Using Intelligent Mobile Agents to Dynamically Determine Itineraries with Time Constraints

Alex Magalhães, Luciana Rech, Lau Cheuk Lung, Rômulo Silva de Oliveira
Informatics and Statistics Department
Department of Automation and Systems Engineering
Federal University of Santa Catarina
{alex, luciana.rech, lau.lung}@inf.ufsc.br, romulo@das.ufsc.br

Abstract

Algorithms to determine the itinerary of intelligent agents are fundamental in the context of distributed applications based on mobile agents with time constraints. In order to establish the agent itinerary it is necessary to consider the trade-offs between high-quality of results and meeting the deadline. In this paper, we describe and evaluate adaptive heuristics able to choose the best behaviour to be used for decision making in the itinerary definition. This decision-making is based on a log of benefits collected by the mobile agent in past executions. The paper's objective is to propose a new heuristic able to realize the change in the environment and adapt itself dynamically, changing its behaviour in order to meet the deadline of the mission and to achieve the greatest possible benefit.

1. Introduction

The mobile agent paradigm is heavily based on the idea of code mobility. Code mobility is an alternative to the traditional approaches based on the exchange of messages. It brings benefits in the context of several application classes. Among the benefits expected from code mobility, it is possible to list [1]: higher level of flexibility, scalability and customization; better use of the infrastructure of communication and the provision of services in an autonomous way without the necessity of permanent connection. Such benefits justify the growing interest in this area of research. Some classes of applications for which code mobility has been applied in the literature are: information recovery, advanced telecommunications services, control of remote devices, workflow management and electronic commerce, among others.

A mobile agent is an element of self-contained software responsible for the execution of a task. It is not limited by the system where it begins its execution, being able to migrate autonomously through a network. Mobile agents reduce the load in the network. They execute asynchronously and adapt dynamically, establishing a new paradigm for programming in distributed environments. In this context, an intelligent agent is an autonomous entity which in real time observes and acts upon an environment and directs its activity towards achieving goals. Intelligent agents may also learn or use knowledge to achieve their goals [2]. In this way, the agents can operate in unknown environments and be able to analyze itself in terms of behaviour (error and success) to become more competent than its initial knowledge alone might allow.

The sequence of nodes visited by a mobile agent, between a node origin and a node destiny, it is called an itinerary. A relevant question for the determination of the itinerary of a mobile agent is its prior knowledge of the nodes that may be visited. In this paper, the timing aspects, along with the feasibility of code mobility, are important requirements.

In the context of distributed applications, there are several approaches that allow for real-time applications to adapt dynamically to different platforms. The imprecise computation technique, for example, can be used for this purpose. In this technique, each application task is able to generate results with different levels of quality or accuracy. With this objective, tasks are divided into mandatory parts and optional parts. The mandatory part is capable of generating a result with the minimum quality necessary to maintain the system operating in a safe manner. The optional part refines the result, until it reaches the desired quality.

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 travelling 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.

This work deals with the use of code mobility along with time requirements in a distributed system. The concept of imprecise mobile agents with time constraints was originally presented in [4] together with some

simple heuristics to guide the agent in defining the route. From this assessment of performance it was clear the need to create heuristics that are adapted to the needs of the mobile agent. The first step toward this behaviour was the creation of adaptive heuristics able to decide the best behaviour to be used for the mission [5]. Based on a log of previous executions, the mobile agent is able of selecting the most appropriate behaviour for the current mission. In [5], adaptation happens on startup, that is, the agent has always the same behaviour throughout the mission.

In this paper we present a new heuristic capable of dynamic adaptation, unlike the works mentioned above. That is, during the itinerary of the mobile agent, there may be changes in the conditions of the network or the nodes to be visited. Then, the intelligent mobile agent - concerned not only with meeting the deadline, but also to achieve the greatest possible benefit for the mission - is capable of changing the behaviour initially adopted. It will be verified that the proposed heuristic is efficient for all ranges of deadlines, but mainly for the tightest ones.

Decision-making techniques used by the heuristics are based only on the observation of the possible next nodes that the agent could visit. The heuristics used are simple (small computational effort) and myopic (working on diagram of resources and diagrams of nodes that are only partially known). In this paper, the intelligent mobile agent has a memory-based log with the information about the previous missions. Based on this knowledge it is able: to analyze the environment, to compare with the mission deadline, to learn and dynamically improve its behaviour in the determination of the itinerary.

The text is divided into 6 sections. Section 2 shows some related work. In Section 3 the computational model used is briefly described. Section 4 contains the adaptive heuristics used in defining the route. Section 5 presents a comparison and evaluation between the heuristics. Finally, conclusions appear in Section 6.

2. Related Work

This section presents some approaches where mobile agents are used in distributed applications with time constraints.

In [6], it is presented an algorithm for routing of mobile agents that considers the cost of connections. To assist in the decision making of the agent, there is a probability distribution of the agent to select one of the mobile nodes neighbors and move to it. The cost of the link between two nodes is defined based on known traffic information and the distribution of probability found. With respect to how the route is defined, in [6] many agents leave in search of the best route, they return to the home node (server), and then select the best route, and only one agent part again to meet its mission and now it already knows its route. In the present article will be dealt with the determination of dynamic routing, where the mobile agent is myopic, that is, only knows the following nodes to visit during your trip. The flexibility in choosing this route will depend on some characteristics of some resources.

In [7], the DMAP (*Dynamic Mobile Agent Planning*) technique is described. The main performance components in the planning of a mobile agent system are: the number of agents, the time spent by the participant agents and total routing time consumed by all of the agents, while keeping the turn-around time to the minimum. In [7] the myopia of the agent is not addressed, since the itinerary is already known at the beginning of the agent's mission, being capable of changing it dynamically due to the current conditions of the network. Differently of [7], in this paper the planning of the mobile agent itinerary happens dynamically. Each mobile agent has one mission and must meet the deadline, the agent does not know the path before its departure.

Monitoring of environments is not a trivial task. Combining the need for a high frequency of real-time data update with the fact that data are not always visible at a distance, the situation may be even more challenging. In [8] it is proposed a technique with mobile agents, which is scalable and able to meet real-time requirements. The problem presented is the monitoring of Bluetooth devices in a university and the mobile agent strategy is divide-and-conquer. The university is divided into levels and the agents working in hierarchies within the deadlines. One of the criticisms of the solution presented in that paper is that despite the high flexibility and scalability of the solution, it is not presented an alternative for problems with very tight deadlines.

In [9] it is presented an algorithm for searching in real-time in unknown surfaces called RTAA*. The proposed algorithm uses past states to keep the focus on its goal. It is a variation of the classic algorithm of Artificial Intelligence A*. To meet the requirement of real-time computing the heuristic presented uses imprecise computation. Differently of [9], this article works with a wider concept of real-time, and the term is applied to the entire mission, and not just to the next part of it, being then applicable to a wider range of problems. The present article is also stricter about the foot-print of the algorithms, having a small and constant computational complexity for every node of the problem, which is not the case for RTAA* (and variations like LRTA*), where the complexity grows along the mission, since it uses every past node to determine the next step.

In [10] it is posed an interesting scenario for lightweight MAS (Mobile Agents Systems), because the traditional TSP (*Travelling Salesman Problem*) cannot be solved by using classical optimization solutions for this kind of problem. The scenario is a MAS on a Manet (*Mobile ad-hoc Network*), where itinerary is unknown at the beginning of the agent's mission, since the mobile

nodes (PDA' s, cell phones, laptops, etc.) can enter and leave the network without further notice, turning optimizations obsolete very quickly. This scenario can be used by the itinerary heuristics presented in this article, because the myopic mobile agent does not think ahead its path. The work in [10], however, focuses on the survival of services within the network. For this, a large biological ecosystem is proposed, a variation of the ant-based solution. Services are replicated in different nodes and it is up to the entire ecosystem, a decentralized form of organization, to find its equilibrium in number of agents and services to guarantee the dependability of the network.

3. Model Description

In this section it is briefly described the computational model presented in [4]. This model describes the behaviour of applications based on imprecise mobile agents with time constraints in a distributed system formed by a set of nodes connected by an underlying communication service.

We assume that agents do not know their itineraries at the starting node. In this computational model, each intelligent mobile agent has a mission associated with a visit to a particular group of nodes of the system. These nodes have the resources necessary to complete its mission. It will be assumed that agents do not communicate. The mission consists of obtaining a group of specific resources within a deadline, e.g., a mission is composed by a set of resources/tasks that must be executed. A resource is an abstraction and may correspond to a processor, file, data structure, etc.

An imprecise intelligent agent is capable of reducing the quality of the result in order to meet the deadline of the mission. In this paper we assume a diagram that shows the precedence relations and options among the resources being used by the agent on its mission. Since many resources may be replicated in the system, it is useful to build a diagram of nodes from the diagram of resources that will show these options.

Each resource represents an additional benefit to the mission of the agent, whose goal is to maximize the sum of benefits along the route, while respecting the precedence relations. However, the mobile agent is supposed myopic [4], i.e., it knows only the immediate options of the diagram of resources. The myopia of the agent prevents the problem to be treated by classical optimization techniques.

Another requirement that must be considered by the mobile agent is the deadline of each mission. The sequence of nodes visited by the mobile agent from the source node to destination node (itinerary) is defined dynamically.

3.1. Problem Formulation

Each intelligent mobile agent has a mission $\boldsymbol{M}$ that defines the benefit function $\boldsymbol{B}$ for each type of resource $\boldsymbol{R}$. This function determines the amount of benefit $\boldsymbol{b_i}$ that each type of resource $\boldsymbol{r_i}$ contributes to the mission of the agent. The benefit $\boldsymbol{b_i}$ indicates the degree of importance of that resource for the mission.

The resource diagram of a mission is an activity diagram of UML (*Unified Modelling Language*) that describes the precedence relations between resources. Any resource in disagreement with this diagram does not bring any benefit to the mission. The diagram of nodes is based on the diagram of resources. Each resource is replaced by the node or nodes where it appears in the system.

The fulfilment of a mission $\boldsymbol{M}$ is related to the choice of an itinerary $\boldsymbol{I}$. For defining the itinerary it is considered: the time of computation for maximum benefit of each task, the communication latency between nodes, the dependencies between the resources described in the mission's resource diagram. Costs should be analyzed so that the agent arrives at a node and acquires that resource ($\boldsymbol{C_i} + \boldsymbol{Q_i}$ where: $\boldsymbol{C_i}$ is the execution time in the node and $\boldsymbol{Q_i}$ is the time in the queue waiting for the local processor).

For every resource $\boldsymbol{r_i} \in \boldsymbol{R}$, the benefit $\boldsymbol{b_{i,t}}$ obtained by using resource $\boldsymbol{r_i}$ during the time interval $\boldsymbol{t}$ is given by:

1) $\boldsymbol{r_i}$ of type *variable*

$$b_{i,t} = b_i * min(1, t/Ci) \qquad (1)$$

where $\boldsymbol{b_i}$ is the maximum benefit obtained from $\boldsymbol{r_i}$ and $\boldsymbol{C_i}$ is the maximum computation time associated with $\boldsymbol{r_i}$ ($\boldsymbol{C_i}$ is the time that the agent should have the resource to receive $\boldsymbol{b_i}$ ant $\boldsymbol{t}$ is the time it actually had the resource).

2) $\boldsymbol{r_i}$ of type *normal*

$$b_{i,t} = b_i \quad \text{when} \quad t \bullet C_i \qquad (2)$$
$$b_{i,t} = 0 \quad \text{when} \quad t < C_i$$

The effective benefit $\boldsymbol{B}$ of the mission is obtained by the sum of the benefits achieved from each resource along the itinerary (R'):

$$B = \sum b_i(y) \quad \text{for every} \quad r_i \quad R'. \qquad (3)$$

We define $\boldsymbol{b_i(y)}$ the benefit achieved by the use of resource $\boldsymbol{r_i}$ that the agent used for the fulfilment of its mission while following itinerary $\boldsymbol{y}$. $\boldsymbol{B}$ is the benefit generated by the use of all resources of the mission. The purpose of the itinerary definition algorithm is to maximize the value of $\boldsymbol{B}$, while meeting deadline $\boldsymbol{D}$ of the mission.

4. Description of the Adaptive Heuristics

This paper presents new heuristics to dynamically determine the itinerary of myopic mobile agents with time constraints under the computational model described in Section 3.

Heuristics' performances are compared with each other using simulation data detailed at Section 5.

The heuristics discussed in this section consider a history of benefits (log) achieved in past executions of

each mobile agent. The agent queries the existing log in order to choose its behaviour to start the mission. The log stores information relating to previous missions, such as the response time, the benefit gained and heuristics used.

The agent will adapt dynamically. Once selected the behaviour for the mission, it may take different behaviours to the end of the mission. In order to make the choice of behaviour it is used conditional probability based on the characteristics of the environment (distributed system) and the past events (history).

Heuristic LG (Lazy-Greedy) is described and evaluated in [5]. The originally chosen behaviour will remain the sole agent behaviour to the end of the mission. The method of defining the itinerary proposed in this paper may be considered an evolution of the versions with adaptation at departure.

The definition of adaptive heuristics uses some of the simple heuristics that are described in detail in [4] and evaluated in [5]. We briefly present the decision making of each simple heuristic, showing the necessary information for understanding the description of adaptive heuristics:

LAZY: This algorithm attempts to execute the resources as quickly as possible, ignoring the maximum benefit of each resource. The optional resources are not used and the resources of type <<variable>> run for the shortest time possible, executing only the mandatory part of each resource. The <<variable>> type indicates a resource that can be partially acquired, i.e., the mobile agent does not have to complete the whole execution time for full benefit.

HIGHER-DENSITY: This algorithm selects as the next resource to be executed that with the best cost/benefit relation. Optional resources will be executed only if their density (ratio between the benefit gained and its cost) is higher than the average density of the mission up to that moment.

GREEDY: When there is an alternative route, this agent will choose that with the greatest benefit to the mission, without any concern for the execution time (deadline). Optional resources will always be executed and resources of type <<variable>> are executed during the computing time necessary to achieve the greatest possible benefit.

4.1. Adaptation at Departure

LAZY-GREEDY (LG): This heuristic is based on the use of two simple heuristics: Lazy and Greedy. It uses a log formed only by the response times of previous executions. For the formation of this log, the deadlines and the information from previous executions about the deadlines being met or not is irrelevant.

In its first execution, the mobile agent always chooses algorithm Greedy - this mission is then saved in the log. When a new mission arrives, its deadline is compared with the response times of previous missions, and value $P(R \leq D)$ is estimated, that is, the probability of the current mission to meet its deadline. This estimate is compared to a threshold and the outcome of this comparison defines the behaviour that the agent will display. When the probability of success is below the threshold, the agent becomes Lazy. But if the probability of success is above the threshold, the behaviour Greedy is adopted. The value of $P(R \leq D)$ is calculated simply by counting the number of missions in the log that would have met the current deadline, divided by the total number of missions in the log. This calculation disregards which heuristic was used by missions in the past.

The value chosen as threshold for the exchange of algorithms between Greedy and Lazy is an arbitrary constant. The quality of the results is directly related to the value of this constant. Despite this disadvantage, the computational complexity of this algorithm is $O(n)$, where n is the size of the log. Moreover, this algorithm shows robustness with respect to the choice of the algorithm by the agent, even with a few executions and a log nearly empty. If a wrong choice is made of algorithm Lazy, the average response time of the log decreases because of the behaviour Lazy. The probability of choosing Greedy increases in its next run. A similar sequence occurs when there is a wrong choice of algorithm Greedy. Figure 1 shows a sketch of the heuristic algorithm LG.

```
Const threshold = 0,9;

Scan_log_to_compute P( R  ≤ D )

if P(R ≤ D ) > threshold
    behaviour = greedy;
else
       behaviour = lazy;
```

Figure 1. Pseudocode for Heuristic LG.

4.2. Adaptation along the Itinerary (Dynamic)

LAZY-ADAPTIVE: This algorithm initially is based on the behaviour of the heuristic Lazy, but with the ability to improve the decision making for the next node using its log. The main objective of the Lazy-Adaptive heuristic is to achieve the greatest possible benefit for a mission without missing the deadline. For this reason it is based on the Lazy Algorithm: it is the fastest heuristic and focuses on meeting the deadline.

The Lazy-Adaptive heuristic builds a log formed by the deadlines met according to the behaviour used in each previous mission. It finds a way to increase the gain of benefits: the heuristic performs a little more aggressively in the next mission with similar deadline, trying to improve on the relation deadline/benefit gradually. That is, the mobile agent will try to get more benefits with some resources of the mission.

Initially, the agent always chooses behaviour Lazy. In future executions, the agent uses the log to improve its

choices; it can analyze the possibility of changing its behaviour, during the mission, and to get bigger benefits. That is possible only because the mobile agent calculates the gap between the response time and the mission deadline.

After reaching this node, it will not use the behaviour Lazy anymore, but a more aggressive one in order to get more benefit at that point (as the Higher-Density and the Greedy). This change of behaviour is defined at the beginning of the mission, depending on the deadline, but only during the route will it be possible to confirm if it should use the more aggressive behaviour. This increases the computational complexity of this algorithm in comparison to the Lazy-Greedy heuristic, but it also has a small footprint: its computational complexity is $O(m \bullet log\ n)$, where n is the size of the log and m is the maximum number of redundant services of any kind (nodes of the same type). Figure 2 describes the algorithm of heuristic Lazy-Adaptive.

The method *ArrivalCalcNextStage* always checks whether a new mission has started to choose the behaviour that will be adopted by the agent to complete its mission: Lazy behaviour, a behaviour in the log or a more aggressive behaviour (Higher-Density, for example). It does not mean the procedure relies on adaptation at the departure. It only means that the agent knows at the moment a mission starts if it will have enough time to try new behaviours within the mission or it will get one already used and listed in the log.

After establishing the initial behaviour of the mission, at each new stage of the itinerary, the method calculates the next stage (*calcNextStage*) and also checks whether to mark a stage of the itinerary for a more aggressive behaviour in the future (*calcFutureStage*).

Then there is a new check to see if the mission ended. In that case, it updates the log with the new behaviour and deadline. It also marks a new behaviour more aggressive, aiming at achieving more benefits in future missions, by using the behaviour of the current mission, together with the stage marked as *futureStage*, the stage which is the best for the search of more benefits.

The log array holds all behaviours that were considered aggressive in previous trips. The agent has to consume/execute a sequence of resources and hence it has groups of nodes to visit. Each resource can be found in one or more nodes, so every step of the trip the agent needs to decide which node to visit (among those that contains the current resource to be executed in the mission). During the trip (including the first trip, using the Lazy behaviour), the agent marks a node to be more aggressive in an upcoming trip, whenever there is time left for improvement at the end of the current trip. This node is marked while choosing the next node to be visited for each kind of resource. Besides choosing the node for this trip, it also marks a second node of higher benefit value. This node is not the highest benefit value for that kind of resource, but a node with benefit value immediately higher the node selected for this trip, considering an ascending scale of benefit values.

In other words, the Lazy Adaptive heuristic combines aspects of the lazy heuristic, which considers only the execution time, with aspects of the Higher-Density heuristic, which considers a balanced average of benefits and execution time. After the first trip using the lazy behaviour, the heuristic can always take a more aggressive behaviour, increasing the accumulated benefit of the mission, and respecting its deadline.

5. Evaluation

Heuristics Lazy, Greedy, Higher-Density, LG and Lazy-Adaptive are evaluated by simulations.

The performance of Lazy-Adaptive will be compared to other heuristics in order to confirm its performance with tight deadlines. The simulator used was developed in Java by the research group.

4.3. Simulation Conditions

It is considered a system composed of 10 nodes and 15 resources. The resources are statically allocated to the nodes, and they are replicated in some nodes. We constructed the nodes diagrams from the definition of 10 mission segments (Diagram of Resources).

Each segment contains precedence relations and some resources. For each segment the corresponding Nodes Diagram was built. The mission of the agent is defined by the composition of 6 segments drawn with replacement from this set of 10 diagrams. This generates 1,000,000 (10^6) different resource diagrams.

For the simulation of the total time of the mission it was considered: the length of the processor queue (the value varies from processor to processor with uniform distribution from 0 to 20), time of use of the resource, a fixed value (between resources there is a uniform distribution between 1 and 15), time in queue for communication links. The average link delay varies between 2 and 5 with uniform distribution.

After the use of each resource, some benefit is received and added to the total benefit of the mission. The objective of the mission is to achieve the greatest benefit within the specified deadline. The benefits gained by the use of each resource is fixed, with $B_0 = 0$, $B_1=1$, $B_2=2$, $B_3=3$, $B_4=4$, $B_5=5$, $B_6=6$, $B_7=7$, $B_8=8$, $B_9=9$, $B_{10}=10$, $B_{11}=11$, $B_{12}=12$, $B_{13}=13$, $B_{14}=14$, $B_{15}=15$.

The quality of the algorithm (global benefit of the algorithm G) is calculated using the equation:

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

Where *B_DA* is the benefit obtained by the use of resources while meeting the deadline and NT is the number of attempts.

```
{Agent}
Variables shared between agents:
 1: initialStage = 0;          { initial node of the mission }
 2: untestedBehaviour = lazy; { aggressive behaviour not tested }

Variables local to each agent:
 3: behaviour = null;                { standard behaviour of the agent }
 4: nextStage = null;                { next node }
 5: futureStage = null;              {chosen node for future aggressive behaviour}

procedure on ArrivalCalcNextStage ( currentStage, stageGraph, missionDeadline, log)
 6: if isNewMission(currentStage, initialStage) then
 7:     if isDeadlineReachable(missionDeadline, log) then
 8:             behaviour <- getHistoricalBehaviour(log)
 9:     else
10:             behaviour <- getUntestedMoreAgressiveBehaviour(untestedBehaviour)
11:     end if
12: end if

13: nextStage <- calcNextStage(behaviour, stageGraph)
14: futureStage <- calcFutureStage(nextStage, stageGraph)

15: if isMissionFinished(nextStage, initialStage) then
16:     updateLog(behaviour, log);
17:   untestedBehaviour <- createMoreAgressiveUntestedBehaviour
18: end if
```

Figure 2. Pseudocode for Heuristic Lazy-Adaptive

4.4. Simulation Results

We simulated 100 different diagrams of nodes, each with 14 different deadlines. A total of 100 agents have been launched, with 100 different configurations and 14 different deadlines, which resulted in 1400 executions of each configuration. This represents the execution of 140,000 missions using each of the heuristics.

Table 1 shows the values of the overall benefit achieved by algorithms using the 100 different configurations and 14 different deadlines tested. Values in bold indicate the heuristics that reached the highest levels of performance for each value of deadline.

Figure 3 is based on data described in Table 1 and it shows a graph of the Global Benefit G obtained by each algorithm. Analyzing the graph of Figure 3, it can be noticed that the heuristic Lazy-Adaptive reached the highest levels of benefit in situations where the deadline of the mission is considered tight, thus fulfilling its original purpose. It is also that this heuristic forms the upper envelope of the graph with loose deadlines.

From the moment when the deadlines of all agents are met, the overall benefit obtained by the algorithm is stable, independent of the remaining time before the deadline and the time spent on the mission.

Completing the analysis of the graph of Figure 3, it can be seen clearly the evolution of the heuristics regarding their performance. The simple heuristics Lazy, Greedy and Higher-Density, present good results only for specific deadline ranges, respectively tight, fair and loose deadlines.

The heuristic LG showed a significant improvement on its performance, showing good results for more than a range of deadlines. This advantage is a result of its adaptation at departure. It is important to remember that once the behaviour of the agent is chosen for the current mission it will remain the same until the end of this mission.

The heuristic Lazy-Adaptive supports dynamic adaptation. It is capable of changing its behaviour at runtime, so it is possible that the mobile agent will assume different behaviours during the same mission. Therefore, this heuristic has a curve of evolution that is faster than heuristic LG for tight deadlines. Another interesting situation to be discussed is the issue of response time. Figure 4 shows the graph with the mean response time of the heuristics. Looking at the chart, one can note that the simple heuristics have the same response time for all deadline ranges.

Algorithm Greedy has a higher consumption of time, which proves that the heuristic needs bigger deadlines to get more relevant benefits. The other simple heuristics alternately appear within a time interval with lower values.

In the graph of Figure 3 it is possible to confirm another advantage of the use of adaptive heuristics. In ranges of tight deadlines, these heuristics consume less time. When noticing a change in the range of deadline, the heuristics use their adaptive capability to take the slack in the deadline and, seeking higher rates of benefit, to increase the consumption of time to complete the mission. Table 2 presents these values. Heuristic Lazy-Adaptive consumes the time it is allowed, providing good results from very tight deadlines to loose deadlines.

Based on the data showed in tables and graphs presented, it was noted that the algorithm Lazy-Adaptive presented a very efficient performance for tight deadlines (desirable characteristic for most real-time

applications), showing superior performance when compared with static algorithms (Lazy, Greedy and Higher-Density) in these scenarios.

Table 1. Global Benefit.

	150	200	250	300	350	400	450	500	550	600	650	700	750	800
Lazy	0	0	0	94,55	148,8	155	155	155	155	155	155	155	155	155
Greedy	0	0	0	0	17,28	103,68	**167,04**	**186,24**	**192**	**192**	**192**	**192**	**192**	**192**
Higher-Density	0	0	9,3	86,8	148,8	155	155	155	155	155	155	155	155	155
Lazy-Adaptive	0	0	**12,4**	**114,7**	144	135,32	153,61	170,33	**192**	**192**	**192**	**192**	**192**	**192**
LG	0	0	4,65	89,16	**151,4**	**160,12**	162,88	182,44	**192**	**192**	**192**	**192**	**192**	**192**

Table 2. Average Response Time.

	150	200	250	300	350	400	450	500	550	600	650	700	750	800
Lazy	288,4	289,2	304,7	295,6	286,5	288,8	291,2	293,9	294,2	295,7	294,1	292,7	290,1	290,1
Greedy	410,0	405,5	398,2	403,3	414,6	396,8	404,4	408,9	402,6	407,3	396,9	402,5	411,3	404,0
Higher-Density	294,1	297	291,5	296,5	296,5	292,4	293,5	297,2	289,5	290,0	297,7	297,7	295,5	288,5
Lazy-Adaptive	295,0	298,2	288,9	288,7	309,6	352,8	394,4	410,4	440,9	446,7	442,5	440,1	436,4	448,2
LG	295,5	298,2	302,0	298,2	312,3	325,7	343,4	404,3	413,6	409,1	401,4	400,9	403,1	406,2

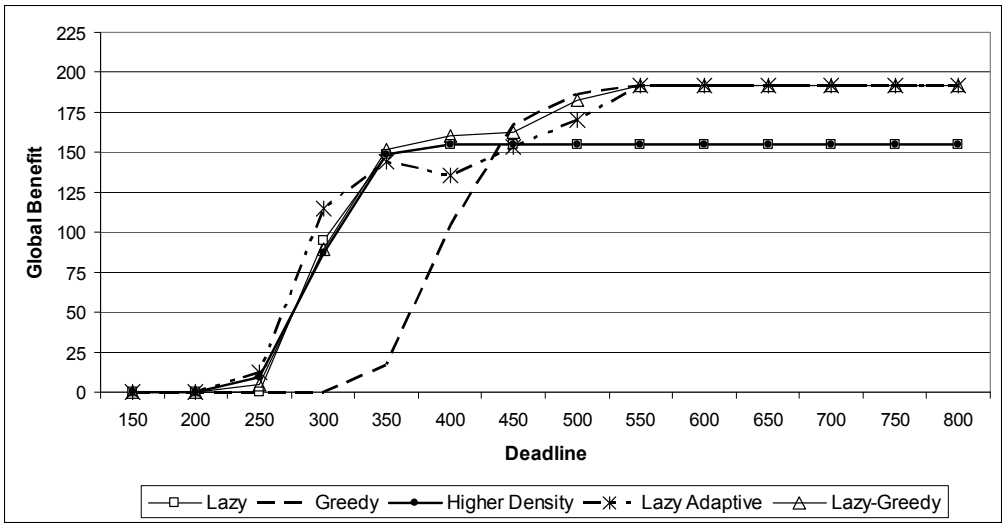

Figure 3. Global Benefit.

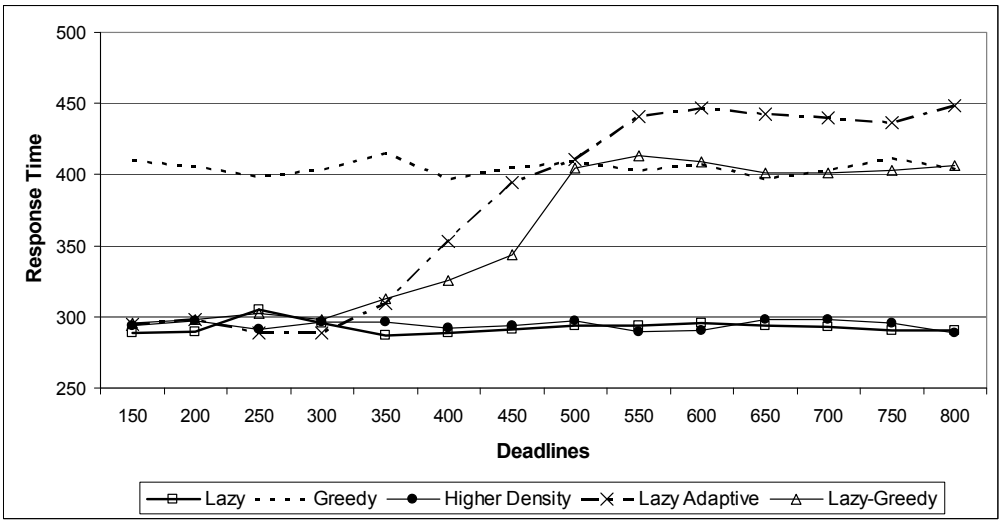

Figure 4. Average Response Time of the Heuristics

Agents using heuristic Lazy-Adaptive follow the line of continuous learning, considering the size of the log (the last 50 executions). The algorithm drops the oldest record of execution in order to save new results, causing the oscillation when faced with new ranges of looser deadlines.

When close to the extremely loose deadlines, algorithm Greedy shows its potential. Heuristic Lazy-Adaptive also managed to increase its performance quickly and obtained similar maximum results.

Heuristic Lazy-Adaptive presented a good result for tight deadlines, achieving the best results in the range. When facing intermediate deadlines it demonstrated its capacity for learning – although not reaching the biggest indices, its result was satisfactory. Finally, with loose deadlines it was able to achieve maximum benefits.

5. Conclusions

In the context of distributed applications based on mobile agents, it is possible to find mobile agents that must meet a deadline with a degree of flexibility in defining the itinerary. This paper proposed and evaluated a new heuristic for definition of itineraries. A detailed comparative analysis has confirmed the good performance of this more elaborated heuristic. The adaptation during the route brought more strength to the algorithm. This heuristic uses a log of previous executions to change the behaviour suggested at the departure of the mission, conforming itself to changes in the environment.

All the heuristics discussed here offer a low computational cost, which is necessary because some nodes to be visited by mobile agents may have low processing capacity.

The behaviour of the proposed heuristics has been examined by simulations. The results were compared for the purpose of electing the heuristic that best adapt to different situations in the distributed system. All heuristics satisfy the assumption of the myopia of mobile agents.

The fact that simple heuristics present a defined behaviour for each deadline range made it possible the suggestion of a more appropriate behaviour for each new mission, as the deadline for this new mission became known. So, for each new mission, taking into account only the current deadline of the task and the experience gained in previous missions, it became possible to choose the most appropriate behaviour for that situation. The results presented after the simulation of adaptive approaches showed very satisfactory performance, so that to achieve the highest levels of benefit within the conditions of each mission. The possibility of changing their behaviour at run time together with the flexibility allowed by the characteristics of some of the resources that make the mission provided good results.

The purpose of the use of adaptive approaches is to create a heuristic able to adjust to the situation of the system and the needs imposed by the mission to be executed. Therefore, they do not have the goal to overcome the original heuristics in all deadline ranges, but to maintain good results (close to the best result obtained by each heuristic on each range of deadline) for all ranges of deadline. This objective was achieved with the heuristic Lazy-Adaptive.

6. References

[1] Lange, D. and Oshima, M. Seven Good Reasons for Mobile Agents. Communication of the ACM. Vol.42. pages 88-89. 1999.

[2] Russell, Stuart J.; Norvig, Peter. “Artificial Intelligence: A Modern Approach”. 2nd ed., Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-790395-2. 2003.

[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] Rech, L., Oliveira, R. and Montez, C. “Dynamic Determination of the Itinerary of Mobile Agents with Timing Constraints”. IAT 2005 IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology. Compiègne. França. pages 45-50. 2005.

[5] Rech, L., de Oliveira, R. and C. Montez, et. al. “Determination of the Itinerary of Imprecise Mobile Agents using an Adaptive Approach”, ETFA 2008 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Germany. 2008.

[6] Qu, W., Shen, H., and Jin, Y. “Theoretical Analysis on a Traffic-Based Routing Algorithm of Mobile Agents”. IAT 2005 IEEE/WIC/ACM. Compiègne. France. pag 520-526. 2005.

[7] Jin-Wook Baek; Jae-Heung Yeo; Heon-Young Yeom, "Agent chaining: an approach to dynamic mobile agent planning,". in the proceedings of the 22nd International Conference on Distributed Computing Systems, 2002, pp. 579-586, 2002

[8] Ilarri, S., Mena, E., Illarramendi, A., “Using cooperative mobile agents to monitor distributed and dynamic environments”, Information Sciences: an International Journal, Volume 178, Issue 9. 2008.

[9] Koenig, S. and Likhachev, M. “Real-time adaptive A*”, Int. Conf. on Autonomous Agents. Japan., pages 281-288. 2006.

[10] Cicirello V., Peysakhov M., Anderson G., Naik G., Tsang K., Regli W., Kam M., "Designing dependable agent systems for mobile wireless networks", IEEE Intelligent Systems, , September/October, 2004.