Dynamic Determination of the Itinerary of Mobile Agents with Timing Constraints

Luciana Rech
LCMI-DAS-UFSC
lrech@das.ufsc.br

Romulo Silva de Oliveira
LCMI-DAS-UFSC
romulo@das.ufsc.br

Carlos Montez
LCMI-DAS-UFSC
montez@das.ufsc.br

Abstract

The mobile agent technology is an important research area in the scope of code mobility. As part of an agent mission objective, it may exist the necessity of meeting an end-to-end deadline, although with some flexibility about the itinerary the agent should follow. The objective of this paper is to define an execution model for mobile agents that include timing constraints. We also show some simple heuristics that are able to guide these agents in the itinerary definition, considering the existence of an end-to-end deadline for the mission.

1. Introduction

Within the context of code mobility, an important research area is the mobile agents technology [1, 2]. A mobile agent is an element of self-contained software that is responsible for the execution of a task, and that is not limited to the system where it starts execution. It is able to migrate in an autonomous way through the network. A mobile agent must be able to interrupt its execution in a place, to move to another one, and to retake its execution there. Mobile agents reduce the load in the network. They execute in a way both assynchronous and autonomous, and they can dynamically adapt, thus establishing a new paradigm for the programming in distributed environments. An important requirement that can be associated to mobile agents is the timing aspect.

A possible scenario that illustrates the use of mobile agents with time constraints is a system of generation and distribution of electrical power where, by following a hierarchic structure, the data is widely spread physically. In the context of the electrical power industry, a wide geographic scale and response in real- time are only two requirements for the supervision and control system, which is also characterized by the heterogeneity of the equipment. The scale of this type of system can be of thousand of nodes [3].

Considering a power generating plant and the substations that take part in the electrical power transmission, the mobile agent can substitute the technician (or engineer), with his laptop, visiting section by section of the system. The agent has as function: to collect data, to form an image of the problem and to decide the next visit for collection of data. The diagnosis of a fault is achieved by visiting some nodes and by collecting data for the decision taking.

This paper deals with code mobility in a distributed system associated with timing constraints. Its objective is to define an execution model for applications based on imprecise mobile agents with real time constraints. Also, this paper presents some basic algorithms for dynamic determination of itinerary. An imprecise mobile agent is able to negotiate the quality of the result in order to meet the mission deadline.

This text is divided in 6 sections. In section 2 we review some previous work found in the literature related to this subject. In section 3 the adopted execution model is described. In section 4 it is introduced some simple heuristics for the definition of the itinerary. Section 5 presents a comparison of this heuristics and section 6 has the conclusions.

2. Related Work

There are some previous work in the literature about the agents technology and time constraints. In [4] is described a routing system based on agents (ARS - Agent-based Routing System) that allocates network resources for tasks that requested point-to-point real-time communication. Mobile agents are supported by a resource management mechanism, they collect current link status of the system to examine the availability of possible routes to each node.

Another work that associates timing constrains to mobile agents is [5], where the decision taking is implemented through negotiation between intelligent and independent mobile agents. A negotiation protocol based on mobile agents (MANPro) is applied for real-time scheduling of a production line system.

The RT Monitor is a real-time data management system that deals with traffic data for applications in network navigation [6]. Navigation 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. To minimize the overhead, a distributed cooperative method is adopted using mobile agents, aiming at to improve system scalability and to reduce communication traffic.

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

Among the above works, only in [2] there is the concern of choosing an itinerary where timing constraints are respected. However, in the execution model considered in this paper, there is the concern of choosing the best itinerary considering the performance of the task execution and the difficulty in meeting the deadline. In this context, the definition of "best" itinerary may vary, in accordance with the metric used to evaluate itineraries. Adaptive scheduling techniques will be used to establish a compromise between the visits to nodes and the timing constraints of the mission.

The imprecise computation [7], for example, is a technique able to establish a solid conceptual base for the decisions that will be made by the agent and/or the support system, with respect to its scheduling. The itinerary of the mobile agent must be constructed so as all the mandatory resources are executed. Moreover, it should execute optional resources so as to maximize the value added to the mission of the agent, without missing the end-to-end deadline.

The problem considered in this work is also similar to that presented in [8], entitled "The prize collecting traveling salesman problem". However, algorithms as presented in [8, 9] are computationally expensive and directed toward the static determination of the itinerary, when there are no real-time constraints.

3. Execution Model

The proposed execution model describes the behavior of an application based on mobile agents, with timing constraints, in a distributed system, formed by a set of nodes connected through a communication underlying service. In this execution model, each mobile agent has a mission. The mission is described in terms of the visit to a determined set of nodes, which includes the necessary resources for the agent. It is assumed that this agent does not communicate with other agents. The agent mission is formed by a task set that will be executed respecting a single deadline. A task represents the use of a resource by the agent in a node.

A resource is an abstraction and can correspond to a processor, device, file, data structure, etc. Each resource has an associated benefit that is derived from its functional importance for the mission. In each system node there is a set of resources. There can be many instances of the same type of resource in distinct nodes. The processor is always a necessary resource for the agent and it is presented in every node. So, it will be treated implicitly.

The itinerary is the route defined by a sequence of nodes. Each agent in our model has some flexibility for defining its itinerary. This flexibility comes from the resources that can: i) to be used in any order, ii) to appear duplicated, iii) to be discarded or iv) to be partially used (with respect to its utilization time).

3.1. Problem Formulation

A computational system is characterized by a set N of nodes and a set R of resources types. The set N of nodes of the system has cardinality m, that is, it is composed by m nodes N_i, $N_i \in N$, where $1 \leq i \leq m$.

In each node N_i of the system there is a set R_i of one or more types of resources of the set R. Each type of resource of set R can be found in one or more nodes, that is, there can be many instances (duplicates) of a same type of resource, as long as they are in distinct nodes. The Figure 1 illustrates a situation where it is possible to observe that instances of the resource type r_3 appear in node N_2 and in node N_6.

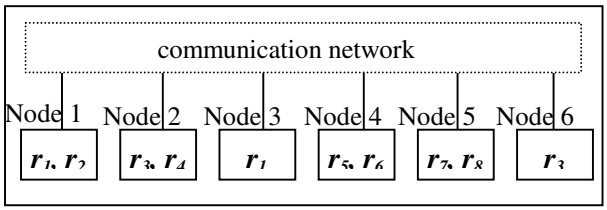

Figure 1. Resources in nodes of a distributed system

An itinerary I is defined as a route in a computational system, it is described as a node sequence N_i pertaining to the set N. An itinerary does not need to include all the nodes of the system, and it may include in its sequence the same node more that one time.

Each mobile agent Z has a mission M that defines a benefit function B for each type of resource. This function determines how much of benefit B_i each type of resource r_i contributes to the agent individual mission.

The agent has as objective to add benefits through the use of resources spread among nodes interconnected by a communication network. The objective function of the agent is to maximize the collected total benefit throughout its itinerary in the system. The agent starts at system external node N_0 (origin node) and must come back to it at the end of the mission. The node N_0 does not has instances of the types of resources that form the set R. It is

assumed clock synchronization in the system. Every mission ***M*** has a firm deadline ***D*** associated.

The agent ***Z*** must conclude the mission (and to return to node N_0) before ***D***, or it will lose all the benefit collected along its itinerary.

The resource diagram of a mission ***M*** (Figure 2) corresponds to an UML activity diagram that describes the existing precedence relations about the use of resources. Any resource used in disagreement with this diagram adds no benefit to the mission.

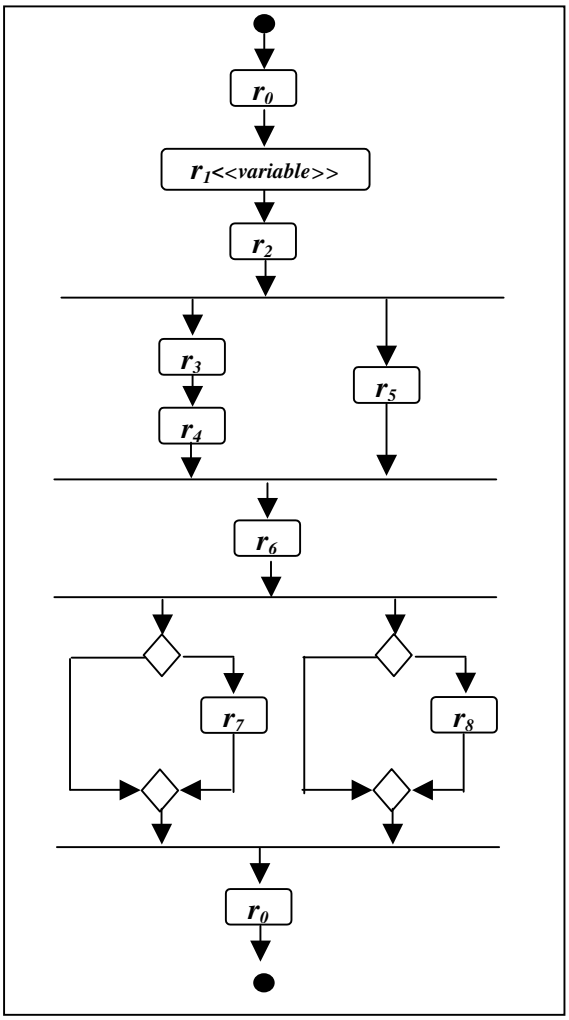

Figure 2. Resource diagram of a mission ***M***

The resource diagram of the mission describes the flexibility and the agent options when carrying through the mission. Figure 2 describes a mission where eight resources are used. Resource type r_0 is a pseudo-resource, only found in node N_0, it is used to indicate the agent must return to the original node. In the diagram, the stereotype <<variable>> indicates that the time of usage of resource r_1 is variable, it may be released before the agent gets its maximum benefit.

The flexibility in composing the itinerary is associated with the resources of set ***R***, since some of them may be used without a previously defined order (for instance: r_5 and r_3 in Figure 2), may appear duplicated (for instance: r_1 in N_1 and N_3, in Figure 1), may be discarded (for instance: r_7 in Figure 2) or may be used partially (for instance: r_1, in Figure 2).

The node diagram of a mission ***M*** corresponds to an UML activity diagram constructed from the resource diagram of mission ***M***, where each resource is substituted by the node or nodes where it appears in the system. This diagram presents the set of all possible itineraries for the agent. For example, assuming that the mission described in Figure 2 was executed in the system described in Figure 1, Figure 3 contains the resultant node diagram of this mission, and it represents the possibility of different itineraries that the agent can follow to conclude its mission.

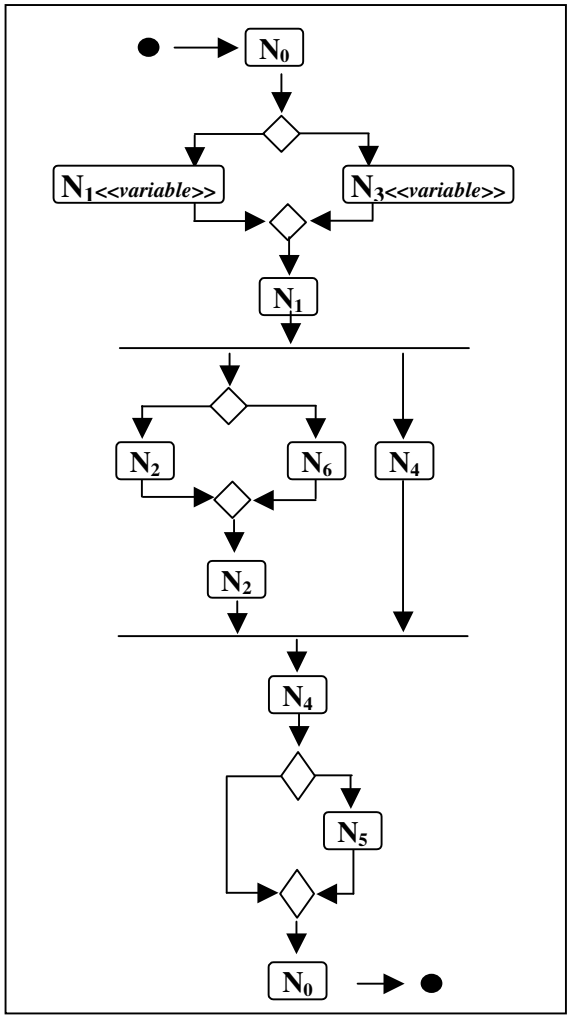

Figure 3. Node diagram of a mission ***M***

The execution of mission M can be seen as an aperiodic activity A. An activity is defined as the task set X and network latencies L that compose an itinerary I in particular. A task x_i represents the use of a resource r_i by an agent Z in a node N_i. Each task x_i belongs to set X. The activations of these tasks (x_1, x_2, ..., x_n. where n = cardinality of set X) occur in irregular intervals of time, characterizing the aperiodical aspect of the activity and the tasks. The deadline D of the activity A is associated with the conclusion of all the tasks that belong to activity A.

The conclusion of a mission M depends on the choice of an itinerary I. The definition of the itinerary should take into account: i) the time of computation for maximum benefit of each task x_i, called C_i; ii) the communication latency between nodes N_k and Ny, called Ly; iii) the dependences between the resources that appear in the resource diagram of the mission.

It should be considered the cost for an agent to arrive at a node (coming from the previous node) and to acquire the resource ($K_i + Q_i$ where: K_i is the execution time in the node and Q_i is the waiting time at the local processor).

For all $r_i \in R$, the benefit $B_{i,t}$ collected for using the resource r_i during the interval of time t is showed by:

1) r_i at the variable type

$$B_{i,t} = B_i \; x \; min\,(1, t/C_i) \qquad [1]$$

where B_i is the collected maximum benefit of r_i and C_i is the maximum computation time associated with r_i (C_i is the time interval that the agent must hold the resource to gain B_i and t is the time interval that the agent actually holded it).

2) r_i the normal type

$$B_{i,t} = B_i \quad \text{se} \quad t \geq C_i \qquad [2]$$
$$B_{i,t} = 0 \quad \text{se} \quad t < C_i$$

For the equations above we must consider optionality and precedence relations:

- if r_i precedes r_j, and r_i was not used, then $B_j=0$;

- if r_i' and r_i'' are duplicates of r_i, and r_i' was already used, then the maximum benefit of r_i" becomes zero.

The effective benefit of the mission is defined as the sum of the benefits collected from each resource along the itinerary:

$$B = \sum B_i(y) \quad \text{for every} \quad r_i \in R. \qquad [3]$$

where $Bi(y)$ is the benefit collected by using resource r_i that the agent visited when following the itinerary y, B is the benefit generated for the use of all the resources of the mission.

4. Algorithms for definition of the itinerary

This section presents some simple heuristics, that can be used with the execution model presented in this work. These simple heuristics make their decisions based only on the data associated with the possible immediate nodes to visit, it does not consider the future unfolding of their short term decisions. When the agent needs to decide between two routes that, in the perspective of the used heuristic, are equivalent, it takes a random decision.

4.1. Lazy Algorithm

The lazy algorithm tries to execute all resources as quickly as possible. It ignore the benefit of each resource. The optional resources are not executed, and resources with stereotype <<variable>> are executed during the minimum possible time. When there are alternative routes, it will choose the one that has the minimum execution time, that is, when there is no precedence relation between two resources, it will be chosen the resource that presents the smallest computation time. Thus, the collected benefit will be minimum, even zero sometimes. In the case of the existence of duplicates of a resource, initially it is verified if there is an instance of this resource in the node where the agent is at the moment (the node where it used the previous resource). In negative case, it will choose the closest node to visit.

4.2. Greedy Algorithm

Every time that there is an alternative, the greedy algorithm will choose the one that generates the greatest benefit. For example, when there is no precedence relation between two resources, it will choose the resource that presents the greatest benefit for the mission. This algorithm always executes optional nodes. In cases where the resource is of type <<variable>>, the greedy algorithm executes the requested time to get the maximum benefit. Like the previous algorithm, in case of the existence of duplicated resources, the preference is for the resource situated in the nearest node.

4.3. Random Algorithm

This algorithm randomly chooses the next node to be visited by the agent. This behavior is only possible if there are duplicates of one resource or there is no precedence relation between the next resource and some of the others resources to be used. When the resource is of type <<variable>>, the random algorithm will decide randomly between zero and the computation time necessary to collect the maximum benefit from this resource.

4.4. Higher Density Algorithm

The higher density algorithm - based on policy AVDT (Average Density Threshold) [10] - chooses as the next resource to be executed the one that presents the best relation benefit/time. This is only possible when there is no precedence relation between two resources. The density of the resource benefit is defined by the equation:

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

where: dbi = density of the benefit of resource i, B_i = benefit collected from the execution of resource i, C_i = time of computation used for resource i, $L_{j,i}$ = time necessary to move to node N_i, and Q_i = waiting time at the local processor. Estimated values can be used when the actual ones are not known.

The agent keeps the average value db observed until the current moment. When a resource is optional, it will only be executed if its density is bigger than the mean density of the mission until that moment.

4.5. Versions with Clock

A possible variation of the above algorithms is to estimate the time necessary to execute the remaining mandatory resources of the mission, before deciding the next resource to be executed. The existing slack is calculated by the difference between the deadline and the execution time of the mandatory resources, added to the time necessary to move the agent between nodes. Optional resources will only be executed if the slack is big enough. This estimate can be applied to the algorithms already described, with the exception of the lazy algorithm, that is not affected by it.

5. Comparison of the Heuristics

Table 1 presents a comparison of the performance of the described algorithms using the system example of section 3. For the calculation of the mission total computation time, the computation time of each resource has been considered and a fixed time of communication L between the nodes ($L=1$).

In the presented models the diagrams of Figures 2 and 3 will be considered, along with the computation times (t_1=10, t_2=2, t_3=7, t_4=2, t_5=8, t_6=5, t_7=7, t_8=3) and the benefits (B_1=10, B_2=3, B_3=3, B_4=7, B_5=5, B_6=3, B_7=6, B_8=8). In this study we assume that the communication and computation times are previously known by the agent. The variable resources have minimum execution time equal to zero.

In this example, the total benefit of the mission when using the greedy algorithm with deadline 50 was 0, since that is a firm deadline and total computation time was 51. Observing the table, we notice that the greedy algorithm with clock does not execute the optional resource r_8 (respecting the mission deadline), with the purpose of keeping the total benefit collected so far.

When the random algorithm is used (and observing the diagram of Figure 3), a possible itinerary is: N_3, N_1, N_4, N_2, N_4, N_5. As the criterion of decision used to choose the next resource is random, there are many possible itineraries.

Observing the diagram of Figure 3 and the resultant itinerary (Table 1), it was estimated the relation between the collected benefit and the time expended to get this benefit for each resource (E_1=10/10, E_2=3/2, E_3=3/7, E_4=7/2, E_5=5/8, E_6=3/5, E_7=6/7, E_8=8/3).

According to table 1: Alg1 represents Lazy Algorithm; Alg2, Greedy Algorithm; Alg3, Random Algorithm; Alg4, Higher Density Algorithm; Alg5, Greedy with clock; Alg6, Random with clock; and Alg7 Higher Density with clock.

Table 1. Comparison of the itinerary definition algorithms

	Itinerary	D=20	D=30	D=40	D=50	D=60
Alg 1	$N_1, N_6, N_2, N_4 N_4$	T=29 B=0 (21)	T=29 B=21	T=29 B=21	T=29 B=21	T=29 B=21
Alg 2	$N_1, N_1, N_4, N_6, N_2, N_4, N_5, N_5$	T=51 B=0 (45)	T=51 B=0 (45)	T=51 B=0 (45)	T=51 B=0 (45)	T=51 B=45
Alg 3	$N_3, N_1, N_4, N_2, N_2, N_4, N_5$	T=48 B=0 (37)	T=48 B=0 (37)	T=48 B=0 (37)	T=48 B=37	T=48 B=37
Alg 4	$N_1, N_1, N_4, N_2 N_4, N_5$	T=43 B=0 (39)	T=43 B=0 (39)	T=43 B=0 (39)	T=43 B=39	T=43 B=39
Alg 5	$N_1, N_1, N_4, N_6, N_2, N_4, N_5$	T=30 B=0 (21)	T=30 B=21	T=40 B=31	T=48 B=37	T=51 B=45
Alg 6	$N_3, N_1, N_4, N_2, N_2, N_4, N_5, N_5$	T=29 B=0 (21)	T=29 B=21	T=37 B=28	T=48 B=42	T=48 B=42
Alg 7	$N_1, N_1, N_4, N_2 N_4, N_5$	T=29 B=0 (21)	T=30 B=22	T=39 B=31	T=43 B=39	T=43 B=39

When using the Higher Density Algorithm, through the analysis of the relations B/C_i, we notice that the agent executed r_5 before r_3. As the average density of the mission until the moment of the execution of r_7 and r_8 was $E_m = 1,275$ and the densities of the optional resources r_7

and r_8 are $E_7=0,85$ and $E_8=2,67$ respectively, only optional resource r_8 was executed.

The Figure 4 shows the algorithms behavior considering the collected benefits. Observing the graphic, it can be noticed that, among the algorithm versions without clock, the Lazy Algorithm is the only one that obtained benefit with deadlines very short (for instance: $D=30$). The algorithm versions with clock deal better with short deadlines if compared with the ones without clock. When larger deadlines are defined, the biggest benefits are collected by the greedy algorithm, considering both versions (with and without clock).

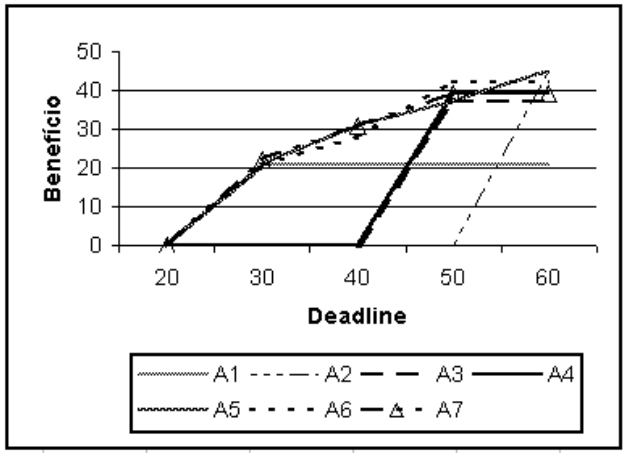

Figure 4. Comparison among the itinerary definition algorithms

6. Conclusions

This paper defined an execution model for imprecise mobile agents with time constraints and some flexibility in the definition of its itinerary. The paper also described some simple heuristics that can be used by the mobile agent for the process of choosing the itinerary. Although simple, the heuristics presented can be used as a starting point for more complex algorithms. The execution model presented here is innovative in associating time constraints and the concept of optional resources to the technology of mobile agents.

A comparison between the heuristics showed that there is a necessity to establish a compromise between mission value and execution time. As future work we include the study of more elaborate algorithms for this problem, by combining characteristics of several of the simple heuristics presented here. Also, a performance evaluation considering the probabilistic behavior of network and nodes is necessary.

7. Acknowledgements

The authors would like to thank CNPQ (Counsel National of Development Scientific and Techonological) by the financial support that allowed the realization of this work.

8. References

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

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

[3] Tolbert, L.M., Qi, H, and Peng, F. Z., "*Scalable Multi-Agent System for Real-Time Electric Power Management*" IEEE Power Engineering Society Summer Meeting. Vancouver, Canada, July, 2001, pp. 1676-1679.

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

[5] Shin, M., Jung, M, and Riu, K., "*A Novel Negotiation Protocol for Agent-based Control Architecture" 5th Int. Conference on Engineering & Automation.* Las Vegas, USA, 2001.

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

[7] Liu, J. W. S., Shih, W.-K., Lin, K.-J., et al. "*Imprecise Computations"* Proceedings of the IEEE, Vol. 82. nº 1, January, 1994, pp. 83-94.

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

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

[10] Davis, A., et al., "*Flexible Scheduling for Adaptable Real-Time Systems"* Proc. IEEE Real-Time Tech. and App. Symp, May, 1995.