Optimal Planning for Autonomous Agents under Time and Resource Uncertainty

Aurélie Beynier, Laurent Jeanpierre, Abdel-Illah Mouaddib
Bd Marechal Juin, Campus II
BP5186
14032 Caen Cedex, France
{abeynier, laurent, mouaddib}@info.unicaen.fr

Keywords:

Abstract: In this paper we develop an approach for optimal planning under time and resource uncertainty with complex task dependencies. We follow the line of research described by (Bresina et al., 2002), overcoming limitations of existing approaches: they only handle simple time constraints and they assume a simple model of uncertainty concerning action durations and resource consumptions. In many domains such as space applications (rovers, satellites), these assumptions are not valid. We present an approach that considers temporal and resource constraints and deals with uncertainty about the durations and resource consumptions. From an acyclic mission graph, we automatically build a Markov Decision Process that can be optimally solved by dynamic programming. Experimental results prove that this approach allows for considering large mission graphs.

1 Introduction

In this paper we develop an approach of planning under time and resource uncertainty along the research lines described in a challenge paper of (Bresina et al., 2002). In that paper, authors claim that existing methods fail because they suffer from many limitations where some of them consist of: (1) they only handle simple time constraints, (2) they assume a simple model of uncertainty concerning action durations and resource consumption. In many domains such as space applications (rovers, satellites), these assumptions are not valid. We present an approach that relaxes those assumptions by using a model of tasks with complex dependencies, uncertain execution time and probabilistic resource consumption. In this approach, we consider the following: (1) a mission is an acyclic graph of tasks that expresses complex dependencies between tasks, (2) a task has a temporal interval during which it can be executed, (3) durations and resource consumptions of tasks are uncertain. This class of problems is found in some rover scenarios where the objective of the rover is to maximize the overall value of the mission: given the mission graph, we want the mission to be executed optimally by an autonomous agent.

The major objectives of future space missions (Estlin et al., 1999; Bresina et al., 2002) will consist of maximizing science return and enabling certain types of science activities by using a robust approach. To show the significance of problems we deal with, let us give some examples of robotic vehicles where we express the different constraints we consider in this paper:

- *Time windows*: a number of tasks need to be done at particular but approximate times: for example "about noon", "at sunrise". There is no explicit time window but we can represent these using time windows or soft constraints on a precise time. Examples include atmospheric measurements (look at the sun through the atmosphere, must be done at sunrise, sunset, and noon). Since the rover will be power-constrained, most of the rover's operations will occur between 10:00 and 15:00 to make sure that there is enough sunlight to operate. Driving will certainly happen during that part of the day. Communication also has to start at a particular time since this requires synchronization with Earth. There are operations that cannot be done outside of a time window – night-time operation of cameras or daytime operation of the night-time spectrometer. Other constraints can be considered such as illumination constraints (which involves time of day and position), setup times (warm-up or cool-down delays for instruments), time separation between images to infer 3D shape from shadows.

- *Bounded resources*: many activities require power (moving from a location to another one) and

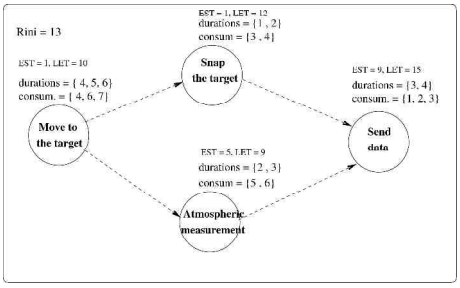

Figure 1: An acyclic graph of tasks

data storage (storing pictures with different resolutions).

In summary, these scenarios share common characteristics such as:

1. Activities have an associated temporal window.
2. Task durations and resource consumptions are uncertain.
3. The temporal constraints are, in general, soft.
4. Precedence dependencies between tasks exist.
5. The mission can be represented by an acyclic graph.

These scenarios are well suited to space-rovers because of the communication delay which prohibits any interactive behavior. For example, communication from earth to mars suffers from several minutes delays, and requires synchronization. Therefore missions generally include a whole workday, that is around one hundred tasks to accomplish. In consequence, mission plans are generally computed on earth, transmitted to the distant agent and then executed autonomously. However, even if we focus mainly on rovers, the algorithms we present in this paper can be applied to any problem formalized as a dependency graph. For example, we could apply them to cooking a dinner; in this case, we would have a recipe to follow, ingredients to buy and mix in the proper order, and choices on which meal to prepare depending on the remaining time before the guests arrival...

Existing works on planning under uncertainty for rovers encounter difficulties in their application to real problems because most of them handle only simple time constraints and instantaneous actions. More sophisticated techniques have been developed to deal with uncertain duration (Tsamardinos et al., 2003; Vidal and Ghallab, 1996; Morris et al., 2001) but they fail to optimally control the execution.

The requirements for a richer model of time and actions are more problematic for those planning techniques (Bresina et al., 2002). The techniques based on MDP and POMDP representations can be used to represent actions with uncertainty on their outcomes (Mouaddib and Zilberstein, 1998; Cardon et al., 2001) but they have difficulties when those actions involve complex temporal dependencies and uncertain durations. Given the temporal constraints of the rover problem, which are not purely Markovian, the approach should handle irregular multi-step transitions.

In this paper, we proposed a formalism that can deal with temporal and resource constraints to represent realistic problems. Our approach deals with uncertainty on task durations and resource consumptions. Like Semi-Markov Decision Processes (SMDPs) (Sutton et al., 1999), it considers different possible durations for each task. We propose to formalize the problem using a MDP that allows for temporal and resource constraints. This MDP is solved optimally by dynamic programming. We demonstrate that problems of a hundred tasks can be considered.

In the next section, we present the important features of the approach based on the construction of an MDP taking temporal constraints and uncertainties into account. This MDP uses a state representation that can express the temporal information needed for the decision-making process.

2 Principles of the approach

Let's consider a planetary rover that have a mission to complete. First, it must move to a target. Then, depending on the time and on available resources, it can snap the target or complete atmospheric measurements. To end its mission, the rover must send the data it has collected. As shown in Figure 1, this small mission can be represented by an acyclic graph. Each node stands for a task t_i and edges represent temporal constraints: the rover must move to the target before it can snap it. "Rini" is the initial resource rate. We consider only "or" nodes : the agent must snap the target or complete atmospheric measurements. In order to send the data, it must have completed one of these predecessor tasks. Temporal constraints are indicated by the Earliest Start Time (EST) and the Latest End Time (LET) of each task.

In the rest of the paper, we assume that the mission graph is given. We will use the terms task, activity, and node interchangeably. A task will be denoted by t_i. Nonetheless, a task t_i differs from an action a: an action consists of a task and a start time.

Each task t_i is characterized by its time window and by the uncertainty about its duration $\delta_i(t_i)$ and about its resource consumption $\delta_r(t_i)$. When it is unambiguous, we will suppress the activity argument t_i for conciseness.

Temporal window of Tasks Each task is assigned a temporal window [EST, LET] during which it must be executed. The execution of the activity (both start and end times) must be included in this interval for the task to succeed.

Uncertain execution time The uncertainty on durations has been considered in several approaches developed in (Mouaddib and Zilberstein, 1998; Zilberstein and Mouaddib, 1999; Cardon et al., 2001). All of them ignore the uncertainty on the start time. We show in this paper how extensions can be considered, taking different temporal constraints into account.

Definition 1 *A probabilistic duration distribution, $P_c(\Delta t) = Pr(\delta_i = \Delta t)$, is the probability that the execution lasts Δt time units.*

Uncertain resource consumption The consumption of resources (energy, memory, etc ...) is uncertain. As the agent has limited initial resources, it must consider resource constraints.

Definition 2 *A probabilistic resource consumption distribution, $P_r(\Delta r) = Pr(\delta_r = \Delta r)$, is the probability that the activity consumes Δr units of resources.*

The representation adopted for these distributions is discrete. We use sets of pairs $(\Delta t, p)$ and $(\Delta r, p)$ where each pair means that there is a probability p that the execution lasts Δt time units or consumes Δr units of resources. We assume that resource consumption and execution time are independent, but this assumption does not affect the genericity of the model (we can use a probability distribution of $(\Delta r, \Delta t)$ such that $P((\Delta r, \Delta t))$ is the probability that the activity lasts Δt time units and consumes Δr resources).

Overview In order to formalize our problem as a MDP, we need further information about the tasks. In fact, we must know the possible execution intervals of each task and their probabilities. Once these information are known, the MDP can be defined and solved.

Our approach can be divided into several steps. An algorithm first propagates the temporal constraints and computes the set of possible time intervals for each task. It also computes the probabilities of these intervals. The MDP is then constructed using these information and solved by the Policy Iteration algorithm (**?**).

3 Temporal interval and probability propagation

The decision process bases its decision on the remaining resources, on the state of the latest executed task and on the execution interval of this task.

Many possible execution intervals exist, and many resource levels can be available for each task, as shown on Figure 2. In order to explore the whole state space, we need to know the entire set of possible execution intervals. Therefore, we developed an algorithm that computes all the possible time intervals from the mission graph, weighted with a probability. This probabilistic weight allows us to know the probability that an activity will be executed during a given interval of time. We first describe an algorithm to compute the possible execution intervals of each task. We then explain how to compute their probabilities.

The set of possible start times is a subset of $\{EST, EST+1, \ldots, LET - \min \delta_i\}$ (EST: Earliest Start Time, LET: Latest End Time). We denote the Latest Start Time as $LST = LET - \min \delta_i$ where $\min \delta_i$ is the minimum duration of the task.

We can compute off-line all the possible end times of an activity's predecessors and consequently compute the possible start times of the activity. The algorithm is similar to the one described in (Bresina and Washington, 2000). The possible execution intervals *I* are determined by a simple forward propagation of temporal constraints in the graph. This propagation organizes the graph into levels such that: l_0 is the root of the graph, l_1 contains all nodes that are constrained only by the root node, ..., l_i contains all nodes whose predecessors include nodes at level l_{i-1} and all of whose predecessors are at level l_{i-1} or before. For each node in a given level l_i, we compute all its possible execution intervals from its predecessors.

- *level* l_0: the start time and the end times of the root node (the first task of the mission) are computed as follows:

  - *start time*: $st(root) = EST(root)$

  - *end times*: $ET(root) = \{st(root) + \delta_i(root), \forall \delta_i(root)\}$

  Possible execution intervals of the root are given by $I = [st(root), et(root)]$ with $et(root) \in ET(root)$. Note that there is potentially a non-zero probability that some end times violate the deadline LET.

- *level* l_i: for each node in level l_i, the possible start times are the times at which the predecessor activities can finish. We first compute all the possible start times, and then we eliminate the start times that do not respect the constraints of earliest start time: $st < EST$, and the latest start time: $st > LST$.

  For each possible start time, we compute the possible end times, accounting for the possible durations of the node. Note that potentially $st(node) + \delta_i(node) > LET(node)$

  Figure 2 gives an example of temporal interval

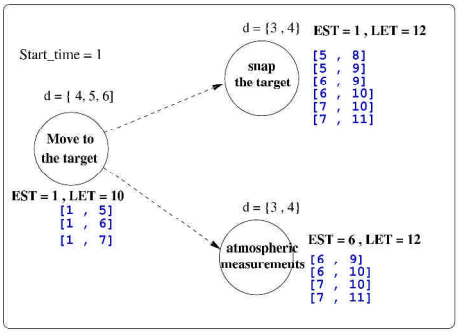

Figure 2: Temporal interval propagation example

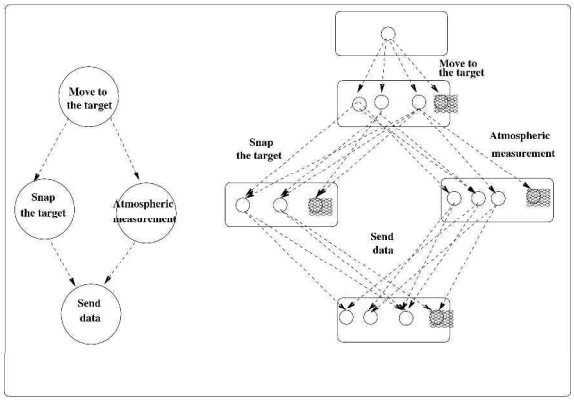

Figure 3: Relationship between the original graph structure and the MDP state space

propagation. Many classical algorithms exist in literature, like PERT, but most of them don't deal with uncertainties on execution time and resource constraints.

As explained in the next section, we must compute the possible execution intervals to define the state space. Probabilities on these intervals must also be known to compute the transition probabilities.

The probability of an execution interval P_w depends on its start time and on the probability of its duration P_c.

A task cannot start before its predecessors finish. Therefore, the probability on the start time depends on the predecessors' end time. We will consider a task t_i. $P_w(I|et(I'))$ denotes the probability that t_i is executed in the interval I when its predecessor finishes at $et(I')$. This value measures the probability that an activity starts at $st(I)$ and ends at $et(I)$.

In order to respect temporal constraints, an agent executes the next task as soon as possible: if it delays the execution of the next task, it will increase the probability of violating the deadline with no expected gain. If $st(I) \geq et(I')$ and there is no possible start time for t_i in $]et(I'), st(I)[$, the agent will start the execution at $st(I)$.

The probability $P_w(I|et(I'))$ of an execution interval I is defined as follows:

- *If $st(I)$ is the first possible start time such that $st(I) \geq et(I')$:* $P_w(I|et(I')) = P_c(et(I) - st(I))$
- *Otherwise:* $P_w(I|et(I')) = 0$

These probabilities are computed off-line during the forward propagation of temporal constraints. By propagating temporal constraints and their probabilities through the mission graph, the algorithm allows for developping the entire state space.

4 A decision model: MDP

As mentioned above, we model this problem with a Markov Decision Process.

Definition 3 *A **Markov Decision Process (MDP)** is defined by a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$:*

- *$\mathcal{S}$ is a finite set of states s*
- *$\mathcal{A}$ is a finite set of stochastic actions a*
- *$\mathcal{P} = \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a Markovian transition function that gives the probability the agent moves to a state s when it executes action a from s'.*
- *$\mathcal{R}$ is a reward function*

To solve our problem, we need to define the state space, the action set, the transition function and the reward function of our model. These components can be deduced from the initial mission graph.

Figure 3 gives a representation of the relationship between the original graph structure, the state space and the transitions of the MDP. The left part of the figure stands for the mission graph. The right part represents the state space of the MDP and its transitions. Each box groups the states associated with a task t_i. Each node stands for a state. An edge represents a transition: it links a state associated to a task t_i with a failure state or with a state associated to a task t_{i+1}, where t_{i+1} is a successor of t_i in the mission graph. The transitions between two states depend on the executed task t_i, its start time, duration and resource consumption. Some nodes have no successor: they are terminal states.

State Space At each decision step, the agent must decide for the execution of a task t_{i+1}. The success of this execution depends on the temporal constraints and the available resources. If temporal constraints are not respected or if the agent lacks resources, the execution of the task fails and the agent moves to a failure state. Otherwise, the execution succeeds and the agent moves to a success state. The decision relies on three parameters: the latest executed task t_i, the

available resources r, and the interval of time I during which task t_i has been executed. A state is therefore defined by a triplet $[t_i, r, I]$. States are fully observable since the temporal intervals are determined off-line and the resource consumption is observed by the agent as an environment feedback.

For each task, we develop a set of states by combining the execution intervals and the remaining resources. Possible resource rates are computed by propagating resource consumptions through the mission graph. The consequence of representing all the intervals and resource levels is that the state space becomes fully observable. The decision process can perform its action selection using the Bellman equation defined below.

Action Space From each non terminal state, the agent should make a decision on which task to execute and when to start its execution. Note that the decision depends only on the current state and thus this process has the Markov property. The set of actions $a = E_i(st)$ to perform consists of *Executing task* t_i at time st, where t_i is a successor of the latest executed task. This action a is probabilistic since the duration and resource consumption of the task are uncertain. Standard MDPs used to represent one time unit actions. Our approach extends this model and allows for representing various possible durations for each action.

Transition model When the agent starts executing an action from a state $[t_i, r, I]$, it can move to various states. If temporal and resource constraints are respected, the execution succeeds. If the agent lacks resources, starts too late or violates the deadline, it moves to a failure state. Failure states are considered as terminal states. It is straightforward to adapt our system to non terminal failure states. States associated with the last task of the mission are also terminal states.

Four kinds of transitions have to be considered:

- *Successful Transition:* The action allows the agent to move to $[t_{i+1}, r', I']$ where task t_{i+1} has been achieved during the interval I', respecting the EST and LET of this task, and r' are the remaining resources. The probability P_{SUC} of moving to the state $[t_{i+1}, r', I']$ is :

$$P_{SUC} = \sum_{\Delta_r \leq r} \sum_{et(I') \leq LET} P_r(\Delta_r).P_w(I'|st(I') = et(I))$$

- *Too late start time (TLST) Transition:* The agent starts too late (after LST). In such a case, the agent moves to a $[failure, r, [st, +\infty]]$ (in fact, the resource and interval arguments are unimportant since the state is terminal). The transition probability P_{TLST} is defined as: $P_{TLST} = Pr(st > LST)$.

- *Deadline met Transition:* The agent starts to execute the task at time st but the duration δ_i is so long that the deadline is met. This transition moves to the state $[failure, r, [st, +\infty]]$. The probability P_{DMT} of moving to this state is: $P_{DMT} =$

$$Pr(st \leq LST) \cdot \sum_{\Delta r < r} \sum_{LET - t_c < st < LST} P_r(\Delta_r) \cdot P_c(t_c)$$

- *Insufficient resource Transition:* The execution requires more resources than available. The agent moves to a state $[failure, 0, [st, +\infty]]$. The probability P_{IRT} of moving to this state is: $P_{IRT} = \sum_{\Delta_r > r} P_r(\Delta_r)$

Reward function Each time the agent finishes to execute a task within temporal, precedence and resource constraints, it obtains a reward which depends on the executed task. R_i is the reward the agent obtains when it has accomplished the task t_i

As the number of tasks and start times are finite, the horizon of the MDP is finite. Moreover, there is no loop in the mission graph: we cannot have "A must be executed before B, B must be executed before C and C must be executed before A". Then, the MDP have no loop.

MDP resolution Once the MDP is defined, it can be solved optimally. To compute an optimal policy for the agent, we use dynamic programing and Bellman principle of optimality. A policy is a mapping from states to actions: for each state it dictates which action to execute. It allows the agent to execute its tasks autonomously. The value of each state relies on the immediate gain (reward) and the expected value that takes into account the various possible transitions. Each state is valued as follows:

$$V([t_i, r, I]) = R_i + \max_{\mathsf{E_k(st \geq et(I)), k=successor(t_i)}} (V')$$

where V' is the expected value of future tasks. We decompose V' as the:

$$V' = V_{SUC} + V_{FAIL}$$

such that:

- *Expected Value of successful transition:*

$$V_{SUC} = P_{SUC} \cdot V([t_{i+1}, r - \Delta_r, I'])$$

This value is used when the task starts and finishes with success: there is enough resources and I' is the execution interval of the next activity where its start time $st(I') \geq et(I)$ of the last executed task.

• *Expected Value of failure transition:* Otherwise, the execution fails because the agent lacks resources or temporal constraints are violated.

$$V_{FAIL} = P_{FAIL} \cdot V([\textit{failure}, r, [st, +\infty]])$$

where $P_{FAIL} = P_{TLST} + P_{DMT} + P_{IRT} = 1 - P_{SUC}$

The policy π of a state $[t_i, r, I]$ is given by:

$$\pi([t_i, r, I]) = R_i + \arg \max_{\mathsf{E_k(st=et(l)),k=successor(a_i)}} (V')$$

Optimality The MDP is easily solved using the standard dynamic programming algorithm *policy iteration*. Therefore, the obtained policy is optimal. Precedence constraints allows for ordering the policy revision: the tasks are considered from the leaves to the node. There is no need to iterate the algorithm since there is no loop.

5 Experimental Results

The proposed model overcomes the difficulties we described in the introduction by proposing a rich model that can deal with complex temporal constraints, limited resources, and uncertainty. However, we need to determine the scale of the problems we can solve. The objective as described in (Bresina et al., 2002) for this experiment is to overcome the problem with a hundred tasks. In the following, we demonstrate that our approach is powerful enough to deal with the hundred tasks required by robotic applications (we can deal with more).

One iteration of policy iteration is polynomial in the number of states and actions. The number of actions depends on the number of tasks and on the number of possible start times for each task. In the worst case, we have:

$$\#n_{actions} = \#n_{tasks}.\#n_{start\_time}$$

(Boutilier et al., 1999) asserts that MDPs composed of one million states can be easily solved. We have therefore studied the state space size of the MDPs we generate. A benchmark composed of several missions of various sizes has been used. For instance, the mission composed of four tasks is presented by Figure 1.

We have computed the state space size of each mission. It relies on:

- the number of tasks
- the number of intervals for each task
- the number of resources available after the execution of each task

These parameters affect the complexity of the problem. In the worst case, the state space size is given by the following equation:

$$\#n_{states} = \#n_{tasks}.\#n_{Max\_Interv}.\#n_{Max\_Res}$$

where $\#n_{tasks}$ is the number of tasks of the mission, $\#n_{Max\_Interv}$ is the maximum number of intervals per task, $\#n_{Max\_Res}$ is the maximum number of resource levels per task.

Number of tasks : As it can be seen in the above equation, if the number of tasks increases, the state space size grows. Figure 5 illustrates this evolution.

Number of intervals for each task : In the worst case, the number of intervals for a task is given by:

$$\#n_{Interv} = (LET - min\ \delta_i - EST).\#n_{durations}$$

where $\#n_{durations}$ is the number of possible execution durations for the task.

If the temporal constraints are tighter, the temporal window [EST, LET] is reduced. Then we compute fewer intervals and the number of states decreases. When we increase the size of the temporal windows, the state space grows: looser temporal constraints allows for more possible plans, involving new states. Figure 4 gives an example of this evolution, considering a graph of one hundred tasks. 100% is the initial size of the temporal windows; 200% stands for temporal windows twice larger than the initial ones. On Figure 4, the state space size rises and then levels off at 150%. Indeed, as temporal windows become larger and larger, temporal constraints relax more and more. At 150%, temporal windows don't constraint any more the execution of the mission. All the possible plans (and possible states) are considered; the maximum number of states is reached. Keeping on relaxing the constraints doesn't increase the state space size anymore. The tighter the constraints are, the less execution intervals are possible and the smaller the state space is.

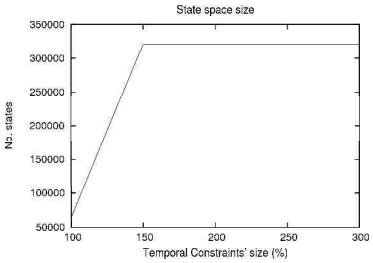

Figure 4: State space size while relaxing the temporal constraints

Number of resource levels: In the worst case, the number of resource levels for a task t_i is given by:

$$\#n_{Res}(t_i) = \sum_{t_k \in pred(t_i)} \#n_{Res}(t_k)).\#n_{consum}$$

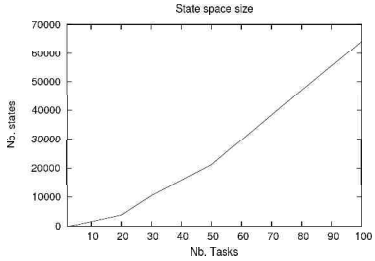

Figure 5: State space size for different mission sizes

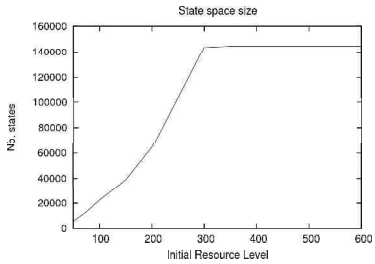

Figure 6: State space size for different initial resource level

where $\#n_{consum}$ is the number of possible resource consumptions for a task.

If the initial resource level is large, a lot of resource levels are available for each task since we only consider positive resource levels. If the initial resources increases, the number of negative resource levels decreases and more levels must be accounted for. Figure 6 shows the evolution of the state space size for a one hundred task mission when the initial resources increases. When the initial resources are greater than 350 units, we can see that the state space size levels off. In this situation, the agent has a large initial resource level and never lacks resources: each level reached is positive; increasing the initial level does not increase the number of levels anymore. If the initial resources get scarce, many possible resource levels are pruned: the state space size diminishes.

Branching factor As we increase the number of predecessors for task t_i, the number of possible resource levels available before t_i's execution increases, as well as the number of possible start times. Therefore, the number of branches (alternative paths) in the mission graph influences the state space size. More experiments about branching are under development.

6 Related Work

There has been considerable work in planning under uncertainty that leads to two categories of planners: conformant planners and contingent planners. These planners are characterized by 2 important criteria: *representation of uncertainty* and *observability*. The first criterion, the *uncertainty representation*, has been addressed in two ways in many planners using *disjunction* or *probability* while the second criterion is composed of *Non-observability* (NO), *partial observability* (PO) or *a full observability* (FO) of planners. A survey on all classes of planners can be found in Blythe (Blythe, 1999) and Boutilier (Boutilier et al., 1999) where details are given on NO, PO, or FO disjunctive planners and on NO, PO or FO probabilistic planners. Let us just recall some of those planners: C-PLAN NO-disjunctive planner, Puccini PO-disjunctive planner, Warplan FO-disjunctive planner, Buridan NO probabilistic planner, POMDP, C-MAXPLAN PO probabilistic planners and JIC, MDP FO probabilistic planners. In this section we focus on why those planners are unsuitable for our concern and why our work is a contribution to overcome those limits.

These planners encounter some difficulties in our domain of interest:

- *Model of time*: the existing planners do not support explicit time constraints nor complex temporal dependencies.
- *Model of actions*: the existing planners assume that actions are instantaneous.
- *Scalability*: the existing planners don't scale to large problems. For rover operations, a daily plan can involve on the order of a hundred operations, many of which having uncertain outcomes.

The approach we present in this paper meets the requirements for a rich model of time and actions and for scalability. It complements the work initiated in (Bresina and Washington, 2000) by using a similar model of time and utility distribution and by using a decision-theoretic approach. The advantage of using such an approach is to achieve optimality. Another contribution consists of handling uncertainty on resource consumption combined with uncertainty on execution time.

In the MDP we present, actions are not instantaneous as in the previous planners and can deal with complex time constraints such as a temporal window of execution and temporal precedence constraints. We also show that our approach can solve large problems with a hundred operations, many of which are uncertain. Another requirement needed by the rover applications consists of continuous time and resources. We experimentally show that when we use different approaches for resource units, our approach has a minor errors at run-time while scalability is better. The tradeoff between the scalability and the time granularity shows that we can discretize the time and the outcomes of actions regarding a small error in execution. Another requirement mentioned for the rover

applications is concurrent actions. This problem is under development, taking advantage of some multiagent concepts : as soon as the rover needs to execute two actions, we consider those actions are concurrent agents. This new line of research allows for bridging the gap between the multiagent systems and the distributed MDP.

7 Conclusion and future work

In this paper we presented an MDP planning technique that allows for a plan where operations have complex dependencies and complex time and resource constraints. The operations are organized in an acyclic graph where each operation has a temporal window during which it can be executed and an uncertain resource consumption and execution time. This approach is based on an MDP using a rich model of time and resources and complex dependencies between tasks. This technique allows us to deal with the variable duration of actions. We presented experimental results showing that our approach can scale to large robotic problems (a hundred of operations). Our approach also overcomes some of the limitations described in (Bresina et al., 2002). Indeed, our model is able to handle more complex time constraints and uncertainty on durations and resource consumptions. Moreover, as required in (Bresina et al., 2002), our system can consider plans of more than a hundred tasks.

In this paper, we have focused on planteray rover apllications Nonethless, our approach is not restricted to space applications. It can be applied to any scenario formalized by an acyclic graph with temporal constraints.

However, this approach needs to be extended to other requirements such as continuous variables. In our current version of the approach we use a discrete representation of time and resources. We show experimentally that with such a representation the execution errors are small and this representation can be tolerated. We continue experiments in this line of work to reduce the errors. We are specially interested in finding tradeoffs between the scalability, execution errors and discretization. The other extension we are developing consists of the use of multiagent systems with complex temporal dependencies using a distributed MDP.

Future work may concern the construction of the graphs that we consider as given in this paper.

REFERENCES

Blythe, J. (1999). *Planning under uncertainty in Dynamic domains*. PhD, Carnegie Mellon University.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning: Structural asumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1:1–93.

Boyan, J. and Littman, M. (2000). Exact solutions to time-dependent mdps. In *NIPS*.

Bresina, J., Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D., and Washington, R. (2002). Planning under continuous time and resource uncertainty : A challenge for ai. In *UAI*.

Bresina, J. and Washington, R. (2000). Expected utility distributions for flexible contingent execution. In *AAAI Workshop on Representation issues for Real World Planning system*.

Cardon, S., Mouaddib, A., Zilberstein, S., and Washington, R. (2001). Adaptive control of acyclic progressive processing task structures. In *IJCAI*, pages 701–706.

Estlin, T., Tobias, A., Rabideau, G., Castana, R., Chien, S., and Mjolsness, E. (1999). An integrated system for multi-rover scientific exploration. In *AAAI*, pages 613–613.

Morris, P., Muscettola, N., and Vidal, T. (2001). Dynamic control of plans with temporal uncertainty. In 17^{th} *IJCAI-01*.

Mouaddib, A.-I. and Zilberstein, S. (1998). Optimal scheduling for dynamic progressive processing. In *ECAI-98*, pages 499–503.

Sutton, R., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps : A framework of temporal abstraction in reinforcement learning. In *Report*.

Tsamardinos, I., Pollack, M., and Ramakrishnan, S. (2003). Assessing the probability of legal execution of plans with temporal uncertainty. In *ICAPS Workshop on Planning under uncertainty and Incomplete information*.

Vidal, T. and Ghallab, M. (1996). Dealing with uncertain durations in temporal constraints networks dedicated to planning. In 12^{th} *ECAI*.

Zilberstein, S. and Mouaddib, A.-I. (1999). Reactive control for dynamic progressive processing. In *IJCAI-99*, pages 1269–1273.