Ensuring Time in Real-Time Commitments

Martí Navarro, Stella Heras, and Vicente Julián

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera S/N 46022 Valencia (Spain)
{mnavarro,sheras,vinglada}@dsic.upv.es

Abstract. This paper presents a framework for the estimation of the temporal cost of agent commitments. The work here presented focuses on the development of a commitment manager as a special module in a real-time agent. This manager has been constructed for the previous analysis of a commitment request in a temporal bounded way. The proposed commitment manager incorporates as CBR module whose aim is to decide if the agent will be able to perform a requested service without exceeding a specified deadline. The proposal has been tested over a mobile robot example and this paper presents the results obtained.

1 Introduction

Nowadays, a way of characterising MAS is to employ the notion of *commitments*. Commitments are viewed as responsibilities acquired by an agent for the fulfilment of some action under certain conditions concerning other agent [1]. If we apply the notion of commitments in real-time environments, the responsibility acquired by an agent for the accomplishment of some action under, possibly hard, temporal conditions increases the complexity of this kind of systems. So, we can define a *real-time commitment* as a commitment characterised by the fact that an agent delegates a task to another with a determined execution time or deadline. Therefore, the agent who commits itself to developing this task must not fail to fulfil this commitment on time. Otherwise, a deadline violation in real-time environments may cause serious or catastrophic effects in the system or produce an important decrease in the quality of the response.

This work proposes a commitment-based framework for real-time agents (agents specially designed for real-time environments –RT-Agents–) [2] based on the Case-Based Reasoning (CBR) methodology. To do this, our proposal includes bounded CBR techniques providing temporal estimations based on previous experiences. This temporal bounded CBR allows a feasibility analysis checking if the agent has enough time to do the commitment, while guaranteeing the possible real-time constraints of the agent. The rest of the paper is structured as follows: section 2 shows the main features of the commitment-based framework; section 3 presents the Temporal Constraints Manager; section 4 proposes the bounded CBR technique; section 5 summarises the results obtained in an application example based on mobile robots and finally, some conclusions are explained in section 6.

H. Geffner et al. (Eds.): IBERAMIA 2008, LNAI 5290, pp. 183–192, 2008.

2 Real-Time Commitment Management

The Commitment Manager is a module of a real-time agent aimed at improving the agent behaviour when it offers services in a real-time environment. This manager is integrated in the real-time agent and it is used to analyse whether or not the agent can satisfy the service requests of other agents before certain deadline. Once a service request is accepted, the agent is committed to its performance within the Multi-Agent System (MAS) where the agent is located. Also, the agent manages other different commitments that it has acquired before. As a last resort, if the agent integrity is in danger due to any unexpected error, the manager can cancel a commitment. The Commitment Manager is composed of two independent modules:

- **The Resource Manager:** with this module the agent can check if it has the necessary resources to execute the related tasks to achieve the goal associated to the service when its request arrives. Otherwise, the module can determine when the agent would have the resources available. This analysis calculates when the agent should start the task execution with all the available resources.
- **The Temporal Constraint Manager (TCM):** before the agent is committed to performing a service, verifying if it can complete the service before its deadline is necessary. The TCM module performs this verification. This module uses dynamic real-time scheduler techniques to determine if executing the task ensuring its temporal constraints is feasible.

The real-time agent uses the Resources Manager and the Temporal Constraint Manager modules to determine if it can commit to performing the service on time. Due to space restrictions, this paper is focused on the temporal analysis performed by the TCM.

3 Temporal Constraint Manager

As pointed out before, the TCM is in charge of deciding if an agent can commit itself to performing some service without exceeding the maximum time that it has been assigned for performing that service.

To determine whether a service can be executed on time, knowing the worst-case execution time (WCET) for each service to complete its execution is necessary. In some cases, the execution time of the service is known and limited. In these bounded services to determine the necessary tasks to fulfil the service and the maximum time needed to perform it is relatively easy.

Moreover, there are services whose number of necessary tasks to complete them cannot be determined, and therefore, to calculate the needed time to execute those services is not possible. In this type of services, a time estimation is the unique time measure that can be made. Figure 1 shows the execution phases of the TCM. The module is launched when the agent begins its execution. At the begining, the manager controls if a new service request has arrived

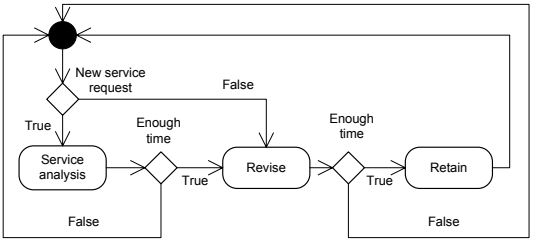

Fig. 1. Temporal Constraints Manager Algorithm

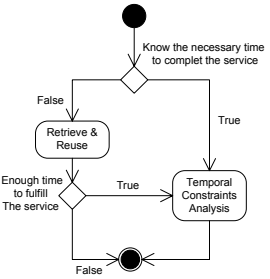

Fig. 2. Service Analysis state

(Figure 2). If the new request is an unbounded service request, the manager must estimate the time required to execute that service. As explained in section 4, to determine if the service can be completed before the deadline specified in the request is necessary. When the estimated time is obtained and the service execution is possible, the necessary tasks to perform the service are analysed at low-level using a real-time scheduler. The WCET of each phase of the TCM is known and therefore the phases are temporally bounded. This feature is crucial to allow the TCM to have a precise time control of each execution phase. As can be seen in Figure 1, the TCM execution is cyclical. When there is no request, the manager can employ its time to perform the revision and retention phases (below commented). The CBR methodology employed to obtain the temporal estimations is explained in the following section.

4 RT-CBR

As shown in the previous section, the TCM must decide if an agent can commit itself to performing a specific service. A possible way of performing such decision-making functionality is to use the knowledge that the manager has gained about previous commitments that it undertook in the past. This fits the main assumption of Case-Based Reasoning systems, which adapt previous problem solving cases to cope with current similar problems [3]. Therefore, in the absence of unpredictable circumstances, we assume that an agent can commit itself to performing a service within certain time if it has already succeeded in doing so in a similar situation.

To carry out the decision-making about contracting or not a commitment, the TCM has been enhanced with a RT-CBR module, following a soft Real-Time approach. This module estimates the time that the performance of a service could entail by using the time spent in performing similar services. With this aim, the RT-CBR module follows the typical steps of the CBR methodology: *retrieve* similar experiences from a *case-base*, *reuse* the knowledge acquired in them, *revise* such knowledge to fit the current situation and finally, *retain* the knowledge learnt from this problem-solving process.

The cases of the RT-CBR module have the following general structure:

$$C =< S, \{T\} > \tag{1}$$

where S represents the features that characterise a service (or one of its subtasks) that the agent performed in the past and T is the time that the agent spent in the execution of that previous service. In addition, since the same task could be executed several times with different duration, this value T can also be a series of temporal values. Therefore, the RT-CBR module estimates the duration of new services by means of a function $t : T \rightarrow f(T)$ computed over the temporal values that last similar previous services. The expected time T_s to perform a service that consists of a set of tasks is the aggregation of the estimated time for each of its tasks:

$$T_s = \sum_{i=0}^{I} t_i \tag{2}$$

Finally, the series of observed execution times could also allow the RT-CBR module to estimate a *success probability* $P(T_s)$ for a service to be performed within a specified time. This is an interesting data for agents, which could use this probability to make strategic decisions about their potential commitments. Setting a *confidence factor* (CF) that represents a minimum threshold for the success probability, agents would commit themselves to fulfilling a service if:

$$\exists T_s / P(T_s) \geq CF \wedge Ts \leq deadline \tag{3}$$

Thus, agents with more risky strategies could undertake commitments with lower confidence values than more cautious agents.

4.1 Real-Time Case-Based Commitment Management

As commented before, the CBR cycle consists of four steps: *Retrieve*, *Reuse*, *Revise* and *Retain*. After the first two steps, a case-based answer to the query that started the reasoning cycle can be proposed. The last two steps are more related with the learning ability of the system, that revises the devised answer and learns from the experience. In our framework, the retrieve and reuse phases must observe soft real-time constraints and thus, its execution time must be bounded. Otherwise, the RT-CBR module could provide the TCM with useless time estimations about services whose deadline have already expired.

To bound the response time of the module, the RT-CBR case-base must have a structure that eases the cases retrieval (e.g. indexing the case-base as a *Hash Structure* with a worst case temporal cost for the search linear with the number of cases stored). Anyway, independently of the choice made about the indexation of the case-base, the temporal cost of most retrieval (and reuse) algorithms depend on its size. This entails to specify a maximum number of cases that can be stored in the case-base and to perform a constant maintenance and updating of the information stored. However, the revise and retain phases can be performed off-line, since their execution does not hinder RT-CBR in providing the TCM with an estimation about the service duration.

5 Application Example

A prototype of a mail robot example has been developed for our proposal. The problem to solve consists in the automated management of the internal and external mail (physical mail, non-electronic) in a department plant. The system created by this automation must be able to request the shipment of a letter or package from an office on one floor to another office on 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 Pioneer 2 mobile robot must gather the shipment and direct it to the destination. Note that each mail or package distribution must be finalised before a maximum time, specified in the shipment request. One of the services offered by the robot agent is mail delivery, which involves its movement from an initial to a final position. In order for an agent to commit itself to that delivery in a bounded time, a temporal estimation, as accurate as possible, is required. In this system, the TCM makes use of the CBR methodology to deal with this requirement.

5.1 RT-CBR Reasoning Cycle

In the previous example, the RT-CBR module has been integrated in the TCM of the robot agent. By means of this module, the manager can decide if an agent could perform a specific service before a deadline and hence, to commit the robot agent to the execution of that service. Therefore, the cases of the module store the information about previous shipment experiences. This information will be used to decide if the agent should undertake a new commitment. The cases are structured as follows:

$$C =< I, F, N_t, N_s, \{T\} > \tag{4}$$

where I and F represent the coordinates of a path from the initial position I to the final position F that the robot travelled (one or several N_t times) straight ahead in the past, N_s stands for the number of times that the robot successfully completed the path within the case-based estimated time. These features define the service. In addition, T shows the series of time values that the robot spent to cover that route. Note that only straight routes are stored as cases, since we assume that it is the quickest way between two points. This design decision should minimise the time for travelling an unvisited route that the RT-CBR module would try to compose by reusing several known routes (old cases).

The RT-CBR reasoning cycle starts when the TCM must decide if an agent could fulfil a shipment service within the time assigned to do it. In that case, the manager is also in charge of checking if the agent has enough power resources to travel the path. In what follows the operation of each reasoning phase of the module is described.

Retrieval and Reuse: Due to the temporal constraints that the CBR process has to keep, we have followed an anytime approach [4] in the design of the algorithm that implements the retrieval and reuse phases of the RT-CBR module. In

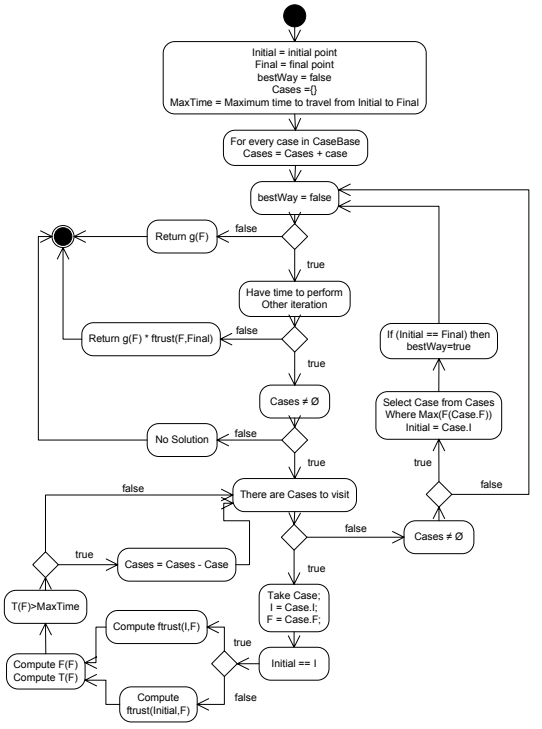

Fig. 3. Retrieval-Reuse Diagram

our design, both phases are coupled in the algorithm, reusing the time estimations about several paths to retrieve the most suitable case(s) to travel current routes (the composition of cases that minimises the travelling time). At the end of each iteration, the algorithm provides the manager with a probability of being able to perform the shipment service on time. If more time is still available, the algorithm computes better estimations on subsequent iterations. In Figure 3 a diagram of the RT-CBR retrieval-reuse algorithm is shown.

First, the RT-CBR module searches its case-base to retrieve a case that represents a similar path that the robot agent travelled in the past. Then, for each retrieved case, the algorithm uses a *confidence function* to compute the probability of travelling from an initial to a final point in an area without diverting the agent's direction. Shortest paths are assumed to have less probability that an unpredictable circumstance could deviate the agent from its route and hence, they are preferred from longer ones. In the best case, there will be a case in the case-base that exactly or very approximately covers the same path that the agent must travel. Then, the necessary time to perform the shipment can be estimated by using the time spent in that previous case. Otherwise, the route could be covered by aggregating a set of cases and estimating the global time by adding the time estimation for each case. If the route can be composed with the cases of the case-base, the following confidence function will be used:

$$f_{trust}(i,j) = 1 - \frac{dist_{ij}}{maxDist} * \frac{N_s}{N_t} \text{ where } dist_{ij} \leq maxDist \tag{5}$$

being $dist_{ij}$ the distance travelled, N_t times between the points <i,j>, N_s the number of times that the robot has travelled the path within the case-based estimated time and *maxDist* the maximum distance above which the agent is unlikely to reach its objective without finding obstacles. In the worst case, the agent will not ever travelled a similar path and hence, it cannot be composed with the cases stored. Then, a confidence function that takes into account the distance that separates both points will be used:

$$f_{trust}(i,j) = \begin{cases} 1 - \frac{dist_{ij}}{const_1} & \text{if } 0 \leq dist \leq dist_1 \\ 1 - const_2 * dist_{ij} & \text{if } dist_1 < dist \leq dist_2 \\ \frac{dist_{ij}}{dist_{ij}^2} & \text{if } dist_2 < dist \end{cases} \tag{6}$$

where *const1* and *const2* are normalisation parameters defined by the user, $dist_{ij}$ is the Euclidean distance between the initial and final points of the path <i,j> and $dist_1$ and $dist_2$ are distance bounds that represent the thresholds that delimit near, medium and far distances from the initial point. This function computes a smoothed probability that the robot can travel its path straight ahead. As the distance between the initial and the final point increases, the confidence on travelling without obstacles decreases.

Once the probability to reach the robot's objective is computed for each case, the complete route from the initial to the final position with the maximum probability of success must be selected. This route is composed by using a selection function $F(n)$ (7), which follows an $A*$ heuristic search approach [5]. The function consists of two sub-functions: $g(n)$ (8) that computes the case-based confidence of travelling from the initial point to a certain point n and $h(n)$ (9) that computes an estimated confidence of travelling from the point n to the final point (always better than the real confidence). Finally, the function $T(n)$ (10) checks if the robot agent has enough time to complete the shipment service by travelling across this specific route. Else, the algorithm prunes the route. The function consists of two sub-functions: *time(n)* (11) that computes the case-based time of travelling from the initial point to a certain point n and *E(n)* (12) that computes an estimated time of travelling from the point n to the final point. In (11) $dist_{mn}$ represents the distance between the last point m visited by the algorithm to the current point n, V_{robot} is the speed of the robot, $f_{trust}(m,n)$ corresponds to (5) or (6) (depending on the possibility of composing the route by using the cases of the case-base) and the constant $const_{trust} \in [0,10]$ shows the caution degree of the robot agent. Bigger values of this constant are chosen for more cautious agents.

Finally, if the RT-CBR algorithm is able to compose the entire route with the information stored in the case-base, it returns the case-based probability to perform the shipment service on time. Otherwise, it returns the product of the probability accumulated to that moment and a pessimistic probability to travel from the last point that could be reached by using the cases of the case-base to

the final point of the route. Finally, in case that all possible solutions computed by the algorithm exceed the time assigned to fulfil the commitment, it returns a null probability to perform successfully the service.

$$F(n) = g(n) * h(n) \tag{7}$$

$$g(n) = g(m) * f_{trust}(m, n) \tag{8}$$

$$h(n) = 1 - \frac{dist_{nf}}{maxDist} \text{ where } dist \leq maxDist \tag{9}$$

$$T(n) = time(n) + E(n) \tag{10}$$

$$time(n) = time(m) + \frac{dist_{mn}}{V_{robot}} + \frac{const_{trust}}{f_{trust}(m, n)} \tag{11}$$

$$E(n) = \frac{dist_{nf}}{V_{robot}} \tag{12}$$

Revision. Once the robot agent has finished the shipment service, it reports to the TCM the coordinates of each path that it finally travelled straight ahead and the time that it spent in doing so. In that way, the manager can check the performance of the RT-CBR module by comparing the time estimated by the module and the real time that finally took the service. Note that if the agent has not changed its navigation algorithm, it will likely try to perform the shipment by following the same route that ended in a success in the past. However, due to that some new obstacles could be found during the route, the design decision of reporting the specific paths that the robot agent has travelled has been taken.

Retention. The last step of the reasoning cycle considers the addition of new knowledge in the case-base of the RT-CBR module. As pointed out before, the size of the case-base must be controlled and therefore, only useful cases must be added (and correspondingly, out-of-date cases must be eliminated). Therefore, the decision about the addition of a new case in our model is crucial. At the moment, we have taken a simple but effective procedure by defining a threshold α below which two points must be considered as nearby in our shipment domain. Let us consider a new case c with coordinates (x_i^c, y_i^c) (initial point) and (x_f^c, y_f^c) (final point) to be added in the case-base. Following an *Euclidean* approach, the distance (dissimilarity) between the case c and each case z of the case-base can be computed with the formula:

$$dist(c, z) = max(\sqrt{(x_i^c - x_i^z)^2 + (y_i^c - y_i^z)^2}, \sqrt{(x_f^c - x_f^z)^2 + (y_f^c - y_f^z)^2}) \tag{13}$$

Therefore, the new case will be included in the case-base iff:

$$\forall z \in caseBase \ / \ dist(c, z) < \alpha \tag{14}$$

In that case, the new case $< (x_i^c, y_i^c),\ (x_f^c, y_f^c),\ 1,\ 1,\ time >$ will be added in the case-base (1 values stand for this first time that the path has been travelled and that it was done within the case-based estimated time). Note that the addition of new cases is always conditioned at the existence of 'free space' to add cases

in the case-base. Otherwise, a maintenance cycle will be triggered, deleting, for instance, those old cases that have low use ratings. Else, if a similar case has been identified, the number of times that the agent has travelled the path that represents the case (N_t) will be increased by 1 and the time spent in travelling the current path will be added to the time series of that case.

5.2 Tests and Results

To develop and execute the proposed real-time MAS, we have use the jART platform [6] that is especially designed for systems of this type and RT-Java [7] as the programming language. Once the example was implemented over the jART platform, several simulation experiments were conducted to evaluate different parameters in order to verify the use of the proposed commitment framework. A simulation prototype was implemented using a Pioneer 2 mobile robot simulation software (specifically, the Webots simulator [8]). The simulation experiments were conducted to evaluate different aspects and to try to show the benefits of the integration of the commitment framework in a real-time agent. A series of tests to check the proper operation of the TCM have been performed. The first test analyses the behaviour of the TCM as it receives new requests by increasing the number of queries. As shown in figure 4, the number of estimations that the TCM performs decreases as new requests are queried. This demonstrates

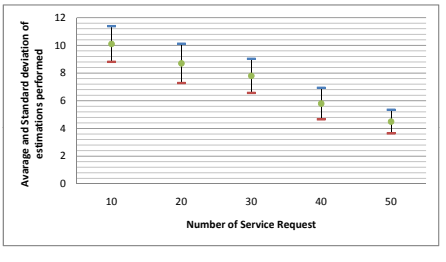

Fig. 4. Average and standard deviation of the number of estimations performed vs number of service requests

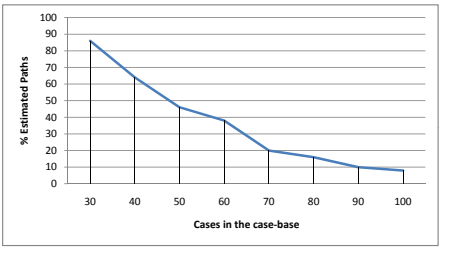

Fig. 5. Percentage of estimated paths in a complete route vs number of cases in the case-base

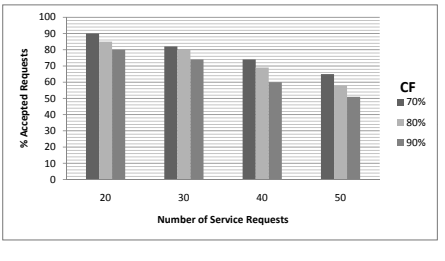

Fig. 6. Percentage of accepted requests vs number of service requests

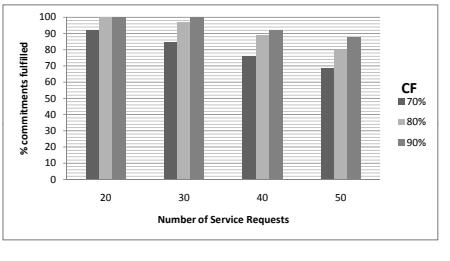

Fig. 7. Percentage of commitments fulfilled vs number of service requests

that as the number of requests increases, the case-base learns properly the new information and hence, the number of routes that can be composed with the cases increases (and an estimation is not necessary). Figure 5, which shows the relation between the number of cases in the case-base and the percentage of estimated routes, also supports this conclusion. Finally, the percentage of distrust from which an agent can commit itself to performing a service was also checked (modifying the confidence values from 70%, 80% and 90%). As expected, bigger confidence percentages resulted in agents committing themselves to performing more tasks (Figure 6). However, in such cases the percentage of services accepted and completed on time decreases, since the agent committed itself to the performance of a big amount of services (Figure 7).

6 Conclusions

A module to analyse whether an agent has enough time to perform a service has been developed. A CBR approach for deciding if an agent can commit itself to performing some service without exceeding the maximum time assigned for performing that service has been used. From now, we have focused our work on the design and implementation of the different phases of the CBR cycle for the mobile robot problem. This work has been tested and evaluated by using a simulated scenario. The results obtained support the expectations.

References

1. Winikoff, M.: Designing commitment-based agent. In: International Conference on Intelligent Agent Technology, pp. 18–22 (2006)
2. Julián, V., Botti, V.: Developing Real-Time Multi-agent Systems. Integrated Computer-Aided Engineering 11(2), 135–149 (2004)
3. Aamodt, A., Plaza, E.: Case-based reasoning; Foundational issues, methodological variations, and system approaches. AI Comm. 1(7), 39–59 (1994)
4. Dean, T., Boddy, M.: An analysis of time-dependent planning. In: Proceedings of the seventh National Conference on AI, pp. 49–54 (1988)
5. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basic for the heuristic determination of minimum cost paths. IEEE Transactions on SSC 4, 100–107 (1968)
6. Navarro, M., Julián, V., Soler, J., Botti, V.: jART: A Real-Time Multi-Agent Platform with RT-Java. In: Proc. 3rd IWPAAMS 2004, pp. 73–82 (2004)
7. RTJ: The Real-Time for Java Expert Group, `http://www.rtsj.org`
8. Webots: `http://www.cyberbotics.com/`