Planning Under Temporal Uncertainty Using Hindsight Optimization

Scott Kiesel[1] and **Wheeler Ruml**[1,2]

[1]Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA

[2]LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse FRANCE

Abstract

A robot task planner must be able to tolerate uncertainty in the durations of commanded actions and uncertainty in the time of occurrence of exogenous events. Sophisticated temporal reasoning techniques have been proposed to deal with such issues, although few existing planners support them. In this paper, we demonstrate the capabilities of a much simpler technique, hindsight optimization, in which uncertainty is handled by using sampling to generate deterministic planning problems that can be solved quickly. We find that sophisticated temporal reasoning is not required to handle many simple tasks. In comparison with a traditional temporal planner architecture, hindsight optimization is much simpler to implement while staying closely integrated with execution. It serves as a flexible baseline against which more complex methods can be compared.

Introduction

In many real-world domains, such as robotics, a planning agent does not have complete knowledge of the world state or precise control over action outcomes of actors and processes. Incomplete world knowledge, stochastic actions and exogenous events are obstacles a planning agent must tackle in many useful robotic applications.

Many planners make the assumption that an action, such as *pickup*, will have a deterministic outcome and duration. This is a fine assumption that makes planning much easier to reason about. However when the resulting plan is executed, actions can fail or take longer than anticipated.

Consider the domain of a robot office assistant. When issuing the simple task of picking up a set of keys from your desk, the planner will quite quickly emit the plan: *pickup(keys)*. When this plan is executed, the pickup action might fail. If the action fails, then the execution certainly will not result in a goal state. The ability to reason about action outcome uncertainty becomes very important when a plan will is intended to be executed.

Perhaps the goal state is slightly more interesting and the agent should pickup your keys and give them to you when you leave to go home. The other half of the assumption about actions that many planners make is that their duration is a known constant. However, depending on the starting pose of the robot office assistant or the position of the keys, this simple *pickup* can have a varying range of execution durations. If it could take anywhere between 1 minute and 10 minutes for your keys to be picked up, you might appreciate a planner that can take this range into consideration, instead of causing you to be 10 minutes late going home. If a planner only assumes the best case for action durations, it is easy to see that the agent could be late to a rendezvous. On the other hand, if the planner only assumes the worst case for action durations, the agent may spend time waiting at a rendezvous point unnecessarily. It could instead be performing more productive tasks with this time. The ability to handle action duration uncertainty becomes very important if a productive and punctual agent is desired.

An even more realistic situation for a robot office assistant to face is the task of retrieving your keys from a set of possible locations. It is not uncommon to forget exactly where you left your keys. Many planners however, require that the exact location of the keys be known when planning begins. If you need to manually search out your keys to simply set the initial state for your planner you might as well skip using your planner because you have already found your keys. It is important for a planner to be able to handle location uncertainty if the exact state of the world is not known.

As hinted at in the previous example scenarios, the agent may not be the only entity in its world. Other agents may exist, such as yourself, and interaction with these other agents is only natural. These other agents are not necessarily under the control of the same planner as your robot office assistant. The world can be affected and changed outside of the control of the planner. Maybe you found your keys while trying to fully annotate an initial state for your planner and those are no longer a goal for the agent. Let's consider a new goal, before you leave the office you would like your robot office assistant to give you a coffee for the trip home. The coffee is not essential for you to get home, but it makes the trip significantly more pleasant. As such, you are willing to wait for 10 minutes before you depart without your coffee. As a human, your timing is not always exact so you might leave sometime between 5pm and 5:30pm. If you would like to get your coffee frequently before you leave the office it is important for a planner to be able to reason about the temporal uncertainty associated with interactions involving other agents and events exogenous to the planner.

Adding a single one of these three aspects of uncertainty

to a domain renders many planners inapplicable. Adding all three types of uncertainty further reduces the number of applicable planners. Those algorithms previously proposed to handle these uncertainties are complicated and can rely complex data-structures such as a Simple Temporal Network With Uncertainty (STNU) (Morris, Muscettola, and Vidal 2001). Many of them also require reasoning about all of the unknown factors at once requiring complex world representations.

In this paper we introduce the Temporal-Uncertainty Hindsight Optimization Planner (TU-HOP), a simple and straightforward approach that extends previous work on hindsight optimization. Instead of trying to manage the various uncertainties directly, we employ a basic sampling strategy and a deterministic planner. TU-HOP begins with a set of beliefs about the initial world state. This belief state manages certain and uncertain information about locations, arrivals and departures of objects and agents and expected action outcomes and durations. Given the current belief, a set of deterministic world samples consistent with the belief state are generated and then solved by the deterministic planner. Using the resulting solutions, the next action is chosen based on solutions maximizing overall expected reward. This action is executed, the belief about the world is updated based on the action's result and the process starts again.

We show that TU-HOP is simple to understand and implement. We also show that TU-HOP is very capable of solving problems containing uncertain action outcomes and durations, uncertain object and agent locations as well as exogenous events. Its simplicity and capability make it a flexible baseline against which future temporal uncertainty planning research can be compared.

Previous Work

There is wealth of literature on deterministic domain dependent and independent planners. We are able to leverage this previous work, as others have, to incorporate well researched deterministic planning ideas and concepts into a larger framework (Yoon, Fern, and Givan 2007; Shani and Brafman 2011).

Yoon, Fern, and Givan (2007) incorporate a classical domain independent planner called FF (Hoffmann and Nebel 2001) into their planning framework called FF-Replan to solve problems with uncertain action effects. FF-Replan uses FF to find a plan to carefully constructed deterministic version of the problem. It then executes actions according to the plan until the executed action has an unexpected effect or the goal is achieved. If an unexpected effect is observed before achieving the goal, FF is called once again to construct a new plan from the current state.

Temporal Planning and Execution

EUROPA (Barreiro et al. 2012) is class library and tool set for building planners within a temporal planning paradigm. It is a complicated architecture that handles time and resources and constructs plans offline. It is able to handle many temporal events using its modeling language NDDL. It relies on many handcoded internal modules and does not directly handle the uncertainties of object locations or action outcomes.

IxTeT (Ghallab and Laruelle 1994; Laborie and Ghallab 1995) is an complex offline planning and scheduling system that can handle time and resources by constructing partial order plans and resolving *threats* to the achievement of goals during planning. It strives to find a balance between the planning (what to do) and the scheduling (in what order to do it) addressing many domains in the intermediate spectrum between planning and scheduling. It however does not take into account the uncertainties that become evident during execution of plans.

Procedural Reasoning System (PRS) (Ingrand et al. 1996) is a system for supervision and control of autonomous mobile robots. This system is explicitly able to monitor plan execution and provide feedback on action execution to an underlying planning system. However, PRS still relies on a high level planner to provide the high level actions for it to execute. Integration of TU-HOP with PRS is a promising avenue for future research.

IxTeT-EXEC (Lemai and Ingrand 2003) is complex system that allows for execution control, plan repair and replanning. IxTeT-EXEC is an extension of the IxTeT planner integrated with PRS (and several other layers). IxTeT-EXEC is able to handle temporal constraints (inherited from IxTeT) as well as action failures and unpredicted action outcomes as reported by PRS. However, this is a very complicated system that is non-trivial to implement and does not address the aspects of temporal uncertainty of interest in this paper.

Simple Temporal Network with Uncertainty (STNU) (Morris, Muscettola, and Vidal 2001) are an extension of Simple Temporal Networks (STN) (Dechter, Meiri, and Pearl 1991). In many cases of interest in planning, an STNU would be used to determine dynamic controllability. If an STNU is dynamically controllable, then we can be assured that from the current state, the actions we plan to execute will not cause us to violate any future temporal constraints regardless of their outcome durations. Computing dynamic controllability, however, is N^3 in the general case.

Hindsight Optimization

Hindsight Optimization was originally developed for scheduling and networking problems (Chong, Givan, and Chang 2000; Mercier and van Hentenryck 2007; Wu, Chong, and Givan 2002) and has been used in a probabilistic planning setting (Yoon et al. 2008; 2010). In these previous applications, sampling was used to resolve uncertainty in the outcome of actions. Burns et al. (2012) used hindsight optimization to solve a problem where exogenous goals are arriving, which requires the agent to plan ahead and anticipate these arrival events. Most recently, hindsight optimization was applied to open world planning where sampling was used to resolve uncertainty in object existence and location as well as uncertainty in navigation graph connectivity (Kiesel et al. 2013).

Similar to most sampling techniques, the samples of generated possible worlds used in this paper are intentionally not exhaustive. They are intended to provide useful relative judgments on the expected value of actions. In hindsight

optimization, given a current world state, we are faced with the choice between all applicable actions. The first step in hindsight optimization is to generate possible world samples that could be true according to the current world belief state. Then in order to estimate the value of an action, we apply that action in each of the sampled possible worlds, find deterministic plans from each of the resulting states, and average over the resulting reward yielded by each plan. The action with the highest average plan reward over the sampled worlds is chosen to be executed.

More formally, we define the value of being in a state s_1 as the maximum expected reward over plans that extend from s_1. That is, the maximum reward over all possible future action sequences of the total reward over all possible future states:

$$V^*(s_1) = \max_{A=\langle a_1,\ldots,a_{|A|}\rangle} \; \underset{\langle s_2,\ldots,s_{|A|}\rangle}{E} \left[\sum_{i=1}^{|A|} R(s_i, a_i) \right]$$

where $R(s, a)$ represents the reward of performing action a in state s. Given our expectations about the uncertainties in the world, we would like to find the action sequence $A = \langle a_1, ..., a_{|A|}\rangle$ that maximizes the expected sum of action rewards. To compute V^* exactly, we would need to compute the expectation for each of exponentially many plans.

In hindsight optimization, we approximate the value function by exchanging expectation and maximization, so that we are taking the expected value of maximum-reward plans instead of the maximum over expected-reward plans:

$$\hat{V}(s_1) = \underset{\langle s_2,s_3,\ldots\rangle}{E} \left[\max_{A=\langle a_1,\ldots,a_{|A|}\rangle} \sum_{i=1}^{|A|} R(s_i, a_i) \right]$$

The expectation in this approximation of $V^*(s)$ can be approximated using sampling and fixed values for each of the uncertainties in each maximization. As in other applications of hindsight optimization, the stochastic elements have been reduced to known values by sampling. For each possible value that an uncertainty could take on in the expectation, the problem is to maximize reward given a known world, i.e., standard, deterministic, reward-maximizing planning. As the underlying deterministic problem becomes more difficult to solve with a standard deterministic planner, TU-HOP can employ a limited horizon planner. A limited horizon planner uses a time horizon which is simply a temporal value by which search depth is bounded. Setting this bound to infinity results an informed, full solution to the maximization problem, decreasing the horizon results in greedier behavior, only considering more immediate reward.

We define the Q-value to be the cumulative expected reward of taking an action a_1 in state s_1:

$$Q(s_1, a_1) = R(s_1, a_1) + \underset{\langle s_2,s_3,\ldots\rangle}{E} \left[\max_{A=\langle a_2,\ldots,a_{|A|+1}\rangle} \sum_{i=2}^{|A|+1} R(s_i, a_i) \right]$$

From this, we estimate the best action choice in s as $\max_a Q(s, a)$. Using this technique, we are said to be performing optimization with the benefit of 'hindsight' knowledge about how future uncertainty will be resolved. It is important to point out that this action choice strategy is an unsound reasoning technique (Yoon et al. 2010).

TU-HOP(W, N, H)
1. while true
2. for i from 1 to N do
3. $w_i \leftarrow sample\_world(W)$
4. foreach action a
5. $s' \leftarrow a(s)$
6. $r \leftarrow (\sum_{i=1} solve(s', w_i, H))/N$
7. $Q(s, a) \leftarrow R(s, a) + r$
8. $a_{best} \leftarrow argmax_a\, Q(s, a)$
9. res = execute(a_{best})
10. update(W, a_{best}, res)

Figure 1: The TU-HOP planner.

After the best action is executed, hindsight optimization updates its current belief state based on the outcome of its action and how it has affected the world. This newly updated belief state will be used in the next planning step.

Approach

In this paper, we extend the line of work on hindsight optimization to handle the three types of uncertainties previously discussed; uncertain action outcomes and durations, uncertain object and agent locations and exogenous events. TU-HOP is an online planner that interleaves search and execution, emitting single actions for the agents to execute at a time.

The pseudo-code in figure 1 provides a high level summary of the TU-HOP planner. The planner first receives three parameters, the first is the current belief about the world state, the second is the number of samples to be used and the third is the horizon with which to bound the deterministic solver. First, we generate a set of N possible worlds that are consistent with the planner's current belief about the world (lines 2–3). Each of these sampled worlds are deterministic versions of the current belief where all objects have known locations, actions have known outcomes and events have known start and end times. Next, for each action a in the domain, we consider the resulting state $s' = a(s)$ (line 5). Next, for each action a, we consider the resulting state $s' = a(s)$ (line 5). Then, each possible world w_i is initialized with the state s', generating a fully-known deterministic planning problem. Solving this problem provides an estimate of the reward from s'. The mean reward across the set of samples (line 6) along with the reward of the action $R(s, a)$ is used as the Q-value for each action a in the original state s (line 7). We then select the action with the maximum Q-value (line 8), this action is then executed (line 9. The result of this action, success, failure or an inaccurate belief is returned and the current belief of the world state is updated with this information (line 10). With this new belief about the state of the world, the planner returns to the beginning of the loop and executes another iteration.

Robot Office Assistant Domain

In this paper we focus on a specific domain where a set of controllable and non-controllable agents are able to navigate around a topological map containing objects. Controllable agents are able to *pickup*, *putdown*, and *give* objects. The give action is the exchange of one object in an agent's possession to another agent. All of the information about the domain and belief of the world state will now be discussed.

The first major entity in the world belief is a representation of the topological navigation graph. Each node in the graph is named and connected to another node in the graph via an edge. Each edge has a traversal success rate, a minimum and successful traversal time and a minimum and maximum failure time. The success rate represents the probability that traversing this edge will succeed. The minimum and maximum successful traversal times provide an expected interval of how long it will take to traverse the edge if the traversal is successful. Similarly, the minimum and maximum failure interval describes how long it will take for the attempted traversal to report failure.

The objects present in the world each have a unique name, two associated temporal intervals and a set of node locations. The first interval is the minimum and maximum expected arrival time for the object. This can be used to represent an object being dropped off by an agent outside the control of the planner. The second interval is the minimum and maximum duration the object is expected to remain usable in the world. This can be used to impose a deadline on when an object may need to arrive at its goal destination. The set of nodes following the intervals contains at least one node name. A single node represents complete certainty of the object's starting location. A set whose size is greater than one represents uncertainty of the object's starting location.

Agents are are very similar to objects with two major differences. The first is that an agent can be marked as outside of the control of the planner. This is useful if an agent is only "stopping by" to receive an object from another agent that *is* under the planner's control. The second difference is that each agent also has a number of grippers available to hold objects.

Goals can be one of three types different types. The first is a simple *goto* goal which tells the planner which agent needs to be moved to which location and what the reward for this goal is. The second type of goal is *move*, which tells the planner which object needs to be moved to which location and what the associated reward is for completing this goal. The last type of goal is *give* and tells the planner which object should be given to which agent and how much reward will be received for achieving this goal. Not all agents and objects need bee involved with a goal.

Lastly, there are four actions in the domain; *pickup*, *putdown*, *give* and *no-op*. Please keep in mind that the implicit *move* action is defined on an edge by edge basis in the graph. The motivation behind this is that some edges may be more difficult or simply take more time to traverse. Each action in the list has a success rate, minimum and maximum successful execution duration and a minimum and maximum failure duration with the exception of the *no-op* action. The success rate represents the probability that executing this action will succeed. The minimum and maximum successful execution times provide an expected interval of how long it will take to complete the action if the execution is successful. Similarly, the minimum and maximum failure interval describes how long it will take for the attempted execution to report failure. The *no-op* action is always successful and has a deterministic execution duration equal to the planned duration. The duration for each *no-op* is determined during planning. It is set to be the time from the current state, until the next occurring event. An event is simply the arrival or departure of an agent or object, or another agent completing its current action. Setting the duration in such a manner can render a deterministic planner incomplete in certain domains, but in this simple Robot Office Assistant Domain completeness is maintained when planning in the deterministic world samples.

A Closer Look

All of the domain instance information is managed and updated in the belief state of the TU-HOP planner throughout its execution. Before emitting any action to be executed, the TU-HOP enters its first planning iteration.

It begins by generating a set of possible deterministic world samples consistent with the current belief state. This means that in each sampled world any and all uncertainty is removed. This is achieved for the location uncertainty by picking, at random, one of the locations in each location list for the agents and objects. The action outcome uncertainty is resolved by using the success rate, success interval and failure interval for each action and constructing a deterministic mapping of times to outcomes and durations. This is accomplished by lazily querying the action in the deterministic world sample when needed for its outcome and duration given the current time. If the time has no mapping, the outcome is randomly computed given the success and failure values, then stored in a lookup table. If the time has a mapping already, that mapping is returned. These time values are rounded to a hundredth of a second before doing the lookup. The arrival and departure times of any object or agent are also resolved by taking a random sample from the arrival interval and the duration interval to construct an exact arrival and departure time.

Following the hindsight optimization framework, in each world sample, TU-HOP examines each available action from the current state. In the most simple *goto* (navigation) case with no objects, TU-HOP will evaluate what will result after moving to each node adjacent to its current node. In figure 2 the agent is currently in location (a) and is considering moving to location (b) or (c). Each move action can either succeed or fail as depicted. Each of these outcomes, $\{s_1, s_2, s_3, s_4\}$, is generated and then reward is maximized individually in each outcome. In our implementation we use a very simple temporal horizon bounded breadth first search to evaluate reward in each outcome. The reward for execution of the action in a single deterministic sample is then a weighted mean of the reward achieved in the success and failure case weighted by the likelihood of the action succeeding and the likelihood of the action failing. For example, if the achievable reward under s_1 is 1, the achievable

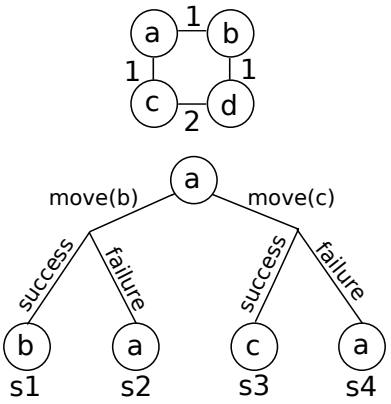

Figure 2: A simple example of the first step of hindsight optimization.

reward under s_2 is 0.5, and the the success rate for that action is 0.9 (0.1 failure rate), then the reward achievable for executing *move(b)* is 0.95 by TU-HOP's reckoning.

This procedure is executed for each generated deterministic sample. Then the reward is averaged over all the samples yielding an estimate of achievable reward. The action with the highest expected reward across all samples is then selected for execution.

The action selected is then executed and the result of the action is used to update the belief state of the planner. This would include increasing the current time, removing a location from an object's possible location list, decreasing the size of an agent's arrival interval, and so on.

Experimental Results

We now evaluate TU-HOP by stressing each of the three types of uncertainties (location, action outcome and temporal uncertainty). All experiments were planned and executed in simulation on a Lenovo W520 with an Intel Core i7 and 8GB of RAM. For each instance problem considered, a set of 25 seeds were used when generating a grounded simulation world as well during planning in each world sample constructed from the current belief. All plots presented show a line representing the mean y-value across the instances and vertical lines representing the 95% confidence interval at that point on the line.

Location Uncertainty

The first set of experiments begin by issuing the goal of moving an object from its start location to a goal location. The easiest instance starts with the object's location known exactly. We increase the difficulty of the instances by adding uncertainty about the objects location to possibly two locations, then possibly three locations and so on. As the uncertainty about the object's start location increases, the agent should be forced to search out the true start location. The results from this experiment using planner configurations of a horizon of 90 seconds and 1, 5 and 10 samples are shown in figure 3.

In all instances, from where the object is in a known location and up to 5 possible locations, the planner is able to find the object and deliver it to its destination. In figure 3 (b) and (c) the lines representing 5 and 10 samples show that these configurations required fewer actions to achieve the goal than the single sample configuration and also have an overall shorter goal achievement time. Since the planner was able to achieve the goal in all instances, we can see that by increasing the number of samples taken, the agent will visit the possible locations in a more reasonable order. First visiting the closest possible location, then moving to the next closest, and the next, until the object's true location is found. Figure 3 (a) shows the time required by the planner at each iteration. Even on the hardest instance with 10 samples the planner takes much less than a second on average before emitting an action for execution.

We also ran a small experiment to show the underlying planner's ability to scale with the number of object relocation goals. We start with a single goal of moving one object to a location and slowly increase the number of goals and objects in the world. The results from this experiment using planner configurations of a horizon of 90 seconds and 1, 5 and 10 samples are shown in figure 4.

In all instances the planner was able to move all the objects to their goal locations. We can see that as expected in figure 4 (a), using more samples does increase the overall planning time at each planning step. However, even in the hardest considered instance with 5 objects using 10 samples the planner only took on average 0.6 seconds before returning the next action. The scaling could be significantly improved by using a heuristic or a more intelligent search technique than breadth first search. In figure 4 (b) and (c) a positive trend is shown where using more samples results in fewer actions in the final execution and also earlier goal achievement times.

Action Outcome Uncertainty

In the second set of experiments we issue a similar goal of moving an object from its start location to a goal location. However in these instances the object's location is known exactly and the topological edge traversals required complete achieve the goal will have increasing traversal failure rates. We start with a failure rate of 0% and increase it all the way to 50%. By increasing the edge traversal failure rates the agent will be forced to re-plan and accommodate for the failure. The results from this experiment using planner configurations of a horizon of 90 seconds and 1, 5 and 10 samples are shown in figure 5.

The planner was able to achieve all goals in all instances during this experiment. The x-axis in figure 5 is numbered by instance difficulty where a difficulty of 1 represents fully reliable edges. This means the outcome of traversing them has 100% probability of success. As the number increases, the success probability decreases to {90%, 80%, 70%, 60%, 50%}. In figure 5 (a) we can see that the planning times for the 1, 5 and 10 sample cases are all quite similar until the edges become quite unreliable (50% success rate). As the failure rate increases, the plan lengths will simply increase and planning with more samples magnifies this in its overall

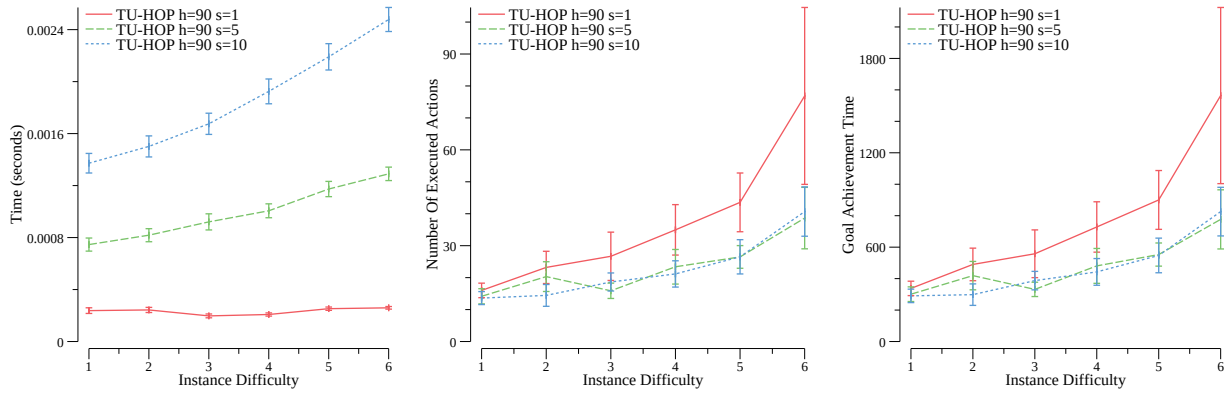

Figure 3: Increasing the amount of uncertainty in an object's location in the world between 1 and 5 locations using a horizon of 90 seconds and 1, 5 and 10 deterministic samples.

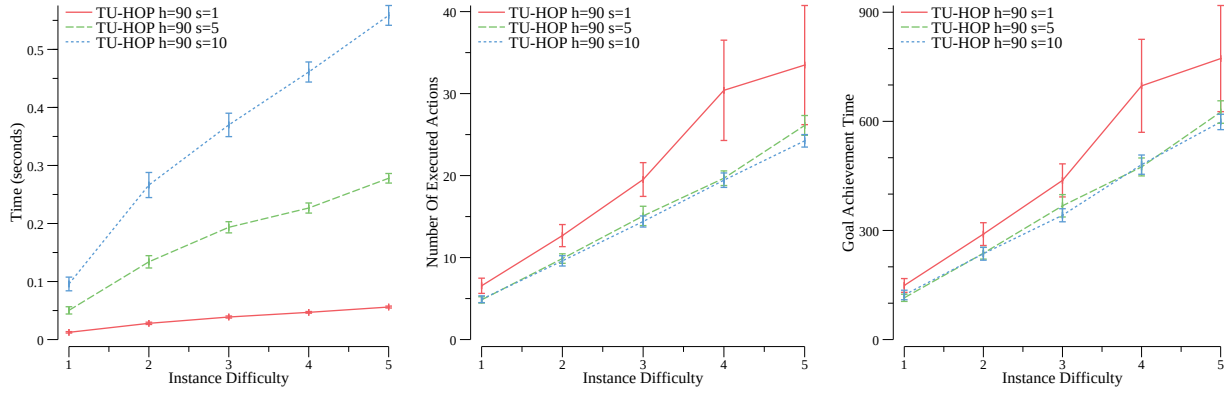

Figure 4: Scaling the number of objects in the world between 1 and 5 using a horizon of 90 seconds and 1, 5 and 10 deterministic samples.

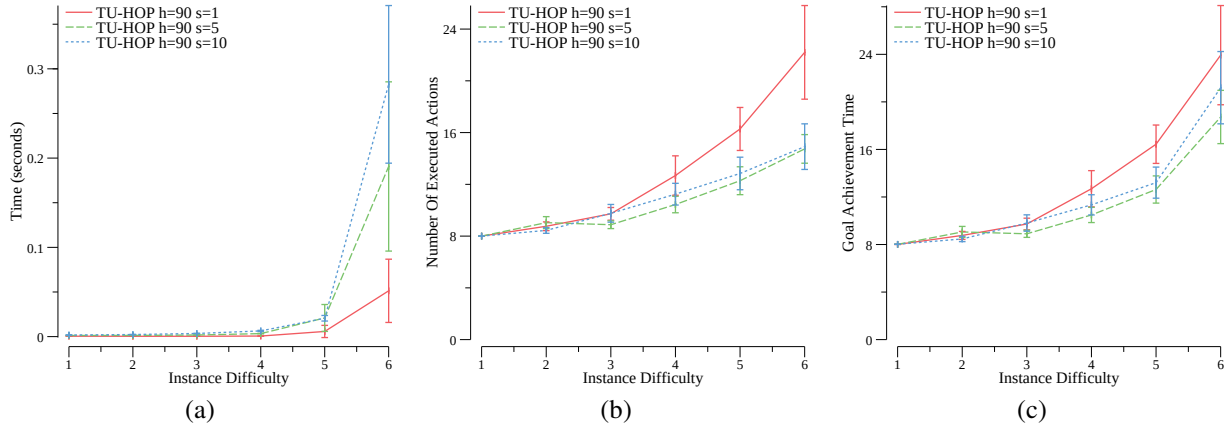

Figure 5: Decreasing the reliability of the only edges available to achieve an issued goal using a horizon of 90 seconds and 1, 5 and 10 deterministic samples.

planning time. In figure 5 (b) and (c) we see a trend similar to the last experiment. This is most likely caused by the same issue. The noise between sampled worlds is minimized by generating more samples.

A simple third set of experiments extending the second set were also performed. The instance is created with a set of inexpensive topological edge traversals between the agent and the object's start locations with high action failure rates. A secondary set of expensive edge traversals with very low failure rates are also created. As the inexpensive route becomes more unreliable throughout the experiments, the agent should choose to take the more reliable expensive route. The results from this experiment are shown in figure 6.

Again, in this experiment the planner was able to achieve all goals in all instances during his experiment. Figure 6 (a) shows a predictable trend where increasing the number of samples causes the planning step between action executions to increase. However, planning times with 10 samples on the most difficult instance are still well below 0.1 seconds on average. Figure 6 (b) and (c) show the realization of our prediction. When using only a single sample the number of actions in the final execution is lower for the first three instances, but the overall goal achievement time for those instances is higher for than when using 5 an 10 samples. Once the reliability of the cheap edges decreases significantly in the last three instances, the single sample case starts to have longer execution lengths and continues to have later goal achievement times than the 5 and 10 sample cases.

Temporal Uncertainty

In this fourth set of experiments we involve a second agent outside of the control of the planner. The goal issued in this set of experiments is to give an object to this second agent and also relocate a second object. Adding a second object goal forces the planner to choose an ordering for the two goals which becomes very important in this experiment. The second agent does not begin at any location but is scheduled to arrive at one during a predefined interval and will only remain for a duration between some minimum and maximum value. We begin by starting with a small arrival interval and a long duration before departing for the second agent. This makes it very easy for the planner to achieve both goals. We increase the difficultly of these instances by making the arrival interval larger (more uncertain) and decreasing the duration the agent remains before departing. By increasing the uncertainty, it becomes much more important for the planner to consider the possible arrival and departure times of the second agent if it is to catch it before it departs. The results from this experiment using planner configurations of a horizon of 30 seconds and 1, 5 and 10 samples are shown in figure 7.

Figure 7 (a) shows that planning times between action executions are still less than 0.5 seconds which is certainly acceptable when the action executions times for a robot can be in the range of full seconds to a minute for some actions. In figure 7 (d) the reliability of both goals being achieved is illustrated. At first with the larger delivery window, the 1, 5 and 10 sample cases are able to achieve both goals reliably. However when the window is reduced the planner clearly benefits from more samples as the 10 sample case is able to achieve all goals in all but the hardest instance. In the hardest instance none of the planner configurations are able to deliver the object to the second agent. Figure 7 (b) and (c) are slightly more difficult to interpret because if the planner incorrectly chooses to relocate the second object first and misses the delivery window, the planner will terminate with fewer actions and an earlier goal achievement time than an algorithm that achieves both goals.

Discussion

The main two attractive features of hindsight optimization are its simplicity and generality. There are no obvious impediments to combining the current work with previous efforts that use hindsight optimization to address other forms of uncertainty such as stochastic action effects, arrival of additional goals, partial observability, or open worlds (Yoon et al. 2008; 2010; Burns et al. 2012; Kiesel et al. 2013).

Compared to a planner using an STNU, TU-HOP's handling of intervals is imprecise, thus the combinations of circumstances that are anticipated is incomplete. This limitation gets more serious as the number of combinations of stochastic events that need to be considered increases. However, in many applications, it is not necessary to reason about such long chains of events in order to act successfully.

TU-HOP demonstrates one way of very tightly coupling planning and acting, namely to ensure reactivity by planing after every state transition and never explicitly committing to actions beyond the one that is currently executing.

Unlike many other task planners, TU-HOP does explicitly consider action failure when selecting actions.

It does not output a complete plan that can be shared with other collaborating agents. However, it should be possible to merge together the actions selected in each rollout to form a branching contingent plan that could be shared. Such sharing could then be represented in the planner by increasing the cost of actions that do not correspond to those in the shared plan. This directly models the coordination costs that the group would sustain if the plan were to be changed.

Hindsight optimization is often used with a limited horizon planner. When this is done, it places some responsibility on the heuristic evaluation function used at the leaf nodes of the search to correctly identify promising states. An alternative is to use hierarchical planning, in which a complete plan exists at some level of abstraction, and detailed planning is then done on those parts that are ready for execution. Such an approach has been proposed by Kaelbling and Lozano-Pérez (2011).

Hindsight optimization is an unsound reasoning technique. UCT is a popular sampling-based technique that is sound, in the sense that it is guaranteed to select the optimal action given an infinite number of samples. Eyerich, Keller, and Helmert (2010) compare hindsight optimization with UCT on the Canadian Traveler's Problem. While they find that UCT does indeed converge better in the limit of many samples, hindsight optimization performed well when the methods were given only a moderate number of samples.

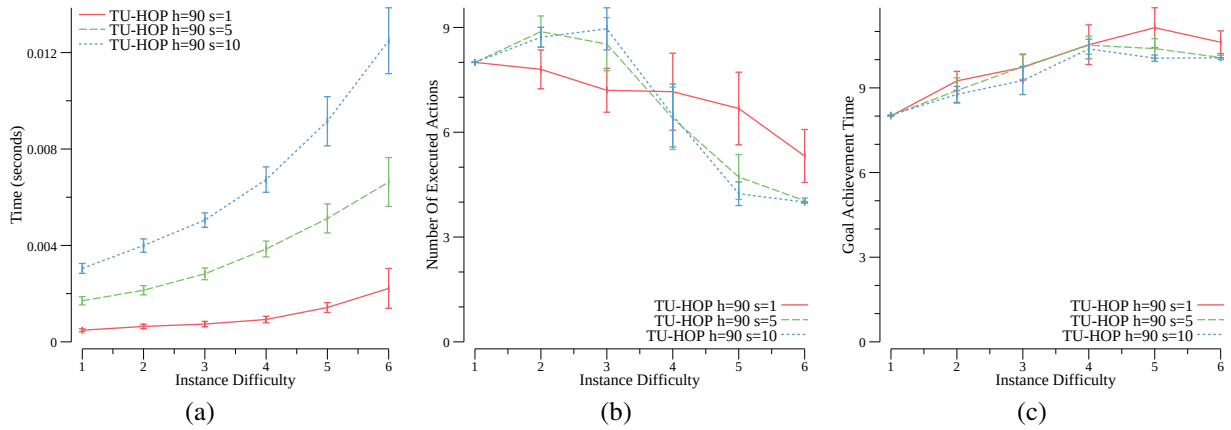

Figure 6: Decreasing the reliability of low cost edges forcing more reliable expensive edges to be utilized using a horizon of 90 seconds and 1, 5 and 10 deterministic samples.

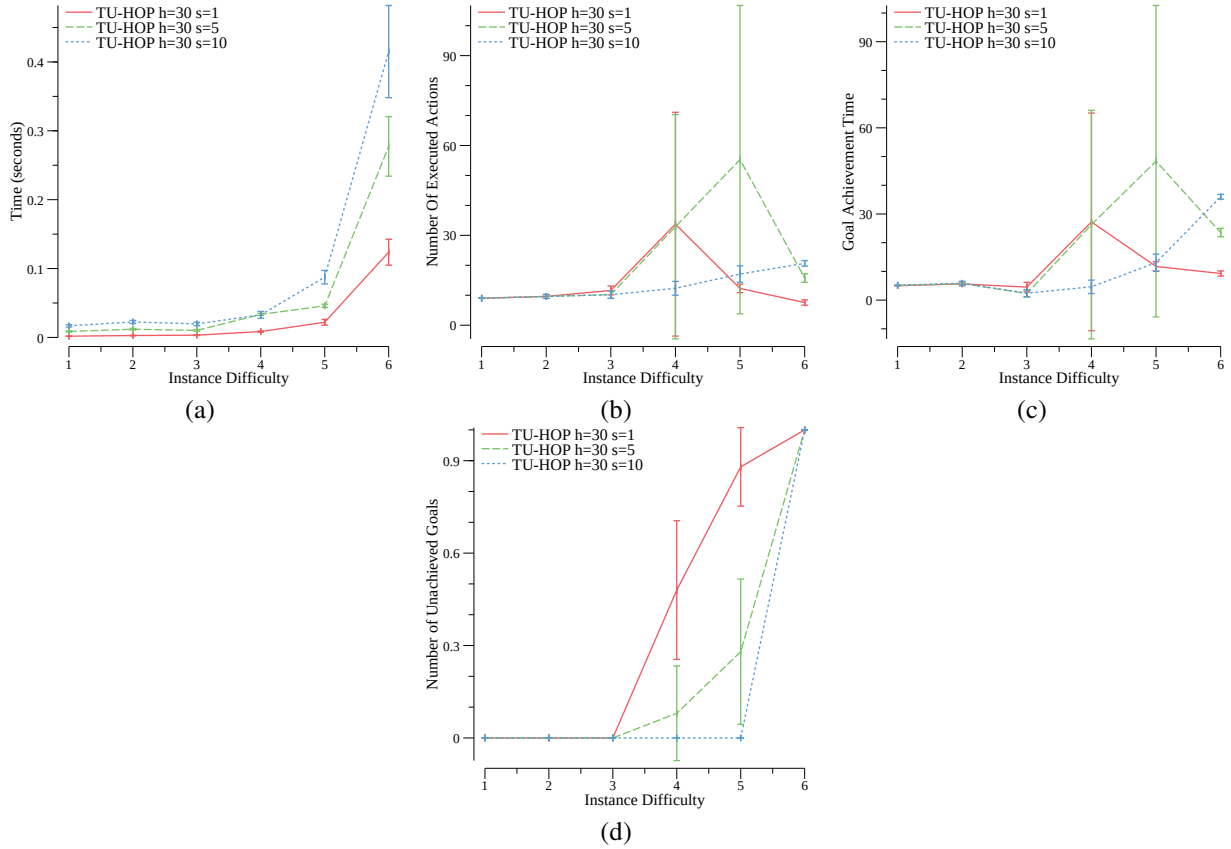

Figure 7: Increasing the uncertainty about when an agent will arrive while also decreasing the duration the agent will wait after arriving using a horizon of 30 seconds and 1, 5 and 10 deterministic samples.

Conclusion

Uncertainty is an unavoidable piece of real-world robotic applications. We have shown how hindsight optimization yields a simple and general approach to planning with location, action outcome and temporal uncertainty. While the technique is approximate, it is easy to implement and our results suggest that it can be successful in practice. Its simplicity and capability make it a flexible baseline against which future temporal uncertainty planning research can be compared.

References

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; et al. 2012. Europa: A platform for ai planning, scheduling, constraint programming, and optimization. *Proceedings of the 4th International Competition on Knowledge Engineering for Planning and Scheduling ICKEPS*.

Burns, E.; Benton, J.; Ruml, W.; Yoon, S.; and Do, M. 2012. Anticipatory on-line planning. In *Proceedings of the Twenty-second International Conference on Automated Planning and Scheduling (ICAPS-12)*.

Chong, E.; Givan, R.; and Chang, H. 2000. A framework for simulation-based network control via hindsight optimization. In *IEEE Conference on Decision and Control*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial intelligence* 49(1):61–95.

Eyerich, P.; Keller, T.; and Helmert, M. 2010. High-quality policies for the canadian travelers problem. In *Third Annual Symposium on Combinatorial Search SoCS-10*.

Ghallab, M., and Laruelle, H. 1994. Representation and control in ixtet, a temporal planner. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems AIPS*, 61–67.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Ingrand, F.; Chatila, R.; Alami, R.; and Robert, F. 1996. Prs: A high level supervision and control language for autonomous mobile robots. In *IEEE Conference on Robotics and Automation ICRA*.

Kaelbling, L. P., and Lozano-Pérez, T. 2011. Hierarchical planning in the now. In *IEEE Conference on Robotics and Automation ICRA*.

Kiesel, S.; Burns, E.; Ruml, W.; Benton, J.; and Kreimendahl, F. 2013. Open world planning for robots via hindsight optimization. In *Proceedings of the ICAPS-13 Workshop on Planning for Robotics (PlanRob-13)*.

Laborie, P., and Ghallab, M. 1995. Ixtet: an integrated approach for plan generation and scheduling. In *INRIA/IEEE Symposium on Emerging Technologies and Factory Automation ETFA*, volume 1, 485–495 vol.1.

Lemai, S., and Ingrand, F. 2003. Interleaving temporal planning and execution: Ixtet-exec. In *Proceedings of the ICAPS Workshop on Plan Execution*.

Mercier, L., and van Hentenryck, P. 2007. Performance analysis of online anticipatory algorithms for large multistage stochastic programs. In *Proceedings of IJCAI*.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, 494–499.

Shani, G., and Brafman, R. 2011. Replanning in domains with partial information and sensing actions. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, 2021–2026.

Wu, G.; Chong, E.; and Givan, R. 2002. Burst-level congestion control using hindsight optimization. *IEEE Transactions on Automatic Control*.

Yoon, S.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic planning via determinization in hindsight. In *Proceedings of Conference on Artificial Intelligence (AAAI)*.

Yoon, S.; Ruml, W.; Benton, J.; and Do, M. B. 2010. Improving determinization in hindsight for on-line probabilistic planning. In *Proceedings of the Tenth International Conference on Automated Planning and Scheduling (ICAPS-10)*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Ff-replan: A baseline for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS-07)*.