Techniques for Generating Optimal, Robust Plans in the Presence of Temporal Uncertainty

Janae N. Foss*
Department of Computer Science
Michigan Technological University
1400 Townsend Drive
Houghton, MI 49931
jnfoss@mtu.edu

Abstract

Planning under uncertainty has been well studied, but usually the uncertainty is in action outcomes. This work instead investigates uncertainty in the amount of time that actions require to execute. In addition to this temporal uncertainty, the problems being studied must have robust solution plans that are optimized based on an objective function. This extended abstract details two iterative approaches that have been used to solve these type of problems and discusses future work including over-subscription of goals and MDP approaches.

Introduction

Uncertainty applies to several aspects of planning problems and many planners have been built that prepare contingency plans when actions may affect the world in uncertain ways (Bresina *et al.* 2002). However, less work has been done with planners that assume action durations are uncertain. One approach to dealing with this type of uncertainty is to take a pessimistic view of the world, assume a worst case scenario, and find conservative plans that are likely to execute to completion regardless of the amount time consumed by the actions in the plan. This approach is often undesirable as it leads to missed opportunities and slack time in the plan when actions complete quickly (Bresina *et al.* 2002). For example, assume that a Mars rover has to move from point *a* to point *b* and use either a slow, high resolution camera or a fast, low resolution camera to take an image of a rock at point *b*. Given that travel time is uncertain, a conservative planner may recognize that in the worst case there will not be enough time to use the high resolution camera, and thus choose to always use the low resolution camera. This plan is robust, but when the rover travels quickly the opportunity of getting a high resolution image is not realized and the rover may undesirably be left idle for some period of time. My research focuses on finding ways to create robust plans where suboptimal actions are taken when time dictates, but optimal actions are executed when time allows.

*Supported by NASA Harriett G. Jenkins Pre-Doctoral Fellowship Program.

Problem Specification

I am considering a class of problems with solutions that combine temporal uncertainty, optimality, and robustness, each of which is difficult to deal with individually and more so in combination. In this class of problems, action durations cannot be specified exactly and are represented by a closed interval [*min-d*, *max-d*], specifying the lower and upper bounds for the duration. Under this model, the actual duration required for an action is only known through observation after the action has executed. These duration intervals complicate the problem because solution plans are ranked by an objective function and the optimal solution is only attainable when actions complete quickly. This means that solutions found with the pessimistic assumption that all actions require *max-d* will be suboptimal, but optimal solutions found under the optimistic assumption that all actions require *min-d* (or any value less than *max-d*) are not guaranteed to execute to completion. The best solutions for these problems must be robust plans that are guaranteed to run to completion regardless of the amount of time actions require to complete. Plans that are robust in this sense are classified as *safe*. Considering all of these attributes, a *temporally uncertain planning problem* is defined as a quadruple <**D**, **I**, **G**, **M**>, where **D** is a domain description that lists the available actions (including interval durations and temporal constraints), **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 for ranking plans.

Temporal Contingency Planning

One way to create optimal plans that are also robust in the face of temporal uncertainty is to build temporal contingency plans (i.e., plans with contingency branches that are taken based on the observed time at execution). At present, I have developed two related iterative approaches for generating temporal contingency plans. They differ in that one is a greedy algorithm and the other is a hill climbing algorithm. These algorithms are implemented in the planners PHOCUS-G (Foss & Onder 2005) and PHOCUS-HC (Foss & Onder 2006).

Both approaches follow a Just-In-Case style algorithm (Drummond, Bresina, & Swanson 1994) where a seed plan is generated, the points where it is likely to fail are located, and then contingency branches are inserted (when available)

at those points (Fig. 2). The two algorithms differ in the way that repairs are found when failure is possible. To generate the seed plan (line 1 in Fig. 2), temporal uncertainty is removed from the problem. This allows generation of plans using any planner that can handle durative actions, timed initial literals, and optimize based on an objective function[1]. Because it is assumed that the optimal plan is only attainable when actions complete quickly, *min-d* is assigned as the duration of each action. The resulting seed plan *P* returned by such a planner is temporally deterministic. My algorithm factors temporal uncertainty back in by converting *P* to a directed, edge-weighted graph called a *distance graph DG*, thus expressing *P* as a simple temporal network (STN) (Dechter, Meiri, & Pearl 1991). Figure 1 (b) shows a distance graph for a plan from a simplified rover domain. This conversion is described in detail in earlier work (Foss & Onder 2005).

Since *DG* contains all temporal constraints given in the domain, it can be used to determine when *P* becomes unsafe (line 11) (Dechter, Meiri, & Pearl 1991). In the loop that contains line 11, the plan is analyzed one step at a time to find the latest action *i* which makes the rest of the plan unsafe. If an action is found to be safe in line 11, the domain and the corresponding distance graph are updated to provide topmost flexibility to the earlier actions (lines 12,13), assuming the action requires its maximum duration. Otherwise, modifications are made so that *i* minimally uses the duration that causes the plan to fail and a new plan meeting the new constraints is sought for in one of the REPAIR-PLAN-* algorithms.

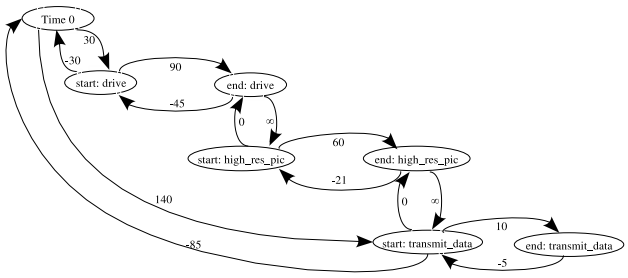
Figure 1 (c) shows how the distance graph in (b) has changed after several actions have been analyzed. First, it was found that the transmit_data action could execute to completion if it required its maximum duration of 10 time units. The distance graph was then updated to constrain transmit_data to always take 10 time units by changing the weight of the arc from end:transmit_data to start:transmit_data to -10. Next, it was found that even with the updated transmit_data constraint, high_res_pic could execute safely with any duration in its interval. The figure shows that this action was also then constrained to require its maximum duration. However, when the drive action was analyzed, it was found that the shortest path from start:drive to end:drive had a weight of 50 (this path is bolded in the figure). This indicates that high_res_pic and/or transmit_data may not have enough time to complete if drive executes for longer than 50 time units. At this point, a repair function must be called.

To apply the greedy approach, REPAIR-PLAN-G is called (Fig. 3). In this version, the initial conditions of the world are changed to represent the state of the world after all actions up to and including *i* are executed, assuming that *i* requires the amount of time that would cause failure in the original plan. Then, an attempt is made to generate a new plan which could be added as a temporal contingency branch on the original plan. If no such plan is found, this algorithm returns null and thus finds no solution. When applying the greedy approach, the seed plan is optimal in respect to the objective function. This optimal plan is never abandoned and it is augmented with branches that are each optimal, given the constraints used when generating them.

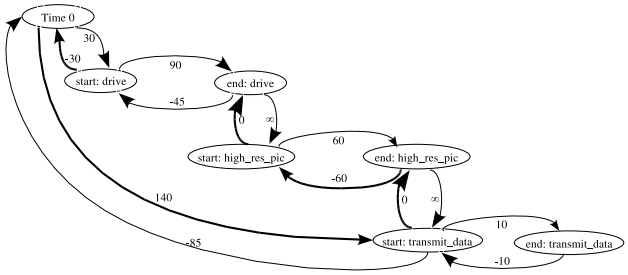
For the hill-climbing approach, REPAIR-PLAN-HC is

Execution Time	Action
30	drive_to_target
76	take_high_res_pic
107	transmit_data

(a)



(b)



(c)

Temporal Contingency Plan

at time 30: drive
if time < 81
  high_res_pic
  transmit_data
else
  low_res_pic
  transmit_data

(d)

Figure 1: (a) A seed plan for a problem from the rover domain. 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. (c) The updated version of the distance graph in (b) after the transmit_data, high_res_pic, and drive actions have been analyzed. The bold arcs show the shortest path from start:drive to end:drive. (d) The temporal contingency plan generated by both the greedy and hill-climbing approaches.

[1]Currently LPG-td(Gerevini *et al.* 2004) is being used for this step because it handles durational actions and timed initial literals (used for specifying deadlines), creates parallel plans, and considers the objective function at planning time. It has also performed well in the International Planning Competition.

called (Fig. 4). Instead of modifying initial conditions, in this case the domain is modified so that *i* minimally requires the amount of time that would cause failure in the original plan. Then, an entire new plan is generated. If the new plan shares a head with the current plan, a contingency plan is formed. Otherwise, the new plan is returned and replaces the seed plan. As with the greedy approach, the initial seed plan is optimal with respect to the object function. However, the hill-climbing approach will abandon and replace the original seed plan either if no branches can be added to the seed plan to make it safe, or if a new seed plan has higher utility than the plan created by adding a branch to the old seed plan. In this way, the safest branch of the plan is optimized.

Both the greedy and hill-climbing versions of the algorithm benefit from the fact that they allow parallelism. This is especially important when deadlines are taken into consideration. Each approach has individual advantages, also. Intuitively, the greedy approach is faster when contingency branches can be added to repair the optimal plan. There are two related factors that contribute to this. First, the domain is modified so that the head of the plan will not be regenerated, restricting the search space. Second, because a contingency branch is shorter than a full plan, it is faster to generate it than to regenerate the entire plan as is done in the hill-climbing algorithm. However, the greedy approach fails to find any solution when no contingency branches can be added to the optimal plan. Since the hill-climbing approach always regenerates the whole plan, it is able to escape local minima/maxima. Also, the greedy algorithm may start with an optimal plan that is unlikely to be executed and augment this plan with very undesirable branches that are likely to be executed. In this situation the hill-climbing algorithm would abandon the optimal plan and find a sub-optimal, but likely to succeed plan that would have higher utility than the branches in the greedy algorithm's plan. Each algorithm has been independently implemented and tested (Foss & Onder 2005; 2006) and more experiments are planned to verify that these intuitive conclusions hold.

Related Work

The main framework of this 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. My work extends this framework to multiple planner goals, parallel plans, and nontemporal metrics, but does not currently consider probability of failure.

Several planners dealing with problems similar to those I am working with have been developed recently. 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.

```
PHOCUS-* (D, I, G, M)
1: P_0 ← GENERATE-SEED-PLAN (D, I, G, M)
2: P_current ← P_0
3: loop do
4:    DG ← CONSTRUCT-DISTANCE-GRAPH(P_current,D,I)
5:    if SAFE-PLAN (P_current, DG, D, I, G, M) return P_current
6:    P_next ← MAKE-PLAN-SAFE (P_current, DG, D, I, G, M)
7:    if P_next is null return failure
8:    P_current ← P_next
MAKE-PLAN-SAFE (Plan P, DistanceGraph DG, D, I, G, M)
9: for i = downto 1 in P
10:   maxAllowedDuration ← SHORTEST-PATH-DISTANCE(s_i, e_i, DG)
11:   if maxAllowedDuration ≥ max-d of i
12:      DG, D ← DG, D updated to constrain i to always require max-d of i
13:      DG, D ← DG, D updated to constrain i to always start at latest possible time that allows max-d of i
14:   else
15:      return REPAIR-PLAN-*(i,Plan P, D, I, G, M)
```

Figure 2: The shared PHOCUS-* algorithms.

```
REPAIR-PLAN-G (i, Plan P, D, I, G, M)
1: newMinDuration ← maxAllowedDuration + 1
2: I_mod ← I modified to represent the world after all steps up to i
          have completed and i has consumed newMinDuration
3: P_new ← generate plan with D, I_mod, G, M
4: if P_new is not null
5:    return a contingency plan created out of P and P_new
6: else
7:    return null
```

Figure 3: The REPAIR-PLAN-G algorithm. A greedy algorithm for finding temporal contingency branches.

Mausam and Weld (2005) describe a planner that can handle actions that are concurrent, durative and probabilistic. They use novel heuristics with sampled realtime 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. Schaffer, Bradley and Chien (2005) developed a probabilistic approach for reasoning about uncertainty in continuous activity duration and resource usage. Their approach does not include contingency planning. They have shown robustness improvements over traditional non-probabilistic methods.

Future Work

Temporal contingency planning improves on conservative planning techniques by including the most conservative plan as the least desirable contingency branch, executed only

REPAIR-PLAN-HC (i, Plan P, D, I, G, M)

1: $newMinDuration \leftarrow maxAllowedDuration + 1$
2: $D_{mod} \leftarrow D$ modified so that action i requires $newMinDuration$
3: $P_{new} \leftarrow$ generate plan with D_{mod}, I, G, M
4: **if** P and P_{new} have the same steps through step i
5: **return** a contingency plan created out of P and P_{new}
6: **else**
7: **return** P_{new}

Figure 4: The REPAIR-PLAN-HC algorithm. A hill-climbing algorithm for finding temporal contingency branches.

when more desirable options may cause failure. The techniques currently implemented begin with an optimistic assumption that actions complete quickly and assume a uniform distribution over the uncertain duration interval. As I continue to work on these iterative approaches, I plan to consider what happens when the distribution is not uniform. The most likely case is that action durations will have a Gaussian distribution where most of the probability mass lies in the center of the interval. Considering this, it does not make sense to start with the assumption that each action requires only its minimum duration because that will result in a plan that is unlikely to execute to completion. Instead, it will be better to start with a value from the duration that is likely to occur, based on the given distribution. In this situation, opportunity branches can be added for when actions complete faster than expected, and contingency branches can be added for when actions run long. This may be a good anytime approach to be applied when there is a limited amount of time available for planning. In this circumstance it is important to spend the time available for planning to generate branches that will improve the plan in a significant way. By incorporating non-uniform distributions, I will be able to better determine when to stop branching because the expected utility gained is too small.

In addition to the rover domain, I have been working with problems from a travel domain and an evacuation domain. In the travel domain, the goal is to travel from home to some destination within a given time constraint. There are several different ways to reach the destination, but some modes of transportation are more expensive and the objective function in this domain is to minimize the amount of money spent. The challenge is that more expensive options, such as taking a taxi, are faster than less expensive options, like taking a bus. Optimally, the bus would be taken, but if this action comes after a flight that is running late, there may only be enough time to take the taxi. In the evacuation domain, the goal is to evacuate as many people as possible within a given period of time. This is further complicated by the fact that there are intermediate deadlines for rescuing different groups of people. As such, it is easy to create problems where it is not possible to evacuate all people, resulting in over-subscribed goals (Smith 2004).

Over-subscription is also an issue in the rover domain and most real world problems. I would like to develop techniques that directly address this issue. One approach is to simply achieve more goals when actions complete quickly and only the highest priority goals, otherwise. Another possibility is that entirely disjoint sets of goals may be attained on different branches of the plan.

Finally, I would like to investigate MDP approaches to solving planning problems with temporal uncertainty. Unlike the iterative planning approaches, MDPs do not naturally allow parallel actions. Even so, MDPs can be useful in this context because they naturally deal with uncertainty and take cost and rewards into account. One challenge in using MDPs to solve these type of problems is how to represent states when time is a factor. A naive approach is to include time in the state and thus have one state for each possible time increment. However, this would very quickly cause a blow-up in the size of the state space. It is likely that many states in this naive approach would be identical, only differing in time stamp. I plan to investigate ways to group states by time to reduce the number of states without sacrificing quality in the solution policy.

References

Bresina, J.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; and Washington, R. 2002. Continuous time and resource uncertainty: A challenge for AI. In *18th Conference on Uncertainty in Artificial Intelligence*.

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

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

Foss, J., and Onder, N. 2005. Generating temporally contingent plans. In *IJCAI 2005 Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains*.

Foss, J., and Onder, N. 2006. A hill-climbing approach for planning with temporal uncertainty. In *FLAIRS 2006 Conference*. To appear.

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)*.

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)*.

Schaffer, S. R.; Clement, B. J.; and Chien, S. A. 2005. Probabilistic reasoning for plan robustness. In *Proc. IJCAI 2005*.

Smith, D. 2004. Choosing objectives in over-subscription planning. In *Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 393–401.

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)*.