The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

A Probabilistic Action Duration Model for Plan Selection and Monitoring

V. A. Ziparo, L. Iocchi, M. Leonetti, and D. Nardi

Abstract— **The execution of tasks for a robotic agent embedded in a dynamic environment brings about several challenges, due to unpredictable (or unobservable) events, and to inaccurate perception. Moreover, the agent can perform multiple tasks and each task can be achieved by applying different plans, therefore the decision about which strategy is the most convenient, given the current situation of the world, is important for assessing an intelligent overall behavior of the agent. This paper tackles the problem of on-line execution monitoring in a novel way with respect to previous work, since: 1) it considers uncertainty in the duration of actions with a probabilistic model of action duration; 2) it evaluates the cost of each possible plan at run-time in terms of probability of successful termination within a desired expected time. The approach has been evaluated both in a robotic soccer and a surveillance scenario.**

I. Introduction

There are great demand and expectations for the new intelligent robots. Unfortunately, while the specific features of robots are improving at a good pass, the embodiment of intelligent behavior is still achieved in very ad hoc ways. The AI symbolic approaches to Cognitive Robotics, although in principle targeted towards robotic systems, often fail to show substantial improvements when they are experimentally evaluated on robots (truth to be told, not very often). We believe that this state of affairs is caused by a mismatch between planning and robot control. Robots' plans are often simple, but rather tricky to formalize in terms of conditions and state properties that must be verified by the perceptive system. Robot control, at the level of behaviors, does not suitably scale to complex intentional goals that require to look-ahead in order to successfully pursue intelligent objectives.

The relationship between planning, or reasoning about actions, and robot control is usually defined in the so-called *execution monitoring* component of a deliberative architecture [1], [2]. There is a whole spectrum of approaches in the design of *execution monitoring*, ranging from the simple "select next action in the chosen plan until a failure is reported", to "on-line planning", where a new plan is devised at each step and the first action in the plan is executed. While the former approaches are applicable when the environment allows a correct and successful execution of robot's actions, the latter are often not practical since they can easily become computationally infeasible.

A key issue in the design of cognitive robots, that has been extensively addressed by the researchers, is the duration of actions, which seems to be needed in order to face the uncertainties that are typical of the execution of actions by robots. The explicit representation of time, however, typically involves a too fine level of description [3], which is not well suited for the abstractions that are used to devise plans that involve high level actions or behaviors. Moreover, estimating the duration of actions at planning time is often implausible.

Based on the above considerations, we propose an approach for execution monitoring which uses a probabilistic representation of the duration of actions, in order to properly evaluate the chance of a successful execution of a plan within a given time limit. Unlike the previous approaches for planning under temporal uncertainty and with continuous resources, the actions' duration is considered only at execution time, when a library of plans is assumed to be available and designed or generated off-line. A temporal analysis of the utility of plans allows for on-line estimation of the probability of successful plan execution, based on the robot perception of the environment.

The proposed model does not require a specific action representation, nor an ad hoc planning approach; rather, it is inspired by a BDI (Belief Desires Intentions) architecture [4], where a set of plans are available for on-line plan selection. More specifically, we devise an execution monitoring algorithm that, in order to choose the next action to be executed, evaluates the cost of each possible plan in terms of the probability of successful termination within a desired expected time.

II. Related work

Execution monitoring is an essential activity for robust action execution in robotics due to the intrinsic uncertainty of the real world. Uncertainty stems from different sources among which: partial observability, sensor noise, actuator failures and unpredictable exogenous events (e.g. caused by other agents).

Traditionally, monitoring methods can be categorized as either *model-based* or *model-free*. In *model-based* methods, expectations about the future state of the environment are generated from a model which can be analytical [5], [6] or knowledge-based [7]. *Model-free* approaches, on the other hand, compute statistics by directly exploiting data [8]. The present paper is more concerned with *model-based* methods monitoring at the action or plan level, rather than the analytical approaches developed in control theory.

Execution monitoring has been investigated within the framework of both planning [1] and cognitive robotics [2]. We specifically address the temporal knowledge about the

The Authors are with the Department of Computer and System Sciences, Sapienza University of Rome, 00185 Rome, Italy {lastname}@dis.uniroma1.it

duration of actions, in order to enrich the capabilities of the system to predict the successful and timely evolution of the plan execution. On the other hand, we do not want to introduce an explicit representation of time [3], since it requires a very fine level of description that is difficult to achieve a priori. There has been some work in modeling durative actions with continuous change [9], although the approach assumes deterministic duration of actions. Given the uncertainty associated with the duration of robotic complex actions, we argue that this aspect should be modeled by a probabilistic representation.

Regarding probabilistic time models and computation of a probability density function for plan durations, some work has been developed in the framework of MDPs [10]. The computation, however, is carried out off-line and there is no execution monitoring process.

Time is generally not explicitly addressed by behaviour-based approaches, but it is implicit in some notions that are used to model the activity of robots. In particular, *impatience* and *acquiescence* in ALLIANCE [11] are used to detect failures, and switch behaviors, by taking duration into account. While the aim is very closely related to execution monitoring, time is not explicitly represented and the approach is obviously focused on behaviors.

Execution monitoring is also addressed by the work on deliberative architectures, which specifically deals with the problem of selecting goals to be pursued and plans to achieve them. Plans are typically not generated during the execution, but available in a plan library. In particular, in BDI architectures [4], intention pursuing is a process that needs to be monitored in order for the agent to realize when an intention is not achievable: in this case, it must reconsider desires and commit to a different one. The problem of checking execution progress and, if necessary, drop some intentions is a key aspect of BDI and is known as *intention reconsideration* [12], [13]. The proposed solutions to this problem span over different *degrees of boldness*: from a *bold* agent that never reconsiders its intentions until it fulfills them, to *caution* agents that reconsider the intentions after every action. In the cited work by Kinny and Georgoff, that was later extended by Schut and Wooldridge [14], it is argued that the need for intention reconsideration depends on the dynamics, observability, and stochasticity of the environment.

Considering the relevant literature on execution monitoring, our approach generalizes the previous work by providing a framework to deal with on-line uncertainty about the duration of actions, that is completely decoupled from (i.e., does not pose any constraint on) the underlying planning formalism.

III. Probabilistic representation of action duration

In this section, we describe a probabilistic representation of the duration of actions. The duration of an action α is denoted by a χ^2 probability distribution [15] starting at time t_0 and with σ degrees of freedom:

$$\chi^2(t-t_0|\sigma) = \begin{cases} \frac{1}{2^{\sigma/2}\Gamma(\sigma/2)} x^{(\sigma/2)-1} e^{(t_0-t)/2} & \text{for } t > t_0 \\ 0 & \text{for } t \leq t_0 \end{cases}$$

where $\Gamma(\cdot)$ denotes the Gamma function. Figure 1 shows some examples of χ^2 distributions.

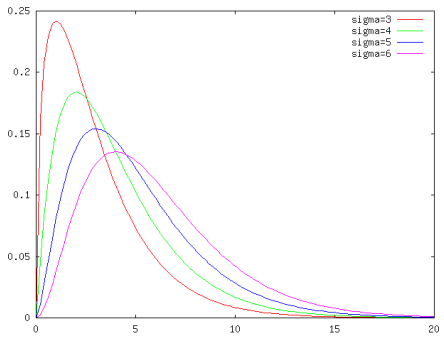


Fig. 1. Examples of χ^2 distributions, when $t_0 = 0$ and σ varies from 3 to 6.

The motivation underlying the choice of the χ^2 distribution for modeling action durations is manifold:

- it can model the fact that the execution of an action for a robot requires at least some minimum time (i.e., t_0), which is a strictly positive quantity. This marks an important difference with the Normal distribution that would assign a non-zero probability to an unrealistic negative time;
- most of the distribution can be found in a small range of time values, depending on σ, which describe the duration of the action under normal conditions;
- the distribution has a tail for $t \to \infty$ which captures the duration of the action under abnormal conditions: actions can take, with low probability, a long time if things go wrong.

Moreover, χ^2 distributions have nice computational properties, which allow to implement the procedures described in the following in a very efficient way. In particular, they are closed under convolution, they can be described by two parameters, and have a closed form for computing convolution and cumulative distribution functions.

As described in the next sections, the evaluation of the probability distribution of the action duration will be performed on-line, during the execution of the plan. Thus, the parameters σ and t_0 are determined on-line, at the beginning and during the execution of the action itself, depending on the current situation perceived by the robot.

A. Probabilistic evaluation of plan duration

Given a temporal characterization of action duration, we would like to infer the probability that a plan terminates within a given deadline T_{max} from the current time t_c. The probabilistic evaluation of plan duration is obtained by combining the probability distributions of action duration with the plan operators. In this paper, we consider plans

only as sequences of actions, although the formalism can be extended to consider more complex structures, such as conditional plans with sensing actions.

The duration of a plan is, thus, computed by combining the duration of the actions in the sequence. The probability distribution of the duration of a plan can be defined as follows. Given a plan Π_i of a plan library Π and an action α, the probability distribution of the plan $\{\Pi_i; \alpha\}$ (i.e., the sequence of Π_i and α) is computed with a convolution of the two probability distributions associated to Π_i and α:

$$p(t|\Pi_i; \alpha) = p(t|\Pi_i) * p(t|\alpha)$$

The probability distribution χ^2 is closed under convolution, thus, the distribution describing a plan duration is again a χ^2 distribution. In particular, given two χ^2 distributions $\chi^2(t - t_1|\sigma_1)$ and $\chi^2(t - t_2|\sigma_2)$, their convolution can be easily, and efficiently, computed as $\chi^2(t - t_1 - t_2|\sigma_1 + \sigma_2)$.

This probability distribution can be used to evaluate the probability of successful termination of the plan starting from current time t_c to a deadline T_{max}:

$$P(t_{end}(\Pi_i, t_c) < T_{max}) = \int_{t_c}^{T_{max}} p(t|\Pi_i)dt \qquad (1)$$

where $t_{end}(\Pi_i, t_c)$ is the random variable denoting the time of termination of the plan Π_i starting from the current time t_c. Computing Equation 1 amounts to compute the cumulative distribution function (cdf) of $p(t|\Pi_i)$. Given that we know that $p(t|\Pi_i)$ is a χ^2 pdf, its cdf is computed as follows:

$$cdf(t_c, \sigma) = \Gamma_r((t - t_c)/2, \sigma/2)$$

The function Γ_r is known as the regularized Gamma function and is available in tabular form in many statistical packages.

Note that the desired maximum execution time for a plan T_{max} depends both on the goal achieved by the plan G_{Π_i} and on the current situation in which the robot is, represented by its current *execution state* S_c. Such a state will be much richer than the corresponding state in the plan, because it will include information gathered at execution time. Therefore, we write this term as $T_{max}(G_{\Pi_i}, S_c)$ and define the probability of successful termination of plan Π_i from the current execution state S_c as:

$$\Lambda(\Pi_i, t_c, S_c) = P(t_{end}(\Pi_i, t_c) < T_{max}(G_{\Pi_i}, S_c))$$

IV. Utility of a plan

In this section, we present a method to derive an expected utility of a plan, which depends on the execution state of the robot and the expected duration of the actions of the plan. In particular, we want to define the utility of a plan to achieve a goal by considering two terms:

1) $U(G_{\Pi_i}, t_c, S_c)$: the utility of achieving the goal G_{Π_i}, given the robot's execution state S_c at time t_c;
2) $\Lambda(\Pi_i, t_c, S_c)$: the probability of successfully terminating the plan within a given deadline T_{max}, given the robot's execution state S_c at time t_c.

It is important to notice that these two terms depend on both the current time t_c and the robot's execution state S_c; hence, they vary during plan execution either because the execution state changes or just because time flows.

The expected utility of executing plan Π_i, given S, is defined as:

$$U(\Pi_i, t_c, S_c) = U(G_{\Pi_i}, t_c, S_c) \cdot \Lambda(\Pi_i, t_c, S_c)$$

During the execution of the robot task, and consequently the evolution of the current state S_c, utility values of all the plans in the library vary because of changes in the time variables t_c and T_{max}, as well as in the parameters of the χ^2 function regulating the actions in the plans. This is an important feature of our approach, and special care must be taken when applying this computation to the plan that is currently under execution, because for the current plan we want to measure also its progress towards the goal. Therefore a different utility function is needed for the current plan: it is denoted with $U^*(\Pi_{\bar{k}}, t_c, S_c)$ and is computed by considering time variables, but not updating the χ^2 functions for the actions under execution. In this way the evaluation of the current plan takes also into consideration its progress towards the goal.

Example Consider a soccer robot r which is facing an opponent o. The ball is between the two robots, and r has to evaluate if it is worth to pursue an attack goal. For example, the utility of attacking at some time $t_c = 0$ $U(G_{Att}, 0, \langle g_o, g_r \rangle)$ could be: the difference of goals scored by the two teams (i.e., $g_o - g_r$). The utility function tells the robot that the more he is losing, the more he should attack (if he is winning he should rather defend). The plan attack could be a sequence of two actions/behaviors:

1) `goToBall` which moves the robot towards the ball. Given the current distance of r from the ball d_b, the average speed of the robot s_r, the estimated duration of the action is computed as follows: $t_0 = c_{t_0} \cdot \frac{d_b}{s_r}$ and $\sigma = c_\sigma \cdot d_b$, where c_{t_0} and c_σ are two parameters, which can be automatically learned or empirically determined.
2) `kick`, which kicks the ball towards the goal when the robot reaches it. Given that the kick action takes more or less the same amount of time in any situation, we can characterize its duration by the two constants: $t_0 = c_{t_0}$ and $\sigma = c_\sigma$.

Now, consider the case in which the robot r is losing 2 to 1 and it estimates at time $t_c = 0$ that the opponent will reach the ball in $T_{max} = 30s$. Moreover, his moving speed is of $1\,m/s$ and the ball is $10m$ away. For simplicity, also assume that all the c_{t_0} constants have a value of 1, and the c_σ constants have a value of .5. In this case, the utility of Attacking is

$$U(Att, 0, s) =$$

$$U(G_{Att}, 0, \langle 2, 1 \rangle) \cdot \int_0^{30} \chi^2(t - 10|5) * \chi^2(t - 1|.5) =$$

$$\int_0^{30} \chi^2(t - 11|5.5) = \Gamma_r(5.5/2, 11/2) = 0.93174$$

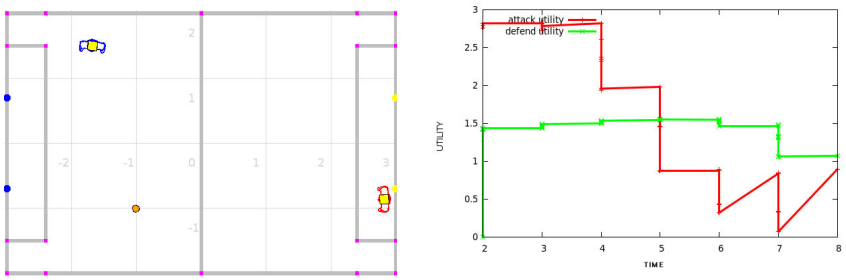


Fig. 2. Example of an early plan switch.

V. On-line execution monitoring

The on-line execution monitoring method described in this paper is based on a probabilistic representation of action duration, and on evaluation of the utility of plans by considering both the utility of goal achievement and the probability to successfully complete the plan within a desired maximum time. The probability distribution of the duration of a plan is continually evaluated on-line by the robot, based on the current robot execution state, and does not require any re-planning or plan repair procedure. Thus, during execution, we monitor the utility of plans in order to dynamically decide which plan must be executed.

We assume the system to be composed of at least two modules, running in different threads: the plan monitor and the plan executor. The plan executor by default runs the empty plan $\emptyset$, which has a utility $U(\emptyset, \cdot, \cdot) = -\infty$ independently of the robot's execution state. Asynchronously and continually, the plan monitor checks the plan library Π to select the most appropriate plan for execution, and communicates it to the plan executor. Given a non-empty plan, the plan executor selects the next action in the plan and activates the corresponding behavior. During execution, it checks for action termination until it can move to the next action or until it receives a new plan from the execution monitor. Notice that the knowledge represented by the robot's execution state is collected asynchronously, with respect to the plan executor and the plan monitor.

Algorithm 1 describes in detail the plan execution monitor. The procedure keeps checking the internal state S of the robot (line 3), in order to find the set of executable plans $\Pi^\star$(line 4), i.e. the plans which are applicable given the current execution state. Then, it computes the utility of each plan and selects the one with the highest utility (line 5). Notice that this step requires to recompute the parameters t_0 and σ of the χ^2 distribution according to the execution state of the robot. The computation of the parameters depends on the implementation of the actions, and it is, in general, domain dependent. Once the best plan is found, if this is not the current one, a further step is required: checking whether it is worth to switch from the current plan $\Pi_{\bar{k}}$ to another plan Π_{best} in the current situation S, considered the cost $(CS(\Pi_{\bar{k}}, \Pi_{best}, S))$ of changing plan (line 6). If it is worth to change the plan, then the current plan is updated with the new plan (line 7), which is then communicated to the plan executor (line 8).

Algorithm 1 PLAN EXECUTION MONITOR

Variables:

Π : plan library
$\Pi_{\bar{k}} \in \Pi$: current plan being executed
t_c : current time
S_c : current state of the robot

procedure $PlanMonitor(\Pi, t_c)$

1: $\Pi_{\bar{k}} = \emptyset$
2: **while** $True$ **do**
3: $S_c = getCurrentState(t_c)$
4: $\Pi^\star = getExecutablePlans(\Pi, S_c)$
5: $\Pi_{best} = \mathbf{argmax}_{\Pi_k \in \Pi^\star \wedge k \neq \bar{k}} U(\Pi_k, t_c, S_c)$
6: **if** $U(\Pi_{best}, t_c, S_c) - CS(\Pi_{\bar{k}}, \Pi_{best}, S_c) > U^*(\Pi_{\bar{k}}, t_c, S_c)$ **then**
7: $\Pi_{\bar{k}} = \Pi_{best}$
8: $setCurrentPlan(\Pi_{\bar{k}})$
9: **end if**
10: **end while**

VI. Experimental evaluation

In this section we provide an experimental analysis of our execution monitor, using a simulated robotic environment implemented with the Player/Stage simulator. For each scenario, we will describe the experimental setting, an example used to highlight some features of the proposed method, and some experimental results on the use of our method in the mentioned robotic scenario. The execution times are not reported because these are always negligible. De facto, the computation amounts to a linear (in the length of the plan) number of sums and multiplications. This is possible because the χ^2 distribution allows for closed form computations. Such low computational costs guarantee an effective real-time monitoring.

A. Robotic soccer scenario

As a first scenario, we consider two simulated soccer robots that compete for reaching the ball and kicking it in the opponent goal. We consider two plans, Π_1 (`Attack`) and Π_2 (`Defend`):

1) Π_1= {`goToBall`; `grab`; `rotateTowardsOpponentGoal`; `shoot`}.

	Match 1	Match 2	Match 3	Match 4	Match 5
Ball distance [m]	0.74 - 0.79	0.89 - 0.93	0.94 - 1.14	1.07 - 1.21	0.89 - 1.06
Ball possession [%]	58 - 42	49 - 51	53 - 47	47 - 53	54 - 46
Opponent half [%]	40 - 60	83 - 17	75 - 25	88 - 12	78 - 22
Kicks	80 - 11	109 - 16	162 - 32	122 - 37	133 - 29
Score	4 - 3	8 - 2	11 - 3	9 - 8	10 - 3

TABLE I

RESULTS OF 5 TPEM VS. OTHER SIMULATED SOCCER MATCHES.

2) Π_2= {`goBetweenBallAndOwnGoal`; `standInDefensivePosition` }

The robots use the same set of actions with the same parameters, and they differ only in the plan execution monitoring: one agent (the blue robot) uses the temporal plan execution monitoring, while the other (the red robot) uses an execution monitoring based on utility functions not depending on time.

a) Example: early plan switch: We consider the situation in which the two robots are both approaching the ball from different directions (see Figure 2). Although the blue robot is closer to the ball, it is slower than the red one (but it does not know it). The temporal plan execution monitoring is used to evaluate whether and when it is the case to interrupt the `Attack` plan, because it will not reach the ball before the opponent, and to switch to the defensive plan, in order to save the goal. The utility of the two plans is shown in the lower graph in Figure 2. We repeated this experiment in two cases: 1) using the temporal plan execution monitor; 2) using a static utility function based only on ball and opponent player distances, but not on time. Notice how the blue robot is able to promptly understand that it is more convenient to switch to the plan `Defend`, because of the combined effect of the dynamic update of the deadline T_{max} and the monitoring of elapsed time. In this experiment, we have measured that the temporal plan execution monitor allows for detecting the utility of switching plans 3 seconds before the standard method. Therefore, in this case, a robot not using an appropriate plan execution monitoring can understand too late that attacking is not as effective as defending, losing in general the opportunity of saving a goal.

b) Experiment: one vs. one soccer match: In order to provide also a quantitative analysis of the results obtained with the temporal plan execution monitoring, we have devised some performance metrics for the soccer task and we repeated the experiments several times to measure such variables. In particular, we simulated a one vs. one soccer match in our experimental setting with two robots: one making use of temporal plan execution monitoring, and one with a fixed non-temporal monitor. Each simulated game was run for 10 minutes and we have measured for each robot: 1) the average distance to the ball, 2) the percentage of ball possession, 3) percentage of time with the ball in the opponent half, 4) the number of kicks, 5) the score.

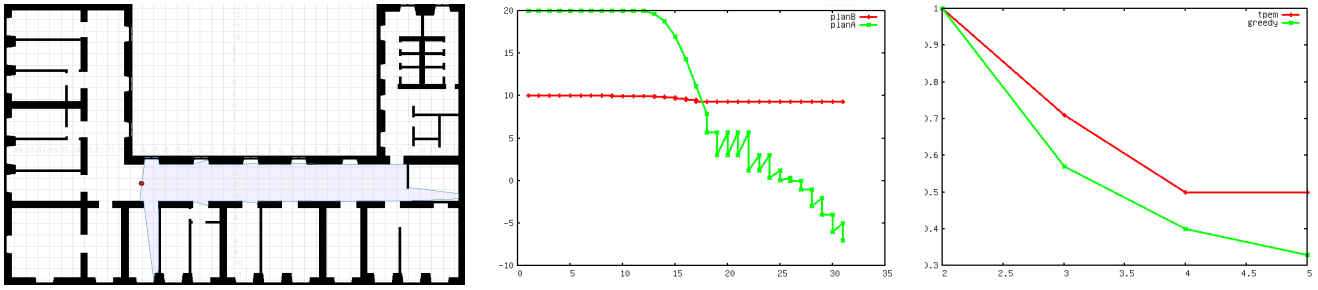
Table I presents the results of 5 experiments, i.e. five 10 minutes matches. The results clearly show a slight advantage of the robot using the temporal analysis in ball distance, ball possession, and opponent half, that is justified by the fact that the two robots were performing the same actions with the same parameters; but also a significant improvement in the number of kicks (and in the score) that shows that even a small amount of time gained to take the right decision can result in a significant improvement in the robot performance.

B. Robotic surveillance scenario

In robotic surveillance robots operate in an environment in order to monitor it and to detect interesting events. In this scenario, we consider a robot that can receive a set of target points with different priorities in a known environment and can reach these points for taking some information (e.g., photos) to be reported to some human surveillance center. Nevertheless, the robot has a limited battery time and may require more time than expected to complete tasks, because of unpredicted events (obstacles, such as chairs in the way) or navigation failures. In this case, plans are considerably longer than the previous case study. In practice, they are paths of an (internal) topological representation of the environment. The Player/Stage environment simulation is depicted in Figure 3(a).

c) Example: identifying failure: Let's consider a situation where a robot, while approaching a high valued target, finds an obstacle (for example, something that blocks its wheels) on its way, but it cannot detect the obstacle with its on-board sensors. The plan execution monitoring continually evaluates the utility of the plans (shown in Figure 3(b)), and is able to understand that it is making no progress. The green line shows how the utility of the most valuable target decreases over time, until the utility values become smaller than the ones of the secondary target (red line). Thus, in this case, the monitor correctly identifies that the current plan is making no progress and switches to a target with lower utility, but with higher chances of success. It is important to observe that, without sensing the failure situation or without an appropriate temporal plan execution monitoring, a robot would not be able to solve this situation and to escape from the stall.

d) Experiment: multi-target surveillance: In order to provide quantitative data we compare our approach with a greedy selection method which at any task completion selects the target with the highest utility. In each episode, the robot has two minutes to complete as many tasks as it can obtaining the corresponding utility along the way. Figure 3(c) shows a comparison of TPEM vs the greedy approach, varying the number of target locations available. The results measure the average ratio between the utility



(a) (b) (c)

Fig. 3. (a) Surveillance scenario, (b) Recognition of no progress in a plan , and (c) Experiments on the domain, reporting the number of tasks accomplished (y axis) over the given time (x axis).

actually gathered and the total utility available if all the tasks were accomplished (y axis), for each number of locations available (x axis). Two minutes are enough to complete any couple of tasks, therefore both algorithms achieve the highest possible result independently of the order in which those are selected. However, as the number of locations increases, the choice becomes more and more relevant and taking time into account proves to be critical. The results clearly show that time execution monitoring yields to consistent improvements in the performance of the robot, especially as the time constraint becomes tighter.

VII. Conclusions

The uncertainty associated with the execution of actions by robotic platforms is very high and the clean, abstract, representations of robot plans do not lead to satisfactory performance.

In this paper, we have presented an approach to execution monitoring that introduces a probabilistic representation of action duration, specifically aiming at monitoring the successful execution of robot plans. We have chosen to deal with the temporal analysis of actions at execution time, rather than at planning time, and therefore the approach can be applied independently of a specific approach to plan design/generation. Moreover, the proposed execution monitoring can be embodied in a BDI architecture, where it combines intention selection and plan selection. In this respect, it specifically addresses some of the challenges that have been raised in the dynamic reconsideration of intentions.

The proposed approach can also be extended and generalized in directions that are not investigated in the present paper. One issue that might be considered is the choice of the probability distribution for representing the uncertainty of action duration. Another aspect that might be addressed is the use of plan structures including hierarchical plans or conditional plans. The proposed approach could also be applicable to monitor the execution of tasks in multi-robot systems, where the temporal analysis of the plan, chosen to accomplish a given task, may be used to realize that one robot can no longer effectively pursue the task assigned to it, and cause a change in the task assignment. Finally, we aim at addressing the use of learning techniques to produce execution time estimation of action duration.

References

[1] K. Z. Haigh and M. M. Veloso, "Interleaving planning and robot execution for asynchronous user requests," in *Autonomous Robots*, 1996, pp. 148–155.

[2] G. D. Giacomo, R. Reiter, and M. Soutchanski, "Execution monitoring of high-level robot programs," in *KR*, 1998, pp. 453–465.

[3] R. Reiter, "Sequential, temporal golog," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, Trento, Italy, 1998, pp. 547–556. [Online]. Available: reiterkr98.pdf

[4] M. Bratman, *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[5] R. Isermann, "Estimation of physical parameters for dynamic processes with application to an industrial robot," in *Proceedings of the American Control Conference*, 1990, pp. 1396–1401.

[6] W. Dixon, I. Walker, D. Dawson, and J. Hartranft, "Fault detection for robot manipulators with parametric uncertainty: a prediction-error-based approach," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 689–699, Dec 2000.

[7] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Semantic knowledge-based execution monitoring for mobile robots," *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3693–3698, April 2007.

[8] O. Pettersson, L. Karlsson, and A. Saffiotti, "Model-free execution monitoring by learning from simulation," *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*, pp. 505–511, June 2005.

[9] J. Claßen, Y. Hu, and G. Lakemeyer, "A situation-calculus semantics for an expressive fragment of pddl," in *Twenty-Second Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.

[10] J. Marecki, Z. Topol, and M. Tambe, "A fast analytical algorithm for Markov decision process with continuous state spaces," in *Proceedings of the Eight Workshop on Game Theoretic and Decision Theoretic Agents (GTDT) held at the Fifth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, May 2006, pp. 2536–2541.

[11] L. E. Parker, "Lifelong adaption in heterogeneous multi-robot teams: Response to continual variation in individual robot performance," *Autonomous Robots*, vol. 8, no. 3, pp. 239–267, 2000.

[12] D. N. Kinny and M. Georgeff, "Commitment and effectiveness of situated agents," in *In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91*, 1991, pp. 82–88.

[13] M. Wooldridge and S. Parsons, "Intention reconsideration reconsidered," in *Intelligent Agents V (LNAI Volume 1555*. Springer-Verlag, 1999, pp. 63–80.

[14] M. Schut and M. Wooldridge, "Intention reconsideration in complex environments," in *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*. New York, NY, USA: ACM, 2000, pp. 209–216.

[15] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, ninth dover printing, tenth gpo printing ed. New York: Dover, 1964.