Computer Science Technical Report

A Hill-Climbing Approach for Planning with Temporal Uncertainty

Janae N. Foss Nilufer Onder
{jnfoss,nilufer}@mtu.edu

Michigan Technological University
Computer Science Technical Report
CS-TR-06-02
February 13, 2006

Department of Computer Science
Houghton, MI 49931-1295
www.cs.mtu.edu

A Hill-Climbing Approach for Planning with Temporal Uncertainty

Janae N. Foss Nilufer Onder
{jnfoss,nilufer}@mtu.edu

Abstract

We approach the problem of finding temporally contingent plans, i.e., plans with branches that are based on the duration of an action at execution time, using a hill-climbing algorithm. We find an optimistic plan that is valid when all actions complete quickly. We then use efficient temporal reasoning techniques to determine when the plan may fail. At time points that cause an unsafe situation, we insert temporal contingency branches. We describe our implemented planner PHOCUS-HC and provide experimental results.

Introduction

Constructing optimal plans for domains with uncertainty is a challenging problem. Such domains often include uncertain discrete or continuous effects, oversubscribed goals, and possibly parallel actions. Many planners have been built that prepare contingency plans when actions may affect the world in uncertain ways (see Bresina et al's paper (2002) for a survey). However, most work in planners that assume uncertain resources has focused on finding plans that are safe considering maximal resource usage. In other words, the choice of actions during execution time does not depend on resource levels. Recent work (Mausam *et al.* 2005) relaxes this assumption by explicitly representing resources as part of the state in the context of the AO* (Nilsson 1980) algorithm. The advantage of this approach is its ability to handle any uncertain continuous resource as well as time. One drawback of using AO* is that the solution plans are sequential whereas executing actions in parallel is a valuable way to optimize plans.

We approach the problem of finding temporally contingent plans, i.e., plans with branches that are based on the duration of an action at execution time, using a hill-climbing algorithm. Such an algorithm was shown to be effective for generating policies for continuous-time stochastic domains (Younes & Simmons 2004). We take an optimistic approach by first finding a plan that is valid when all actions complete quickly. We then use the methods described in (Dechter, Meiri, & Pearl 1991) to determine when the plan may fail. At time points that cause an unsafe situation, temporal contingency branches are inserted. The problems we are studying satisfy the following criteria: (1) there is more than one solution plan, (2) solution plans are ranked by an objective function that is not fully based on makespan, (3) actions have uncertain durations, (4) the start and/or end times of some actions are constrained, (5) as actions require more time to complete, plans with high utility become invalid, and (6) actions can be executed in parallel unless they are ordered explicitly.

As an example, consider the problem of traveling from home to a conference. One solution plan is to drive to the airport, fly to the destination city, take a shuttle to the conference venue, and finally register for the conference. Another solution plan could involve taking a taxi instead of a shuttle to the venue. Assuming the objective is to minimize money spent, the plan with the shuttle action would be preferred. However, the taxi may be faster than the shuttle and because there exist constraints on the time one can register for the conference, there may only be enough time for the more expensive taxi option depending on how long the flight takes. To always have a safe plan, and be able to save money when possible, our approach would generate a temporally contingent plan: drive to the airport, fly to the destination, take the shuttle if there is enough time, otherwise take the taxi, and register for the conference. In addition, the utility of the plan can be increased by having parallel tasks such as reading a paper or grading exams during flights or airport wait periods. Throughout this paper our running example will be this conference domain.

We have several contributions: (1) we define the notion of temporally contingent plans, (2) we provide a hill-climbing algorithm that uses efficient temporal reasoning techniques to insert branches based on time rather than world conditions, (3) we show that plans with maximal expected utility can be generated in this framework by using

our implemented planner PHOCUS-HC[1], (4) we provide example domains including a disaster evacuation domain which can benefit from our approach. In the remainder of this paper we first define temporal uncertainty and what constitutes a solution plan. We then explain our algorithm for creating temporally contingent plans. This is followed by a description of experimental results, related work, and future work.

Planning problems with Temporal Uncertainty

Our framework deals with several temporal aspects of planning problems. First, we deal with problems where action durations are uncertain. We define this uncertainty by assigning each action a closed interval duration [*min-d*, *max-d*], where *min-d* and *max-d* are the minimum and maximum reasonable durations required by the action, respectively. We assume that a probability distribution function such as a Gaussian or Weibull distribution is associated with each interval. When intervals are used to define action durations, there are two possible interpretations (Vidal & Fargier 1999). The first and most common interpretation is to assume that the agent can determine the duration of the action by choosing any value from the interval. We call these *assignable durations*. (Vidal and Fargier refer to these as *free constraints* (1999).)

The second interpretation is to assume that the agent has no control over the duration and the actual duration of an action is only known after it completes execution. This is the interpretation that our work centers around and we call these *unassignable durations*. (Vidal and Fargier refer to these as *contingent constraints* (1999).)

Unassignable durations are more difficult to plan for because the agent must be prepared for any action duration. Therefore, the simplest approach to dealing with unassignable durations is to assume that every action always requires its maximum duration, and plan accordingly. However, it is unlikely that all actions will require that much time, so taking this pessimistic approach results in a safe plan where resources (money, fuel, time, etc.) are often not used efficiently. We define a plan as *safe* if it is guaranteed to execute to completion, regardless of how long its actions take. A conservative approach to conference travel is to always take a taxi because it is faster than the bus, when in reality there is often enough time to take the bus to save money. Another example is planetary exploration by a rover, where planning conservatively could result in much idle time, lowering the number of experiments (Bresina *et al.* 2002). Our approach combines the need to have a safe plan with the desire to use resources as efficiently as possible by creating a plan with branches based on temporal conditions.

Deadlines are another temporal aspect that our planner considers. Rather than assuming that any action can be executed at any time, actions can be defined to occur only within given temporal constraints. This complicates the problem further because an efficient plan may include an action whose execution is only possible when previous actions in the plan complete quickly. Our approach of creating a plan with branches based on temporal conditions allows us to include the efficient action while also adding a less efficient branch to be executed when necessary. Finally, our planner optimizes solutions based on an objective function which includes time and other metrics.

We have extended PDDL2.2 (Edelkamp & Hoffman 2004) to represent *interval durative* actions as opposed to single point durative actions. In Figure 1 we show a coding of the conference domain. In this example the flight takes between 45 to 90 time units and starts at time 30. Notice that the `eat-meal` action has an assignable duration but the other actions have unassignable durations. For coding convenience, we have also added a syntax that associates actions with their execution time constraints more directly than the *timed initial literals* of PDDL2.2.

Formally, a *planning problem* is defined as a quadruple <**D**, **I**, **G**, **M**>, where **D** is a domain description that lists the actions that are available, **I** is a description of the initial state, **G** is a description of the goals, and **M** is a plan metric that represents the objective function. In the next section, we describe our solution approach.

Creating Temporal Contingency Plans

When creating a temporal contingency plan, it is important to find a plan that is both safe and has high utility. We do this by using a Just-In-Case style algorithm (Drummond, Bresina, & Swanson 1994) where we generate a seed plan, find points where it is likely to fail, and then insert contingency branches at those points (Figure 2).

To generate the seed plan (line 1 in Figure 2), we assign *min-d* as the duration of each action. This yields a plan with high utility. Next, we analyze the seed plan to find out, temporally speaking, when it becomes unsafe (lines 4 through 6). At any time point where the seed plan becomes unsafe, we generate and insert a branch that is safe. This technique creates a plan that includes a path that can be safely executed when all of its actions require their maximum duration, but also includes branches that yield a more desirable result.

Setting the duration of each action to *min-d* removes all uncertainty at planning time. This allows the use of any planner that understands PDDL2.2. A plan *P* returned by such a planner will be temporally deterministic. Our

[1]Phocus means "planner with an algorithmic focus." PHOCUS-HC has a hill-climbing algorithm focus and is one of our set of planners which have different algorithmic foci for dealing with temporal uncertainty.

Domain description

```
(define (domain conference-travel)
  (:requirements :fluents :equality :execution-times
                 :interval-durative-actions)
  (:predicates (at_airport1) (at_airport2) (at_hotel)
               (not_hungry) (attending_conference))
  (:functions   (money_spent))

  (:interval-durative-action fly_airport2_airport1
     :unassignable-interval-duration
          (and (min ?duration 45) (max ?duration 90))
     :condition (at start (at_airport1))
     :effect (and (at end (at_airport2))
             (at start (not (at_airport1)))
             (at start (increase (money_spent) 200)))
     :execution-time (start at 30))

  (:interval-durative-action taxi_hotel_airport2
     :unassignable-interval-duration
          (and (min ?duration 15) (max ?duration 20))
     :condition (at start (at_airport2))
     :effect (and (at end (at_hotel))
             (at start (not (at_airport2)))
             (at start (increase (money_spent) 120))))

  (:interval-durative-action shuttle_hotel_airport2
     :unassignable-interval-duration
          (and (min ?duration 30) (max ?duration 60))
     :condition (at start (at_airport2))
     :effect (and (at end (at_hotel))
             (at start (not (at_airport2)))
             (at start (increase (money_spent) 20))))

  (:interval-durative-action eat_meal
     :assignable-interval-duration
          (and (min ?duration 20) (max ?duration 60))
     :condition (at start (attending_conference))
     :effect (at end (not_hungry))
             (at start (increase (money_spent) 20))))

 (:interval-durative-action register_for_conference
    :unassignable-interval-duration
         (and (min ?duration 5) (max ?duration 10))
    :condition (over all (at_hotel))
    :effect (at end (attending_conference))
    :execution-time (and (start after 84) (start before 141))))
```

Problem description

```
(define (problem conference-travel-1)
  (:domain conference-travel)
  (:init        (at_airport1)
                (= (money-spent) 0))
  (:goal        (attending_conference))
  (:metric      minimize (money-spent)))
```

Figure 1: Conference travel domain and problem.

PHOCUS-HC (*D, I, G, M*)

1: $P_0 \leftarrow$ GENERATE-SEED-PLAN (*D, I, G, M*)

2: $P_{current} \leftarrow P_0$

3: **loop do**

4: $DG \leftarrow$ CONSTRUCT-DISTANCE-GRAPH($P_{current}$,*D*,*I*)

5: **if** SAFE-PLAN ($P_{current}$, *DG, D, I, G, M*) **return** $P_{current}$

6: $P_{next} \leftarrow$ MAKE-PLAN-SAFE ($P_{current}$, *DG, D, I, G, M*)

7: **if** P_{next} is null **return** failure

8: $P_{current} \leftarrow P_{next}$

Figure 2: The top-level algorithm.

algorithm factors temporal uncertainty back in by converting *P* to a directed, edge-weighted graph called a *distance graph*, thus expressing *P* as a simple temporal network (STN) (Dechter, Meiri, & Pearl 1991). STNs are widely used in temporal reasoning and include nodes representing time points and edges between pairs of nodes representing temporal constraints between time points. Figure 3 shows (a) the seed plan that would be generated for the problem in Figure 1 and (b) the corresponding distance graph as computed in step 4 of the algorithm.

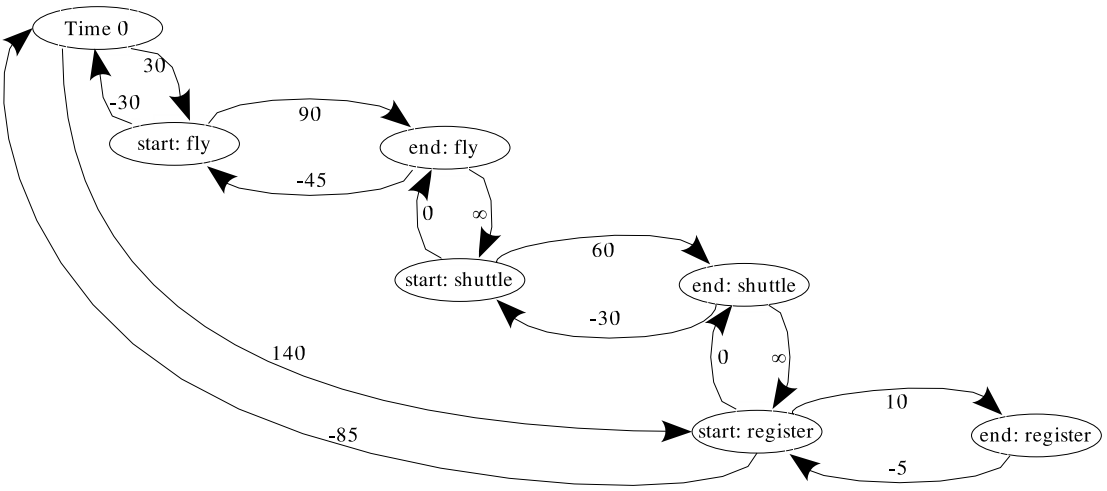
In construction of the distance graph *DG*, each action is dealt with individually to allow any possible concurrency in *P* to be present in *DG*. The first step is to add a node s_0 representing time 0, and two nodes for each action *i*, one for its start time s_i and one for its end time e_i. Edges are then added in pairs to represent temporal relations. For each action *i*, a pair of edges is added between s_i and e_i. The edge $s_i \rightarrow e_i$ is weighted with (*max-d* of *i*) and the edge $e_i \rightarrow s_i$ is weighted with -1 $\times$ (*min-d* of *i*). Next, pairs of edges are added between s_0 and each s_i node to represent constrained start times. This is shown with the fly and register actions in Figure 3(b). When an action does not have a constrained start time, the edge $s_0 \rightarrow s_i$ is weighted with ∞ and the edge $s_i \rightarrow s_0$ is weighted with 0, signifying that the start of action *i* comes after time 0, but there are no other constraints. For clarity, these edges are not included in Figure 3(b).

The final step in constructing the distance graph is to insert edges that represent relationships between actions. Though *P* contains a sequence of steps, some concurrency may be possible. To properly discover and encode this in *D*, causal links and threats in *P* must be identified. This is done using an algorithm similar to the one described by (R-Moreno *et al.* 2002). For every condition *c* of each action *i*, a causal link is added to the closest action *j* in the plan that appears before *i* and produces *c* as an effect. The causal link forces the producer action *j* to occur before the consumer action *i*. Threats occur when the effect of one action negates the precondition of another action. A threat link is added between an action *i* and an action *j* when an effect of *j* negates a precondition of *i*. This is done to ensure the precondition is true when *i* executes. This algorithm discovers no knowledge about temporal distance, so pairs of edges labeled with 0 and ∞ are added to the graph simply expressing that one action must occur before the other. There are no threats or concurrency in the plan of Figure 3, so edges representing causal links are added from the start of each action to the end of the previous action.

Since *D* contains all temporal constraints given in the domain, it can be used to determine when *P* becomes unsafe. This procedure is given in Figure 4. In (Dechter, Meiri, & Pearl 1991) it is proved that the absolute bounds on the temporal distance between any two time points represented by nodes *a* and *b* (assuming $a \prec b$) in *D*, is given by the interval [-1 $\times$ *(weight of shortest path from b to a), weight of shortest path from a to b*]. The shortest path can be found using an algorithm such as the *Bellman-Ford* single source shortest path algorithm with a runtime of $O(|V||E|)$ (Cormen *et al.* 2001). In Figure 3(b) we see that the duration of the fly action is expressed by the interval [45, 90]. However, using the shortest path method (step 2 in Figure 4), it is found that the absolute bounds on the duration of the fly action are expressed by the interval [45, 80]. This indicates that if the fly action takes more than 80 time units, the rest of the plan becomes unsafe. To have a safe solution, a contingency must be generated that can reach the goal safely when the fly action takes more than 80 time units. The last action which makes the rest of the plan unsafe is found by looping through the steps in reverse order as shown in step 1 in Figure 4. If an action is found to be safe in line 3 the domain and the corresponding distance graph are updated to provide topmost flexibility to the earlier actions. If the execution of an action is not safe for a certain duration, the domain and problem are modified so that the action minimally takes that duration. A new plan that meets the new constraints is sought for at lines 7 through 9. If the new plan shares a head with the current plan, a contingency plan is formed and returned to the top level algorithm (at step 11). Otherwise, the new plan is returned to the top level algorithm to becomes a new seed plan (step 13). In the

Execution Time	Action
30	fly_airport2_airport1
76	shuttle_hotel_airport2
107	register_for_conference

(a)



(b)

Figure 3: (a) A seed plan for the problem in Figure 1. Note that the times given by the seed plan assume actions require their minimum durations. (b) The distance graph for the seed plan in *(a)*, incorporating temporal uncertainty. For clarity, only the most important edges are shown.

example problem, a contingency plan is formed (Figure 5(a)).

To verify that the the plans generated are sound assume that there is a plan P which has been generated. Every time a possibility of failure is detected in P, the domain is modified in such a way that P is invalidated. Therefore, P will not be generated again. This leaves 2 possibilities: either a different plan P' will be found, or no plan will be found. If no plan is found, then there is not a 100% safe solution for this problem. If P' is found, it will either be a new seed plan (totally replacing P) or contain actions that can be used to form a branch on P. The ability to replace the seed plan with a new one in line 13 allows the hill-climbing to escape local maxima because it might not always be possible to repair the first seed plan by adding contingency branches and it is better to start with a new plan. Once all actions in P have been verified, the temporal contingency plan (TCP) is safe. In the next section, we formally explain a data structure that can be used to represent TCPs.

Temporal Contingency Networks

We represent TCPs using a new data structure called a *Temporal Contingency Planning Network* (TCPN). TCPNs are an extension of STNs and are inspired by the STPU model defined in (Vidal & Fargier 1999). TCPNs extend STNs in two dimensions. First, interval durations are labeled as user assignable or not; second, some nodes represent decisions based on observations of time to enable the representation of TCPs. Figure 5(b) depicts a TCPN for the TCP in part (a).

Formally, a TCPN is a quadruple <**T, O, E, B**>. **T** is a set of nodes representing start and end times of actions. A node representing the absolute start time is also included in **T**. Each node in **T** is referred to as a *time point*. Nodes in **T** that are not included in all paths of execution contain a context label (Peot & Smith 1992) identifying the branch of execution they belong to. The oval nodes in the figure belong to **T**. The shuttle and taxi nodes contain context labels because these actions do not belong to all paths of execution. **O** is a (possibly empty) set of observation nodes representing decisions about which subsequent actions to execute. Observations of time are assumed to be executable at any time (no preconditions) and instantaneous; and should be executed immediately after the preceding time point. In the figure, the diamond represents an observation node. **E** is a set of interval labeled edges representing constraints between time points. Edges in **E** can be marked as unassignable, assignable, or unmarked. A TCPN with observation

```
MAKE-PLAN-SAFE (Plan P, DistanceGraph DG, D, I, G, M)
1:  for i = downto 1 in P
2:    maxAllowedDuration ← SHORTEST-PATH-DISTANCE(s_i, e_i, DG)
3:    if maxAllowedDuration ≥ max-d of i
4:      DG, D ← DG, D updated to constrain i to always require max-d of i
5:      DG, D ← DG, D updated to constrain i to always start at latest possible time that allows max-d of i
6:    else
7:      newMinDuration ← maxAllowedDuration + 1
8:      D ← D modified so that action i requires newMinDuration
9:      P_new ← generate plan with D_mod
10:     if P and P_new have the same steps through step i
11:       return a contingency plan created out of P and P_new
12:     else
13:       return P_new
```

Figure 4: The MAKE-PLAN-SAFE algorithm.

nodes is safe if all the possible paths are safe. The non-bold edges in the figure belong to **E**. Edges representing assignable durations are marked with *a* and those representing unassignable durations are marked with *u*. Edges with intervals representing an exact amount of time (such as *Time 0 → start: fly*) are unmarked. **B** is a set of temporally labeled edges leaving observation nodes. The bold edges in the figure belong to **B**. As shown, these edges are given a label indicating when each branch can safely be taken. This data structure provides a rich context for reasoning about TCPs.

Experiments and Discussion

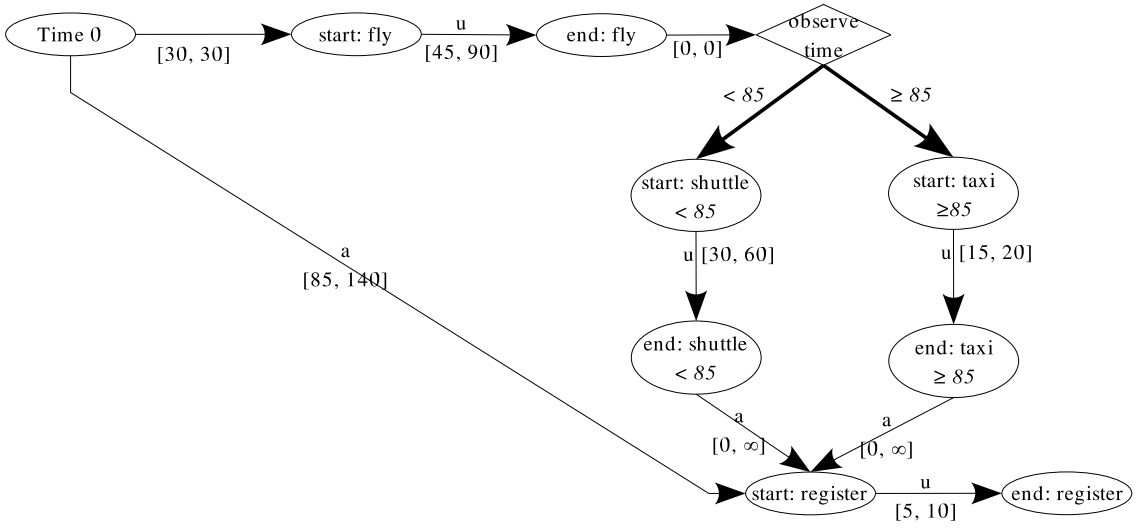
In this section we provide preliminary experimental results. To the best of our knowledge, there are no planners that prepare contingency branches based on time. We therefore designed our experiments to show that our algorithm works and to help identify the ways in which it can be improved. LPG-*TD* (Gerevini *et al.* 2004) was the planner that we used for generating seed plans and branches. We chose LPG-*TD* because it can handle the timed initial literals of PDDL2.2 and can optimize for temporal and nontemporal metrics. All the experiments were performed on a machine with a 3.0GHz Pentium 4 CPU and 1GB of RAM. We tested PHOCUS-HC with 2 different domains, the conference domain and an evacuation domain. Less than two seconds of time were required to produce any of the conditional plans which contain from 1-3 branches and up to 13 steps. Small domains and problems were used in these preliminary experiments to more easily control branching in the plans and verify the correctness of the results.

We tested with both a sequential and a parallel version of the conference domain. The sequential version consists of traveling from home to a conference and requires both ground and air transportation with the objective of arriving at the conference venue in time to register while minimizing money spent. The domain used for testing includes two different flight paths from the home airport to the conference city airport and three different options for ground transportation from the conference city airport to the conference venue. The parallel version of the conference domain also includes actions to read a short and/or long paper and grade exams, which must complete before registering for the conference. Also, the objective for the parallel problems is modified to prefer reading the long paper. The costs and durations of actions were varied to produce differing conditional plans.

The second domain that we tested is an evacuation domain motivated by the difficulty involved in planning to successfully evacuate during a natural disaster, like flooding. Complications arise when deciding the best way to allocate available resources (buses, emergency vehicles, helicopters, etc.) to save the most lives. Because it is not always possible to save all lives, oversubscription becomes an issue in this domain. In our simplified version of the evacuation domain there are one bus and one helicopter available to evacuate a school and a hospital with the objective being to save the largest number of lives. The children at school and most of the patients in the hospital must be evacuated by bus while the critical patients at the hospital can only be evacuated by helicopter. It takes two trips to evacuate all of the critical patients at the hospital and if the first trip takes too long, there are less critical patients to

```
At time 30: fly_airport2_airport1
IF (time < 85)
   Before time 85: shuttle_hotel_airport2
   Before time 140: register_for_conference
ELSE
   Before time 120: taxi_hotel_airport2
   Before time 140: register_for_conference
```

(a)



(b)

Figure 5: (a) A TCP for the problem in Figure 1. (b) The TCPN for the plan in *(a)*.

evacuate on the second trip. The amount of time required for each action was varied to produce different conditional plans.

In addition to generating the conditional plan using PHOCUS-HC, we also generated plans using LPG-*TD*, assuming all actions always require their minimum, maximum, and average durations respectively (min, max and average plans respectively). The max plan and the conditional plan always succeed, but may have a lower utility than the min and average plans. However, the min and average plans are riskier. To factor in both safety and utility, expected utility (EU) was calculated for each plan as $\sum_b probability(b) \times utility(b)$ where b is a complete branch or path that can be taken in a plan. Table 1 shows a comparison of the EU for the four different plans generated for each problem. In each instance, the conditional plan had the highest EU. In cases where probability of success was very low, as in the min version of c-2a, the EU was negative because the largest amount of utility was gained by reaching the final goal which would rarely happen with this plan. It should also be noted that in this domain, the conditional plan gains only a small amount of EU over the avg and max plans because the reward for achieving the goal is much larger than the transportation expenses.

Related work

The main framework of our algorithm is very close to Just-In-Case (JIC) scheduling (Drummond, Bresina, & Swanson 1994). The JIC scheduler analyzes a seed schedule, finds possible failure points, and inserts contingency branches so that valuable equipment time is not lost when an experiment fails. Our work extends this framework to multiple planner goals, parallel plans, and nontemporal metrics, but does not consider probability of failure.

There are a number of domain independent planners that can handle durative actions. We used LPG-*TD* because it can optimize based on a nontemporal metric. Other temporal planners include TGP, a planner that uses mutual exclusion reasoning in a temporal context (Smith & Weld 1999); SAPA, a heuristic forward chaining planner (Do & Kambhampati 2002); HSP, a heuristic planner with time and resources (Haslum & Geffner 2002); and CPT, an optimal temporal POCL planner based on constraint programming (Vidal & Geffner 2004). Tsamardinos et al. describe an

prob	min	avg	max	cond
c-1	480.00	480.00	480.00	480.00
c-2a	-467.33 *(0.03)*	450.00	450.00	451.00
c-2b	177.78 *(0.55)*	177.78 *(0.55)*	280.00	391.11
c-3a	-467.33*(0.03)*	189.50 *(0.70)*	380.00	430.00
c-3b	-348.51 *(0.02)*	161.11 *(0.55)*	280.00	541.66
c-3c	177.78 *(0.55)*	177.78 *(0.55)*	250.00	378.22
pc-1	510.00	510.00	510.00	510.00
pc-2a	-437.33 *(0.03)*	480.00	480.00	481.00
pc-2b	207.78 *(0.55)*	207.78 *(0.55)*	310.00	421.11
pc-2c	504.71 *(0.82)*	504.71 *(0.82)*	490.00	506.47
pc-3a	-437.33 *(0.03)*	219.50 *(0.70)*	410.00	460.00
pc-3b	-318.51 *(0.02)*	191.11 *(0.55)*	310.00	571.66
pe-1	140.00	140.00	140.00	140.00
pe-2a	84.89 *(0.31)*	84.89*(0.31)*	80.00	98.67
pe-2b	132.67 *(0.27)*	138.00	138.00	138.53
pe-3a	91.33 *(0.08)*	78.00	78.00	97.20
pe-3b	130.50 *(0.05)*	136.90 *(0.09)*	137.00	137.96

Table 1: EU for min, avg, max, and conditional plans for each problem. When probability of success is less than 1, it is given in parenthesis after EU. Number in problem name denotes number of branches in conditional plan, assuming a plan without conditions has 1 branch. Problems beginning with p contain parallel steps. Problems with a c are conference problems, while problems with an e are from the evacuation domain.

algorithm for merging existing plans with assignable durations and nontemporal conditional branches (2000). We plan to extend our algorithm with their plan merging framework.

Tempastic (Younes & Simmons 2004) is a planner that models continuous time, probabilistic effects, probabilistic exogenous events and both achievement and maintenance goals. It uses a "generate-test-debug" algorithm that generates an initial policy and fixes the policy after analyzing the failure paths. In producing a better plan, the objective is to decrease the probability of failure. Nontemporal resources are not modeled. Mausam and Weld (2005) describe a planner that can handle actions that are concurrent, durative and probabilistic. They use novel heuristics with sampled real-time dynamic programming in this framework to generate policies that are highly optimal. The quality metric includes makespan but nontemporal resources are not modeled in the planning problem. Prottle (Little, Aberdeen, & Thiebaux 2005) is a planner that allows concurrent actions that have probabilistic effects and probabilistic effect times. Prottle uses effective planning graph based heuristics to search a probabilistic AND/OR graph consisting of advancement and placement nodes. Prottle's plan metric includes probability of failure but not makespan or metric resources.

Conclusions and Future Work

We have presented a framework for characterizing and directly dealing with temporal uncertainty. We define temporal uncertainty by assigning actions interval durations, rather than single point durations. We have implemented our hill-climbing approach in a planner called PHOCUS-HC. The planner starts by making an optimistic assumption that all actions complete as quickly as possible and generates a seed plan with high utility that may become invalid when the assumption proves wrong. It then analyzes the plan and generates more costly contingency branches to be executed only when actions in the seed plan run long enough that an unsafe situation occurs. In addition to generating contingency branches, our hill-climbing approach has the advantage of being able to replace the entire seed plan when adding a contingency branch is not possible, or when starting with a new seed plan yields a better expected utility. In the current version of PHOCUS-HC, a uniform distribution is assumed over all uncertain action durations. In the future we plan to further develop the implementation to allow user specified distributions. Also, the current implementation always searches until a plan with 100% safety is found. We plan to improve PHOCUS-HC so that the user can choose the level of safety that is required. We would also like to extend our work to be able to handle actions with uncertain effects and uncertain consumption of nontemporal resources.

Acknowledgements

We would like to thank Alfonso Gerevini, Alessandro Saetti, Ivan Serina, and Paolo Toninelli for making LPG-td available. Janae N. Foss' research was supported by the Harriett G. Jenkins Predoctoral Fellowship Program and a grant from the Michigan Council of Women in Technology.

References

Bresina, J. L.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D. E.; and Washington, R. 2002. Planning under continuous time and resource uncertainty: A challenge for ai. In *Proc. 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 77–84.

Cormen, T. H.; Stein, C.; Rivest, R. L.; and Leiserson, C. E. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Do, M. B., and Kambhampati, S. 2002. Planning graph-based heuristics for cost-sensitive temporal planning. In *Proc. 6th Int. Conf. on AI Planning & Scheduling*.

Drummond, M.; Bresina, J.; and Swanson, K. 1994. Just-incase scheduling. In *Proc. 12th National Conf. on Artificial Intelligence*, 1098–1104.

Edelkamp, S., and Hoffman, J. 2004. PDDL2.2: The language for the classical part of the 4th international planning competition. Technical Report 195, Computer Science Department, University of Freiburg.

Gerevini, A.; Saetti, A.; Serina, I.; and Toninelli, P. 2004. Planning in PDDL2.2 domains with LPG-*TD*. In *International Planning Competition booklet (ICAPS-04)*.

Haslum, P., and Geffner, H. 2002. Heuristic planning with time and resources. In *Proc. 6th European Conf. on Planning*.

Little, I.; Aberdeen, D.; and Thiebaux, S. 2005. Prottle: A probabilistic temporal planner. In *Proc. 20th National Conf. on Artificial Intelligence (AAAI-05)*.

Mausam, and Weld, D. S. 2005. Concurrent probabilistic temporal planning. In *Proc. 15th International Conf. on Automated Planning and Scheduling (ICAPS-05)*.

Mausam; Benazara, E.; Brafman, R.; Meuleau, N.; and Hansen, E. 2005. Planning with continuous resources in stochastic domains. In *Proc. 19th International Joint Conference on Artificial Intelligence Scheduling (IJCAI-05)*, 1244–1251.

Nilsson, N. J. 1980. *Principles of Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proc. 1st International Conf. on Artificial Intelligence Planning Systems*, 189–197.

R-Moreno, M. D.; Oddi, A.; Borrajo, D.; Cesta, A.; and Meziat, D. 2002. Integrating hybrid reasoners for planning and scheduling. In *The Twenty-First Workshop of the UK Planning and Scheduling Special Interest Group*, 179–189.

Smith, D. E., and Weld, D. 1999. Temporal planning with mutual exclusion reasoning. In *Proc. 16th International Joint Conf. on Artificial Intelligence*.

Tsamardinos, I.; Pollack, M. E.; and Horty, J. F. 2000. Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In *Proc. 5th International Conf. on Artificial Intelligence Planning and Scheduling*, 264–272.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)* 11(1):23–45.

Vidal, V., and Geffner, H. 2004. Branching and pruning: An optimal temporal POCL planner based on constraint programming. In *Proc. 19th National Conf. on Artificial Intelligence*, 570–577.

Younes, H. L., and Simmons, R. G. 2004. Policy generation for continuous-time stochastic domains with concurrency. In *Proc. 14th International Conf. on Automated Planning and Scheduling (ICAPS-04)*.