing system is described based on agents (ARS Agent-based Routing System), which allocates network resources for tasks that request point to point communication in real-time. Its resource management mechanism is so that mobile agents collect the current state of the connections in the system to examine the availability of possible routes to each node.

Another work that deals with mobile agents and timing restrictions can be seen in [3], where the decision making process is implemented through negotiation among mobile, intelligent and autonomous agents. A negotiation protocol, based on mobile agents (MANPro) is applied in a real-time scheduling system for a production line.

RT Monitor is a system of real-time data management for browsing applications traffic [6]. In that system, browsing requests with timing constraints are originated and the RT Monitor calculates and communicates the best routes to the requesting tasks. The precision of the suggested routes depends on the consistency of the traffic data. In order to minimize the overhead, a cooperative and distributed method is adopted by using mobile agents. It seeks to improve the

schedulability of the system and to reduce communication traffic.

In [7] a routing algorithm based on mobile agents is proposed, where the cost of the traffic is considered. It is defined a traffic cost function for each connection based on well-known information of traffic. A probabilistic distribution is calculated which the mobile agents can use to select the next node where to go. Routing is the key factor for the network performance. In the process of moving packages of data from its origin to its destiny, once a request of sending a package is received, the router should recommend the optimal route (the shortest path) to send this package through the network. Routing algorithms based on mobile agents are a promising option for the use in these environments [8].

In [9] a method is described for fault tolerance that uses mobile agents with timing constraints.

BAEK *et.al*. in [2] propose a method for planning mobile agents with timing restrictions (TMAP - Timed Mobile Agent Planning) in order to find the minimum number of agents and the best itinerary for the agents to accomplish information recovery in a distributed computation environment. Mobile agents should respect timing constraints while they maintain the minimum time of total routing.

Among the mentioned works, just in [2] there is the concern of choosing an itinerary so that timing restrictions are respected. However, in the execution model proposed in this paper, there is the concern of choosing the best itinerary considering the performance of nodes and network and the difficulty of meeting deadline. In this context, the definition of what is the best itinerary may vary according to the metric used to evaluate itineraries. Adaptive scheduling techniques will be used with the objective of establishing a compromise between the visits to nodes and the timing restrictions of the mission.

The problem considered in this work is also similar to that presented in [10], called "The prize collecting traveling salesman problem." However, algorithms such as the one presented in [10, 11] are computationally intense and targeted to the static determination of the itinerary, when there are no real-time requirements.

III. OVERVIEW OF THE EXECUTION MODEL

The execution model considered in this paper was previously defined in [4]. In this section it is presented an overview of that model. The imprecise mobile agent execution model describes the behavior of an application based on mobile agents that have a firm deadline in a distributed system. Each mobile agent has a mission that is associated with the visit to a certain group of nodes of the system. These nodes host the necessary resources for the agent to accomplish its mission. In this paper it is assumed that this agent does not communicate with any other agent.

A resource is an abstraction and it can correspond to a processor, device, file, data structure, etc. There is a benefit associated with each resource based on its functional importance within a particular mission. There is a set of resources in each node of the system. Instances of the same resource type may exist in different nodes. The processor is always a necessary resource for the agent and it is presented in all nodes, so it will be considered in an implicit way.

In each node ***Ni*** of the system there is a set ***Ri*** of one or more types of resources belonging to set ***R***. Each resource type belonging to set ***R*** can be found in one or more nodes, that is, several instances can exist (replicas) of the same resource type, but in different nodes.

Figure 1 illustrates a situation where there are instances of the resource type ***r3*** at nodes ***N1*** and ***N2.*** Also, resource type ***r9*** appears at node ***N1*** and ***N6***.

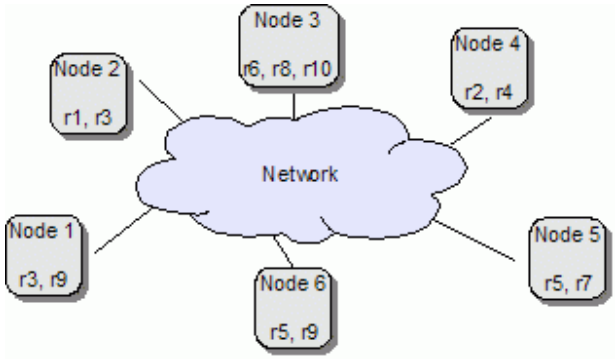

Fig.1. Resources in a distributed system

An itinerary ***I*** is defined as a route traveled in the distributed system. It is described as an orderly sequence of nodes ***Ni*** belonging to set ***N***. An itinerary does not need to include all nodes of the system, and it can include in its defining sequence the same node several times.

Each mobile agent has a mission that defines a benefit function ***B*** for each resource type. This function determines the amount of benefit ***Bi*** each resource type ***ri*** contributes to the agent's mission.

The goal of the agent is to obtain benefits through the use of several resources that are spread through several nodes interconnected by a communication network. The objective of the agent function is to maximize the total benefit collected along its itinerary in the system. The agent starts at an external node ***N0*** (node origin) and it must return to this same node at the end of the mission. Node ***N0*** does not have any instance of the types of resources that make set ***R***. Clock synchronization is assumed in the system; and every mission ***M*** has a firm deadline ***D***. The agent ***Z*** should complete the mission (by returning to node ***N0***) before ***D***, otherwise it loses all the benefit collected along the itinerary.

The resource diagram of a mission (see the Figure 2) corresponds to a UML activity diagram that describes the precedence relations among resources. Any resource used in disagreement with this diagram brings no benefit for the mission. The resource diagram of the mission describes the

flexibility and options of the agent when accomplishing the mission. Resource type ***r0*** is a pseudo-resource, it is found only in node ***N0*** and it indicates that the agent must return to the original node.

The resource diagram is gradually known by the mobile agent along the route, because it is defined by the information that the agent accesses at each node. In that way, the mobile agent is defined as myopic, that is to say, at each instant it only knows the possible immediate next nodes.

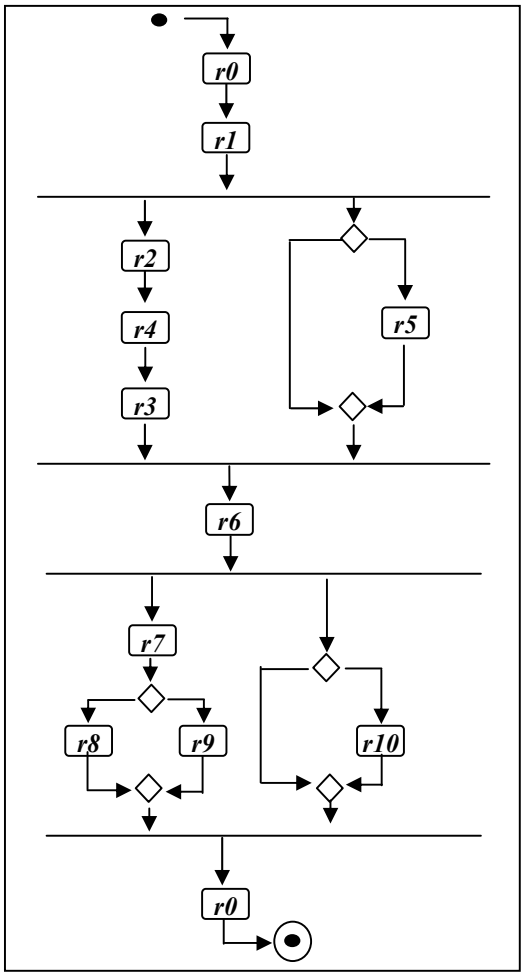

Fig.2. Resource Diagram

The node diagram of a mission (see Figure 3) is a UML Activity Diagram based on the Resource Diagram of the mission. Each resource is replaced by the node or nodes where that resource appears in the system. It presents the possibility of many different itineraries that the agent can take in order to execute its mission.

The execution of a mission is related to the choice of an itinerary ***I***. For each $r_i \in R$, the benefit $B_{i,t}$ obtained by using resource r_i for the time interval t is given by:

$$B_{i,t} = B_i \quad if \quad t \geq C_i \qquad (1)$$

$$B_{i,t} = 0 \quad if \quad t < C_i$$

In the above equation the precedence and optionally relations must be considered:

- if *ri* precede *rj*, and *ri* was not utilized, then $B_j = 0$;
- if *ri'* e *ri"* are replicas of *ri*, and *ri'* was already utilized, the maximum benefit from *ri"* will be zero.

The effective benefit of the mission is obtained by the sum of the benefits collected from each resource in the itinerary:

$$B = \sum B_i(y) \quad \text{for each} \quad ri \in R \qquad (2)$$

where $B_i(y)$ is the benefit realized by the utilization of resource *ri* that the agent visited in order to fulfill its mission through itinerary *y*, *B* is the benefit generated by the utilization of all resources in the mission.

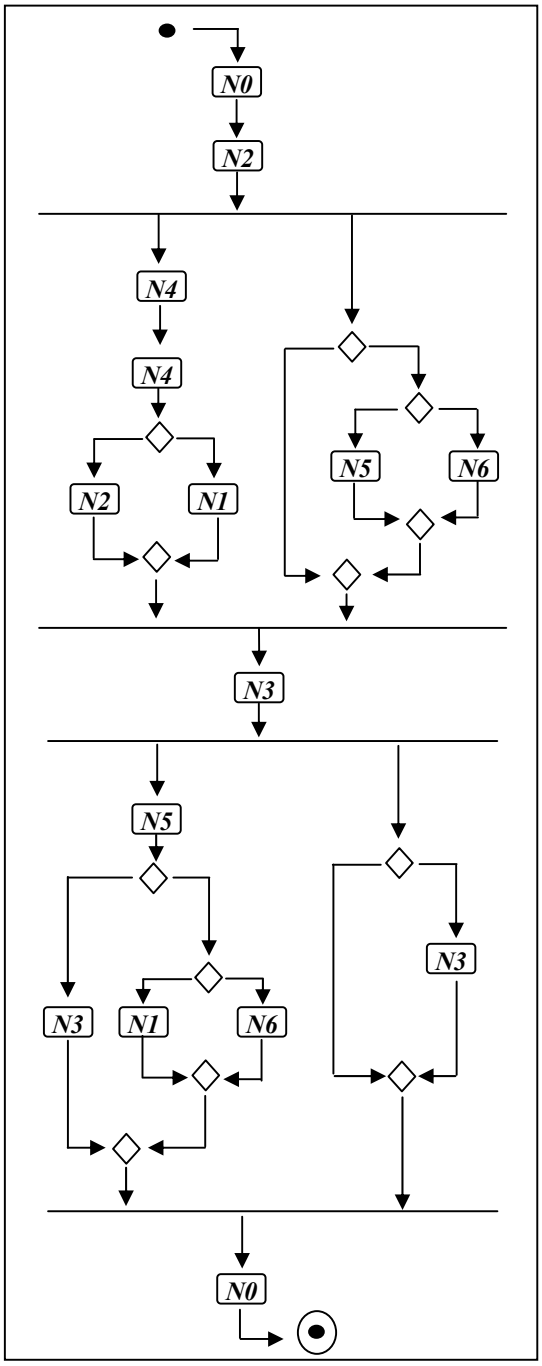

Fig.3. Node Diagram

The problem described in this paper cannot be solved by using optimization approaches, because at the starting node the resource diagram is not completely known; and during the itinerary the visited nodes have limited processing capacity and they cannot make the calculations necessary for an optimization approach.

IV. ALGORITHMS FOR THE DEFINITION OF THE ITINERARY

This section presents some heuristics for the mobile agent itinerary definition. They were originally defined in [4]. The heuristics used are simple (they require minimum computational effort) and myopic (they work based on a partially known resource diagram). Whenever the agent needs to decide between two routes that, in the perspective of the heuristic used, they are equivalent, a random decision is made.

1) Lazy Algorithm

It tries to execute the resources in the fastest possible way, ignoring its benefits. Whenever there is an alternative (when there is not a precedence relation between two resources), it will be chosen the resource that presents the smallest computation time. When there are replicas of a resource, the replica hosted by the closest node is chosen.

2) Greedy Algorithm

Whenever there is an alternative route, it will be chosen the resource that presents the largest benefit for the mission. When there are replicas of a resource, preference will be given to the resource located in the closest node.

3) Random Algorithm

This algorithm chooses randomly the next node to be visited by the agent. This behavior is only possible if there are several instances of a certain resource or there is not a precedence relation between the next resource and some of the other resources yet to be executed.

4) *Higher Density Algorithm*

This algorithm is based on AVDT (Average Density Threshold) [12]. It chooses to be the next resource that one that presents the best benefit/execution time rate. This characteristic is possible only when there is not a precedence relation between two resources. The density of the benefit of the resource is defined by the equation:

$$db_i = B_i / (C_i + L_{j,i} + Q_i) \qquad (3)$$

where: db_i = benefit density of resource i, B_i = benefit obtained by executing resource i, C_i computation time used by resource i, $L_{j,i}$ is the time necessary to move the agent to node Ni, and Q_i is the time spent in the local processor queue. Estimated values can be used when the exact values are not known. The agent maintains the average value until the present moment. When a resource is optional, it will be executed only if its density is higher than the average density of the mission to that moment.

5) *Versions with Clock*

Algorithms with clock in this paper have a probabilistic behavior. In the beginning of the execution they follow their original behavior; but as the deadline of the mission approaches, these algorithms adopt characteristics similar to the Lazy Algorithm. At any moment, the probability of the algorithm to behave like the Lazy Algorithm is directly proportional to the amount of time already consumed.

The heuristics were simulated considering the node diagram of Figure 3. Based on these data, a graph (Figure 4) was created composed by 116 vertexes and 149 edges, where each vertex represents a state in the agent's behavior (for example: arrival in the node, execution of the resource) and the edges represent the network. The time consumed and the benefit for the execution of a certain resource is showed inside the vertexes.

After the execution of each resource, some benefit is received and this benefit is added to the total benefit of the mission. The objective of the mission is to reach the greatest benefit while respecting the specified deadline.

For the calculation of the total computation time of the mission it was considered:

- The computation time of each resource;
- The processor queue time, according to an uniform distribution between 1 and 3 (light load) or between 1 and 10 (heavy load);
- The communication time $\boldsymbol{L}$ between nodes (exponential distribution with average 2).

The quality of the algorithm (global benefit $\boldsymbol{G}$ of the algorithm) is calculated by the following equation:

$$G = \sum B\_DA \ / \ NT \qquad (4)$$

where:

B_DA: it is the benefit obtained by the utilization of resources while meeting the deadline;

NT: it is the number of tries (met deadlines + missed deadlines).

The Figure 5 presents the global benefit $\boldsymbol{G}$ obtained by each algorithm for situations where the system load is respectively light (a) and heavy (b).

Analyzing the graphic of Figure 5 it is noticed that, starting from the moment when all agents meet their deadlines, the global benefit obtained by the algorithms stays stable, independent of the remaining time to the deadline.

The algorithms Lazy and Greedy represent the two extremes. The Lazy algorithm presented the best performance with tight deadlines (for example, 90). The Greedy algorithm obtained the largest benefit with loose deadlines (for example, 165).

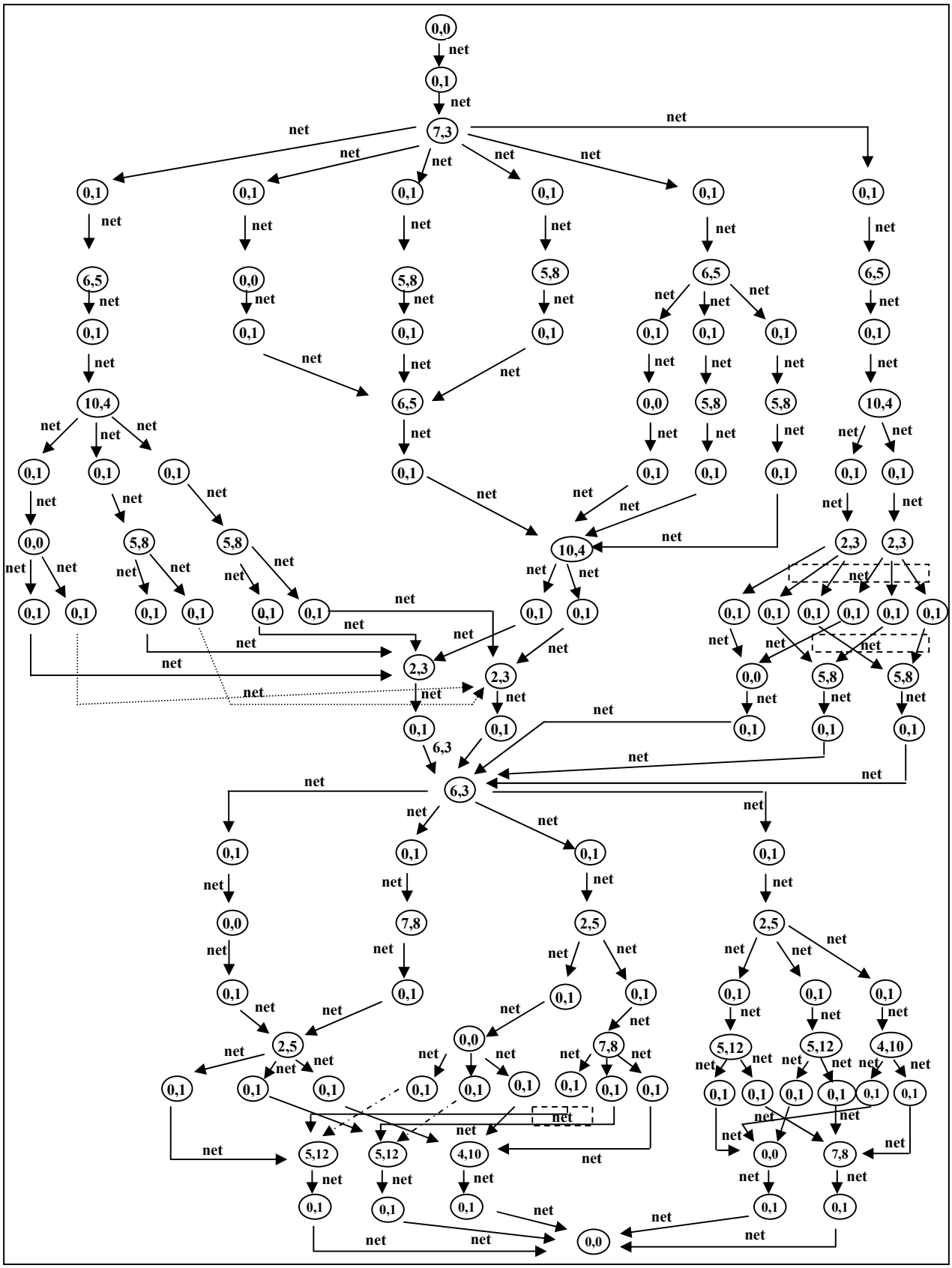

Fig.4. Graph for the example application

The algorithms Lazy and Density presented close results, but for tight deadlines the first presented better results. The algorithms Random, Greedy with clock and Density with clock presented close and average results when compared to the performance of the other heuristics.

Analyzing the results presented in the graphics of Figure 5, it is noticed that the best performance of the heuristic used by the mobile agent is closely related to the distributed system conditions. That is, the load of the network and of the nodes. Therefore, for different situations, different heuristics obtained the best result for the global benefit. For example, with tight deadlines the algorithm Lazy presented better benefit. With loose deadlines the heuristic that obtained the best result was the Greedy. A deadline can be considered loose or tight only by analyzing the description of the distributed system situation at the moment the agent begins its mission.

It is not possible to choose a single best heuristic for all possible scenarios, since their performance is very dependent of the distributed system conditions.

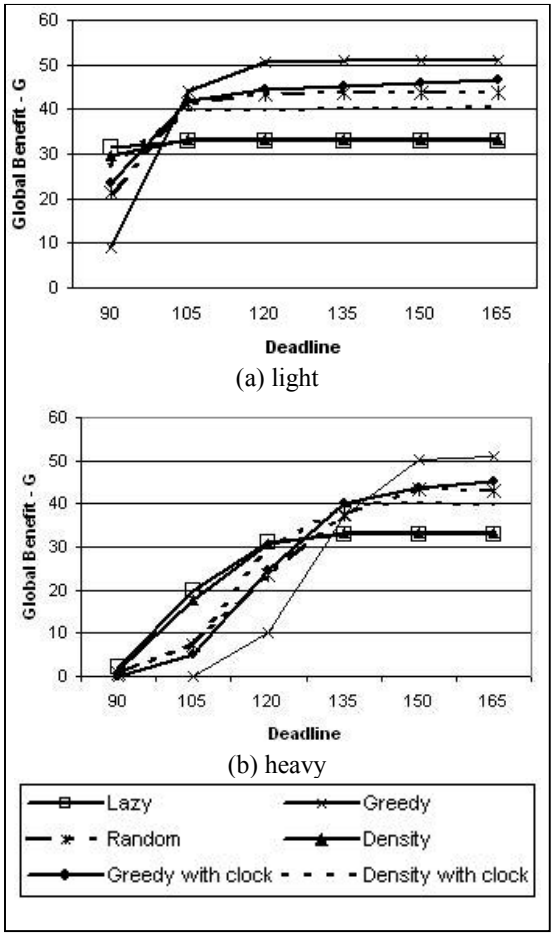

Fig.5. Performance of the heuristics

V. THE CLONE-PAIR APPROACH

Analyzing the results of the simulations described above, it was noticed the need to combine them to reach better results. The myopia of the agent can limit the efficiency of the existing heuristics. A possible solution is the use of an agent clone that adopts a second heuristic.

Many factors influence the decision of which is the best heuristic. These factors are directly related to the deadline of the mission. In the attempt of obtaining the best results in general, we propose the use of mobile agent's clones. At the moment that the mission begins, a pair of mobile agents using different heuristics is launched. The pair (the original mobile agent and the mobile agent clone) leaves following different itineraries in search of the same objective. At the end of the mission both come back to the starting node and then it is made a comparison and selection of the pair's best individual results. This approach will probably increase the global benefit of the mission. It will certainly not decrease the global value. In the case of one of the mobile agents do not return to the starting node within its deadline, the data obtained by the agent that already arrived will be used. We keep the system overhead low by limiting the number of clones to two.

Observing the performance obtained individually by the algorithm Greedy with a heavy load (5b), we noticed that it presented expressive results only with deadlines above 135. When compared to the performance of other heuristics and deadlines smaller than 120 it was a worse solution. We also noticed that in situations where the agent has a larger deadline, the largest benefit among all heuristics was obtained with this algorithm (for example, with the deadlines 150 and 165).

The same happens when the load in the system is considered light. In a first test with deadline of 90 its benefit when compared to the other algorithms was very small, but the same test made with larger deadlines (D = 120, 135, etc) this algorithm reached the highest amount of benefit.

We now analyze the algorithm Lazy that presents the opposite results of the algorithm Greedy. Considering a system with a heavy load (5a), algorithm Lazy obtained the best performance when the deadline defined for the mission was tight (for example, deadline of 90). The same happens with the light load and tight deadlines, until a certain point this algorithm obtained the highest benefits, being surpassed by the other heuristics only when the deadline became loose, that is, deadlines of 135 and above.

In the attempt of finding an algorithm that presents a behavior satisfactory we idealized a flexible heuristic that was capable of dealing with different load situations of the system and different mission deadlines. In search of this flexible heuristic we analyzed the individual performance of each heuristic and we proposed some clone pairs to be tested.

Figure 6 presents the behavior of the algorithms working in pairs. They were simulated for situations in that the system had a heavy load (6b) and a light load (6a). The global benefit obtained by each pair of agents is presented.

After the simulations we concluded that the best heuristic pair for the proposed scenario is the Lazy/Greedy pair. They

obtained a benefit close to the best with tight deadlines. They also achieved the best benefits with average and loose deadlines.

If we analyze the performance of pair Density/Greedy_clock and Lazy/Greedy_clock we can notice that they presented results close to each other and only average results.

The clone pair Lazy/Density-clock obtained the greatest benefit in situations with light load and tight deadline, but only with a small difference from the Lazy/Greedy pair. At the other tests this pair did not present good results when compared to the others pairs.

On the other hand, the pair Lazy/Greedy obtained good results for different load types and different deadline values.

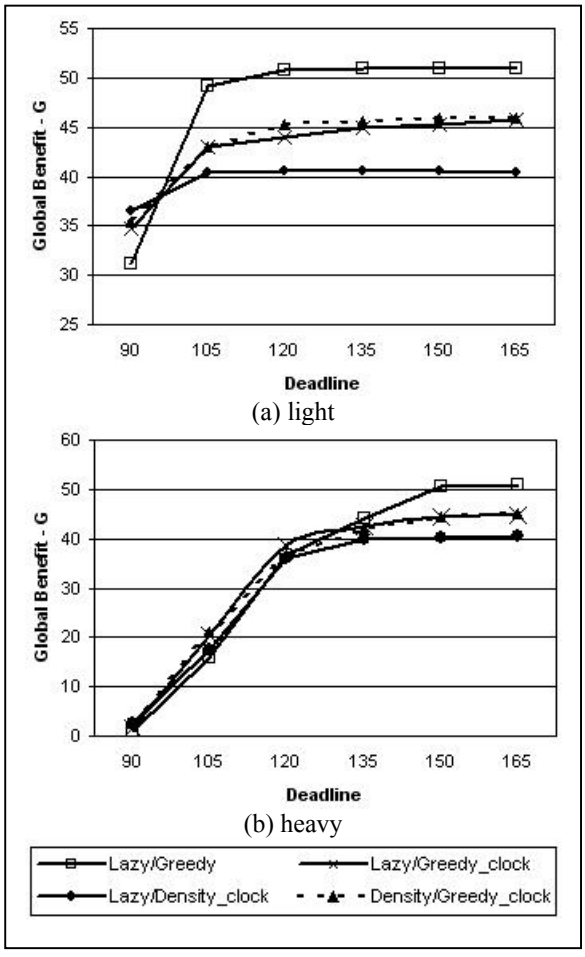

Fig.6. Performance of the agent clones

VI. CONCLUSIONS

This paper considered an execution model based on imprecise mobile agents with firm deadlines but some flexibility in the definition of their itinerary. This execution model associates timing requirements and the concept of optional resources to the technology of mobile agents. It was described several heuristics that can be used by the mobile agent for the determination of its itinerary. A comparison among the several heuristics showed that it is necessary to establish a compromise between mission value and execution time.

In this paper we proposed the use of pairs of mobile agent clones as an approach to deal with the difficulty of having a single heuristic for the problem. By using heuristics with very different behavior we are capable of dealing with different scenarios of load and deadlines. Our experiences showed that a simple combination of Lazy and Greedy agents is very effective.

As future work , we plan to adapt some of these heuristics so as they may use their knowledge (acquired with previous executions) to aid in the decision making regarding the definition of the mobile agent's itinerary. Another open question is how to use the knowledge that the mission is concluded when there is still some time before its deadline, and how to use this remaining time to increase its benefit.

We will continue the search for algorithms that present a satisfactory behavior for varying load and deadline conditions. It may be possible to adapt the heuristic to different situations of system load and mission deadline.

VII. REFERENCES

[1] G. P. Picco, "Understanding, Evaluating, Formalizing, and Exploiting Code Mobility*"* PhD Thesis, Politecno di Torino, Italy, 1998.

[2] J.W. Baek, G.T. Kim, and H.Y. Yeom, "Timed Mobile Agent Planning for Distributed Information Retrieval" Computation and Engineering School, Seoul National University, Proceedings of AGENTS´01, Montreal, Quebec, Canada, June, 2001.

[3] M. Shin, M. Jung, and K. Ryu, "A Novel Negotiation Protocol for Agent-based Control Architecture", 5th International Conference on Engineering & Automation, Las Vegas, EUA, 2001.

[4] L. Rech, R. de Oliveira, and C. Montez, "Dynamic Determination of the Itinerary of Mobile Agents with Timing Constrain*"*, IAT 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 45-50, Compiègne, France, September, 2005.

[5] K. Oida, and M. Sekido, "ARS: An efficient agent-based routing system for QoS guarantees*"*, Elsevier Science, Computer Communications 23, pp. 1437-1447, 2000.

[6] K. Ramamritham, A. Kwan, and K. Y. Lam, "RTMonitor: Real-time Data Monitoring Using Mobile Agent Technologies*"*, Proceedings of 28th VLDB Conference, Hong Kong, China., 2002.

[7] W. Qu, H. Shen, and Y. Jin, "Theoretical Analysis on a Traffic-Based Routing Algorithm of Mobile Agents*"* IAT 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 520-526, Compiègne, France, September, 2005.

[8] G. D. Caro, and M. Dorigo, "Two Ant Colony Algorithms for Best-Effort Routing in Datagram Networks*"* Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 541-546, 1998.

[9] J. Xu, and S. Pears, "A Dynamic Shadow Approach to fault-tolerant Mobile Agents in an Autonomic Environment", Real-time Systems, Springer Science + Business Media, Inc. Manufactured in the Netherlands, December, 2005.

[10] E. Balas, "The Prize Collecting Traveling Salesman Problem" John Wiley & Sons, Inc. Networks, Vol.19, pp.621-636, 1989.

[11] E. Balas, "The Prize Collecting Traveling Salesman Problem: II. Polyhedral Results", John Wiley & Sons, Inc. Networks, Vol.25, pp.199-216, 1995.

[12] A. Davis, *et al*. "Flexible Scheduling for Adaptable Real-Time Systems", Proceedings IEEE Real-Time Tech. and App. Symp, pp. 230-239, May, 1995.