

Using a Classical Forward Search to Solve Temporal Planning Problems under Uncertainty

Éric Beaudry

Université du Québec à Montréal (Québec, Canada)
beaudry.eric@uqam.ca

Froduald Kabanza and Francois Michaud

Université de Sherbrooke (Québec, Canada)
{kabanza, francois.michaud}@usherbrooke.ca

Abstract

Planning with action concurrency under time and resources constraints and uncertainty is a challenging problem. Current approaches which rely on Markov Decision Processes and a discrete model for time and resources are limited by a blow-up of the search state-space. This paper presents a planner which is based on a classical forward search for solving this kind of problems. A continuous model is used for time and resources. The uncertainty on time is represented by continuous random variables which are organized in a dynamically generated Bayesian network. Two versions of the ACTUPLAN planner are presented. As a first step, ACTUPLAN^{nc} performs a forward-search in an augmented state-space to generate ϵ -optimal nonconditional plans which are robust to uncertainty (threshold on the probability of success). ACTUPLAN^{nc} is then adapted to generate a set of nonconditional plans which are characterized by different trade-offs between their probability of success and their expected cost. ACTUPLAN, the second version, builds a conditional plan with a lower expected cost by merging previously generated nonconditional plans. The branches are built by conditioning on the time. Empirical experimentation on standard benchmarks demonstrates the effectiveness of the approach.

Introduction

Significant advances have been made in classical planning in the last decades. However, classical planning approaches are often criticized by their lack of applicability to real-world problems which deal with durative actions, temporal and resources constraints, uncertainty, etc. Fortunately, many techniques originally designed for classical planning can be exploited to solve problems of other classes of planning.

This paper presents the ACTUPLAN planner which exploits classical planning methods to solve a class of planning problems which involves concurrent actions and uncertainty on the duration of actions. This class is motivated by real-world applications such as robotics and space applications. This type of planning is also known as *Concurrent Probabilistic Temporal Planning* (CPTP) (Mausam and Weld 2008).

State-of-the-art approach targeting problems with concurrency and uncertainty is based on Markov Decision Pro-

cesses (MDPs) (Mausam and Weld 2008). A CPTP problem can be translated into a Concurrent MDP by using an interwoven state-space and several methods have been developed to generate optimal and sub-optimal policies. An important assumption required by this approach is the use of discrete time to have a finite interwoven state-space. The use of a discrete time model introduces a blow up in the size of the interwoven state space. The scalability is then limited in practice by size of physical memory and by available CPU time.

Our approach is based on a continuous model of time and resources to address both action concurrency with time and resources uncertainty. The uncertainty on the occurrences of events (the start and end time of actions) is modelled using continuous random variables. In our state representation, each state feature is associated to random variables to mark their valid time and release time (Beaudry, Kabanza, and Michaud 2010). The dependencies between random variables are captured in a dynamically-generated Bayesian network. Two versions of our planner, named ACTUPLAN, are presented.

ACTUPLAN^{nc}, the first presented planner, performs a classical forward-search in an augmented state-space to generate nonconditional plans which are robust to time uncertainty. The robustness of plans is defined by a threshold on the probability of success (α). During the search, a Bayesian network is dynamically generated and the distributions of random variables are incrementally estimated. The probability of success and the expected cost of candidate plans are estimated by querying the Bayesian network. Generated nonconditional plans are ϵ -optimal in terms of expected cost (e.g., makespan).

ACTUPLAN, the second planner, merges several nonconditional plans. The nonconditional planning algorithm is adapted to generate several nonconditional plans which are characterized by different trade-offs between their probability of success and their expected cost. The resulting conditional plan has a lower cost and/or a higher probability of success than individual nonconditional plans. The switch conditions within the plan are built through an analysis of the estimated distribution probability of random variables.

Basic Concepts

This section introduces basic concepts behind ACTUPLAN. These concepts are illustrated with examples on the Transport domain (from the International Planning Competition) on which uncertainty has been added to the duration of actions. In that planning domain a set of trucks have to drive and deliver packages distributed over a given map. A package is either at a location or loaded onto a truck. There is no limit on the number of packages a truck can transport at the same time and on the number of trucks that can be parked at same location.

State Variables

State features are represented as state variables which is equivalent to a classical representation (Nebel 2000). A *state variable* $x \in X$ describes a particular feature of the world state associated to a finite domain $Dom(x)$. A world state is an assignment of values to the state variables, while action effects (state updates) are changes of variable values. This paper only focus on temporal uncertainty. It is assumed that no exogenous events take place; hence only planned actions cause state changes.

For instance, the Transport domain involves the following objects: a set of trucks $R = \{r_1, \dots, r_n\}$, a set of packages $B = \{b_1, \dots, b_m\}$ and a set of locations $L = \{l_1, \dots, l_k\}$ distributed over a map. The set of state variables $X = \{C_r, C_b \mid r \in R, b \in B\}$ specifies the current location of trucks and packages. The domain of variables is defined as $Dom(C_r) = L (\forall r \in R)$ and $Dom(C_b) = L \cup R (\forall b \in B)$.

Time Random Variables

The uncertainty related to time is represented using continuous random variables. A *time random variable* $t \in T$ marks the occurrence of an event, corresponding to either the start or the end of an action. An event induces a change of the values of a subset of state variables. The time random variable $t_0 \in T$ is reserved for the initial time, i.e., the time associated to all state variables in the initial world state. Each action a has a duration represented by a random variable d_a . A time random variable $t \in T$ is defined by an equation specifying the time at which the associated event occurs. For instance, an action a starting at time t_0 will end at time t_1 , the latter being defined by the equation $t_1 = t_0 + d_a$.

States

A state describes the current world state using a set of state features, that is, a set of value assignments for all state variables. A *state* s is defined by $s = (\mathcal{U}, \mathcal{V}, \mathcal{R})$ where:

- \mathcal{U} is a total mapping function $\mathcal{U} : X \rightarrow \cup_{x \in X} Dom(x)$ which retrieves the *current assigned value* for each variable $x \in X$ such that $\mathcal{U}(x) \in Dom(x)$.
- \mathcal{V} is a total mapping function $\mathcal{V} : X \rightarrow T$ which denotes the *valid time* at which the assignment of variables X have become effective.
- \mathcal{R} is a total mapping function $\mathcal{R} : X \rightarrow T$ which indicates the *release time* on state variables X which correspond to the latest time that *over all* conditions expire.

The release times of state variables are used to track *over all* conditions of actions. The time random variable $t = \mathcal{R}(x)$ means that a change of the state variable x cannot be initiated before time t . The valid time of an state variable is always before or equals to its release time, i.e., $\mathcal{V}(x) \leq \mathcal{R}(x) \forall x \in X$. The valid time (\mathcal{V}) and the release time (\mathcal{R}) respectively correspond to the write-time and the read-time in Multiple-Timeline of SHOP2 planner (Do and Kambhampati 2003), with the key difference here being that random variables are used instead of numerical values.

Hence, a state is not associated with a fixed timestamp as in a classical approach for action concurrency (Bacchus and Ady 2001). Only temporal uncertainty is considered in this paper, i.e., there is no uncertainty about the values being assigned to state variables. Dealing with this kind of uncertainty is planned as future work. The only uncertainty on state variables is about **when** their assigned values become valid. The valid time $\mathcal{V}(x)$ models this uncertainty by mapping each state variable to a corresponding time random variable.

Figure 1 illustrates an example of a state in the Transport domain. The left side (a) is illustrated using a graphical representation on a map. The right side (b) presents state s_0 in ACTUPLAN formalism. Two trucks r_1 and r_2 are respectively located at locations l_1 and l_4 . Package b_1 is loaded on r_1 and package b_2 is located at location l_3 . The valid time of all state variables is set to time t_0 .

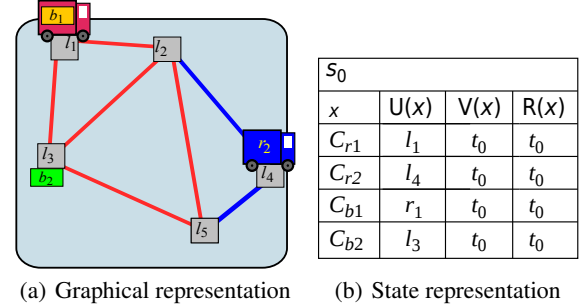


Figure 1: Example of a state for the Transport domain

Actions

The specification of actions follows the extensions introduced in PDDL 2.1 (Fox and Long 2003) for expressing temporal planning domains. The set of all actions is denoted by A . An *action* $a \in A$ is a tuple $a = (\text{params}, \text{cstart}, \text{coverall}, \text{estart}, \text{eend}, d_a)$ where :

- *params* is a list of parameters p_1, \dots, p_n , each one taking a value in the domain of a state variable;
- *cstart* is the set of *at start* conditions that must be satisfied at the beginning of the action;
- *coverall* is the set of persistence conditions that must be satisfied *over all* the duration of the action;
- *estart* and *eend* are respectively the sets of *at start* and *at end* effects on the state variables;

- and $d_a \in D$ is the random variable which models the duration of the action.

A condition is a pair $c = (p, x)$ which is interpreted as a Boolean expression $p = s.U(x)$ where s is an implicit state. The function $vars(c) \rightarrow 2^X$ returns the set of all state variables that are referenced by the condition c .

An effect $e = (x, exp)$ specifies the assignment of the value resulting from the evaluation of expression exp to the state variable x . The expressions $conds(a)$ and $effects(a)$ return, respectively, all conditions and all effects of action a .

The set of *action duration random variables* is defined by $D = \{d_a \mid a \in A\}$ where A is the set of actions. A random variable d_a for an action follows a probability distribution specified by a probability density function $\phi_{d_a} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ and a cumulative distribution function $\Phi_{d_a} : \mathbb{R}^+ \rightarrow [0, 1]$.

An action a is **applicable** in a state s if and only if state s satisfies all **at start** and **over all** conditions of a . A condition $c \in conds(a)$ is satisfied in state s if c is satisfied by the current assigned values of state variables of s .

ACTUPLAN^{nc} : Nonconditional Planner

The ACTUPLAN^{nc} algorithm expands a search graph in the state space and dynamically generates a Bayesian network which contains random variables.

State Transition

Algorithm 1 describes the APPLY function which computes the state resulting from application of an action a to a state s . Time random variables are added to the Bayesian network when new states are generated. The start time of an action is defined as the earliest time at which its requirements are satisfied in the current state. Line 3 calculates the time t_{conds} which is the earliest time at which all *at start* and *over all* conditions are satisfied. This time corresponds to the maximum of all time random variables associated to the state variables referenced in the action's conditions. Line 4 calculates time $t_{release}$ which is the earliest time at which all persistence conditions are released on all state variables modified by an effect. Then at Line 5, the time random variable t_{start} is generated. Its defining equation is the max of all time random variables collected in Lines 3–4. Line 6 generates the time random variable t_{end} with the equation $t_{end} = t_{start} + d_a$. Once generated, the time random variables t_{start} and t_{end} are added to the Bayesian network if they do not already exist. Lines 7–9 set the release time to t_{end} for each state variable involved in an *over all* condition. Lines 10–17 process *at start* and *at end* effects. For each effect on a state variable, they assign this state variable a new value, set the valid and release times to t_{start} and add t_{end} .

Figure 2 illustrates two state transitions. Figure 3 presents the corresponding Bayesian network. Colours are used to differentiate types of random variables. State s_1 is obtained by applying the action $Goto(r_1, l_1, l_2)$ from state s_0 . The APPLY function (see Algorithm 1) works as follows. The action $Goto(r_1, l_1, l_2)$ has the *at start* condition $C_{r_1} = l_1$. Because C_{r_1} is associated to t_0 , we have $t_{conds} = \max(t_0) = t_0$. Since the action modifies the C_{r_1} state variable, Line 4 computes the time $t_{release} = \max(t_0) = t_0$.

Algorithm 1 APPLY action function

```

1. function APPLY( $s, a$ )
2.    $s' \leftarrow s$ 
3.    $t_{conds} \leftarrow \max_{x \in vars(conds(a))} s.V(x)$ 
4.    $t_{release} \leftarrow \max_{x \in vars(effects(a))} s.R(x)$ 
5.    $t_{start} \leftarrow \max(t_{conds}, t_{release})$ 
6.    $t_{end} \leftarrow t_{start} + d_a$ 
7.   for each  $c \in a.coverall$ 
8.     for each  $x \in vars(c)$ 
9.        $s'.R(x) \leftarrow \max(s'.R(x), t_{end})$ 
10.  for each  $e \in a.estart$ 
11.     $s'.U(e.x) \leftarrow eval(e.exp)$ 
12.     $s'.V(e.x) \leftarrow t_{start}$ 
13.     $s'.R(e.x) \leftarrow t_{start}$ 
14.  for each  $e \in a.eend$ 
15.     $s'.U(e.x) \leftarrow eval(e.exp)$ 
16.     $s'.V(e.x) \leftarrow t_{end}$ 
17.     $s'.R(e.x) \leftarrow t_{end}$ 
18.  returns  $s'$ 

```

At Line 5, the action's start time is defined as $t_{start} = \max(t_{conds}, t_{release}) = t_0$, which already exists. Then, at Line 6, the time random variable $t_{end} = t_0 + d_{Goto(r_1, l_1, l_2)}$ is created and added to the Bayesian network with the label t_1 . Next, Lines 13–16 apply effects by performing the assignment $C_{r_1} = l_2$ and by setting time t_1 as the valid time for C_{r_1} . State s_2 is obtained similarly.

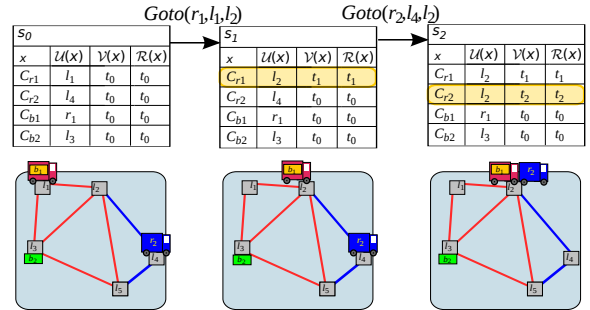


Figure 2: Example of two state transitions in the Transport domain

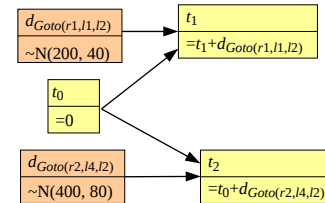
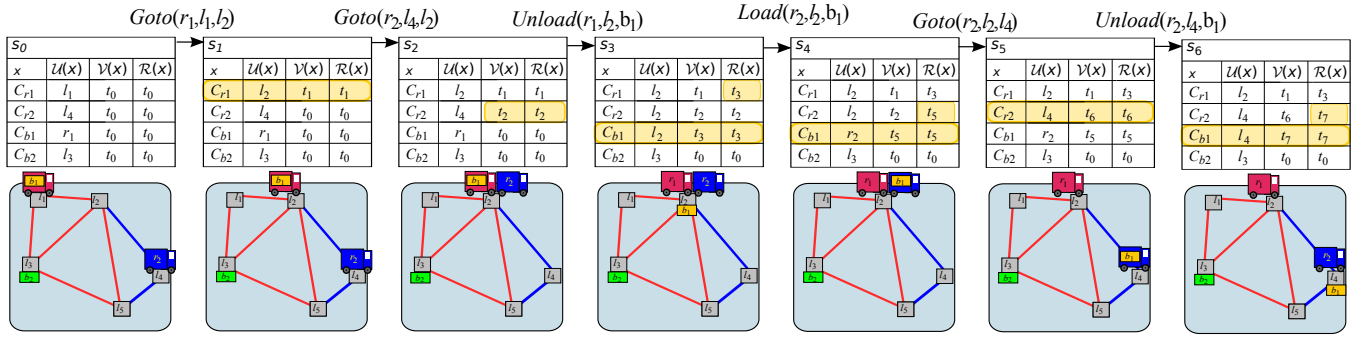
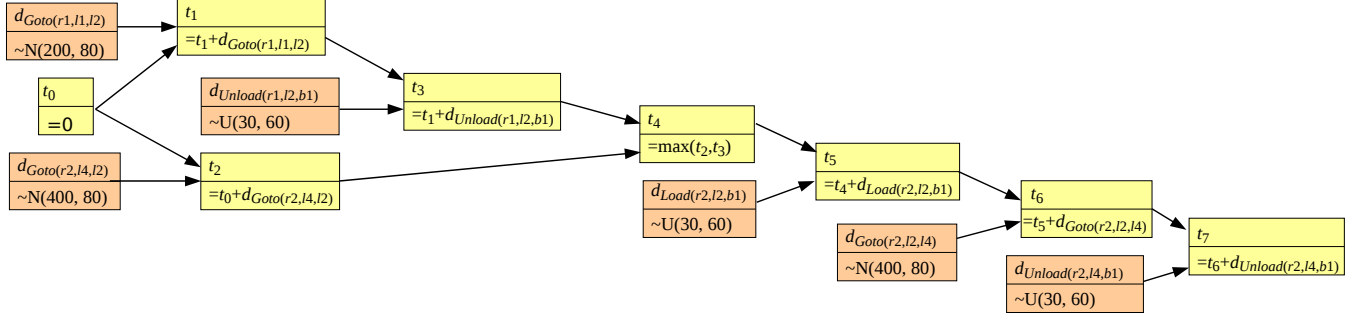


Figure 3: Example of a Bayesian network

Algorithm 2 presents the planning algorithm of ACTUPLAN^{nc} in a recursive form. This planning algorithm performs best-first-search in the state space defined in previous section to find a state which satisfies the goal with a probability of success greater than or equal a given



(a) State-space



(b) Bayesian network

Figure 4: Sample search with the Transport domain

threshold α (Line 2). The α parameter is a hard constraint of the probability of success of output plans and must be interpreted as a tolerance to failures. If $s \models \mathcal{G}$ then a nonconditional plan π is built (Line 3) and returned (Line 4). The choice of an action a at Line 5 is a backtrack point. A heuristic function based on the h_{max} (Haslum and Geffner 2000) is involved to guide this choice. It estimates a lower bound on the expected value of the minimum makespan reachable from a state s . The optimization criteria is implicitly given by a given metric function $cost$. Line 6 applies the chosen action to the current state. At Line 7, an upper bound on the probability that state s can lead to a state which satisfies goal \mathcal{G} is evaluated. The symbol \models^* is used to denote a goal may be reachable from a given state, i.e. their may exist a plan. The symbol \bar{P} denotes an upper bound on the probability of success. If that probability is under the fixed threshold α , the state s is pruned. Line 8 performs a recursive call.

Example on Transport domain

Figure 4 illustrates an example of a partial search carried out by Algorithm 2 on a problem instance of the Transport domain. The goal in that example is defined by $\mathcal{G} = \{C_{b_1} = l_4\}$. Note that trucks can only travel on paths of its color. For instance, truck r_1 cannot move from l_2 to l_4 . A subset of expanded states is shown in (a). The states s_0 , s_1 and s_2 are the same as previous figures except that

State s_3 is obtained by applying $Unload(r_1, l_2, b_1)$ action from state s_2 . This action has two conditions : the *at start*

Algorithm 2 Nonconditional planning algorithm

1. ACTUPLAN^{nc}($s, \mathcal{G}, A, \alpha$)
2. if $P(s \models \mathcal{G}) \geq \alpha$
3. $\pi \leftarrow \text{ExtractNonConditionalPlan}(s)$
4. return π
5. nondeterministically choose $a \in A$
6. $s' \leftarrow \text{APPLY}(s, a)$
7. if $\bar{P}(s' \models^* \mathcal{G}) \geq \alpha$
8. return ACTUPLAN^{nc}($s', \mathcal{G}, A, \alpha$)
9. else return FAILURE

condition $C_{b_1} = r_1$ and the *over all* condition $C_{r_1} = l_2$. The action has the *at end* $C_{b_1} = l_2$ effect. The start time of this action is obtained by computing the maximum time of all valid times of state variables concerned by conditions and all release times of state variables concerned by effects. The start time is then $\max(t_0, t_1) = t_1$. The end time is $t_3 = t_1 + d_{Unload(r_1, l_2, b_1)}$. Because the *over all* condition $C_{r_1} = l_2$ exits, the release time $\mathcal{R}(C_{r_1})$ is updated to t_3 . This means that another action cannot move truck r_1 away from l_2 before t_3 .

State s_4 is obtained by applying $Load(r_2, l_2, b_1)$ action from state s_3 . This action has two conditions : the *at start* condition $C_{b_1} = l_2$ and the *over all* condition $C_{r_2} = l_2$. The action has the *at end* $C_{b_1} = r_2$ effect. The start time is then $t_4 = \max(t_2, t_3)$. The end time is $t_5 = t_4 + d_{Load(r_2, l_2, b_1)}$. The goal \mathcal{G} is finally satisfied in state s_6 . A nonconditional temporal plan is extracted from this state.

ACTUPLAN : Conditional Planner

The conditional planner is built on top of the nonconditional planner presented in previous section. The basic idea behind the conditional planner is that the nonconditional planner can be modified to generate several nonconditional plans with different trade-offs on their probability of success and their cost. A better conditional plan is then obtained by merging multiple nonconditional plans. Time conditions are set to control execution.

Consider the following situation. A truck initially parked in location l_2 has to deliver two packages b_0 and b_1 from locations l_9 and l_8 to locations l_6 and l_8 . Please refer to the map on the left of Figure 5. The package b_0 destined to location l_6 has time $t = 1800$ as a delivery deadline. The cost is defined by the total duration of the plan (makespan).

There exists at least the two following possible nonconditional plans π_a and π_b . The plan π_a delivers package b_0 first while plan π_b delivers package b_1 first. Since package b_0 has the earliest deadline, plan π_a has a higher probability of success than plan π_b . However, plan π_a has a longer makespan than plan π_b because its does not take the shortest path. If the threshold on the probability of success is fixed to $\alpha = 0.8$, the nonconditional plan π_a must be returned by ACTUPLAN^{nc}.

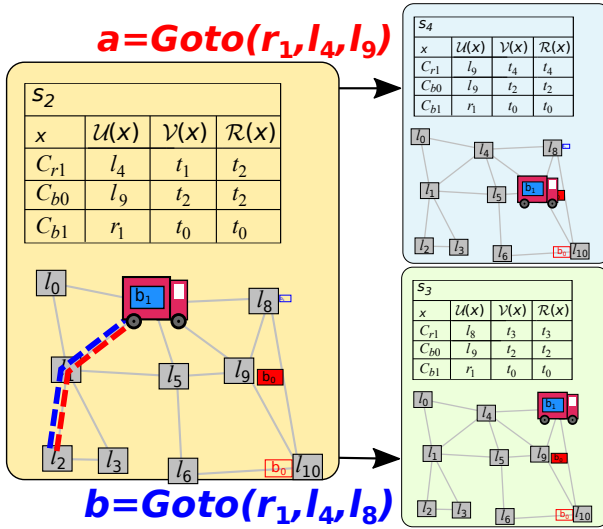


Figure 5: Possible actions in state s_2

A better solution is to delay the decision about which package to deliver first. The common prefix of plans π_a and π_b ($\text{Goto}(r_1, l_2, l_4)$, $\text{Load}(r_1, l_4, b_1)$) is first executed. Let the resulting state of this prefix be s_2 . As illustrated in Figure 5, with the respect of nonconditional plans π_a and π_b , there is two possible actions : $\text{Goto}(r_1, l_4, l_8)$ or $\text{Goto}(r_1, l_4, l_9)$.

Building Conditional Plans

The conditional planner (ACTUPLAN) proceeds in two steps. It first expands a state space using Algorithm 3. This algorithm is a modified best-first-search that generates the

set of all reachable states F which satisfy the goal with a cost (e.g. makespan) of at most t_α . The t_α is obtained by generating a nonconditional plan using α parameter. At Line 17, we store in f attribute the final cost (e.g. makespan) to reach state s' which satisfies the goal with a probability $\geq \beta$. The β parameter controls the quality of the solution. Intuitively, when β is equals to α , only a nonconditional plan could be extracted. Lower β is then more states are generated and better could be the generated conditional plans. When β tends to zero, the generated conditional plans will be approximate solutions of an optimal policy.

Algorithm 3 Algorithm for generating a set of nonconditional plans

```

1. EXPANDGRAPH( $s_0, \mathcal{G}, A, \alpha, \beta$ )
2.  $open \leftarrow \{s_0\}$ 
3.  $close \leftarrow \emptyset$ 
4.  $t_\alpha \leftarrow \sim \mathcal{N}(+\infty, 0)$ 
5. while  $open \neq \emptyset$ 
6.    $s \leftarrow open.\text{RemoveFirst}()$ 
7.   add  $s$  to  $close$ 
8.   if  $\max_{x \in X}(s.\mathcal{V}(x)) > t_\alpha$  exit while loop
9.   if  $P(s \models \mathcal{G}) \geq \beta$ 
10.    add  $s$  to  $F$ 
11.    if  $P(s \models \mathcal{G}) \geq \alpha$ 
12.       $t_\alpha \leftarrow \min(t_\alpha, \max_{x \in X}(s.\mathcal{V}(x)))$ 
13.   else
14.     for each  $a \in A$  such  $a$  is applicable in  $s$ 
15.        $s' \leftarrow \text{Apply}(s, a)$ 
16.       add  $s'$  to  $s.\text{parents}$ 
17.        $s'.f \leftarrow \text{EvaluateHeuristic}(s', \mathcal{G}, \beta)$ 
18.       if  $s' \notin close$  and  $\overline{P}(s' \models^* \mathcal{G}) \geq \beta$ 
19.         add  $s'$  to  $open$ 
20. return  $F$ 

```

Once the graph has been generated by Algorithm 3, the second step is to recursively generate branching conditions from states in set F to the initial state. Conditions are computed by estimating critical times (λ_{min} and λ_{max}) for each actions. The critical time $\lambda_{a,max}$ is obtained by estimating the latest time at which starting action a leads to a probability of success of at least α . The critical time $\lambda_{a,min}$ is obtained by estimating the earliest time at which it better to start action a than waiting for starting another action. Figure 6 illustrates an example how critical times are computed for state s_2 from Figure 5 for the threshold $\alpha = 0.8$. Small solid lines show the estimated probability of success of actions a and b when they are started at a decision time λ . We see that the probability of success of starting action a at time λ drops under 0.8 at $\lambda_{a,max} = 642$. This means that action a should not be selected later than time 642. The large solid lines indicates the expected probability when actions are started before λ . Critical times λ_{min} are obtained similarly.

Empirical Results

We compared our planners to an MDP based planner (Mausam and Weld 2008) on the Transport domain. Table 1 presents empirical results. The nonconditional planner (ACTUPLAN^{nc}) was run with $\alpha = 0.9$. The conditional

Table 1: Empirical results for the ACTUPLAN^{nc} and ACTUPLAN on the Transport domain

Size		ACTUPLAN ^{nc}			ACTUPLAN			MDP Planner		
$ R $	$ \mathcal{G} $	CPU	P	$Cost$	CPU	P	$Cost$	CPU	P	$Cost$
1	2	0.3	1.00	2105.5	0.3	0.96	1994.5	44.9	0.99	1992.4
1	3	0.3	1.00	2157.3	0.6	0.93	1937.9	214.5	0.00	-
1	4	0.3	0.98	2428.0	0.5	0.98	2426.2	192.6	0.00	-
2	2	0.2	1.00	1174.0	0.4	1.00	1179.2	302.3	0.00	-
2	3	0.3	1.00	1798.3	1.8	0.93	1615.2	288.3	0.00	-
2	4	0.3	1.00	1800.0	4.2	0.91	1500.5	282.8	0.00	-
3	6	4.1	0.97	1460.1	300	0.97	1464.5	307.9	0.00	-

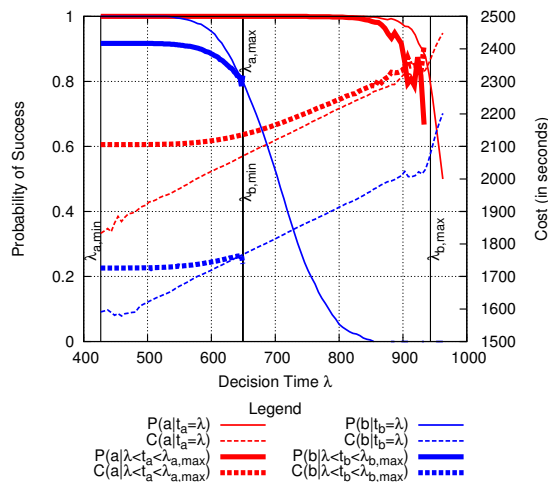


Figure 6: Latest times for starting actions

planner (ACTUPLAN) was run with $\alpha = 0.9$ and $\beta = 0.4$. The first two columns show the size of the problems (number of trucks and packages). Columns under each planner report the CPU time, the probability of success and the cost (makespan). Results show that the conditional planner reduced the expected makespan for most of the problems, when it is possible. For few problems, there does not exist conditional plan which is strictly better than the nonconditional one.

Generated conditional plans are generally better plans but require much more CPU time than ACTUPLAN^{nc}. This time is required because an optimal conditional plan may require the combination of many plans, which requires to explore a much larger part of the state space. In practice, the choice between ACTUPLAN^{nc} and ACTUPLAN should be guided by the potential benefit of having smaller costs. There is a trade-off between plans quality and required time to generate them. When a small improvement on the cost of plans has a significant consequence, then ACTUPLAN should be used. If decisions have to be taken rapidly, then ACTUPLAN^{nc} is more appropriate.

Conclusion

This paper presented ACTUPLAN, which is based a new approach for planning under time constraints and uncertainty.

The first contributions are a new state representation which is based on a continuous model of time. Continuous random variables are used to model the uncertainty on the time. Time random variables define the start and end times of actions, as well as their duration. Since a continuous model is more compact than discrete representations, it helps avoid the state space explosion problem.

Two planning algorithms have been presented. The non-conditional algorithm (ACTUPLAN^{nc}) produces ϵ -optimal nonconditional plans by performing a classical forward-search in the state space. A conditional planning algorithm (ACTUPLAN) generates conditional plans which are an approximation of optimal plans. The conditional plans are obtained by merging several nonconditional plans having different trade-offs between their quality and probability of success. Empirical results showed the efficiency of our approach on planning domains having uncertainty on the duration of actions.

References

- Bacchus, F., and Ady, M. 2001. Planning with resources and concurrency: A forward chaining approach. In *Proc. of the International Joint Conference on Artificial Intelligence*, 417–424.
- Beaudry, É.; Kabanza, F.; and Michaud, F. 2010. Planning for concurrent action executions under action duration uncertainty using dynamically generated bayesian networks. In *Proc. of the International Conference on Automated Planning and Scheduling*, 10–17.
- Do, M., and Kambhampati, S. 2003. SAPA: A scalable multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194.
- Fox, M., and Long, D. 2003. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.
- Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proc. of the Artificial Intelligence Planning Systems*, 140–149.
- Mausam, and Weld, D. S. 2008. Concurrent probabilistic temporal planning. *Journal of Artificial Intelligence Research* 31:33–82.
- Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research* 12:271–315.