On-line Anticipatory Planning

Allen Hubbe[1] and **Wheeler Ruml**[1] and **Sungwook Yoon**[2] and **J. Benton**[2] and **Minh B. Do**[3]

[1]Department of Computer Science	[2]Dept. of Computer Science and Eng.	[3]Embedded Reasoning Area
University of New Hampshire	Arizona State University	Palo Alto Research Center
Durham, NH 03824 USA	Tempe, AZ 85287 USA	Palo Alto, CA 94304 USA
aba4 at unh.edu	sungwook.yoon at gmail.com	minhdo at parc.com
ruml at cs.unh.edu	j.benton at asu.edu	

Abstract

We consider the problem of on-line continual planning, in which additional goals may arrive while plans for previous goals are still executing and plan quality may depend on how quickly goals are achieved. Even in domains with deterministic actions, this is a challenging problem. One common and straightforward approach is *reactionary* planning, in which fresh plans are synthesized when a new goal arrives. In this paper, we adapt the technique of hindsight optimization from on-line scheduling to create an *anticipatory* planning algorithm. Using an estimate of the goal arrival distribution, we sample possible futures and use ordinary deterministic planning techniques to estimate the value of taking each possible action. We present benchmark domains inspired by crisis response and manufacturing and show how an anticipatory approach yields a superior planner that is sensitive not only to which action should be executed, but when.

Introduction

Consider the problem faced by an ambulance dispatcher. The agent wants to minimize the time between when a victim is stricken and when he reaches the hospital. Even when the actions of the ambulance, such as driving particular routes or loading and unloading patients, can be regarded as deterministic, the stochastic arrival of new transport requests can make for a challenging on-line planning problem. Should a dispatched ambulance divert from its previously planned route to tend to a new request that can be served faster? What should the ambulance do when no requests are pending? From a global perspective, it might be worth preemptively parking the ambulance near locations where requests are likely.

A similar problem occurs in situations where significant setup costs are involved. For instance, problems like grocery shopping (or manufacturing) often have relatively predictable actions but the question is whether to immediately plan and execute every time a request arrives. It may be worth the cost of delaying to wait for more requests so as to minimize the makespan and cost of the final plan.

We call these *on-line continual planning* problems. They incorporate aspects of planning (action selection), scheduling (when to perform the actions), and uncertainty (unknown future goals). The traditional approach to such problems is to synthesize plans when goals arrive. After generating a plan when the first goal arrives, there are two possible strategies for handling the arrival of subsequent goals. The first is a greedy strategy, in which previous plans are held fixed and the plans for the new goals are restricted to respect the assumptions of the old ones. This simple approach was taken by Do, Ruml, and Zhou (2008) in their work on fast on-line planning for modular printers and by ter Mors, Zutt, and Witteveen (2007) in their work on airport taxiway routing. The second possible strategy is based on replanning, and considers all the possible future plans for the current goals that are compatible with the actions already executed (Nebel and Koehler 1995; Knight et al. 2001; Fox et al. 2006). Both of these strategies wait to plan until goals actually arrive, and they consider only those goals that have been revealed. We call this general approach *reactionary planning*.

In this paper, we consider an alternative approach that explicitly acknowledges stochastic goal arrival. We call it *anticipatory planning*. It assumes that the probability distribution over incoming goals is either known or learnable and employs the technique of *optimization in hindsight*, previously developed for on-line scheduling and recently introduced for planning with stochastic actions (Mercier and van Hentenryck 2007; Yoon et al. 2008). As we explain in more detail below, this technique first samples from the distribution of possible future goal arrivals and then considers which next action optimizes the expected cost when averaged over the sampled futures. While anticipatory planning involves more computation than reactionary planning, we show that its overhead can be made quite modest and that the resulting planner successfully addresses the challenges raised by on-line continual planning when tested in two benchmarks domains modeled on the two examples above.

On-line Continual Planning

Most academic research on general purpose planning has concentrated on off-line scenarios. However, planners deployed for real-world applications are frequently run in an on-line setting in which goal arrival, plan synthesis, and plan execution happen concurrently. Such domains include manufacturing process control, supply chain management, power distribution network configuration, transportation logistics, mobile robotics, and spacecraft control. Despite the importance of these on-line domains, little investigation ef-

fort has focused on the potential drawbacks of using standard, academic off-line (and therefore reactionary) planners in an on-line environment. In particular, we see three main qualities of on-line planning problems that current state-of-the-art off-line planners lack mechanisms to handle.

Optimizing wall clock end time: A planner must keep a view of the global *wall-clock* time. That is, as pointed out by Benton et. al. (2007), the planner must focus both on *execution* and *planning* time to minimize the total *wall-clock* makespan of plans. This is especially important when the objective is to achieve the goals as quickly as possible, meaning that the figure of merit is the wall-clock time at which the plan's execution finishes.

Asynchronous goal arrival: In many real-world settings, goals are not neatly divided into separate episodes. Rather, new goals may arrive while plans for previous goals are still executing.

Goal distributions: Though goals may be asynchronous, in many on-line planning problems information such as the distribution of possible goals and their arrival times can be obtained either by learning techniques or expert knowledge.

An off-line planner with informedness as to its own planning time may appear to solve these problems at first glance. Unfortunately, this is an ignoratio elenchi. Consider an off-line planner capable of finding an optimal solution instantaneously. It would need to completely replan and at most could use a mechanism for maintaining commitments made in previous plans for plan repair. This can lead to plans that appear *globally* inefficient in hindsight. For example, consider a logistics domain where a vehicle has just passed a warehouse. After several minutes a new pick-up request arrives for that same warehouse, and so the vehicle must revisit the warehouse to achieve the goal. Having foreknowledge of the high possibility that a goal at the warehouse may arrive would allow the planner to generate a plan that waited at the location for a short time.

In this work, we investigate the use of goal arrival distributions to generate plans that exhibit anticipatory behavior for asynchronous goals. By sampling solutions for the potential goals, we can guess the best next action to take, potentially even choosing actions for goals that have not yet been revealed. This can significantly improve plan quality, especially when it depends on the time of goal achievement rather than the makespan from the first action to the last.

Problem Setup

We define an on-line planning problem P as a 7 tuples $\langle F, S, A, G, C, L, T \rangle$, where S is the set of states, and A is the set of actions. As usual, each state consists of a set of facts $s \subset F$. Applying action a to a state s satisfying all of a's preconditions change s as usual: $s' \leftarrow (s \setminus Del(a)) \cup Add(a)$, where $Del(a)$ is the set of delete facts of action a and $Add(a)$ is the set of add facts.

G is a generation function and generates new goals and corresponding state facts at each time point, $\langle g, s \rangle$. We assume G is either stationary or changes deterministically. Each goal g is a pair containing a conjunction of facts and a deadline time, $\langle f, t \rangle$, where $t \in T$ is a time point. T is a

set of integer-valued infinite time points. C is an action cost function that maps each action to a real number. $L(g)$ is the goal cost function that maps each goal to some real number at each time point. We will not explicitly specify the role of the goal deadline time in g, as it is defined by the application. One example of a cost function is a "step function" that increases its value after passing the goal deadline, similar to what is defined by Haddawy and Hanks (1993).

Unlike an off-line planning problem, the goal set of the problem is continuously changing. There are two factors that affect the current goal set: (1) G generates new goals (or possibly no goals) at every time point and (2) an action that achieves a goal g removes it from the current goal set $\mathcal{G}$. Regardless of a goal's deadline, we consider a goal achieved when an action asserts it. Thus, an action $a \in A$ maps the current state s and current goal set $\mathcal{G}$ to another state and another goal, $a(s, \mathcal{G}) \to (s', \mathcal{G}')$.

We use $N(s, \mathcal{G}, a, G) \to (s', \mathcal{G}')$, to advance the current state s to s' through an action a and new goals G. This constitutes advancing one time point, that is, if s is in time t, then s' is in time $t + 1$. Suppose, G generated $\langle g, s_g \rangle$ at the current time step and $a(s, \mathcal{G}) \to (s^a, \mathcal{G}^a)$ then the updated state s' is $s^a \cup s_g$ and the updated goals $\mathcal{G}'$ is $\mathcal{G}^a \cup \{g\}$.

On-line Planning Objective: The objective of an on-line planning problem $O(s, \mathcal{G}_0, H)$ is to minimize the average cost over a predefined horizon H starting from the current state s and the current goal set $\mathcal{G}_0$.

$$O(s, \mathcal{G}_0, H) = \sum_{t=1}^{H} C(a_t) + L(\mathcal{G}_t)$$

where $(s_t, \mathcal{G}_t) \sim N(s_{t-1}, \mathcal{G}_{t-1}, a_t, G)$ and we assume $t = 1$ at the current time. At the current state s, the future goals are not seen and deciding which action to take is not a trivial problem. Since the future is not known, a simple reactionary planning approach plans only for the outstanding goals. This approach does not directly optimize the above equation. In the following sections, we develop an anticipatory algorithm that minimizes $O(s, \mathcal{G}_0, H)$. First, we describe two domains that we use to demonstrate the need for anticipatory planning.

Example Domains

We introduce two on-line planning domains that reflect interesting real life on-line planning scenarios.

Ambulance Domain

The ambulance domain models a scenario faced by hospitals. Namely, where to place ambulances throughout a city such that they can quickly apply first aid and deliver people to the hospital. The domain consists of people, ambulances, locations, and hospitals. The goals involve having each person at the hospital. When an accident happens, a goal is posted that indicates the location of a person and a deadline time of when the person needs to be at the hospital (indicating the survival time for a critically injured patient). Therefore, ambulances need to pick up people and deliver them to the hospital as quickly as possible. There is a stochastic distribution of the potential locations where the accidents
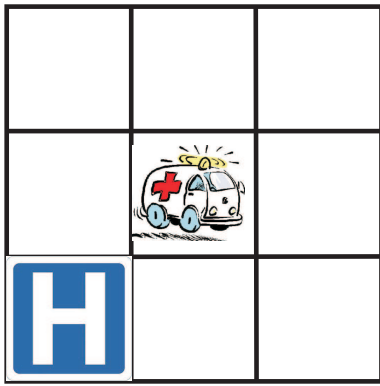
Figure 1: Ambulance Domain Example: There are 9 locations and the distribution of the accident is uniform and the best spot for the ambulance is the center point.
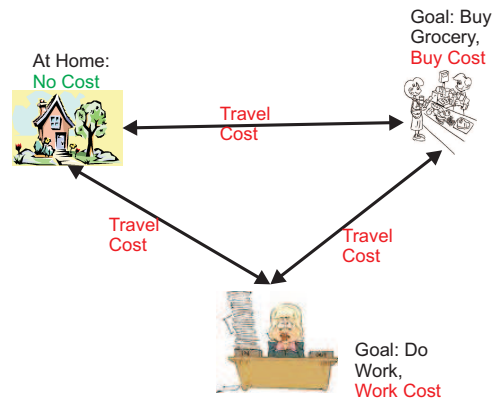

Figure 2: Grocery Domain Example: Staying at home is always preferable, as long as there is no outstanding goals. Since travel is costly, when some more goals are expected to arrive it is often better to wait and achieve them in a single trip to the destination.

might happen. We would like plans that place ambulances near these locations. Reactionary planners cannot immediately identify these strategic locations and will therefore fail to serve the goal in a timely manner. Figure 1 shows an example of this domain in a 3×3 grid. In our example, the probability of an accident occurring is uniform throughout each location. Because of this, it is preferable to have the ambulance at the center location rather than at the hospital.

This domain is analogous to police patrols, which are routed and scheduled so as to encounter predicted problems or criminal activity. Police are commonly dispatched to major gatherings such as sporting events, concerts, and protests, in anticipation of their being needed.

Grocery Domain

This domain models problems where paying setup costs results in plans where delaying action can be better than taking action (as opposed to the ambulance domain, where preemptive action is important). It involves three locations: home, work, and the grocery store. Figure 2 gives a pictorial illustration. Goals involve finishing work (at a workplace) and purchasing groceries (at a grocery store). In contrast to the ambulance domain, where cost is accrued due to unmet deadlines, here we have cost for (1) having unfinished work to do or groceries to purchase (independent of the amount of work or groceries) (2) travelling between locations, (3) being away from home and (4) doing a work task or purchasing groceries. A travel actions has the same cost as purchasing any number of item from the grocery store, and therefore it is often better to wait for more grocery requests to arrive before heading to the store. Similarly, we want to go to work only when doing the work is worth the cost of travelling to work to "relieve" the penalty of not working. The best strategy may involve delaying action until enough goals arrive.

Anticipatory Planning

Anticipatory algorithms were originally developed in the scheduling and networking communities (Chong, Givan, and Chang 2000; Mercier and van Hentenryck 2007) and

recently have been applied to probabilistic planning domains (Yoon et al. 2008). They work by sampling potential future problems and solving them to determine the best "next" actions to take. In this spirit, our algorithm samples for potential goals, solves for them, then chooses the best course of action (or inaction) based upon the resulting solutions.

On-line Anticipatory Planning

Anticipating goal arrival involves some knowledge of possible future events. Essentially, given knowledge of goal arrival distributions, we conjecture the best next actions (including null actions) to take that will likely lead to the lowest cost outcome. Recall that plan cost is defined by the summed cost of actions and the cost associated with the time of goal achievement. Thus the objective is $O(s, \mathcal{G}_0, H) = \sum_{t=1}^{H} C(a_t) + L(\mathcal{G}_t)$.

Using a set of given cost functions on goals and actions along with the distribution of goal arrival, a set of actions should be found that minimizes the expectation of the future goals. That is, we have the function

$$V(s) = \min_{a_1 \dots a_T} E_{g_t \sim G} \sum_{t=1}^{H} C(a_t) + L(\mathcal{G}_t)$$

where $(s_t, \mathcal{G}_t) \sim N(s_{t-1}, \mathcal{G}_{t-1}, a_t, G)$. This function asks us to take a first action, a_1 in an action sequence that minimizes the expectation of the cost over future goals. To compute this exactly, we can perform an expectation minimization search. Unfortunately, this choice offers some severe trade-offs, as its performance degrades exponentially as the horizon H increases. Other alternatives, such as brute force methods involving the enumeration of all plans, offer similarly bleak prospects on scalability.

Instead of these approaches, this work implements a technique used in other anticipatory algorithms that involves

interchanging expectation and minimization (Chong, Givan, and Chang 2000; Mercier and van Hentenryck 2007). Specifically, the expectation function is changed to reflect expectation over minimizing cost for future goals as against minimizing cost for the expected value of future goals. We now have the equation

$$V^a(s) = E_{g\sim G} \min_{a_1 \ldots a_T} \sum_{t=1}^{T} C(a_t) + L(\mathcal{G}_t).$$

Of course this neglects preservation of the current state s. However, this technique allows for large computational efficiency gains. Using fixed future goals, the minimization problem is manageable. The problem becomes the deterministic planning problem of finding an action sequence that minimizes cost given a fixed set of future goals. To find the best action in the current state, we define a Q-value for each action.

$$Q(s,a) = C(a) + E_{g\sim G} \min_{a_1 \ldots a_T} \sum_{t=1}^{H-1} C(a_t) + L(\mathcal{G}_t)$$

The Q-value computes the potential cost of taking each action in the state s. And following this, the best action choice in s is given by $\min_a Q(s,a)$. In this sense, we are performing *optimization in hindsight* of knowledge gained through sampled futures.

Figure 3 summarizes our anticipatory planning algorithm. At each time step, the algorithm is used to find the next action to execute for the current state s. The inputs to the algorithm are the current state, the set of current goals, a goal distribution function, a horizon H and a sampling width W. For each action, we advance to a next state s' then generate a sample of W future goals using the *draw-future()* function for H time points. The resulting possible future $\mathcal{G}_F$ is paired with the state s', generating a deterministic planning problem. Solving this problem provides a solution to the future $\mathcal{G}_F$. The function *solve()* returns this solution and its cost. We repeat this W times, accumulating the potential cost across the sampled futures that could have occurred after an action a is taken. This cumulative value is the Q-value for each action a in s. We select the minimum Q-value action and return it.

Empirical Evaluation

We have tested our on-line anticipatory algorithm on the two example domains that were introduced above. For this demonstration, we have built a simple A^* search-based planning algorithm for both reactionary and anticipatory planning experiments. Our implementation serves as a proof of concept, and our approach can easily be adapted to use any off-line deterministic planner capable of handling the given cost metrics. For comparison we also implemented a reactionary planner that simply generates plans as goals arrive.

Ambulance Domain: The ambulance domain, as previously described, involves getting people to a hospital before a deadline time. For our tests, the probability of a incident with a patient is uniform through a grid (as illustrated in Figure 1). The best strategy for this scenario is to have the

---

Anticipatory Planning $(s, \mathcal{G}, G, H, W)$
    // s the current state
    // $\mathcal{G}$ the current goal set
    // G is the goal distribution function
    // $\mathcal{H}$ the horizon
    // $\mathcal{W}$ future sample width

Best-Action ← null
Best-Action-Cost ← ∞
for-each action a available in s
    $Q(s,a) \leftarrow 0$
    for W times
        $\mathcal{G}_F \leftarrow$ *draw-future(G, H)*
        $Q(s,a) \leftarrow C(a) +$ *solve($a(s), \mathcal{G}, \mathcal{G}_F$)* $+Q(s,a)$
    if $Q(s,a) <$ Best-Action-Cost
        Best-Action ← a
        Best-Action-Cost ← $Q(s,a)$
Return Best-Action

Figure 3: Anticipatory Planning Algorithm. At the current state, it computes the anticipatory Q values for each action, by sampling and planning. The action that achieves the minimal Q value cost is returned.

---

ambulance at a central location where any new patients may be easily reached with minimum expected cost. In this case, this is the center position. There exists a fixed cost for ambulance movement. Outstanding goals have a fixed cost at every time step, therefore total cost increases steadily as goals remain unachieved and one wishes to deliver patients to the hospital as quickly as possible.

We used fixed grid sizes of 3×3, 5×5, and 7×7 cells and set the future sample size to 20 with varying horizon lengths. Figure 4 shows the results of these experiments. The x-axis indicates the probability of goal arrival (i.e., higher values indicate a higher chance that an incident will occur, requiring a patient be transferred to a hospital) and the y-axis indicates the percentage cumulative cost above a lower bound.

For all grid sizes, the anticipatory planning algorithm performed better than the reactionary planning algorithm, independent of the horizon used. During the experiments we observed that the anticipatory algorithm preemptively placed the ambulance at the center location and therefore the generated plans achieved the goal more quickly than the reactionary planner after an accident occurs. The results of this were especially noticeable after a single patient was delivered to the hospital. The reactionary planner would wait at the hospital for the next goal, rather than incur cost by moving to the center location in anticipation of a new patient. Given that we placed the hospital at a corner location, this caused much of the additional cost used.

We see that horizon also changes the performance of the anticipatory planner. Interestingly, as we increase the horizon to 4, the cost of the plans decreases to its best value. It is striking that even a horizon of 1 was more effective than the reactionary planner, as providing a small lookahead as to the arrival of new goals works better than none (a behavior

which is also seen by Natarajan et al. (2007)).

Adding future samples should improve the accuracy of the anticipatory planner, thereby providing, over time, lower cost plans. To test this, we fixed the horizon at 12 with varied numbers of future samples. As can be seen in Figure 5, the size of the future samples correlates well with performance. That is, as we increase the number of samples, the anticipatory planning algorithm generates lower cost plans. Giving more sample futures gives a more reliable estimate on the Q-values (and the possible futures). Surprisingly, even a single sample does significantly better than the reactionary planner. This is even true when we have a low probability of an incident occurring. Note, however, that on the 3×3 grid, in fact the reactionary planner and the anticipatory planner with a single future sample performed about equivalently. Likely the larger grids provided the sample with an incident, providing a more accurate Q-value.

In a separate pilot experiment, we tried re-distributing the goal arrival probabilities in a 5×5 grid such that patients could only arise in a 3×3 area of the grid, located in a corner. In this case, the anticipatory planner placed the ambulance in the center of the 3×3 area, as expected, rather than in the center of the entire grid. This demonstrates the technique's ability to adapt to the particular goal distribution encountered.

Grocery Domain: In the grocery domain, it is better to stay at home when there are no work task or grocery item goals outstanding. Travel to the work place and grocery store has a cost, as does being away from home. Since grocery store errands can be performed with a single visit (and purchase), it can be advantageous to wait for grocery requests to accumulate due to the expense of the trip. Work penalties accrue when tasks remains undone, independent of the amount of work to do, but each individual work task request takes an entire time unit to achieve. It is best to go to work only when enough work accumulates to justify the trip and remove the burden of the penalty.

Figure 6 shows the results of our runs. In the Figure 6(a), we varied the grocery order arrival rate while fixing the the work probability arrival rate to 0.5 (i.e., for every time step there is a 0.5 chance that work will arrive). As the grocery order arrival rate increases, the anticipatory algorithm found much lower cost plans than the reactionary planer (and is always dominating it). However, with a lower probability of a grocery order occurring, the two approaches yield similar results. This can happen when possible futures are sampled that involve grocery orders but in the "true" future, the orders were not actually made. In the end, the anticipatory planner waited for an order but none arrived, and therefore it had worse performance. We see a similar result in Figure 6(c), where we have only grocery store orders and no work orders (by setting the probability of work to 0). In this panel, the lines eventually dip below the baseline lower bound. This is because the 'lower bound' used in the grocery domain is not a true lower bound, as it does not recognize that multiple groceries can be purchased in a single action.

In Figure 6(b), we fixed the grocery order arrival rate to 0.5 and varied the work order arrival rate. Although the anticipatory algorithm perform much better, its performance did not correlate with the work order rate. This is a reasonable result, considering that an entire set of grocery orders can be achieved in one session of shopping session. Thus, as grocery order rate increases, it is better to wait and in this case anticipatory planner found that the "waiting" action was more useful.

Discussion and Related Work

Many real-world scenarios involve on-line continual planning. Despite this, most academic planning systems focus on off-line domain-independent planning techniques that do not address these real-world scenarios. We have identified several challenges associated with on-line planning which off-line planners cannot directly tackle. Namely, on-line continual planning involves optimizing for wall-clock time and handling asynchronous goals with known arrival distributions. Our work takes a step toward handling these issues. By anticipating future goal arrival and acting on it, we can find plans that minimize costs associated goal achievement time.

Anticipatory algorithms have been studied by the optimization and scheduling communities (Mercier and van Hentenryck 2007; Chong, Givan, and Chang 2000; Wu, Chong, and Givan 2002). Some schedulers face challenges similar to ours. For instance, they need to continuously schedule actions given new goals. As an example, consider the problem of routing packets. As new packets arrive for a networking router, it must schedule their routing for speed and lack of contention. However, the scheduler has only a distribution of their possible arrival times and must allot a space for them based on anticipating their future arrival. These anticipatory schedulers do hindsight sampling and use domain specific solvers to find the estimation of the Q-values. The key difference is that planning goals are achieved over time and typically require multiple actions. This highlights preemptive execution and opens the door to delayed execution.

In contrast to Chong, Givan, and Chang (2000), we consider stochastic goal arrival rather than probabilistic action effects. Conceptually, one could consider the current goal set as part of the agent's state and view our approach as a special case of probabilistic effects. However, on-line continual planning deserves to be considered separately for two reasons. First, it arises often in applications where either the environment is static (as with spacecraft) or the agent's hardware is very reliable (as in manufacturing). Practitioners often prefer to approximate their systems as deterministic whenever possible, relying on replanning to handle execution failure. Secondly, problems with deterministic actions and stochastic goal arrival avoid one drawback of optimization in hindsight that was mentioned by Chong, Givan, and Chang (2000). Consider the example from their Figure 1b: a probabilistic domain in which a possible action a leads to state s'. From s', action c achieves the goal with probability ϵ (and leads to a dead end otherwise) and action d leads to a dead end with probability ϵ (and achieves the goal otherwise). When drawing 'common random numbers' in advance (as is common to reduce variance in hindsight optimization), it will appear that the goal is reachable in one
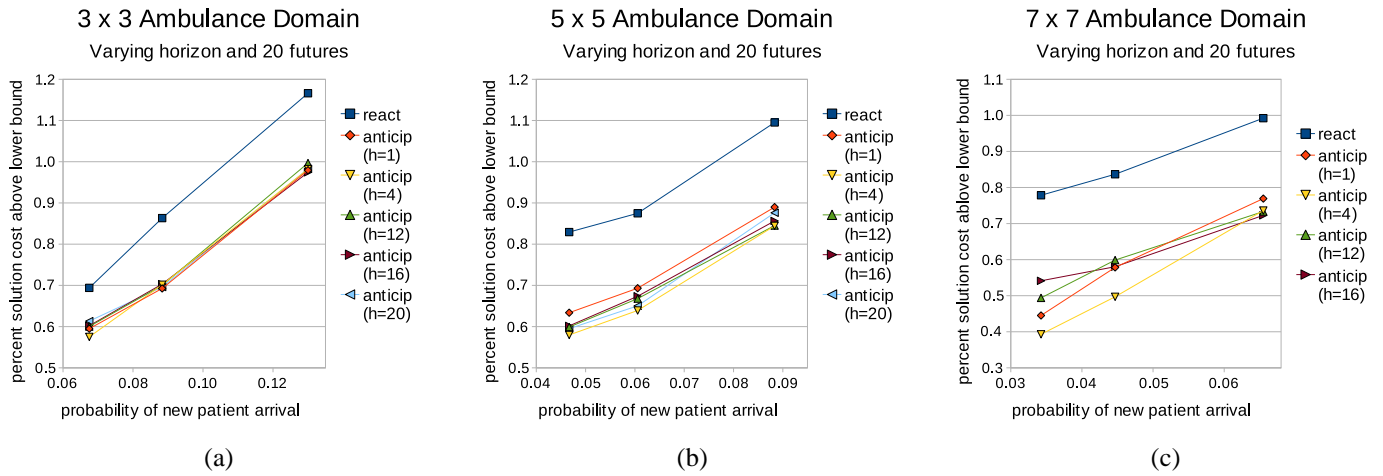
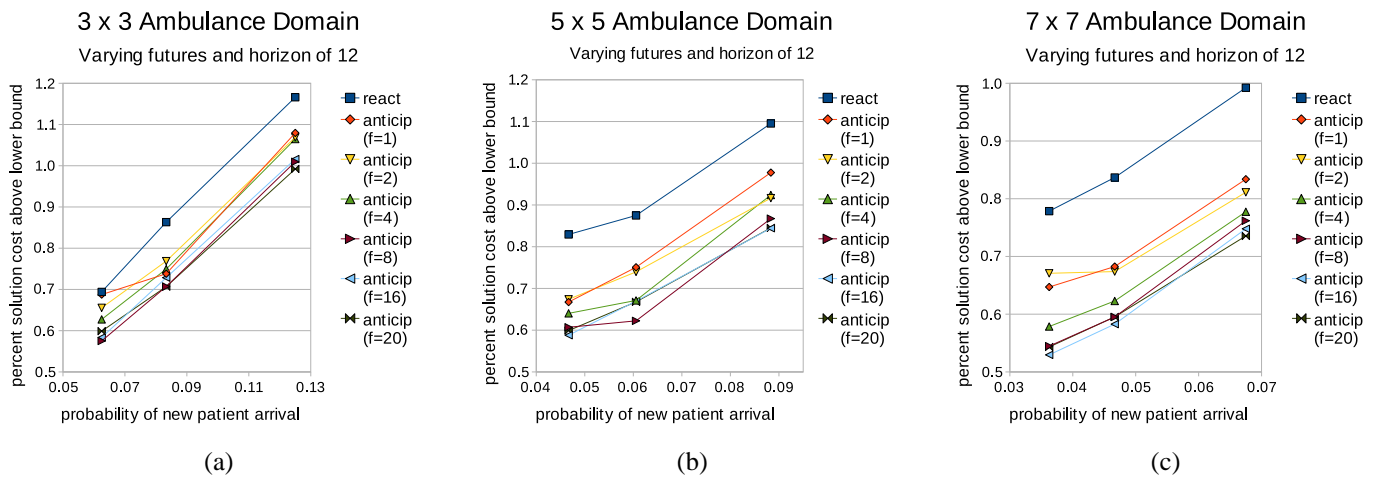Figure 4: Results in the Ambulance domain, varying the horizon length.

Figure 5: Results in the Ambulance domain, varying the number of samples.
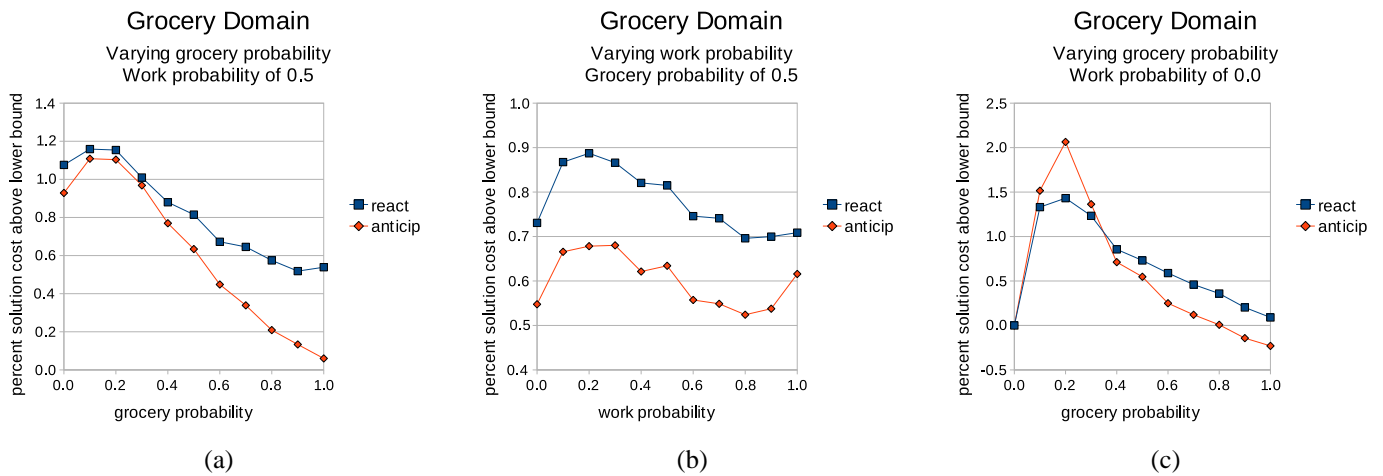

Figure 6: Results in the Grocery domain, varying the goal mix.

step with probability 1 from s', leading the agent to choose action a, even if there exists a series of actions that actually do reach the goal from s with probability 1 but take a little longer. On the other hand, the use of common random numbers in our setting, sampling the possible future goal arrival sequences before considering each possible next action, causes no problems whatsoever. In this sense, the hindsight optimization technique is in fact better suited to our setting than to probabilistic planning.

Future Work

This work serves as an initial investigation into the possibility of using anticipatory algorithms for on-line continual planning settings. As such, there is much room for future work. In particular, we would like to investigate the possibility of handling parallel actions and committed actions that have retraction costs, and explore the best way to allocate plan synthesis time to solve for possible futures.

Our current algorithm generates only sequential plans. Naively extending our algorithm for parallel actions would lead to an increase in computation, requiring every possible combination of actions to be considered for possible futures to investigate. Heuristics that can take into account 'timed goals' would be relevant here.

Planning for new goals when there already exists an executing plan is difficult. Often the decision has to be made to retract previous actions and commitments in order to generate the best plan. These retractions often come with a cost, which is a serious issue for on-line planning scenarios. Models involving such costs have been discussed by Cushing and Kambhampati (2005). We plan to investigate this issue with a simulator that was developed for this purpose (Benton, Do, and Ruml 2007).

Along with these issues, we also wish to investigate how much time to allocate to the planner for its search over sampled future goal arrivals. The baseline planner could take an additional argument of a time limit for each future, trying to find the best solution in an anytime fashion. We are considering the use of a stochastic search with random restart behavior as found in scheduling research in case a plan cannot be found within the time limit.

References

Benton, J.; Do, M.; and Ruml, W. 2007. A simple testbed for on-line planning. In *Proceedings of the ICAPS Workshop on Moving Planning and Scheduling Systems into the Real World*.

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*.

Cushing, W., and Kambhampati, S. 2005. Replanning: A new perpsective. In *Poster Proceedings of the International Conference on Automated Planning and Scheduling*.

Do, M.; Ruml, W.; and Zhou, R. 2008. On-line planning and scheduling: An application to controlling modular printers. In *AAAI*.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In Long, D.; Smith, S. F.; Borrajo, D.; and McCluskey, L., eds., *ICAPS*, 212–221. AAAI.

Haddawy, and Hanks, S. 1993. Utility models for goal-directed decision theoretic planners. Technical Report TR-93-06-04.

Knight, R.; Rabideau, G.; Chien, S.; Engelhardt, B.; and Sherwood, R. 2001. Casper: Space exploration through continuous planning. *IEEE Intelligent Systems* 16(5):70–75.

Mercier, L., and van Hentenryck, P. 2007. Performance analysis of online anticipatory algorithms for large multi-stage stochastic programs. In *International Joint Conference on Artificial Intelligence*.

Natarajan, S.; Judah, K.; Tadepalli, P.; and Fern, A. 2007. A decision-theoretic model of assistance - evaluation, extensions and open problems. In *Interaction Challenges for Intelligent Assistants*, 90–97.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empircal analysis. *Artificial Intelligence* 76:427–454.

ter Mors, A. W.; Zutt, J.; and Witteveen, C. 2007. Context-aware logistic routing and scheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

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)*.