Continual On-line Planning

Sofia Lemons
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
sofia.lemons at cs.unh.edu

Abstract

My research proposes an approach to integrating planning and execution in time-sensitive environments, a setting I call continual on-line planning. New goals arrive stochastically during execution, the agent issues actions for execution one at a time, and the environment is otherwise deterministic. My dissertation will address this setting in three stages: optimizing total goal achievement time, handling on-line goal arrival, and adapting to exogenous changes in state. My approach to these problems is based on incremental heuristic search. The two central issues are the decision of which partial plans to elaborate during search and the decision of when to issue an action for execution. I propose an extension of Russell and Wefald's decision-theoretic A* algorithm to inadmissible heuristics. This algorithm, Decision Theoretic On-line Continual Search (DTOCS), handles the complexities of the on-line setting by balancing deliberative planning and real-time response.

Introduction

The goal of planning is to synthesize a set of actions that, when executed, will achieve the user's goals. Most academic research on general-purpose planning has concentrated on off-line planning, in which plan synthesis is considered separately from plan execution. This separation was originally motivated by the fact that even simplified off-line settings, such as sequential non-temporal planning, are intractable in the general case and it has helped focus research in the field on core algorithmic issues. In the last ten years, tremendous advances have been made in domain-independent plan synthesis and in many domains we now can find parallel temporal plans with hundreds of actions in a few seconds.

However, currently deployed planners for real-world applications are frequently run in an on-line setting in which plan synthesis and execution run concurrently, new goals may arrive, and facts about the world state may change. Such domains include manufacturing process control, supply chain management, power distribution network configuration, transportation logistics, mobile robotics, and spacecraft control. For example, in the CASPER system developed at JPL for the EO-1 satellite, observations of the Earth are analyzed on-board and can trigger recognition of important activity that immediately spawns requests for additional images (Gratch and Chien 1996). When goals arrive, CASPER generates completely new plans to achieve them. Similarly, in the Man-U-Plan system developed at PARC for parallel printing systems, new requests for printed sheets arrive in the system at a rate of several per second interleaving with the printing of other sheets (Ruml, Do, and Fromherz 2005). Likewise, this system handles new goals and failures by generating new plans.

Despite the importance of on-line planning domains, the issues they raise related to total goal achievement time, goal arrival, and asynchronous state changes have not been thoroughly explored. Indeed, during ICAPS community meetings and Festivus events, many prominent planning researchers have pointed out the weak connection of academic work to industrial applications as the most pressing issue facing the community today. One reason for this disconnection is that real-world applications tend to be complex. Another is that evaluating an on-line planner is more complex than running a planner off-line. My dissertation proposes a problem formulation, called *continual on-line planning*, that extends current standard off-line planning into the on-line setting that takes into account wall-clock time. To solve problems in this setting, I am developing a real-time heuristic search approach that extends decision-theoretic A^* algorithm (DTA*) (Russell and Wefald 1988b).

The stages for my dissertation are optimizing total time, handling on-line goal arrival, and handling asynchronous state changes. By total time, I mean the time taken by both plan generation and execution. In addition, cost of a solution can and should be taken into consideration, along with time. This new objective takes into consideration all aspects of planning time, appropriate for agents active in real time. Instead of goals being seen as a pre-defined, static list of desired states, my work will allow for them arriving during plan generation or execution. Goal arrivals are a common concern in industry problems, and have been poorly addressed by efforts such as replanning or plan repair. Finally, my work will be extended to handle unexpected changes in state during planning or execution. This last stage moves planning into a more realistic setting in which the world can change beyond the agent's intended effect and the simplifying assumptions of the previous domains can be broken and my work will be able to handle these situations gracefully.

Optimizing Goal Achievement Time

The first objective for my dissertation is to develop a search algorithm which optimizes total goal achievement time, minimizing both time spent planning and time executing the plan. Because of this, it may be advantageous to begin issuing actions before settling on a complete plan to avoid wasted time when it is clear which action is best from the current state. However, this can cause the search to be suboptimal and even incomplete in domains with dead ends unless complete certainty can be reached. It is important to note that time-aware search is different than anytime search which maintains an improving stream of complete incumbent solutions. While algorithms for this setting might find solutions and improve them, the focus of time-aware search is to be able to reduce the total time taken by the agent in producing and executing a plan, partial or complete.

This setting is very different from conventional off-line planning. It is widely understood that finding cost-optimal (ie, shortest) plans does not optimize goal achievement time. For example, an optimal planner may take two minutes to find a plan that takes 30 seconds to execute whereas a satisficing planner might find a plan within seconds that takes only a few seconds longer to execute than the optimal plan. However, it could also be the case that the satisficing planner quickly finds a plan that takes five minutes, in which case the optimal planner would be preferred. Because conventional satisficing planners ignore the concept of wall time, they can perform worse than optimal planners when time matters.

Furthermore, it may be beneficial to begin acting before an entire plan has been constructed. If it seems clear that all good plans share a common prefix, it can shorten goal achievement time to begin execution of that prefix while the tail of the plan is refined. There has been an enormous amount of work on real-time heuristic search (Korf 1990), but that work actually addresses a different objective: constant planning time per action. If it is not clear which action is best, then it may reduce the time to goal to delay acting rather than taking a long and expensive action that turns out to be unnecessary. Or, it may turn out that the action to take is obvious at every step, and virtually no search is actually necessary. Thus real-time search may be suboptimal. What we need is an algorithm that adjusts its deliberations according to the problem at hand. We will use the term *incremental heuristic search* to differentiate this setting from the more rigid real-time one.

The BUGSY algorithm (Ruml and Do 2007) comes close to one of my work's objectives, trying to optimize total search time and solution cost using a user-provided utility function to weigh one against the other. It returns a complete solution, much like A*, but allows for suboptimality in exchange for the expansion of fewer nodes. It uses estimates of the amount of time it would take to find a solution under a node, much like the cost-to-go heuristic estimates the quality of the best solution under a node. While there may be some correlation between the cost-to-go and the search-effort-to-go, an explicit estimation of both allows the algorithm more freedom to decide the true utility of searching under a node.

Even though the intent of BUGSY is to minimize time and cost, it only considers returning complete plans. This allows it to guarantee good features like gracefully handling irreversible actions and maintaining strict bounds on solution quality, but it may lead to longer total time. Not issuing actions as the planning is generated could lead to longer combined planning and execution time. If we truly wish to minimize total time, algorithms which generate complete plans before acting will likely fail us.

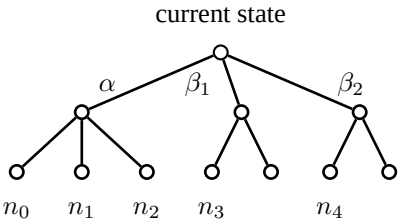

Figure 1: A search tree with three top-level actions.

The Decision-Theoretic A* algorithm (DTA*) of Russell and Wefald, 1991 has many of the properties that we need and is capable of issuing actions before a complete plan has been generated. DTA* uses meta-level search control to decide whether the cost of further search is worth the expected gain in decision quality resulting from having better estimates of action value. If search isn't worth it, then the current best action is executed. To understand how the algorithm works, we will use the simple tree in Figure 1. This tree has three top-level actions applicable in the current state, α, β_1, and β_2. DTA* assumes that the heuristic evaluation function is consistent and thus the value of each top-level action is taken to be the lowest f value of any node on the search frontier of its subtree. We will assume that nodes are arranged left-to-right with the cheapest nodes on the left, so $f(\alpha) = f(n_0) \leq f(\beta_1) = f(n_3) \leq f(\beta_2) = f(n_4)$. Russell and Wefald note that because α is currently the best action, and further search with an admissible heuristic can only cause f values to increase, that further search under β_1 or β_2 cannot change the action that appears best. This means that DTA* need only consider searching under α. In fact, any search under α must expand at least those nodes whose f values are less than or equal to $f(\beta_1)$ in order for the search to have any effect. So DTA* considers expanding supersets of those nodes. For example, if $f(n_1) \leq f(\beta_1) \leq f(n_2)$, then DTA will consider expanding either the set $\{n_0, n_1\}$ or the set $\{n_0, n_1, n_2\}$. For each node to be expanded in a set, DTA* estimates how much its f value is likely to increase. (This estimate is learned off-line from training problems.) With this information, the meta-level control loop can estimate whether $f(\alpha)$ is likely to rise high enough to justify choosing β_1 over α. Because DTA* knows how many expansions it is considering performing, it can weight the expected difference in solution cost against the time required to perform the search.

DTA* is a very elegant algorithm and Russell and Wefald, 1991 present promising results on sliding tile puzzles. Unfortunately, there appears to have been little further work on the topic after the untimely death of Eric Wefald cut short his dissertation work. It has never been applied to more challenging problems, such as AI planning. Nor has it actually been shown to be effective in optimizing true wall-clock time to goal achievement, rather than

1. while the utility of search > 0
2. *action* $\leftarrow$ select action to search
3. repeat k times
4. if a new goal has arrived, then
5. clear the open list and go to line 1
6. if real time has reached the time stamp of the root, then
7. prune everything except the subtree under the *advance time* top-level action
8. go to line 1
9. expand the best node under *action*

Figure 2: The pseudocode for DTOCS

simply abstract nodes generated, a measure that ignores the overhead of meta-reasoning. Likewise, its design depends highly on the assumption of an admissible (or more importantly non-decreasing) cost estimates. When an inadmissible or weighted heuristic is used, the total cost of a node can decrease with more search.

While Russell and Wefald, 1988a make mention of an adaptation of DTA* to an inadmissible heuristic, they only discuss a new calculation of the value of search and ignore the consideration of where search should be done. The natural assumption might be to continue searching under the top-level action with the best estimated cost, but the choice of that expansion order was originally justified by the nodes under that action being the only ones which would cause the agent to change its mind about which action is best. If cost estimates can decrease after search is performed, however, expanding nodes under the second-best action or others may also cause the agent to change its mind. This situation requires could leave DTA* searching much longer under the estimated best action when only a little bit of search under another action would provide clear information about which action is really best.

My proposed algorithm, called Decision Theoretic Online Continual Search (DTOCS) , is capable of using a heuristic function which is not strictly optimistic by measuring the total uncertainty in the believed value of actions. It attempts to decide whether to continue searching or issue the action that appears to be the best. As mentioned above, the objective is to minimize actual time spent for both planning and executing (in addition to minimizing action cost.) Given this novel objective function, the expansion order for nodes during search can not be based solely on the estimated cost of solutions under that particular node. Instead, nodes should be given priority based on the information gained by expanding them. The information that is most important in this setting is which action from the agent's current state lies on the path to a better solution than the others. We want to expand nodes which will decrease the agent's uncertainty about which action is best. Uncertainty in this case must be expressed as a function of heuristic error, allowing the agent to estimate the probability of belief over action values changing in a significant way should more search be performed. Pseudocode for DTOCS is given in Figure 2.

DTOCS decides whether to issue an action by estimating whether the expected benefit of further search outweighs the cost of the time more search would take. Assume we have an action α that we currently believe is the best action, and an

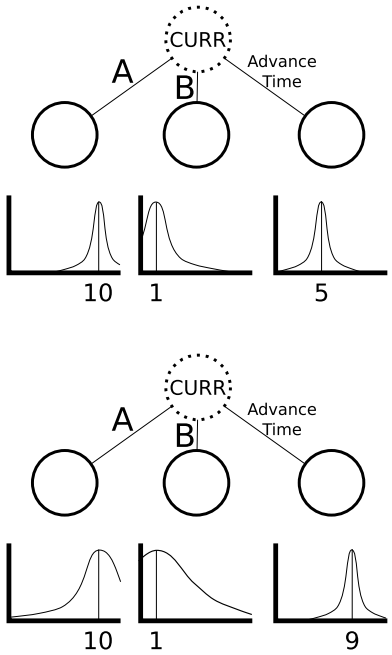

Figure 3: **Top**: The choice of action is clear. **Bottom**: Search may be justified.

action β that we are not completely certain is worse. More search may reveal that β is, in fact, the best and α is not, but it will cause us to issue an action later and, therefore, increase total search time. We issue actions when more search would likely cost more than it would offer in plan improvement.

Figure 3 shows an example for parallel planning, in which the agent can choose to begin new actions or wait until a later point in time. On the left, action A has an expected net utility of 10 and the variance in the belief distribution is small, indicating that we are quite certain about that value. Action B has little overlap with A, and we are certain enough about the value of advancing time that it is unlikely that further search would reveal it to be better than A. Delaying execution for search would be a waste of time, so we choose to issue A. On the right in Figure 3, A still has the best expected value but we are less certain. We are also less certain about B's value, and advance time has substantial overlap with A. Depending on the value we attribute to time versus plan cost, further search may be beneficial to refine the agent's beliefs about A and B. It is important to note that uncertainty about A would not be enough on its own to merit more search. Had we been uncertain about A but very certain that the values of all other actions were lower than A's, we would still be unlikely to change the agent's preference for A after more search. To reduce the overhead of this meta-level control, we assume a fixed 'quantum' of search (notated k on line 3 in Figure 2) between meta-level deliberation.

My work so far has addressed this issue of optimizing total goal achievement time, and resulted in the DTOCS algorithm. Future work will require evaluation of DTOCS in comparison to traditional full-plan algorithms like weighted A* and BUGSY as well as incremental search algorithms such as RTA* and DTA*. I will evaluate DTA* over a variety

of problems, such as pathfinding and domain-independent planning, both sequential and parallel.

On-line Goal Arrival

The next section of my dissertation is to handle the arrival of additional goals during planning and execution. I assume that goal arrival does not change the state in terms of the applicability of actions, meaning that the existence or non-existence of a goal does not invalidate an existing plan, although it may change its reward value. Much like traditional planning, action results are deterministic, as opposed to settings in which executed actions may fail to achieve some or all of their post-conditions or cause additional effects beyond what was expected. One could simply synthesize a complete plan for the current set of goals, replanning whenever new goals arrive (Benton, Do, and Ruml 2007; Gratch and Chien 1996; Ruml, Do, and Fromherz 2005), but the complexity of complete plan synthesis can delay the start of action execution.

Because we assume that the goal arrival distribution is known (or estimated on-line), we want to be able to plan for the future, acting in a way that anticipates any likely future goal arrivals. The agent does so by generating sampled futures using the arrival probability, allowing it to deal with a determinized version of the problem. These samples are created by projecting into the future out to a given horizon, with possible goal arrivals and the times at which they are imagined to have arrived. Planning can then be done for these samples to determine the value of each action. I am collaborating with the creators of an "anticipatory" algorithm for handling on-line goal arrival (Hubbe et al. 2008) and will extend upon their work by generating a single plan for all sampled futures (rather than a plan for each sample), allowing work to be reused after each action is issued.

My work with DTOCS will be extended to handle goal arrival, but several factors need to be taken into consideration: 1) given that goals arrive as we plan and execute, DTOCS must predict and plan ahead for likely future goals; 2) a policy must be established for when to generate new samples, since passing time may make them less accurate or useful; and 3) given heuristic evaluation over several sample futures, there may need to be a change in the expansion order for search.

Asynchronous State Changes

To extend my work to an even more general case, I will work on the problem of state changes outside the agent's control during search. Work on adapting to changing goal sets should provide insight into the problem of changing state. This feature can be viewed, much like goals, as the arrival of new facts according to some probability function. It is general enough to encompass several other interesting features often ignored by traditional planning, such as arriving goals which affect action applicability or even action failure.

This work will also be centered around methods of determinizing the problem and attempting to predict future outcomes using sampling. Samples can similarly be drawn using a probability distribution, as before, but the distribution must be over arrival of general world facts, rather than goals. The number of likely fact arrivals may be different than the number of possible goal arrivals, but it is not clear whether one will necessarily be larger than the other.

All of the issues required for consideration in handling on-line goal arrival are outstanding for fact arrival, as well. The biggest new issue for this stage is the applicability of actions in sampled futures or when facts actually do arrive. The planning methods discussed for anticipating goal arrival will not be directly applicable because an existing plan can be made invalid by the arrival (whether predicted or real) of new facts. Planning must be done in a way that can handle when actions become applicable or inapplicable. This may be as simple as simulating a plan over each sampled future as before and assuming that actions which are not applicable in that future incur cost as normal but fail to achieve their post-conditions. It may also be complex enough to necessitate the algorithm maintaining multiple partial plans or repairing them as necessary.

There is some room to extend the complexity of the arrival function in this stage, as well. A random probability distribution for fact arrivals is the simplest assumption, but it does not allow for causality in the arrival of facts. This means, for instance, that an action with varying effects could not be fully modeled. Some distribution that allows for cause and effect would be an interesting extension to this work.

References

Benton, J.; Do, M. B.; and Ruml, W. 2007. A simple testbed for on-line planning. In *Proceedings of the ICAPS-07 Workshop on Moving Planning and Scheduling Systems into the Real World*.

Gratch, J., and Chien, S. 1996. Adaptive problem-solving for large-scale scheduling problems: A case study. *Journal of Artificial Intelligence Research* 4:365–396.

Hubbe, A.; Ruml, W.; Yoon, S.; Benton, J.; and Do, M. B. 2008. On-line anticipatory planning. In *Proceedings of the ICAPS-08 Workshop on A Reality Check for Planning and Scheduling Under Uncertainty*.

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42:189–211.

Ruml, W., and Do, M. B. 2007. Best-first utility-guided search. In *Proceedings of IJCAI-07*, 2378–2384.

Ruml, W.; Do, M. B.; and Fromherz, M. P. J. 2005. On-line planning and scheduling for high-speed manufacturing. In *Proceedings of ICAPS-05*, 30–39.

Russell, S., and Wefald, E. 1988a. Decision theoretic control of reasoning: General theory and an application to game-playing. Technical Report UCB/CSD 88/345, University of California, Berkeley.

Russell, S., and Wefald, E. 1988b. Multi-level decision-theoretic search. In *Proceedings of the AAAI Symposium on Computer Game-Playing*.

Russell, S., and Wefald, E. 1991. *Do the Right Thing: Studies in Limited Rationality*. MIT Press.