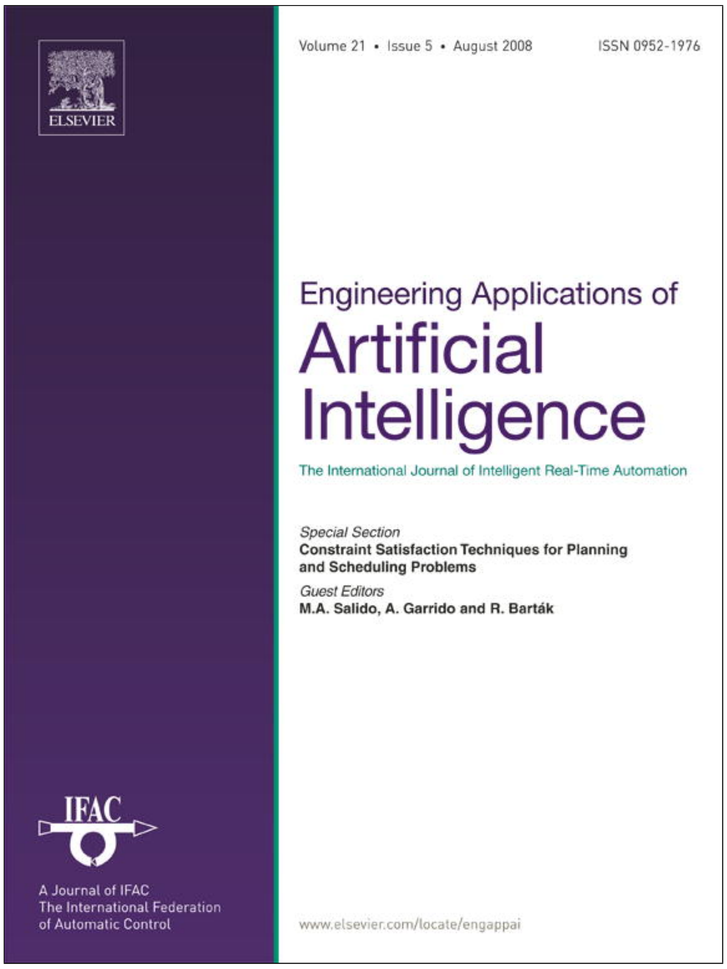

Volume 21 • Issue 5 • August 2008
ISSN 0952-1976
ELSEVIER
Engineering Applications of Artificial Intelligence
The International Journal of Intelligent Real-Time Automation
Special Section
Constraint Satisfaction Techniques for Planning and Scheduling Problems
Guest Editors
M.A. Salido, A. Garrido and R. Barták
IFAC
A Journal of IFAC The International Federation of Automatic Control
www.elsevier.com/locate/engappai

Available online at www.sciencedirect.com
ScienceDirect
Engineering Applications of Artificial Intelligence 21 (2008) 769–784
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

An execution time planner for the ARTIS agent architecture

Javier Bajo[a], Vicente Julián[b], Juan Manuel Corchado[a,*], Carlos Carrascosa[b], Yanira de Paz[a], Vicente Botti[b], Juan Francisco de Paz[a]

[a]*Departamento de Informática y Automática, Universidad de Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain*
[b]*Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 46022 Valencia, Spain*

Received 11 December 2006; received in revised form 9 July 2007; accepted 16 July 2007
Available online 7 September 2007

Abstract

The purpose of this paper is to present an approach for integrating new complex deliberative behaviours in a real-time agent architecture, specifically in the ARTIS agent architecture, which is specially designed for hard real-time environments. The new deliberative agent proposed remakes its plans at runtime conserving the system integrity and its real-time feature. The proposed system has been successfully tested in a robotic test environment. This environment consisted of the automated management of the internal and external mail in a department plant, where the main goal was to ease the workload of a mail-robot. The results obtained increased the flexibility and adaptability of the real-time agent while retaining the temporal restrictions.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Execution time planner; Case-based reasoning; Case-based planning; ARTIS agent; Mail delivery

1. Introduction

The future of multi-agent systems will be determined by how they can be applied to real and complex environments. Such environments may need to satisfy fundamental restrictions for which current agent-based techniques may not be appropriated. That is the case of hard real-time environments, where multi-agent systems seem especially suitable for developing solutions in systems of this kind (Julian et al., 2000). The present study covers the problem of real-time multi-agent system construction, where it is necessary to merge intelligent deliberative techniques with real-time actions in complex and distributed environments. The system development with these integrated features produces an unquestionable increase in complexity and a need for the adaptation of current techniques or deliberative processes and, in some cases, the development of new ones. More specifically, this paper proposes the integration of a new deliberative behaviour, based on bounded case-base reasoning techniques, in the ARTIS hard real-time agent architecture (AA) (Botti et al., 1999).

A real-time system (RTS) may be defined as a system at which correction depends not only on the computation result but also on the moment this result is produced (Burns and Wellings, 2001). In general, a RTS is characterized by the concurrent execution of tasks, the priority of security and reliability, and temporal determinism. Nowadays, the application of agents and multi-agent systems to RTSs provides flexibility and distributed problem solving. The developing of new applications in the real-time distributed systems area has added a great complexity to the systems, especially to requirements related to the dynamic adaptation to changing conditions. The development of a great part of the RTS is realized by handling them as distributed systems, especially because these RTS possess the inherent features of distribution and dynamism (Burns and Wellings, 2001).

The agent theory is considered a very powerful and promising option that can be used to develop complex distributed systems. If multi-agent systems are applied to RTSs, then they may be called real-time multi-agent systems. Furthermore, it is possible to introduce in these systems the concept of real-time agent as an agent that has to deal with timing restrictions to carry out its responsibilities (Botti et al., 1999). Therefore, a real-time agent

*Corresponding author. Tel.: +34 618696589; fax: +34 923294514.
E-mail address: corchado@usal.es (J.M. Corchado).

0952-1976/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2007.07.006

must guarantee the fulfilment of its timing restrictions and, at the same time it has to try to get its goals or objectives (Soler et al., 2002). If a real-time agent is used in a multi-agent system and besides it takes into account that in RTSs the communications and protocols among agents cannot be too many complexes, then a more complete definition of real-time multi-agent system is obtained.

The ARTIS AA (Botti et al., 1999) provides a real-time agent model composed basically of *in-agents* able to model the agent behaviours dedicated to reach its goals. *In-agents* are internal entities that have all the knowledge needed to solve a particular sub-problem. Last but not least, a Control Module in charge of the real-time execution of all the AA *in-agents* is available. According to the AA features shown, it seems to fit for solving hard real-time problems.

Intelligent agents may use a lot of reasoning mechanisms. One of them is based on planning techniques (Martens and Uhrmacher, 2002). Planning-based agents decide the course of an action before it is realized. Thus, a plan represents the structure of such action. A planning-based agent will execute plans allowing it to reach its goals. To do this, the agent goes from an initial state to try to get to a final state or set of states. The mechanism used to reach the goals is to apply a set of operators over the objects composing the agent's environment. The AAs are agents that allow the use of planning techniques. To do this, they have a knowledge base composed of the agent's environment description, a description of the objectives to be obtained and an inference mechanism that allows it to reach the final state from the initial state.

The main aim of this paper is to integrate new bounded deliberative techniques into the ARTIS AA. Such an agent will be able to incorporate a new planning proposal known as case-base planning-beliefs desires intentions (CBP-BDI) derivative from CBR-BDI system (Corchado and Laza, 2003) in order to carry out deliberative planning tasks at the moment where the timing restrictions are not considered critical. A CBP system is a type of case-based reasoning (CBR) system that works with plans instead of cases (Aamodt and Plaza, 1994). This integration makes it possible to improve the ARTIS AA, incorporating a deliberative behaviour. One of the main problems that need to be overcome is the efficient integration of high-level, deliberative planning processes within reactive processes. These complex deliberative processes, which allow the agent to adapt and learn, are unbounded and it is difficult to integrate them in hard RTSs. Typically, in the multi-agent area the processes are carried out by so-called deliberative agents, which decide what to do and how to do it according to their mental attitudes. In a deliberative agent, it is relatively simple to identify decision processes and how to perform them. However, its main drawback lies in finding a mechanism that permits its efficient and temporal bounded execution. Therefore, it would be interesting to integrate complex deliberative processes for decision-making in hard RTSs in a simple and efficient way. This work presents an improvement to the ARTIS AA incorporating a deliberative case-based planning behaviour, which facilitates an innovative hybrid AA with both real-time and deliberative capabilities.

Our case study consists in solving the automation of the internal mail management of a department that is physically distributed in a single floor of a building plant (restricted and well-known test environment). At the department, there is a mail robot in charge of attending sending requests, carried out by a user from a department office through a PDA to send a letter or packet to other office of such department. In this way, the robot will be in charge of picking up and delivering the external mail received by the department or the mail that is going to be sent to the outside. The robot is going to be controlled by an ARTIS agent (Botti et al., 1999). Each ARTIS agent has a reflex server (RS) able to plan tasks at real-time and a second-level deliberative server (DS) in charge of non-critical timing restrictions. So, this agent will be able to replan in situations where the robot is unable to fulfil the assigned plans, such as finding obstacles, closed doors, low battery level, or receiving new requests of picking up or sending mail while the robot is executing a plan.

The case study has been implemented in a simulated environment in order to evaluate the proposal. To do this, different experiments have been carried out investigating, basically, the performance of the system and the planning/replanning behaviour. The results have shown the benefits obtained with the integration of the CBP-BDI deliberative behaviour into the ARTIS agent while maintaining the fulfilment of the critical time restrictions.

The rest of the paper is structured as follows: Section 2 shows a study of related work; Section 3 focuses on the ARTIS AA; Section 4 describes how the ARTIS agent integrates an execution-time planner; a simulated application example is shown in Section 5; finally, the analysis of the results obtained over the example and conclusions are described in Section 6.

2. Related work

Planning based on cases is a kind of planning based on experience. The generation of a new plan is made from plans or fragments of plans that have been previously generated (Hammond, 1990; Veloso et al., 1996; Muñoz-Avila and Aha, 2004). The different planners based on cases differ from each other in the way that they represent and store the cases and the way in which they execute the CBP cycle (in algorithms executed in each of its stages). The case-based planner proposed within the framework of this article incorporates an adaptation algorithm that allows dynamic replanning in execution time. This fact means that our system is unique in terms of the response that it offers to changes in the environment during the execution of the plan.

The applications of the planning agents are increasingly prolific, especially in fields such as the web, games, tourism applications, etc. Case-based Tactician (CAT) introduces a case-based planner with a plan-retrieval algorithm that, by using three key sources of domain knowledge, removes the assumption of a static opponent (Aha et al., 2005). In Muñoz-Avila and Aha (2004)) it is described an application of hierarchical case-based planning that involves reasoning in the context of real-time strategy games. Multiagent planning in the web (MAPWeb) presents a multiagent system for cooperative work among different intelligent software agents whose main goal is to solve user planning problems using the information stored in the Web (Camacho et al., 2006). The RETSINA AA presents a planner module for every task agent, which interleaves HTN planning and process execution (Giampapa and Sycara, 2002). Furthermore, Giampapa and Sycara (2001) describe a prototype in which a conversational case-based reasoner, NaCoDAE, was agentified and inserted in the RETSINA multi-agent system. Some case-based planners have been used in tourism applications, such as the one presented by Corchado (Corchado et al., 2005) in order to improve the traditional tourism techniques. Users of the case-based planner tourism application noticed the utility of the dynamic replanning, since it is quite usual for them change opinions/objectives in the middle of a plan. Another application field is intelligent guidance and suggestions in leisure or shopping. Bohnenberger et al. (2005) present a decision-theoretic location-aware shopping guide in a shopping mall as a kind of virtual shop assistant. Bohnenberger et al. (2005) propose the use of decision-theoretic planning, but their system cannot provide the option of replanning in execution time. SHOMAS (Bajo et al., 2006) uses the CBP-BDI mechanism for replanning in execution time and incorporates RFID technology to automatically asses a user's location. Furthermore SHOMAS uses past experiences to take new decisions, which increases the personalization and adaptation capabilities of the system as well as the success of the guidance. The CBP-BDI mechanisms enables the system to offer efficient plans in execution time that make it possible to choose optimum routes, and to react to changes that may be produced in the execution of the plan, responding with a dynamic replanning that avoids "retracing one's steps".

On the other hand, this article can be framed within the field of "Artificial Intelligence in Real Time", which can be defined as (Musliner et al., 1995): 'combining guaranteed execution methods of RTSs with artificial intelligence (AI) planning, problem resolution and adaptation mechanisms in order to construct an intelligent and flexible control system that can dynamically plan its own behaviours and guarantee that these behaviours satisfy the maximum for strict execution time limits.

In this way, in order to obtain such characteristics of flexibility and adaptability, the use of the multi-agent paradigms would seem the most appropriate for the development of hard RTSs in environments that are clearly characterized by very strong time constraints. As such the paradigm has been applied successfully in architectures such as CIRCA/SA-CIRCA (Goldman et al., 2001; Musliner, 2002) and ARTIS (Garcia-Fornes, 1996; Carrascosa et al., 2006).

One of the problems central to the construction of RTSs is to determine whether an admissible plan exists, in other words, whether there exists an assignation of the resources from the system (processors, memory, network, input/output devices, etc.) to the real-time tasks, in such a way that they meet the response time limits set.

Generally, RTS scheduling is a complete-NP problem. Nevertheless the problem can be simplified by considering solely the processor scheduling and assuming that the rest of the resources will be available when the task requires them. This approximation is realistic in mono-processor systems in which the system is developed in such a way that the tasks available from the memory that are required and the access to common resources are encapsulated in a critical section.

Research in RTSs is aimed specifically at the resolution of this problem, developing methods to guarantee that the reactions of the control system are always produced on time according to when changes are produced within the environment. Normally, these methods operate on a set of fixed tasks upon which a set of time requirements are defined and the execution times for the worst-case scenario. Techniques consist of determining whether there exists an admissible plan for the set of tasks.

Among the approximations to real-time AI described above, the most promising algorithms are "any-time" (Boddy and Dean, 1994) and multiple methods (approximate processing) (Lesser et al., 1988).

3. ARTIS: real-time agent architecture

The ARTIS AA is a hard real-time AA (a more detailed definition may be found at Botti et al. (1999) and Soler et al. (2000)). According to existing agent taxonomies (Wooldridge and Jennings, 1995), the AA architecture can be labelled as vertical, hybrid and specifically designed to work in hard real-time environments. This architecture guarantees an answer satisfying all the agent's hard timing restrictions while trying to obtain the best answer, if one exists, to the current environment state.

The AA architecture can be seen from two different perspectives: the user model (high-level model) and the system model (low-level or implementation model). The user model presents the view of the system's developer, whilst the system model is the execution framework used to build the final real-time running version of the agent.

3.1. User model

From the user model point of view, the AA architecture is an extension of the blackboard model (Nii, 1986) that has been adapted to work in hard real-time environments. This model is formed by the following elements (Fig. 1):

(1) A set of sensors and effectors allowing the agent to interact with the environment. Due to the environment restrictions, the perception and the action in these environments usually is timely restricted.

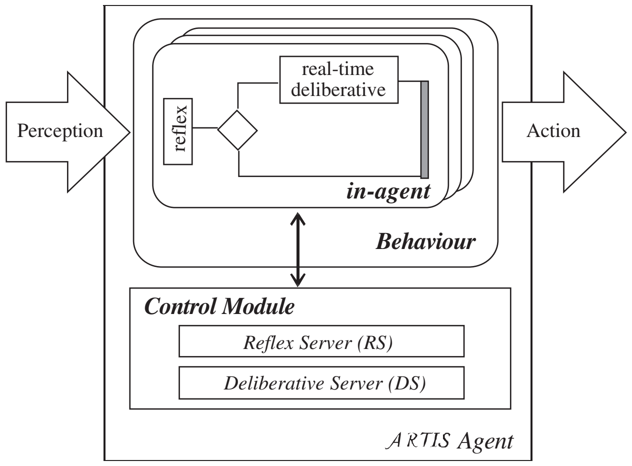


Fig. 1. ARTIS agent architecture. User Model.

(2) A set of behaviours. Behaviours model the alternate ways of facing the environment that are available to the agent. At running time, an AA always uses one behaviour at a time, which is called *active behaviour*.

Each behaviour of an AA is composed of a set of *in-agents*, each one of them dealing with a part of the environment (or solving part of the problem) that the AA is facing. The main reason to split the problem-solving method into smaller parts is to provide an abstraction that organizes the problem-solving knowledge in a modular and gradual way.

Each *in-agent* makes a specific activity related to solving a particular sub-problem, so that all the *in-agents* (in a behaviour) cooperate to solve the whole problem. The cooperation is mainly obtained sharing the calculated outcomes by the different *in-agents* through a global memory.

The *in-agents* composing the different AA's behaviours are structured into two layers. One that is in charge of solving the essential parts of the agent's problem (assuring that a minimum quality response is reached and that the problem is kept under control) and another one that is optional (trying to improve the response obtained by the other layer or dealing with parts of the problem that are not strictly needed). *In-agents* can be classified according to different criteria. According to their mode of activation, there are periodic *in-agents* (their activation is repeated every certain period established at AA's design time) and sporadic *in-agents* (activated as a reaction to a meaningful event, de-activating after only one execution until a future activation to react to such event). A more detailed explanation of the in-agent concept can be found in Botti et al. (1999) and Soler et al. (2000).

(3) A belief set comprising a world model (with all the domain knowledge that is relevant to the agent) and the internal state, that is, the agent mental states.

This set is stored in a framed-based blackboard that is accessible to all the *in-agents*. It is possible to specify that some meaningful changes in a belief may produce an event at running time.

(4) A Control Module that is responsible for the real-time execution of the *in-agents* belonging to the current AA behaviour. The time requirements of the two layers at each *in-agent* (reflex and deliberative) are different. Therefore, the Control Module has to use different execution criteria for each one. In fact, the Control Module is divided into two sub-modules, RS and DS, in charge, respectively, of the reflex and deliberative parts of the AA. Both parts of the Control Module work with the feasibility analysis in a coordinated and coherent way.

The Control Module incorporates a meta-reasoning process (Carrascosa, 2004) in order to adapt the reasoning process of the AA to changing situations. The specification of this process is the only Control Module part that is application-dependent. The specification is composed by a meta-rule set written by the AA designer in a language developed for this purpose.

It is necessary to emphasize that once the above elements' specification has been completed, it has to be validated in order to be considered as an AA's user model specification. This validation must guarantee the agent's real-time restrictions. This is carried out by means of a static feasibility analysis of each one of the different behaviours and of their transitions. This analysis, described in Garcia-Fornes et al. (1997), is based on well-known techniques in the RTSs community. The different components of the agent are shown in Fig. 1.

3.2. System model

The system model provides the software architecture implementing all the high-level features shown in the user model. The main elements of this model (translating the corresponding parts of the user model) are (more detailed explanation in Carrascosa et al. (2004)):

(1) A library to access the hardware devices and/or the hardware devices themselves. This corresponds to the sensors and effectors shown in the user model.
(2) A working mode[1] set that, along with the Mode Change Protocol (Real and Crespo, 2004), implements the behaviour management expressed in the user model. The working mode concept has to be based on a specific task model that guarantees the environment's critical time restrictions. Therefore, the user model's *in-agents* are translated into system model's tasks. Each working mode will consist of the tasks corresponding to the *in-agents* defined in the behaviour related to that mode.

[1]A working mode is defined by the set of tasks which execution is needed in an specific situation (Sha et al., 1989; Tindell et al., 1992). The purpose of this concept, of the real-time systems area, is to reduce the number of tasks that are active at the same time.

(3) A shared memory (implementing the blackboard model) with all the data corresponding to the different beliefs specified in the user model. This memory may be accessed by all the tasks.
(4) Here, both modules of the Control Module of the user model are two separated entities:

The RS includes the real-time task scheduler[2] or first-level scheduler (FLS). The FLS uses a real-time scheduling policy such that, at running time, it decides which task is the next one to be executed. This policy is compatible with the static feasibility test (used at design time to guarantee the fulfilment of the critical restrictions associated to the different working modes). Using a scheduler at run-time helps the AA to adapt itself to environmental changes and to benefit from tasks taking less time than indicated by their worst-case execution time.

The FLS also implements the mode change protocol included in the AA, allowing it to change its working mode according to the transitions specified in the user model. These changes are activated by running specific mechanisms. Furthermore, there exists a slack-stealing algorithm (Davis, 1993; Garcia-Fornes, 1996), based on a specific technique of the real-time community, to calculate at run-time the available time for the DS to execute optional parts belonging to the current working mode.

The DS also includes a real-time task scheduler, the second-level scheduler (SLS). This scheduler executes optional parts of the active tasks (see Carrascosa et al., 2004) in the current mode. So, in the slack intervals, the SLS uses a scheduling policy that will choose the next optional part to execute according to quality criteria.

Though it has no specific counterpart in the user model, the system model also includes a static analysis of the feasibility of the different working modes.

It is important to emphasize that this analysis does not build a plan with the tasks execution sequence. On the contrary, this analysis only assures the capability of the scheduler at run-time (FLS) to execute the real-time tasks guaranteeing their deadlines.

4. Execution-time planner for ARTIS agents

The main purpose of this paper is to develop and to integrate new bounded CBP-based planner techniques inside of the ARTIS AA. This planner allows a more efficient execution time management, according to the agent's goals. It has to be taken into account that this planner activates tasks to fulfil agent's goals that will be deal by the real-time task schedulers in order to be executed guaranteeing the real-time constraints. CBR-based planner (or CBP) has been included as a sporadic *in-agent* that will be activated when a new plan needs to be generated for a new goal. Moreover the *in-agent* will be also activated when replanning because the environment evolution makes it impossible to finish the current plan. The *in-agent*'s initial part reads the planning or replanning event that activated it. According to this event, it checks if the existing current plan is still feasible. If such plan is not yet applicable, it builds a new plan or modifies the existing one. In an optional way, it tries to improve the new plan. Lastly, the action part of this *in-agent* begins the plan.

The CBR-based planner provides planning based on previous experiences. CBR systems use memories (past experiences) to solve new problems. The main concept when working with CBR systems is the concept of case. A case is a past experience that can be represented as a 3-tuple $\langle$P,S(P),R$\rangle$. In this way a case is composed of a problem description (initial state), the solution applied to solve the problem (in CBP the solution is a plan or a set of plans, in other words, the sequences of actions executed in order to achieve the objectives) and the result obtained after applying the solution (the final state an the evaluation of the plan executed). The planner needs to maintain a case memory that will be used to solve new problems. When a new problem is presented the planner executes a CBR cycle to solve it. The CBR cycle is composed of four sequential stages: Retrieve, where those cases with the most similar problem description to the current problem are recovered from the cases memory; Reuse, in which the plans (solutions) corresponding to the similar cases retrieved in the previous stage are reused to construct a new plan; Revise, where the proposed plan is evaluated; and Retain, where the planner learns from the new experience. One of the key points in the CBR-based planning is the notation used to represent the solution (the plans). A solution can be seen as a sequence of intermediate states transited to go from an initial state to the final state. States are usually represented as propositional logic sets. The set of actions can be represented as a set of operators together with an order relationship. Furthermore Carbonell (1986) indicates that additional information is needed on the decisions taken during the plan execution.

According to the mail delivery problem, the test environment is restricted. Let $E=\{e_0,\ldots,e_n\}$ the set of the possible collection points and mail delivery. $e_{j\text{-}}$ $j\in\{0,\ldots,n\}$ represents the point of collection of the external mail provided by the postman.

In each action the agent goes from the delivery point or of mail collection to other one

$$a_j : \begin{array}{ccc} E & \to & E \\ e_i & \to & a_j(e_i)=e_j \end{array} . \qquad (1)$$

Agent plan is the name given to a sequence of actions that, from a current state e_0, defines the path of states through which the agent passes in order to reach the other delivery point or of mail collection. The dynamic relation-

[2]It has to be taken into account that a real-time task scheduler is the system's part in charge of managing the CPU time, indicating which is the task using it at each moment.

ship between the behaviour of the agent and the changes in medium is modelled below.

The behaviour of *agent A* is represented by its *action function* $a_A(t)$ $\forall t$, defined as a correspondence between one moment in time t and the action selected by the agent,

$$\text{Agent } A = \{a_A(t)\}_{t \in T \subseteq N}. \tag{2}$$

From the definition of the *action function* $a_A(t)$ a new relationship that collects the idea of an agent's *action plan* can be defined,

$$p_A : \begin{array}{ccc} TxA & \rightarrow & A \\ (t,a_A(t)) & \rightarrow & p_A(t) \end{array} \tag{3}$$

in the following way, taking into account the dynamic character of our agent

$$p_A(t_n) = \int_{t_0}^{t_n} a_A(t)\,\mathrm{d}t. \tag{4}$$

The variation of the agent plan $p_A(t)$ will be provoked essentially by

(1) The *changes that occur in the medium and that force the initial plan to be modified.*
(2) The knowledge from the success and failure of the plans that were used in the past, and which are favoured or punished via *learning*.

The Efficiency of the plan is the relationship between the objectives attained and the resources consumed

$$\text{Eff} = \frac{\#(O' \cap O)}{\#R'}. \tag{5}$$

where # means cardinal of a set, O indicates the objectives of the agent and O' are the results achieved by the chosen plan ($O' \subseteq O$; O' is a subset of O). R are the total resources and R' are the resources consumed by the agent, after that, the agent carries out the chosen plan ($R' \subseteq R$; R' is a subset of R).

Given a problem E and a plan $p(t)$ it is possible to construct functions Ob and Rc accumulated from the objectives and costs of the plan. For all time points t_i two variables can be associated:

$$Ob(t_i) = \int_a^{t_i} O(t)\,\mathrm{d}t, \quad Rc(t_i) = \int_a^{t_i} R(t)\,\mathrm{d}t. \tag{6}$$

$O(t)$ indicates the objectives of the agent by time t and $R(t)$ are the total resources by time t.

This allows us to construct *a planning space* (or space representing the environment for planning problems) as a vectorial hyperdimensional space where each axis represents the *accumulative variable* associated with each objective and resource.

In the planning space, defined in this way, conform to the following properties:

(1) *Property* 1: The representation of the plans within the planning space are always monotonously growing functions. Given that $Ob(t)$ and $Rc(t)$ are functions defined as positive (see definition), function $p(t)$ expressed at these coordinates is constant or growing.
(2) *Property* 2: In the planning space, the straight lines represent plans of constant efficiency. If the representation of the plans are straight lines, the slope of the function is constant, and coincides with the definition of the efficiency of the plan

$$\frac{\mathrm{d}}{\mathrm{d}t}p(t) = \text{cte} \Leftrightarrow \lim_{\Delta \to 0} \frac{\Delta O(t)}{\Delta R(t)} = \text{cte}. \tag{7}$$

In an *n*-dimensional space, the extension of the straight concept line is called a geodesic curve. In this sense, the notion of *geodesic plans* can be introduced. Geodesic plans are defined as those that maintain efficiency at a constant throughout their development, and therefore, they are the most replanned in the event of changes in the environment in order to complete the desired objectives. This way, only the plans of constant efficiency (geodesic plans) are considered, due to the fact that they represent minimum risk. In an environment that changes unpredictably, any plan that is distal to the geodesic plan means that a certain risk is accepted. Geodesic plans have been used in many other domains. Schramm et al. (2005) use geodesic plans for planning optimization, Peyre and Cohen (2003) apply geodesic plans for adaptive remeshing, Sbeh et al. (2001) propose the use of geodesic reconstruction and image segmentation and Page et al. (2006) apply geodesics for path planning for 3D terrains.

Given a problem, the agent must search for the plan that determines a solution with a series of restrictions $F(O; R) = 0$. If all the possible plans $\{p_1,\ldots,p_n\}$ (which are a collection of points) are represented within the *planning space*, a *subset of states* that the agent has already attained in the past in order to resolve similar problems will be obtained. With the mesh of points obtained (generally irregular) within the planning space and using interpolation techniques, a working hyperplane $h(x)$ (that encapsulates the information on the set of restrictions from restored experiences) can be obtained, from which geodesic plans can be calculated.

In general, the simplest variation problem is given when f_sf is only one point in the space, f_sf, and the geodesic g that links e_0 with e^* is obtained (Fig. 2).

In a problem where the set of end points is $n > 1$, *variation techniques with mobile frontiers* are used to offer us a set of geodesics between the starting point and each one of the points of the final set. Suppose, for simplicity's sake, that a planning space of dimension 3 with coordinates $\{O,\ R_1,\ R_2\}$ is selected. Between the point e_0 and the objective points f_sf and over the interpolation surface $h(x)$, the Euler Theorem Jost and Li-Jost (1998) guarantees that the expression of the geodesic plans will be obtained by resolving the following system of equations:

$$\begin{aligned} \frac{\partial L}{\partial R_1} - \frac{\mathrm{d}}{\mathrm{d}O}\frac{\partial L}{\partial R_1'} &= 0, \\ \frac{\partial L}{\partial R_2} - \frac{\mathrm{d}}{\mathrm{d}O}\frac{\partial L}{\partial R_2'} &= 0, \end{aligned} \tag{8}$$

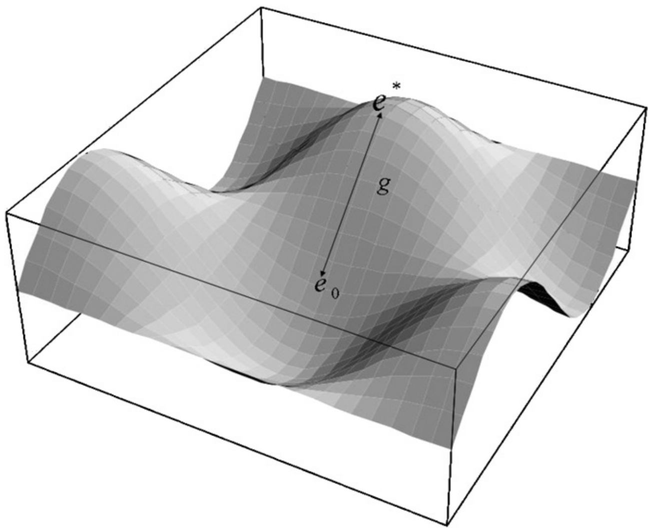


Fig. 2. Geodesic g linking initial and final point.

where R_i is the function accumulated R, O is the function of accumulated O and L is the distance function on the hyperplan $h(x)$

$$L = \int_h \mathrm{d}l. \tag{9}$$

In order to obtain all the geodesic plans that, on the surface $h(x)$ and beginning at e_0, allows us to reach any of the points $e^* \in f_s f$, a condition must be imposed on the surroundings: the initial point is $e_0 = (O_0, R_0)$. Using variation techniques it is possible to obtain expressions for all the geodesic plans that, beginning at e_0 allows us to attain the desired point. Once the plans that will create efficient solutions between the current state and the set of solution states have been obtained, it will be possible to calculate the plan around it (along its trajectory). This is done by using a denser distribution of geodesic plans (in other words, a greater number of geodesic plans in its environment). The tool that allows us to determine this, is called the *minimum Jacobi field associated with the solution set* (Lee, 1997).

Let g_0:[0,1]$\rightarrow S$ be a geodesic over a surface S. Let h:[0,1]$x[-\varepsilon,\ \varepsilon]\rightarrow S$ be a variation of g_0 so that for each $t\in(-\varepsilon,\ \varepsilon)$, the set $\{h_t(s)\}_{t\in(-\varepsilon,\ \varepsilon)}$:

(1) $h_t(s)\ \forall t\in(-\varepsilon,\varepsilon)$ are geodesic in S.
(2) They begin at $g_0(0)$, in other words, they conform to $h_t(0) = g_0(0)\ \forall t\in(-\varepsilon,\varepsilon)$.

In these conditions, taking the variations to a differential limit:

$$\lim_{t\to 0}\{h_t(s) = g_0(s+t)\} = \lim_{t\to 0}\{h(s,t)\} = \left.\frac{\partial g_0}{\partial t}\right|_{(s,0)} = \frac{\mathrm{d}g_0}{\mathrm{d}s} \equiv J_{g_0}(s). \tag{10}$$

The term $J_{g_0}(s)$ is given to the Jacobi Field of the geodesic g_0 for the set $\{g_n(x)\}_{n\in N}$, and in the same way that the definition has been constructed, it is possible to give a measurement for the distribution of the other geodesics of $\{g_n(x)\}_{n\in N}$ around g_0 throughout the trajectory. Given a set of geodesics, some of them in their environment, have a greater distribution than other geodesics in a neighbouring environment. This is equivalent to saying that it presents a variation in the distribution of geodesics lower than the others and therefore the Jacobi Field associated with $\{g_n(x)\}_{n\in N}$ reaches its lowest value at J_{g^*}.

Let us return to the most-re-plan-able (MRP) agent problem that, following the recuperation and variation calculation phase, contains a set of geodesic plans $\{p_1,\ldots,p_n\}$, the *MRP solution* will be the geodesic plan p^* with minimum associated Jacobi field associated with the set $\{g_n(x)\}_{n\in N}$.

In this way, the behaviour model G for the MRP agent is defined. For each problem that it represents, the agent selects the MRP solution defined as that geodesic plan with minimum Jacobi field, that expresses

$$G(e_0, p_1, \ldots, p_n) = p^* \Leftrightarrow \exists n \in N / J_{g_n} \equiv J_{g*} = \operatorname*{Min}_{n\in N} J_{g_n}. \tag{11}$$

If the plan p^* is not interrupted, the agent will reach a desired estate $e_j \equiv e^* \in f_s f_j \in \{1,\ldots,m\}$. Below, in the learning phase, a weighting $w_f(p)$ is stored. With the updating of weighting $w_f(p^*)$, the planning cycle of the cased based planning (CBP) motor is completed. Below, it is possible to see what happens if p^* is interrupted. Let us suppose that the agent has initiated a plan p^* but at a moment $t>t_0$, the plan is interrupted due to a change in the medium. The geodesic planning (the section of plans with a constant slope in the planning space) meets the conditions of the Bellman Principle of Optimality (Bellman, 1957), in other words, each on of the plan's parts is partially geodesic between the selected points. This guarantees that if g_0 is geodesic for interrupted e_0 in t_1, because e_0 changes to e_1, and g_1 is geodesic to e_1 that is begun in the state where g_0 has been interrupted, it follows that

$$g = g_0 + g_1 \text{ is geodesic to } e = e_0(t_1 - t_0) + e_1(t_2 - t_1). \tag{12}$$

If each time the environment changes and interrupts the execution plan, a new geodesic plan is selected; the *overall plan will be geodesic*. The dynamic process follows the CBP cycle recurrently: each time a plan finds itself interrupted, it generates from the state reached so far, the surroundings of the plans from the case base and adjusts them to the new problem. With this it calculates the geodesic plans and selects the one, which meets the minimum conditions of the associated Jacobi field. In this way, the dynamic planning model of the agent $G(t)$ is characterized (Fig. 3). Fig. 3 shows a replanning situation in which an interruption occurs during the execution of a plan. CBR-based planner responds to this situation by identifying a new initial state, the last intermediate state marked as success (e_1 in Fig. 3). The next step is recovering the initial planning philosophy, that is, to execute a CBP cycle and obtain a new plan. Now the set of geodesic plans will be composed of those plans previously considered geodesics in the initial plan (all were optimal and the closest to the initial plan) and new plans (those with the most similar problem description to the current initial problem description, provided by e_1). The new CBP cycle takes into account the new restrictions (some of the tasks have already been executed and there are new environmental conditions).

In the dynamic context the following properties of $G(t)$ are particularly relevant:

(1) *Property* 1: All the Jacobi fields are variations of geodesics.

It can be demonstrated (Milnor, 1973) that there exists a isomorphism between all Jacobi fields that is constructed between the end points.

(2) *Property* 2: All the geodesic variations are Jacobi fields (Milnor, 1973).

These two results allow us to introduce the concept of a *global Jacobi field*. The *Global Jacobi field or Dynamic Jacobi field* $J(t)$ is the Jacobi field formed by a set of partial or successive Jacobi fields. The above properties allow us to ensure that the change from one partial Jacobi field and the next preserves the conditions a Jacobi field because it produces a change between geodesics: It is possible to observe that a minimum global Jacobi field $J(t)$ also meets Bellman's conditions of optimality (Bellman, 1957), in other words, a minimum global Jacobi field, must select minimum Jacobi fields "in pieces"

$$J_{\min}(t) = \{J_{\min}(t_1 - t_0), J_{\min}(t_2 - t_1), \ldots, J_{\min}(t_n - t_{n-1})\}. \tag{13}$$

If on the one hand, successive Jacobi fields generate one Jacobi field, and on the other hand, minimum Jacobi fields generate a minimum Jacobi field, the MRP agent that follows a strategy of replanning $G(t)$ as indicated in order to survive a dynamic environment, it generates a *global plan* $p^*(t)$ that, faced with all possible global plans $\{p_n(t)\}_{n\in N}$, presents a minimum value in its Jacobi field $J_{g^*}(t) \equiv J_{p^*}(t)$.

For example, Fig. 4 shows the representation of tasks of a postman robot in a space $\Re^3$, according to the following three coordinates: time, number of objectives achieved, and number of resources used (coordinates taken from similar cases retrieved). In order to retrieve the cases, firstly those from the beliefs base that have at least the delivery points for the current problem are selected, and that from those, a similarity measurement is used, the cosin, taking into account the restrictions of the current problem and those of previously selected cases. This similarity measurement is based on the angle formed by two vectors (these vectors may represent items, users, keywords, profiles, etc.) in order to determine whether they are similar or not. If the cosin is equal to 1, they will be equal, and totally different if it is −1 and dissimilar if it is 0.

$$\cos(\vec{i}, \vec{j}) = \frac{\langle \vec{i}, \vec{j} \rangle}{\|\vec{i}\| . \|\vec{j}\|},$$

where $\langle .,. \rangle$ signifies scale product and $\|.\|$ module. Specifically, Fig. 4.a shows a hyperplane of restrictions and the plan followed by a case retrieved from the beliefs base, considered to be similar. So that Fig. 4a is not overly large, and in order for the plan to be appreciated at first glance, the time axis has been rescaled (axis z) to [0,1].

For other similar retrieved cases, the same procedure is followed. The new plan is made in such a way that the planner proposes the plan in sections, with the greatest density of plans around it (reflected by the formulae (10) and (11)).

In order to understand the graphical representation, given that the plans are made up of pieces, one initial task e_0 and a final task e_5 are the focus. Between the initial and the final task, the mail robot could carry out other tasks. The idea that the planner presents is to choose as the optimal solution the plan that has the most plans around it, involving these two fixed tasks, (independently of whether or not it includes other tasks, then the mail robots will only do the assigned tasks). In this way, as can be seen in Fig. 4b, the plan chosen is the one represented by a discontinuous line, since it represents the plan that has most other plans around it, and involves other tasks that

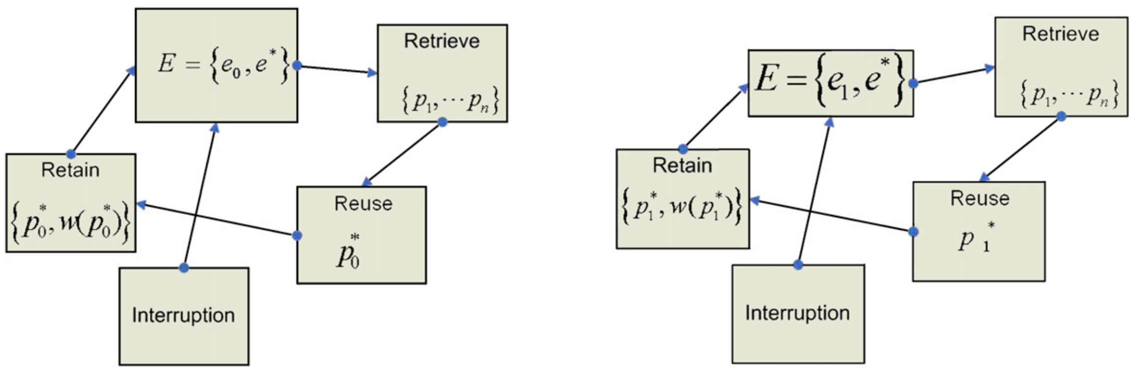


Fig. 3. Model for behaviour $G(t)$.

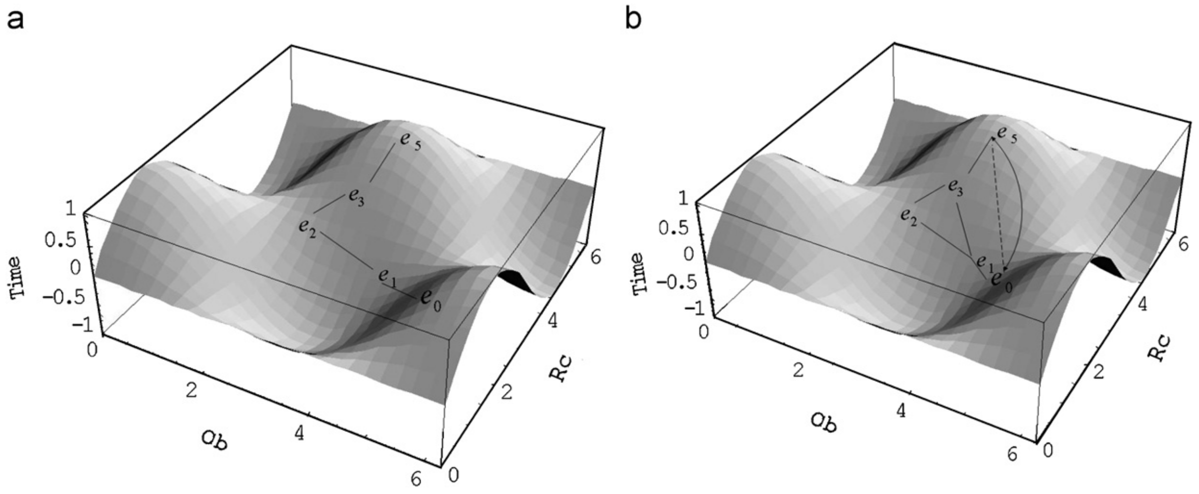


Fig. 4. Hyperplan of restrictions: (a) hyperplan together with the corresponding plan and (b) selection of the most replannable plan.

could be assigned in the even of interruptions to the initial plan (for example due to new assigned tasks).

If the tasks were repetitive and the plan was never interrupted, it would be enough to find the plan once only and no replanning would be necessary, but it does not make sense in dynamic environments, which vary during execution time.

Up until now, an agent that in a dynamic environment seeks plans that lends it greater capacity for replanning has been formally defined.

In a RTS, it is necessary to respect time restrictions, and as such, in the planning system explained above, the time available and the time consumed within the deliberative process is taken into account. In other words, when a plan is given both the time available and the time taken for the planner to make the optimum plan are taken into account (the MRP route in the event that the plan is interrupted). For this purpose, when a new task is presented, similar cases are recovered from the beliefs base and two actions are carried out: one is to consider the plan carried out previously for similar cases and which took less time than the time currently available, while the other consists of taking into account the maximum deliberative time of the cases recovered.

In order to obtain a better understanding of how the planner respects the time restrictions, the application example presented in Section 4 can be observed. In such example, the robot begins to execute the recovered plan (limited by the time available) and the planner begins to deliberate (using the planning method described above), taking into account the time restrictions: the difference between the available time and the maximum time observed from the recovered cases (thereby ensuring that the deliberative process is completed and can offer an optimum plan that is also limited by the available time). Once the deliberative process is completed, the initial plan is replanned taking into account the optimal plan, and adapting itself to this one as such ensuring that the objectives are reached within the available time.

5. Case study

In this section, a prototype of the mail robot example is presented, developed according to our approach, which emphasizes the proposal presented in previous sections. The problem to solve consists of the automated management of the internal and external mail (physical mail, non-electronic) in a department plant. For this purpose, the system created by this automation process, must be able to request the shipment of a letter or package from an office on one floor to another office from the same floor, as well as the reception of external mail at a collection point for later distribution. Once this service has been requested, a mobile robot (or postbot) must gather the shipment and direct it to the destination. It is important to note that each mail or package distribution must be finalized before a maximum time, specified in the shipment request. In order to be able to carry out all this, the resources employed include a mobile robot type Mobile Pioneer 2, and one radio network that allows the communication within the robot around the plant.

According to these resources, this problem will be solved using a real-time agent by means of the ARTIS AA (Botti et al., 1999). This architecture will give the real-time support to the system, considering that it must satisfy critical time restrictions (of forced fulfilment). And will be designed according to the ARTIS hard real-time AA. On the other hand, all the planning processes for the delivery and collection of mail around the floor will be managed by a deliberative planning behaviour integrated inside the real-time agent. This behaviour will give the more appropriate distribution routes to the mobile robot. This planning behaviour will be developed following the CBP-BDI model.

The cases are represented using objects. A case is composed of a problem description, the solution (plan) given to solve the problem and the efficiency obtained after applying the solution. Table 1 shows a case structure.

Table 1
Case attributes

Case field	Measurement
Home	Location
Agentstate	AgentState
Currentloc	Location
Targets	ArrayList of Target
Restricts	ArrayList of Restriction
Environment	Environment
Solution	Solution
Efficiency	Efficiency

A case is composed of the problem description (location—x, y coordinates; AgentState—batery, velocity, current mail, capacity, mission time; target—location and restriction; restriction—required time, forbidden areas, etc.; environment—map, obstacles, etc.) and the solution (solution—ArrayList of plans, decisions, solution data; efficiency—percentage, comments).

A simulation prototype was implemented using the ARTIS architecture on a Pionner 2 mobile robot simulation software. As mentioned above, the robot must deliver a set of letters or packages from an office to another office on the same department floor. Several simulation experiments were conducted to evaluate different aspects and try to show the benefits of the planner integration into the ARTIS architecture. The different experiments were tested on an ARTIS agent without planning behaviour and on one, which includes the planner behaviour proposed above.

A view of the application environment is shown in Fig. 5. The figure presents a space formed by a set of offices connected through different corridors. Initially, the PostBot is informed about the arrival of new mail through a Jade-Leap agent running in a PDA. This notice contains a set of offices to be visited in order to deliver or to gather new mail. Later, the PostBot may be informed of new mail orders, which implies changes in the initial set of offices. It is important to remark that in the case of the ARTIS agent without planning behaviour the different mail orders are sequentially executed in a FIFO order. That is, a new order is put at the end of its order list. On the other hand, the other ARTIS agent will execute its mail orders according to its planning/replanning capability. Moreover, in each case the mail or package distribution must be finalized before a maximum time and the robot control behaviour must guarantee robot integrity, which implies hard real-time constraints.

In the examples it has been supposed that if each unit of time is equivalent to one unit of space, the unit of time is the second; the second 0 has been taken as the starting point (the robot begins to work at 0 s, if it began to work at 8 am, at 9 it would have worked 3600 s); in order to aid comprehension, the Manhattan distance has been used.

$$d_M((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|.$$

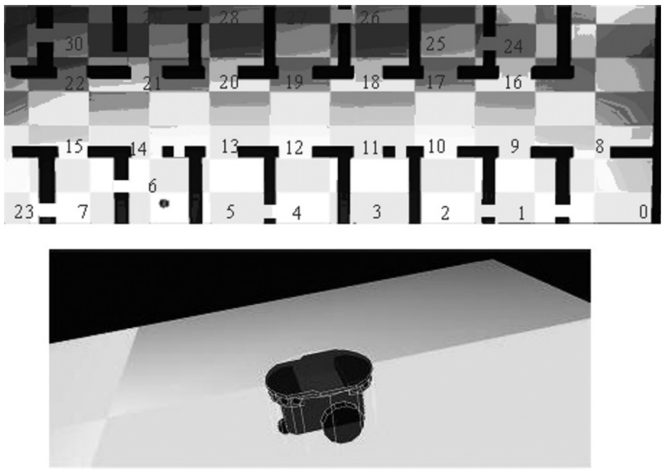


Fig. 5. Different views of the simulation environment.

Let us suppose that the robot is in position 0, the letter pick-up centre; let us also suppose that in the centre there is a letter and that each letter has a delivery point (from a total of 32 delivery points). The delivery points are represented through coordinates in a plan. Let us suppose that there are no restrictions of any type, such as delivery time or battery life. Given that there are 32 delivery/pick-up points, the 32 points should be represented in $\mathfrak{R}^2$, each point represented by its coordinates $\{e_i = (x_i, y_i)\}_{i=0,\ldots,31}$. e_0 is the robot's starting point. In the simple case, there is no type of restriction. The cases with similar delivery descriptions are retrieved from the beliefs base—in other words those cases without restrictions that have at least the same delivery points (Retrieve). Using interpolation techniques a hyperplan is revealed that contains all the delivery points of the retrieved cases, that at a minimum contained the collection/delivery points assigned to the robot in its task. Within this hyperplan the retrieved plans are drawn (that are no more than geodesic curves). The solution selected is that plan with the greatest density of plans around it (formula 10—Reuse stage)—in other words the one that corresponds with the geodesic of least value from the associated Jacobi field, as can be observed in Fig. 4. Graphically it is possible to represent the MRP plan (formula (11)) as shown in Fig. 6.

In Fig. 6 it is possible to see how two possible routes exist for the robot to use initially: the robot must choose to move from position 0 to 1 or from position 0 to 8 (we must choose an action, formula (1)). It should be noted that an initial case free of restrictions has been chosen. The objectives were: Deliver and Collect the letters at the delivery/collection points and the resource is the lifetime of battery. In this case, $O = O'$, since the objectives are being achieved and $R' \subseteq R$, since the battery of the robot has not run out, because we have assumed there are not any restrictions.

In the event that some type of restriction is made, such as battery life, the process would be the same but the only cases retrieved from the beliefs case would be those whose

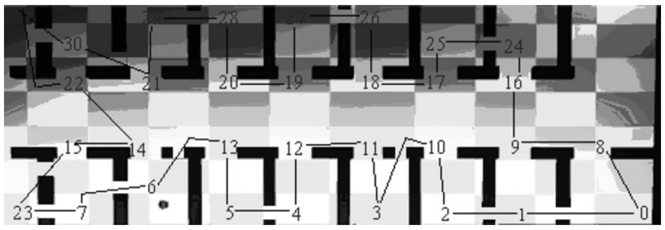


Fig. 6. Representation of the most replannable plan for a case without restrictions.

associated plan are executed within the lifetime of battery z.

In other test it has been supposed that the same description of the problem is given, with the same delivery/collection points, but with a series of initial restrictions. These restrictions are priority delivery, service time according to delivery point (since more than one letter can now be delivered to—or collected from—one delivery point) and battery life. In this case, the initial data comes from those expressed by $\{e_i = (x_i, y_i)\}_{i=0,\ldots,31}$. At each delivery/collection point there is a minimum delivery time, maximum time and service time (which depends on the number of deliveries and collections, for example) $D_i = [\min_i, \max_i - \text{service time}_i]$, $i = 0,\ldots,31$. Another restriction is the battery life. In the beliefs base the time is stored which the robot took to carry out each one of the plan in the past. As such, when the cases with similar descriptions are retrieved, only those cases with an execution time less than the battery life are taken into account (Retrieve). Once cases are retrieved from the beliefs base, a hyperplan of restrictions is created. Upon adding two restrictions, another coordinates axis is required, working in $\Re^3$; on the z-axis, the delivery/collection times are represented in seconds. As such, each delivery/collection point will have its coordinate in x, its coordinate in y and two values on the axis z. For each (x_i, y_i), there is a $\min_i$ and a $\max_i - \text{service time}_i$; $i = 0,\ldots,31$. The hyperplan of restriction is the one that contains the minimum number of delivery/collection points assigned to the robot as a task and also $\forall(x_i, y_i, z_i)$ verifies $z_i \in D_i$. In order to reach the hyperplan of restrictions, interpolation techniques are applied, taking into account that they pass by the points (x_i, y_i), $i = 0, \ldots, 31$ of the retrieved cases; and fixed "i" that

$$z_i = \underset{j \in \left\{ \begin{array}{c} \text{retrievaled cases} \\ \text{associated to } (x_i, y_i) \end{array} \right\}}{\underset{\text{Frequency}}{\text{Max}}} \{z_{i_j}\}$$

with $i = 0,\ldots,31$. The fact that similar cases have been retrieved from the beliefs case that comply with $z_i \in D_i$, $i = 0,\ldots,31$. Once the restrictions hyperplan has been obtained and the retrieved plans represented in it (corresponding to geodesic curves), a plan with most plans around it is chosen (formula (10)), being the geodesic of least value in the associated Jacobi field (formula (11)—reuse). The MRP solution in our example is the one shown in Fig. 7.

Given the restrictions imposed, only one direction is valid. This is shown in the table with explanation below.

According to the MRP plan presented in Table 1, the robot initially moves from delivery/collection point 0 to delivery/collection point 1. The arrival time to point 1 is 15. Given that the minimum collection time for point 1 was 15, the robot does not needs to wait there until this time (until the collection can be made). The time that the robot needs to make the delivery and collection is 10 units of time and later the robot uses 15 units to get to point 4. In this way the time take will be 55 units of time to point 4 (arriving between the minimum and maximum collection time) and so on…

The goals were: Deliver and Collect the letters at the delivery/collection points. The considered resource is the lifetime of the battery. In the time shown in Table 2 the robot did not have problems with the battery life, in this case, $O = O'$, since the goals are being achieved and $R' \subset R$, since the battery of the robot has not run out.

If the first row in the table is observed, for point 0 (robot), the maximum delivery time is 3390 units of time. On the other hand, in the last row of the table, the arrival time to the starting point (point 0) is at 2788 (note, this is obtained adding together arrival time + service time + distance to point 0), with which it can be observed that the battery will not run out.

Let us see now that there is an interruption during the delivery/collection process. Specifically, when the robot was carrying out the delivery/collection tasks in point 17, it received a communication informing it that delivery point 30 had changed its minimum and maximum collection times from 2300 to 2385 units of time. In this case the robot changes its starting point. Now point 0 for the robot will change to point 17. The same will apply to the numbers of delivery/collection points, which the root should go to in order to complete, its task. At this moment the planning reasoning will be repeated, but now with 15 delivery/collection points. The MRP plan obtained from the recall of similar cases taking into account the points that have not been visited and the current restrictions would be the one shown in Fig. 8. The objectives were: Deliver and collect the letters in the delivery/collection points and the

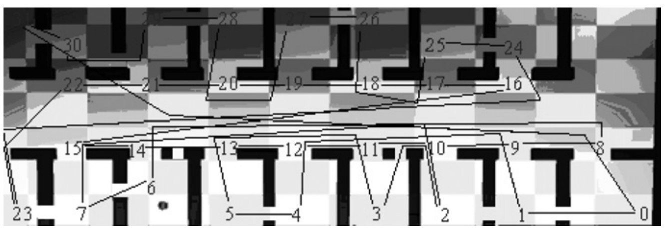


Fig. 7. Representation of the most replannable plan for a case with time restrictions.

Table 2
MRP

Sequence order	Delivery/collection coordinates	Arrival time	Minimum collection time	Maximum collection time	Service time
0	(90.0, 0.0)	0.0	0.0	3390.0	0.0
1	(75.0, 0.0)	15.0	15.0	402.0	10.0
4	(45.0, 0.0)	55.0	23.0	368.0	10.0
5	(35.0, 0.0)	75.0	93.0	484.0	10.0
3	(55.0, 0.0)	123.0	12.0	505.0	20.0
2	(65.0, 0.0)	153.0	10.0	645.0	10.0
7	(15.0, 0.0)	213.0	170.0	595.0	10.0
6	(25.0, 5.0)	238.0	168.0	787.0	10.0
8	(85.0, 10.0)	313.0	331.0	822.0	20.0
9	(75.0, 10.0)	361.0	448.0	897.0	30.0
10	(65.0, 10.0)	488.0	499.0	1030.0	40.0
11	(55.0, 10.0)	549.0	585.0	1136.0	20.0
12	(45.0, 10.0)	615.0	666.0	1245.0	10.0
13	(35.0, 10.0)	686.0	772.0	1329.0	20.0
14	(20.0, 10.0)	807.0	890.0	1395.0	30.0
15	(15.0, 10.0)	925.0	1036.0	1439.0	20.0
16	(75.0, 20.0)	1126.0	1268.0	1785.0	20.0
17	(65.0, 20.0)	1298.0	1330.0	1919.0	10.0
18	(55.0, 20.0)	1350.0	1519.0	1926.0	10.0
19	(45.0, 20.0)	1539.0	1560.0	2083.0	20.0
20	(35.0, 20.0)	1590.0	1689.0	2144.0	10.0
21	(25.0, 20.0)	1709.0	1763.0	2264.0	40.0
22	(15.0, 20.0)	1813.0	1846.0	2365.0	10.0
23	(5.0, 0.0)	1886.0	1985.0	2412.0	20.0
24	(75.0, 25.0)	2100.0	2077.0	2514.0	40.0
25	(65.0, 25.0)	2150.0	2143.0	2638.0	20.0
26	(55.0, 30.0)	2185.0	2310.0	2659.0	30.0
27	(45.0, 30.0)	2350.0	2380.0	2787.0	40.0
28	(35.0, 30.0)	2430.0	2385.0	2976.0	10.0
29	(25.0, 30.0)	2450.0	2603.0	2952.0	10.0
30	(15.0, 25.0)	2628.0	2628.0	3113.0	10.0
31	(5.0, 30.0)	2653.0	2653.0	3280.0	20.0
		2788.0			

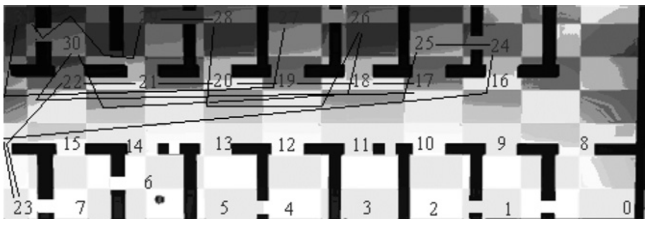


Fig. 8. Representation of the most replannable plan after an interruption.

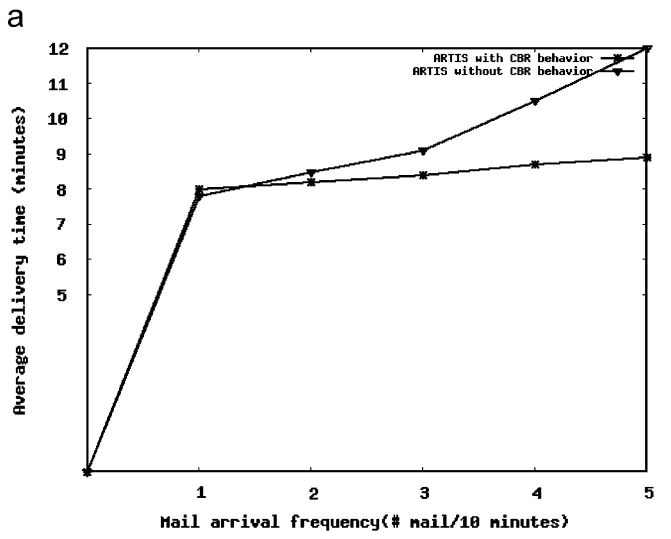
resource is the lifetime of battery. In Table 3, we can note that, $O = O'$, since the objectives are being achieved and $R' \subset R$, since the battery of robot has not run out; because, the maximum delivery time is 3390 units of time and the robot battery ran out at 2783.

6. Results and conclusions

Several simulation experiments were conducted to evaluate different parameters in order to asses the proposal. The first set of experiments investigates the performance of the system according to package or mail arrival frequency. The simulation prototype was tested by increasing this frequency incrementally and by testing two different parameters: average delivery time in the whole system and percentage of deliveries completed on time. Each experiment was repeated one hundred times and the results show the average obtained value.

Fig. 9 illustrates how the average delivery time maintains more or less constant when the CBP behaviour is used. However, in the tests carried out with the ARTIS agent, if the planning/replanning capacity increases in proportion to the arrival frequency, the average delivery time increases considerably. In terms of the percentage of deliveries over time, the CBP behaviour agent maintained a level of 90% even with the maximum foreseen frequency. In the event that the agent's planning/replanning capacity is eliminated, as might be expected, it performs worse, decreasing the delivery time percentage significantly.

The second set of experiments concerns the investigation of the system's replanning behaviour. In order to do this, the simulation was tested introducing events that will cause a replanning in the system and measuring the same parameters as in the previous experiment. Each experiment supposes a delivery by the agent of a set of 10 letters as previously indicated, and the average delivery time and



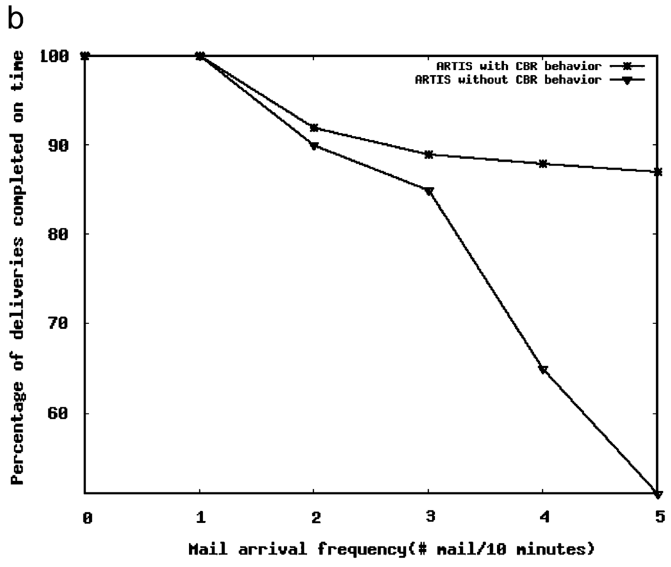


Fig. 9. (a) Delivery average time in the whole system and (b) percentage of deliveries completed on time when increasing the mail arrival frequency.

percentage of deliveries carried out is measured. The behaviour of these parameters is then studied in relation to those events, which lead to replanning (Table 3).

In Fig. 10, the results demonstrate that thanks to the planning/replanning capacity, the average delivery time is hardly affected by increases in number of uncontrolled events. On the other hand, if this capacity is not available the robot agent's average delivery time increases considerably. As far as the percentage of deliveries is concerned, if the agent is capable of planning/replanning behaviour, the behaviour can be considered quite acceptable, while if it is not available the percentage of deliveries carried out is considerably inferior.

The agent and multi-agent system paradigm has changed the development process of certain software systems. Nevertheless, the technologies employed in multi-agent systems must be adapted for their correct use in real-time environments. Accordingly, this paper has proposed the integration of a new deliberative capacity, based on bound case-base reasoning techniques, into a hard real-time ARTIS agent. More specifically, the work has shown how a new temporally bound CBR-based planner (CBP-BDI agent) has been integrated inside the ARTIS AA. This new planner takes into account the time available and the time consumed within the deliberative process, allowing a more efficient execution time management, according to the agent's goals.

The main goal of this approach was to increase flexibility and adaptability of RTS implementations, and it has been achieved. According to the example implementation and the results obtained, in any situation of the real-time environment the PostBot controlled by an ARTIS with planning behaviour has a better performance than the PostBot controlled by an ARTIS agent without planning behaviour. This approach allows to the ARTIS architecture an extremely high degree of flexibility while at the same time retaining the hard or soft time restrictions

Table 3
MRP after the interruption

Sequence order	Delivery/collection coordinates	Arrival time	Minimum collection time	Maximum collection time	Service time
17	(65.0, 20.0)	0.0	1330.0	1919.0	10.0
18	(55.0, 20.0)	1350.0	1519.0	1926.0	10.0
19	(45.0, 20.0)	1539.0	1560.0	2083.0	20.0
20	(35.0, 20.0)	1590.0	1689.0	2144.0	10.0
21	(25.0, 20.0)	1709.0	1763.0	2264.0	40.0
22	(15.0, 20.0)	1813.0	1846.0	2365.0	10.0
23	(5.0, 0.0)	1886.0	1985.0	2412.0	20.0
24	(75.0, 25.0)	2100.0	2077.0	2514.0	40.0
25	(65.0, 25.0)	2150.0	2143.0	2638.0	20.0
30	(15.0, 25.0)	2220.0	2300.0	2385.0	10.0
26	(55.0, 30.0)	2355.0	2310.0	2659.0	30.0
28	(35.0, 30.0)	2405.0	2385.0	2976.0	10.0
29	(25.0, 30.0)	2425.0	2603.0	2952.0	10.0
31	(5.0, 30.0)	2633.0	2653.0	3280.0	20.0
27	(45.0, 30.0)	2713.0	2380.0	2787.0	40.0
		2783.0			

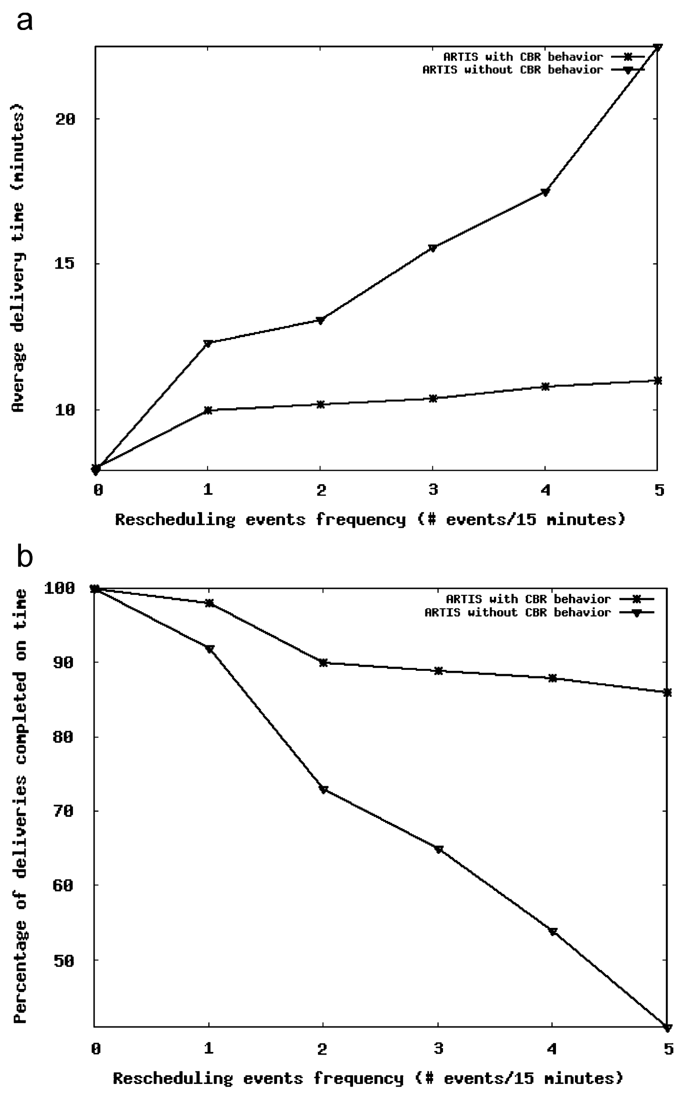


Fig. 10. (a) Delivery average time in the whole system and (b) percentage of deliveries completed on time when increasing the external event arrival frequency.

needed in systems of this kind. On the other hand, some limitations were detected. The case-based planning behaviour needs initial knowledge as well as initial cases to be efficient, so the agent needs some initial data or some initial time to learn about the environment. Furthermore, the case memory needs index and maintenance. By now the CBP uses the plan's efficiency as an index to organize and accesses the case memory.

The results are promising for deployment within a real scenario in the near future. The characteristics of the architecture presented in this paper make it very suitable for application in dynamic environments, in which learning and adaptation to constant changes is required. In this sense our future research work will consist in the application of the proposed planner in everyday dynamic environments, such as the construction of intelligent environments that facilitate care for the sick and elderly, housing, e-commerce, e-learning, entertainment etc. They can also be applied to other domains characterized by unstructured and non-conventional environments such as manufacturing companies, scheduling and control, manipulation of materials, etc.

References

Aamodt, A., Plaza, E., 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, vol. 7, IOS Press, March, pp. 39–59.

Aha, D.W., Molineaux, M., Ponsen, M., 2005. Learning to win: case-based plan selection in a real-time strategy game. In: Case-Based Reasoning Research and Development, Lecture Notes in Computer Science, vol. 3620, Springer, Berlin, pp. 5–20.

Bajo, J., Corchado, J.M., de Paz, Y., de Paz, J.F., Martín, Q., 2006. A multiagent recommending system for shopping centres. In: Proceedings of the ECAI 2006 Workshop on Recommender Systems, August 28–29, 2006, Riva del Garda, Italy, pp. 92–96.

Bellman, R.E., 1957. Dynamic Programming. Princeton University Press, Princeton, NJ.

Boddy, M., Dean, T., 1994. Deliberation scheduling for problem solving in time-constrained environments. Artificial Intelligence 67, 245–285.

Bohnenberger, T., Jacobs, O., Jameson, A., 2005. DTP meets user requirements: enhancements and studies of an intelligent shopping guide. In: Proceedings of PERVASIVE-05, Lecture Notes in Computer Science, vol. 3468, Springer, Berlin, pp. 279–296.

Botti, V., Carrascosa, C., Julian, V., Soler, J., 1999. Modelling agents in hard real-time environments. In: Proceedings of MAAMAW'99, Lecture Notes in Artificial Intelligence, vol. 1647, Springer, Berlin, pp. 63–76.

Burns, A., Wellings, A., 2001. Real-Time Systems and Programming Languages. Addison-Wesley, Longman, Reading, MA.

Camacho, D., Aler, R., Borrajo, D., Molina, J.M., 2006. Multi-agent plan based information gathering. In: Applied Intelligence, vol. 25(1). Springer, Netherlands, pp. 59–71.

Carbonell, J.G., 1986. Derivational analogy: a theory of reconstructive problem solving and expertise acquisition. In: Machine Learning: An Artificial Intelligence Approach, vol. 2. Morgan Kaufmann, Los Altos, CA, pp. 371–392.

Carrascosa, C., 2004. Meta-razonamiento en Agentes con restricciones temporales criticas. Ph.D. Thesis, Dept. Sistemas Informaticos y Computación, Univ. Politécnica, Valencia.

Carrascosa, C., Terrasa, A., Fabregat, J., Botti, V., 2004. Behaviour management in real-time agents. In: Proceedings of Fifth Iberoamerican Workshop on Multi-Agent Systems, pp. 1–11.

Carrascosa, C., Terrasa, A., García-Fornes, A., Espinosa, A., Botti, V., 2006. A meta-reasoning model for hard real-time agents. In: Selected Papers from the 11th Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2005), vol. 4177, pp. 42–51.

Corchado, J.M., Laza, R., 2003. Constructing deliberative agents with case-based reasoning technology. International Journal of Intelligent Systems 18 (12), 1227–1241 ISSN:0884-8173.

Corchado, J.M., Pavón, J., Corchado, E.S, Castillo, L.F., 2005. Development of CBR-BDI agents: a tourist guide application. In: ECCBR 2004, Lecture Notes in Artificial Intelligence, vol. 3155, Springer, Berlin, pp. 547–559.

Davis, R.I., 1993. Approximate slack stealing algorithms for fixed priority preemptive systems. Technical Report YCS217, Department of Computer Science, University of York.

Garcia-Fornes, A., 1996. ARTIS: Un modelo y una arquitectura para sistemas de tiempo real inteligentes. Ph.D. Dissertation, Dept. Sistemas Informaticos y Computacion. Univ. Politecnica Valencia.

Garcia-Fornes, A., Terrasa, A., Botti, V., Crespo, A., 1997. Analyzing the schedulability of hard real-time artificial intelligence systems. Engineering Applications of Artificial Intelligence, 369–377.

Giampapa, J.A., Sycara, K., 2001. Conversational case-based planning for agent team coordination. In: Case-based Reasoning Research and

Development: Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR 2001, vol. 2080, Springer, Berlin, Heidelberg, July 2001, pp. 189–203.
Giampapa, J.A., Sycara, K., 2002. Team-oriented agent coordination in the RETSINA multi-agent system. In: The Paper Presented at AAMAS 2002 Workshop on Teamwork and Coalition Formation. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
Goldman, R.P., Musliner, D.J., Krebsbach, K.D., 2001. Managing online self-adaptation in real-time environments. In: Proceedings of Second International Workshop on Self-Adaptive Software, Balatonfured, Hungary.
Hammond, K., 1990. Case-based planning: a framework for planning from experience. Cognitive Science 14 (3), 385–443.
Jost, J., Li-Jost, X., 1998. Calculus of Variations. Cambridge University Press, UK.
Julian, V., Carrascosa, C., Rebollo, M., Soler, J., Botti, V., 2000. SIMBA: an approach for real-time multi-agent systems. In: Lecture Notes in Computer Science, vol. 2504(1), Springer, Berlin, pp. 282–293.
Lee, J.M., 1997. Riemannian Manifolds. An Introduction to Curvature. Springer, New York.
Lesser, V.R., Pavlin, J., Durfee, E., 1988. Approximate processing in real-time problem solving. AI Magazine 9 (1), 49–61 Spring.
Martens, A., Uhrmacher, A.M., 2002. Adaptative tutoring processes and mental plans. In: Cerri, S.A., Gouardères, G., Paraguaçu, F. (Eds.), Proceedings of Intelligent Tutoring Systems—ITS 2002, Springer, Berlin, pp. 71–80.
Milnor, J., 1973. Morse Theory. Annals of Mathematical Studies. Princeton University Press, Princeton, NJ.
Muñoz-Avila, H., Aha, D.W., 2004. On the role of explanation for hierarchical case-based planning in real-time strategy games. In: Gervás, P., Gupta, K.M. (Eds.) Proceedings of the ECCBR 2004 Workshops (Technical Report 142-04), Departamento di Sistemos Informáticos y Programación, Universidad Complutense Madrid, Madrid, Spain.
Musliner, D.J., 2002. Safe learning in mission-critical domains: time is of the essence. Working Notes of the AAAI Spring Symposium on Safe Learning Agents, Stanford, California.
Musliner, D.J., Hendler, J.A., Agrawala, A.K., Durfee, E.H., Strosnider, J.K., Paul, C.J., 1995. The challenge of real-time in AI. IEEE Computer (January), 58–66.
Nii, P., 1986. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. AI Magazine, 38–53.
Page, D.L., Koschan, A.F., Abidi, M.A., Overholt, J.L., 2006. Ridge-valley path planning for 3D terrains. In: Proceedings of International Conference on Robotics and Automation 2006, pp. 119–124.
Peyre, G., Cohen, L.D., 2003. Geodesic re-meshing and parameterization using front propagation. In: Proceedings of Second IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision (VLSM'03).
Real, J.V., Crespo, A., 2004. Mode change protocols for real-time systems: a survey and a new proposal. Real-Time Systems 26(2), ISSN:0922-6443.
Sbeh, Z.B., Cohen, L.D., Mimoun, G., Coscas, G.A., 2001. New approach for geodesic reconstruction in mathematical morphology and application to image segmentation and tracking in ophtalmology. IEEE Transactions on Medical Imaging 20 (12), 1321–1333.
Sha, L., Rajkumar, R., Lehoczky, J., Ramamritham, K., 1989. Mode Change Protocols for Priority-Driven Preemptive Scheduling, UM-CS-1989-060, p. 31.
Schramm, F., Micaelli, A., Morel, G., 2005. Calibration free path planning for visual serving yielding straight line behaviour both in image and work space. In: IROS 2005, Edmonton, Canada, pp. 2216–2221.
Soler, J., Julian, V., Carrascosa, C., Botti, V., 2000. Applying the ARTIS agent architecture to mobile robot control. In: Proceedings of IBERAMIA'2000, Atibaia, Sao Paulo, Brazil, vol. I, Springer, Berlin, pp. 359–368.
Soler, J., Julian, V., Rebollo, M., Carrascosa, C., Botti, V., 2002. Towards a real-time multi-agent system architecture. In: Proceedings of the First International Workshop on Challenges in Open Agent Systems, Bologna, Italy.
Tindell, K.W., Burns, A., Wellings, A.J., 1992. Mode changes in priority pre-emptively scheduled systems. In: IEEE Real-Time Systems Symposium, pp. 100–109.
Veloso, M., Muñoz-Avila, H., Bergmann, R., 1996. Case-based planning: selected methods and systems. AI Communication 9 (3), 128–137.
Wooldridge, M., Jennings, N.R., 1995. Intelligent agents: theory and practice. The Knowledge Engineering Review 10 (2), 115–152.

Javier Bajo is at present a Ph.D. student and an Assistant Professor at the University of Salamanca (Spain), he obtained his Information Technology degree at the University of Valladolid (Spain) in 2001 and an Engineering in Computer Sciences degree at the Pontifical University of Salamanca in 2003. He has been Member of the organising and scientific committee of several international symposiums such as CAEPIA, IDEAL, HAIS, etc. and co-author of papers published in recognized journals, workshops and symposiums.

Vicente Julián (Ph.D.) is originally from Valencia (Spain), he received his B.S. and M.S. degrees in Computing Engineering from the Polytechnic University of Valencia in 1992 and 1995, respectively. He is a Lecturer and obtained his Ph.D. at the Computer Science Department at the Polytechnic University of Valencia in 2002. His current research interests are in multi-agent systems, agent design, information retrieval and real-time systems.

Juan M. Corchado (Ph.D.) received his Ph.D. in Computer Science from the University of Salamanca in 1998 and Ph.D. in Artificial Intelligence (AI) from the University of Paisley, Glasgow (UK) in 2000. At present he is Associate Professor, Director of the Intelligent Information System Group (http://bisite.usal.es) and Director of the M.Sc. programs in Computer Science at the University of Salamanca (Spain). Previously he was sub-director of the Computer Science School at the University of Vigo (Spain, 1999–2000) and Researcher at the University of Paisley (UK, 1995–1998). He has been a research collaborator with the Plymouth Marine Laboratory (UK) since 1993. He has led several Artificial Intelligence research projects sponsored by Spanish and European public and private institutions and has supervised seven Ph.D. students. He is the co-author of over 130 books, book chapters, journal papers, technical reports, etc. published by organizations such as Elsevier, IEEE, IEE, ACM, AAAI, Springer Verlag, Morgan Kaufmann, etc., most of these present practical and theoretical achievements of hybrid AI and distributed systems. He has been President of the organising and scientific committee of several international symposiums.

Carlos Carrascosa (Ph.D.) was born in Valencia (Spain) and received his M.S. degree in Computer Science from the Polytechnic University of Valencia in 1995. Currently, he is a Lecturer and obtained his Ph.D. from the Computer Science Department at the Polytechnic University of Valencia. His research interests include multi-agent systems, learning, information retrieval and real-time systems.

Yanira de Paz is at present a Ph.D. student, she holds a scholarship provided by the Spanish Minister of Education to complete a Ph.D. program at the University of Salamanca (Spain). She obtained her Mathematics degree in 2002 and a Statistic degree in 2003 at the University of Salamanca (Spain). She is an Assistant Professor at the Faculty of Economy at the University of Salamanca and co-author of several mathematical and statistical books. She has also been Lecturer in the Faculty of Mathematics at the Complutense University of Madrid. She has been co-author of published papers in several journals.

Vicente J. Botti (Ph.D.) is an Electrical Engineer and Doctorate in Computer Science. He is currently a Full Professor at the Universidad Politécnica de Valencia (Spain), where he has also been the Head of the Department of Informatics Systems and Computation. His fields of study are focused mainly on multi-agent systems, methodologies for developing multi-agent sytems, artificial societies, and more specifically, real-time multi-agent systems, real-time systems, mobile robotics (in which he has developed his own models, architectures and applications) in addition to the field of knowledge engineering and softcomputing techniques. He is leader of an extensive research group whose general line of research is Artificial Intelligence and has published about 200 scientific articles. He has been and is a principal researcher on nationally and internationally funded projects (CICYT, MC&T, ESPRIT, etc.), and on technology transfer agreements, as well as sitting on various scientific committees in his areas of interest.

Juan Francisco De Paz is currently a Ph.D. student of Computer Science at the University of Salamanca (Spain). He obtained his Technical Engineering in Systems Computer Sciences degree in 2003, an Engineering in Computer Sciences degree in 2005 at the University of Salamanca and at this moment is finishing Statistics in the same University. He has been co-author of published papers in several journals.