When to Commit to an Action in Online Planning and Search

Tianyi Gu,[1] Wheeler Ruml,[1] Shahaf S. Shperberg,[2, 4] Eyal Shlomo Shimony,[2] Erez Karpas[3]
[1]University of New Hampshire, USA
[2]Ben-Gurion University of the Negev, Israel
[3]Technion, Israel
[4]University of Texas at Austin, USA
gu@cs.unh.edu, ruml@cs.unh.edu, shperbsh@post.bgu.ac.il, shimony@cs.bgu.ac.il, karpase@technion.ac.il

Abstract

In online planning, search is concurrent with execution. Under the formulation of planning as heuristic search, when a planner commits to an action, it re-roots its search tree at the node representing the outcome of that action. For the system to remain controlled, the planner must commit to a new action (possibly a no-op) before the previously chosen action completes. This time pressure results in a real-time search. In this time-bounded setting, it can be beneficial to commit early, in order to perform more lookahead search focused below an upcoming state. In this paper, we propose a principled method for making this commitment decision. Our experimental evaluation shows that our scheme can outperform previously-proposed fixed strategies.

Introduction

Many applications of planning involve time pressure. Often, we wish to achieve the goal as soon as possible, minimizing the so-called Goal Achievement Time (GAT) (Hernández et al. 2012; Kiesel, Burns, and Ruml 2015). This can be thought of as the sum of the planning time before the agent starts executing plus the execution time until the agent reaches the goal. In situated temporal planning, external time constraints, such as buses or trains that depart at scheduled times, cause plans to become infeasible if planning takes too long (Cashmore et al. 2018; Shperberg et al. 2021). Having a plan to take a certain bus does an agent no good if the bus has already left. One way to address time pressure is to design faster planning algorithms. But many planning problems are inherently intractable. The most direct way to address planning under time pressure is to allow actions to begin executing before a complete plan has been found. Further planning can then overlap with action execution. In other words, the planning time before execution starts may be very short, but the rest of the planning time becomes 'free', as it overlaps with the execution time. The fundamental question raised by such concurrent planning and execution settings, beyond those already present in offline planning, is: when should the planner commit to an action?

Existing methods offer simple fixed answers to this question. Some methods use a fixed amount of lookahead for every action selection decision and commit to exactly one action given this lookahead. This is the scheme originally proposed in Real-Time A* (Korf 1990). We call this commitment strategy a *commit one* strategy. This is conservative, in that the single action is chosen on the basis of significant lookahead, but also incurs some node re-generation overhead as the entire subtree below that action may be re-explored during the next planning iteration. Other methods commit to multiple actions, possibly even the entire sequence of actions leading all the way to the search frontier. We call the latter extreme commitment a *commit all* strategy. The commit all strategy is riskier, as little lookahead has been done below the actions near the search frontier. However, the agent now has much more time to select its subsequent actions, as many actions have been queued for execution.

These two approaches form the extreme points of a spectrum that exhibits an interesting tradeoff. On the one hand, committing to only one action as late as possible preserves the most flexibility for the planner, as it only prunes plans as a last resort. However, this comes at the expense of the need to perform more search, to cover the larger space of possible plans, as well as less search time to make a decision for each action commitment. On the other hand, committing to long sequences of actions early allows the planner to have a longer time to perform a deeper lookahead before its next commitment decision. However, this comes at the expense of a higher risk of committing to a dead-end, and thus never reaching the goal.

In this paper, we propose what is, to our knowledge, the first principled approach to action commitment in online planning. We develop and test a method that uses heuristic information to assess the planner's uncertainty about action values. This uncertainty then drives the decision of whether to commit to an action or whether to perform additional lookahead before deciding. Our results indicate that the approach has promise, as it outperforms previous non-adaptive strategies in three challenging scenarios.

Background

Problem Setting

Our problem setting requires the system to be controlled at all times. That is, some action must be executing at any given time, even if it is just a no-op that leaves the state unchanged. Some domains, such as fixed-wing aircraft control, do not have no-op actions. We make the additional simplifying assumptions that actions are serial and that the world is fully observable and deterministic. Thus, we have a planner searching for a sequence of actions under the constraint that at all times at least one action (beyond those that have already completed execution) has been computed, committed to, and has begun execution. The objective of the system is to achieve a goal as soon as possible. Because we are addressing concurrent planning and execution, we use GAT as our main evaluation metric. This is the total time taken from the start of planning to the arrival of the agent at a goal.

We take a heuristic search perspective, in which planning explores an incrementally-generated tree of feasible action sequences, with the root of the tree representing the state resulting from the execution of all the actions that have been committed to up until now. The planner is allowed to commit to actions earlier than required, in order to allow it to re-root the tree at a deeper node, thereby focusing the search later on into the future. A commitment queue records all the committed actions. How and when to make such additional commitments, so as to reduce the expected time to reach the goal, is the focus of this paper. Following Russell and Wefald (1991), we aim to pose and solve this question as a decision-theoretic metareasoning problem. However, even this limited focus is too general to formalize, hence we make additional metareasoning assumptions about the search process:

1. The order of decisions in the planner is a fixed search tree structure, from early actions to later actions.
2. No replanning is permitted after action commitment: a decision to commit to an action in the sequence means that it will eventually be executed in the order specified.
3. The only question we address is when to 'reroot the tree' at a successor of the root, that is, should we do this before it is necessary?
4. We assume a given expansion strategy that is not modified by the commitment strategy, other than by pruning the parts of the search tree inconsistent with the action commitments.

Note that the metareasoning assumptions are meant to define the constraints on the decision-making at the metareasoning level, rather than representing assumptions about the domain or the planner. They are used to define the distributions and utilities. In fact, in an actual implementation the planner may deliberately act in a way that does not conform to the assumptions, especially when it is obvious that better performance can be achieved by violating the assumptions. For example, it is possible, due to several commitment decisions, to get a commitment queue containing a sequence of actions that makes the agent walk in a loop. In such cases, if such a loop is detected by the planner before beginning to execute these actions, the agent may decide to remove such a loop from the action queue, even though this may not be allowed by the metareasoning assumptions. Alternatively, we may re-start the search at a new state if necessary, for example, if the controlled system departs from our assumption of determinism.

Periodically during the search, perhaps after each expansion or periodically after a set of expansions, a metareasoning process decides between two options:

- commit to the current seemingly-best top-level action now and re-root the search tree accordingly, or
- postpone the commitment and continue the current search.

Note that if we always decide to postpone, eventually action execution will reach the current root node state and force us to commit to a next action.

Previous Work

The seminal work of Korf (1990) defined the problem setting of single-agent real-time search, in which a fixed number of expansions (or equivalently, amount of time) is allowed for lookahead node expansions, after which the search must commit to the next action to take and re-root the search tree. Korf's RTA* and LRTA* algorithms back up h values from the lookahead frontier to inform the action choice, caching the backed up values at every node to allow the heuristic information to become more accurate over time and provably prevent the search from becoming stuck in infinite loops. The LRTA* variant provably converges to the optimal h values. These algorithms were designed to be simple and amenable to analysis. They commit only to a single action, which means that the lookahead of one iteration can have significant overlap with the nodes visited in the previous iteration, depending on the state space connectivity and the heuristic function.

The widely popular LSS-LRTA* algorithm (Koenig and Sun 2008) takes a different approach, committing to the entire sequence of actions leading to the most promising frontier node. This reduces the re-generation of nodes seen during the previous lookahead and reduces the overall overhead of the search per executed action, but also commits the agent to certain actions, such as those at or near the frontier, for which little lookahead has been performed and for which the heuristic value of their resulting successor state is their only attractive attribute.

The Dynamic $\hat{f}$ algorithm (Kiesel, Burns, and Ruml 2015) modifies LSS-LRTA* in two ways. First, rather than idling the planner for $k - 1$ time steps after committing to k actions, Dynamic $\hat{f}$ uses the entire time until all the committed actions have finished executing to perform lookahead search. The amount of lookahead is thus adjusted dynamically, rather than being fixed from the start. This often results in the next iteration having a sequence of more than k actions to the best node on the search frontier, leading to a virtuous cycle of larger and larger lookahead. Second, rather than expanding the frontier node with the lowest f value, the algorithm computes an inadmissible heuristic $\hat{h}$, which when added to g yields the inadmissible (but possibly more accurate) total plan cost estimate $\hat{f}$. By selecting the node with lowest $\hat{f}$, Dynamic $\hat{f}$ tries to avoid being tempted by shallow nodes whose admissible f values are low merely because they haven't been explored as deeply as others.

The stark contrast between the two fixed commitment strategies of LRTA* (one action) and LSS-LRTA* and Dynamic $\hat{f}$ (all the way to the frontier) raises the question of whether a principled adaptive strategy can be found to decide when to commit to an action. The first approach in this direction was Decision-Theoretic A* (DTA*) (Russell and Wefald 1991), which attempts to optimize GAT by periodically deciding whether to continue the current lookahead search or commit to an action and re-root the tree. This is done by estimating whether the improvement in decision quality, measured by reduction in plan length, that is likely to result from further search would outweigh the time required to do the further search itself. In the implementation used for their experiments, training data was used to gather statistics on how often, and by how much, heuristic estimates tend to change as a results of further search. DTA* is not a real-time search algorithm, in that it does not respect or consider a time bound on lookahead. There is no requirement that the system constantly be executing an action and it is always permissible to deliberate further. Thus DTA* is capable of emulating A* and planning all the way to a goal before committing to its first action. DTA* is based on the less-performant depth-based lookahead of RTA* rather than the f-based lookahead of LSS-LRTA*, but it pioneered the deliberative metareasoning approach to action commitment.

The Mo'RTS algorithm of O'Ceallaigh and Ruml (2015) is basically a modification of DTA* into a true real-time search algorithm based on LSS-LRTA*. We focus here on its action commitment strategy, called $\hat{f}_{PMR}$. It assumes that a no-op 'identity' action is available in every state, which allows the planner to continue searching from the same root. Once the path from the root to the most promising frontier node has been identified, $\hat{f}_{PMR}$ considers each node in turn, asking whether additional search would be worthwhile, and stopping at the first node for which this appears true. However, $\hat{f}_{PMR}$ does not offer a principled way to evaluate this decision at each node. It estimates the benefit of search as the expected reduction in time-to-goal resulting from more certain estimates of action cost, which seems reasonable. However, it is much harder to assess the costs of stopping the re-rooting process short of the frontier. The $\hat{f}_{PMR}$ method uses the time required to regenerate the path from the node to the frontier, which, as the authors note, is not particularly reasonable because this repeated work would likely happen concurrently with execution, not affecting the goal achievement time directly at all. This leaves the approach fundamentally unsatisfying.

In this work, we propose what we believe to be a more principled metareasoning scheme for action commitment, which we call Flexible Action Commitment Search (FACS). We integrate FACS into Dynamic $\hat{f}$ and assess its behavior using three challenging grid pathfinding scenarios specially designed to stress real-time search in different ways.

Metareasoning for Action Commitment

Our objective is to minimize GAT. Thus g, h, and f values in the state space will represent the duration of actions and the total utility of a final outcome is exactly the sum of action costs/durations taken to reach the achieved goal state. So optimizing f directly optimizes total utility.

The metareasoning problem of heuristic search can be conceptualized as a POMDP in which each state represents an entire state space graph, complete with costs on every arc and h values at every vertex. To avoid confusion in this discussion, we will use the term 'vertex' for a node in the state space graph and the term 'state' for a state in the POMDP. The search does not know which exact state space graph it is dealing with, thus its situation is captured by a belief distribution over states. Every node expansion action results in an observation that rules out those state space graphs that are inconsistent with the vertices, action costs, and h values that are generated. The action of expanding a node is stochastic in that the search does not know in advance which new nodes, actions costs, and h values will be observed, so there are many possible belief distributions resulting from every expansion. The action of committing to an action and re-rooting the search tree at a new vertex is deterministic, as it does not yield new information. A goal in the POMDP is a belief that has positive support only on state spaces that all share the same path from the initial vertex to a goal vertex, providing a solution to the original problem but potentially harboring remaining uncertainty about the unseen portions of the graph. A policy for the POMDP corresponds to a search strategy, as it would prescribe an action for the search to take at every reachable belief state. Solving the POMDP for a policy that, for example, minimizes expected solution length would give a heuristic search strategy that finds a solution as quickly as possible by minimizing the expected number of expansions. Approaching such a problem in practice depends crucially on exploiting structure in the h values, the arc costs, and the distance to the nearest goal.

It is not feasible to solve this POMDP, or even to find a reliable approximation of its solution using standard approximation methods. Therefore, we propose a myopic metareasoning scheme that only considers the next action commitment decision. In this formulation, one of the following two options need to be chosen:

- Commit to the action with the best (least) estimated $\hat{f}$-value among all children of the current root node. We denote the node that corresponds to this action by α.
- Do not commit to α yet, and spend more time searching before deciding which action to take next.

Prematurely committing to α might reduce the quality of the solution. For example, if α leads to a dead-end and the search algorithm has failed to figured that out before committing, then it would be forced to turn around eventually, which would result in a solution with an increased cost. On the other hand, by gaining additional search time before making consequent decisions the search algorithm might be able to avoid future dead-ends or pitfalls, which would not have been possible to avoid otherwise. Thus, the utility of committing to α or not committing to α should depend in part on our certainty regarding the $\hat{f}$-value of α.

The Effect of Committing

Let $P_s^d(x)$ be the predicted probability of having the belief that $\hat{f}(s) = x$ given d more node expansions of search under node s. Denote by X_s^d the random variable distributed as P_s^d.

We begin with several additional simplifying assumptions:

1. Each node has exactly two children (a branching factor of 2), α and β, where α is the node with the highest expected utility (lowest expected $\hat{f}$-value); we will relax this assumption in the next subsection.
2. The time d_f required to fully execute an action is identical for all actions; this assumption will also be relaxed later.
3. When searching under a node s, the search time is evenly divided among all of its children.
4. The X_s^d random variables are independent for all d and s.

Under the above assumptions, we can now estimate the utility of committing to and of not committing to α. In Figure 1(a), we show the current search tree rooted at node s. The available search time consists of the remaining time d_r induced by previous commitments and d_f, the time required to execute a full action α or β (see the the top red time line in Figure 1(b)). By committing to α, the agent would be able to invest all of the available search time to search under the children of α (the bottom red time line in Figure 1(b), starting with the word "commit"). We denote the children of α as $\alpha\alpha$ and $\alpha\beta$ (again, see the search tree in Figure 1(a)). Since we assume that search time is evenly divided between $\alpha\alpha$ and $\alpha\beta$, each of them receives a search duration budget of $d = \frac{d_r + d_f}{2}$. Thus, the utility estimate of committing to α (denoted as U_{commit}) can be defined as the expectation of the minimum $\hat{f}$-value of $\alpha\alpha$ and $\alpha\beta$, after searching d time units under each of them:

$$U_{\text{commit}} = \mathbb{E}\left[\min(X_{\alpha\alpha}^d, X_{\alpha\beta}^d)\right] \tag{1}$$

If the agent chooses not to commit yet (commit later), the remaining time d_r will be used to search under current root s (see the middle red time line in Figure 1(b), starting with the words "do not commit"). Thus half of d_r ($\frac{d_r}{2}$) will be used to search under each child of the root. Even though α is initially estimated to have the lowest $\hat{f}$-value among the children of the root, this estimation can change after searching for $\frac{d_r}{2}$ time under α and β. In essence, the new $\hat{f}$-value estimation of α, induced by the additional search, can be greater than the new $\hat{f}$-value estimation of β. Thus, the rest of the time line (d_f) is used for searching under whichever child of the root is judged most promising at that time (again, see the middle red time line in Figure 1(b), starting with the words "do not commit"). As a result, the search duration under each grandchild of the current most promising child (either α or β) will be $d' = \frac{\frac{d_r}{2} + d_f}{2}$. In our simplification, the branching factor is 2, so:

Case 1: after $\frac{d_r}{2}$ time spent searching under α and β, we will believe that $\hat{f}(\alpha) \leq \hat{f}(\beta)$. In this case, the rest of the search time would be invested under α:

$$U_\alpha = \mathbb{E}\left[\min(X_{\alpha\alpha}^{d'}, X_{\alpha\beta}^{d'})\right] \tag{2}$$

Case 2: after $\frac{d_r}{2}$ time spent searching under α and β, we will believe $\hat{f}(\alpha) > \hat{f}(\beta)$. Symmetrically to the previous case, here the rest of the search time would be invested under β:

$$U_\beta = \mathbb{E}\left[\min(X_{\beta\alpha}^{d'}, X_{\beta\beta}^{d'})\right] \tag{3}$$

Then, we can estimate the overall utility of committing later by weighting the probability of α and β to become the most-promising nodes after the initial search time with their corresponding utilities. The probability of α becoming the most-promising child (choosing to commit to α) can be defined as follows:

$$P_{\text{choose } \alpha} = P((X_\alpha^{\frac{d_r}{2}} - X_\beta^{\frac{d_r}{2}}) < 0) \tag{4}$$

The utility of not committing at t' denoted $U_{\text{do not commit}}^{t'}$ can be estimated as:

$$U_{\text{do not commit}} = P_{\text{choose } \alpha} \cdot U_\alpha + (1 - P_{\text{choose } \alpha}) \cdot U_\beta \tag{5}$$

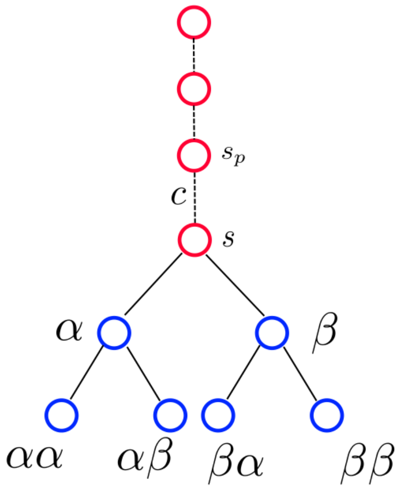
Using these equations, the metareasoning scheme simply needs to compute the utility of committing to α (Equation 1) and not committing to α (Equation 5), and to choose the metalevel action with the highest utility (lowest expected cost).

A Conceptual Example

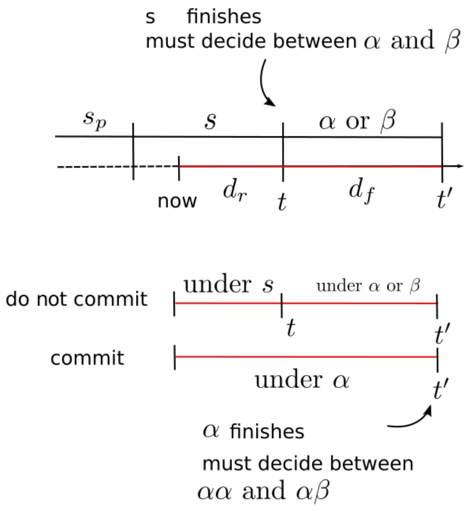
In Figure 1, suppose that after the search, we obtain the expected cost under each leaf node, so we have $\hat{f}_{\alpha\alpha} = 3$, $\hat{f}_{\alpha\beta} = 5$, $\hat{f}_{\beta\alpha} = 4$, $\hat{f}_{\beta\beta} = 6$. And we also have $\hat{f}_\alpha = 3$, $\hat{f}_\beta = 4$ simply by backing-up from their best child node $\alpha\alpha$ and $\beta\alpha$ respectively. We are at the root node s and want to decide whether to commit to the current best action and re-root the search at α or not commit and keep searching under s. Suppose further that the expansion rate is 10 expansions per action duration, and that the action c leading to s is currently 5 expansions from completing execution. In this case, $d_r = 5$ and $d_f = 10$.

If we choose to commit, the total 15 expansions will be used to perform search under α, so $\alpha\alpha$ and $\alpha\beta$ both gain 7.5 expansions under our even division search time assumption. Now we can obtain the belief distribution for the future $\hat{f}$-value after search via Equation 10 (discussed below): $X_{\alpha\alpha}^{7.5} \sim \mathcal{N}(3, 0.4)$, $X_{\alpha\beta}^{7.5} \sim \mathcal{N}(5, 2.0)$. Then by applying Equation 1, we get the $U_{commit} = 3.2$. This can be calculated directly using the closed-form formula for the minimum of two normally distributed random variables (Nadarajah and Kotz 2008).

If we choose not to commit, we have two search phases: before and after c completes. In the first phase, we still search sub-trees under both α and β, so both gain $d_r/2 = 2.5$ expansions. Because the system can not be left uncontrolled, we are forced to commit when c completes. So in the second phase, after c completes, the search will only expand nodes either under α or β with $df = 10$ expansions. Thus we have $d' = (2.5 + 10)/2 = 6.25$ expansions



(a)



(b)

Figure 1: Committing vs not committing.

for each leaf node. To compute U_α, now we can again obtain the belief distribution of future $\hat{f}$-value by Equation 10: $X^{6.25}_{\alpha\alpha} \sim \mathcal{N}(3, 0.2)$, $X^{6.25}_{\alpha\beta} \sim \mathcal{N}(5, 1.5)$. Equation 2 can give us $U_\alpha = 3.1$. The same computation can be applied to the β subtree to get $X^{6.25}_{\beta\alpha} \sim \mathcal{N}(4, 0.1)$, $X^{6.25}_{\beta\beta} \sim \mathcal{N}(6, 1.3)$, and $U_\beta = 4.2$. By Equation 4, say we get $P_{choose\alpha} = 0.7$, then we can have $U_{\text{do not commit}} = 0.7 \times 3.1 + 0.3 \times 4.2 = 3.43$. In this case, the meta-level decision is to commit since it results in the lowest expected cost of 3.2.

Relaxing the Assumptions

In order to relax the branching factor 2 and the identical action duration assumptions, we make the following modifications. Let $Children(x)$ be the set of children of node x, let $b = |Children(root)|$, and let d_a be the duration of action a. First, the search times d and d' needs to be updated with respect to b as follows: $d = \frac{d_r + d_\alpha}{b}$, $d'(a) = \frac{\frac{d_r}{b} + d_a}{b}$. Note that now d' is a function of the action chosen to be taken from the root. Then, the utility functions need to be updated. The utility of committing (Equation 1) should be generalized to:

$$U_{\text{commit}} = \mathbb{E}\left[\min_{c \in Children(\alpha)} X^d_c\right] \quad (6)$$

The utility of searching d' time under node c (generalization of equations 2 and 3):

$$U_c = \mathbb{E}\left[\min_{c' \in Children(c)} X^{d'(c)}_{c'}\right] \quad (7)$$

The probability of choosing node c after searching d' time under each child of the root is:

$$P_{\text{choose } c} = P(\operatorname*{argmin}_{c' \in Children(root)} X^{\frac{d_r}{b}}_{c'} = X^{\frac{d_r}{b}}_c) \quad (8)$$

Thus, the utility of not committing (generalization of Equation 5) becomes:

$$U_{\text{do not commit}} = \prod_{c \in Children(root)} P_{\text{choose } c} \cdot U^{t'}_c \quad (9)$$

Defining the $P^d_s(x)$ Distributions

The $P^d_s(x)$ distributions should reflect the effect of search effort under nodes given d more node expansions. Specifically, the more we search under a node, the more likely that our estimation of its $\hat{f}$ value will change and get closer to its true value. In addition, we assume that the closer a node is to a goal, the more accurate the original estimation of its $\hat{f}$ value. Finally, the average heuristic error on the path which leads to s from the root, $\bar{\epsilon_s}$, can be used as an indicator of the quality of the $\hat{f}$ value estimation of s. We estimate heuristic error by averaging the errors experienced at each expansion. The heuristic error experienced at state s is defined as $h(s) - (h(bc(s)) + c(s, bc(s))$, where $bc(s)$ is the best child of s, that is, the successor of s with the lower heuristic estimate. We take this approach from (Thayer, Dionne, and Ruml 2011).

So the variance of $P^d_s(x)$ should grow proportionally to d and $\bar{\epsilon_s}$, and the distance-to-go estimation of s. Let $dtg(s)$ be the distance-to-goal estimation of node s, measured in number of 'search steps' (expansions necessary for the search to generate the path). When each successor extends the previous plan by exactly one action, dtg is equivalent to an estimate of the number of actions remaining, and dtg plus depth equals estimated solution length. When actions are unit cost, these are the same, but that is not always the case. In most domains, $dtg(s)$ can be computed similarly to $h(s)$. Let ed be the average *expansion delay*, which measures the number of node expansions from the moment a node is generated until it is expanded (Thayer, Dionne, and Ruml 2011). This can be estimated by assigning every node a serial number when it is generated and comparing each expanded node's number with the current counter. If ed is the expansion delay, it takes (on average) expanding ed nodes to advance one layer in the search tree, so d/ed gives us the number of layers we advance (distance explored) when performing d expansions.

For an initial estimate of the variance, we use the square of the heuristic error multiplied by the distance-to-goal of s. This uncertainty estimate is modeled as being reduced according to the fraction of the distance to the goal that we expect to explore using d node expansions (d divided by the expansion delay represents the additional depth that we estimate the lookahead will afford us). Since $dtg(s)$ represents the estimated depth remaining, $(d/ed)/dtg(s)$ represents the fraction of the distance to the goal that we estimate the lookahead being able to explore. If the expected exploration depth surpasses the estimated distance-to-go, we clamp the fraction at 1. The mean of the distribution is the current cost-to-goal estimation, $\hat{f}(s)$. Putting it all together, we model $P_s^d(x)$ as a normal distribution:

$$P_s^d = \mathcal{N}(\hat{f}(s), (\bar{\epsilon}_s \cdot dtg(s))^2 \cdot \min(1, \frac{\frac{d}{ed}}{dtg(s)})) \qquad (10)$$

To summarize, our Flexible Action Commitment Search (FACS) approach uses the P_s^d estimates about how the planner's beliefs about α and β will change after search in order to estimate U_{commit} and $U_{\text{do not commit}}$ and hence decide whether to commit to α or continue searching.

Empirical Evaluation

Although our approach seems to be a more principled commitment strategy, when all of its assumptions hold, than the fixed approaches in previous work, it remains to be seen how it performs in practice. In this section, we integrate FACS into Dynamic $\hat{f}$ and empirically evaluate it against three baselines: original LSS-LRTA* (i.e., commit-all), LSS-LRTA* with commit-one, and Dynamic $\hat{f}$ (i.e., commit-all with dynamic lookahead). The LSS-LRTA* variants also use $\hat{f}$ expansion strategy.

Synthetic Grid Pathfinding Domain

We implemented all real-time search agents in a synthetic 4-connected grid pathfinding domain. Both cost- and distance-to-go are estimated using the Euclidean distance heuristic. Figure 2 shows a schematic view of a tricky instance of our novel variant of the classic grid pathfinding problem, specially contrived to challenge real-time search. The black areas are the obstacles. The red patches are 'tar pits', cells for which the cost of moving to an adjacent empty cell is very high (i.e., high cost for stepping out from a tar pit). With this setting, there will be a high cost if an agent commits to an action that steps into a pit, as the agent would have to step out of it in order to reach the goal. Note that the admissible heuristic function does not take these costs into account. Therefore, the agent has to be very careful about its commitment decisions. The small red tar pits are very common in the left part of the map, so a search's lookahead frontier will have a high probability of including at least one, possibly even as the best node. Thus, we expect an agent with a strategy that commits all the way to the frontier to be fairly likely to step into a tar pit at some point. In the middle, we have a long empty area. Since in this area there are no traps or mazes, algorithms can safely commit and re-root the search to the frontier nodes in order to gain search time for the future. In contrast, algorithms that conservatively commit only to one action at a time and re-root the search tree at every step cannot benefit from such gains in future search time. On the right side of the map, we have a corridor setting. Again, the red region is a large tar pit, where there is a small cost of stepping into this area, but a large cost of getting out of it. If agents do not have sufficient lookahead to observe that the red region is a high cost local minimum, they are likely to be tempted to get into this large tar pit, as it will seem to be a shorter path to the goal because we sample the goal position from the lower rows and sample the entrance of the corridor from upper rows so that the agent must go against the Euclidean heuristic to enter the upper corridor. However, with a sufficiently large lookahead, agents can detect that this tar pit is a dead-end and avoid stepping into it altogether. Therefore, we expect agents that accumulate search time (i.e., lookahead) during the middle empty region to be able to utilize it to avoid the large tar pit. In the next section, we show results for maps that only have the left-side tar pits, maps that only have the right-side corridor and pit, and maps with both left-side and right-side pits.

Experiments

All algorithms were implemented in C++ [1] and run on 64-bit Linux systems with 3.16 GHz Intel E8500 processors and 8 GB of RAM. We used grid maps of 50 rows and 200 columns. For each map, we set the start in the left-most column and goal in the right-most column, randomizing the row numbers of the start position and goal position to generate 100 problem instances. We used lookahead limits of 4, 10, 30, 100, and 300 expanded nodes per action (i.e., the relative search vs action execution speed), shown in the x-axis of each plot in Figures 3-5. We set the cost of stepping out of a tar pit as 1,000 expansions. The y-axis shows GAT, normalized as a factor of the GAT of a clairvoyant agent that immediately commits to an optimal plan without searching. Error bars show 95% confidence intervals on the mean over all the instances. The legends are sorted by the geometric mean across all lookahead limits.

Figure 3 shows the result of grid pathfinding problems with tar pits near the start. FACS performs consistently close to clairvoyant across all search speeds. The commit-one strategy is also very competitive at low search speed, due to its conservative commitment strategy that helps the agent avoid stepping into tar pits. The Dynamic $\hat{f}$ and commit-all strategies are both far from optimal in this map, with dynamic $\hat{f}$ stepping less frequently into tar pits as it accumulates a slightly longer lookahead by the time it reaches the tar pit field.

[1] See https://github.com/gtianyi/https://github.com/gtianyi/metaReasoningRealTimePlanning.

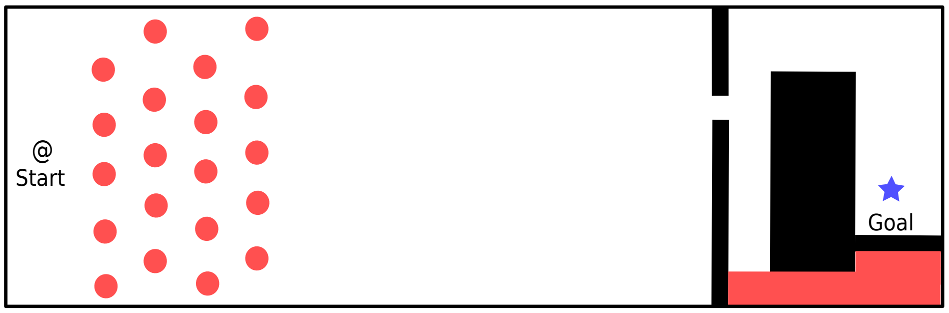


Figure 2: Schematics of grid benchmarks with tar pits.

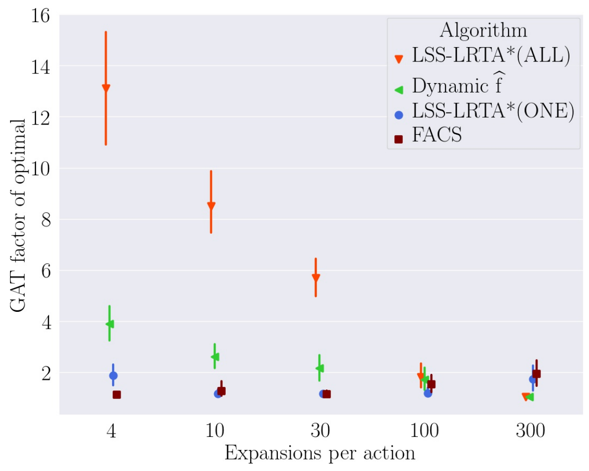


Figure 3: Goal achievement time as a function of search speed in grid pathfinding with tar pits near the start.

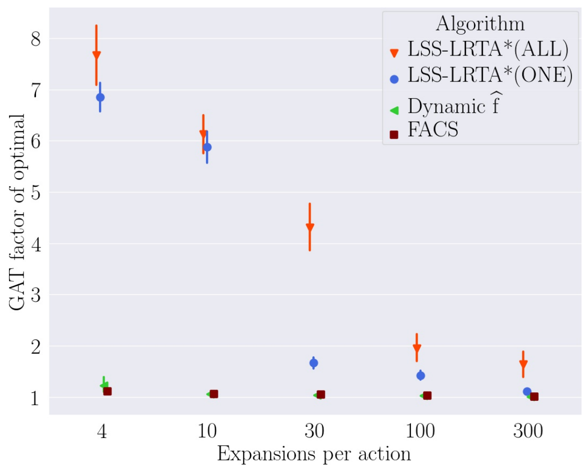


Figure 4: GAT with corridor and tar pit near the goal.

Figure 4 shows results for maps with a corridor area near the goal. Both FACS and dynamic $\hat{f}$ perform close to clairvoyant since they take advantage of accumulating search time in the empty area and thus have a lookahead that is large enough to detect the dead-end and avoid stepping into the high-cost trap area. Both variants of LSS-LRTA* are unable to detect the dead-end with lookahead limit below 30.

In Figure 5, we show the results for maps with tar pits both near the start and near the goal. FACS is able to survive both the tar pit field and corridor field, with a conservative commitment strategy initially, adapting to an aggressive commitment strategy in the empty area, and having a sufficiently large lookahead to avoid the large tar pit when reaching the corridor.

Discussion

We have suggested an approximate metareasoning scheme for action commitment geared at focusing a real-time search, in order to get additional time to search farther ahead in the search tree. Our scheme involves some domain assumptions, as well as several metareasoning assumptions. while FACS performed well in our contrived grid pathfinding domain, further experiments are necessary to characterize when its assumptions lead to poor behavior.

Possible Extensions

Typically, metareasoning assumptions are not true limitations. Instead, these are just a way to simplify the semantics and computation of expected utilities for the search. The independence assumption falls in this category; it is made so that we can have an easy-to-compute estimate, even though in practice it does not hold.

Our treatment of action commitment is in a different category. As mentioned above, if we observe that we have committed to a of set actions that will lead us through a loop in the path, it is clear that this sequence of actions achieves nothing, except a delay. Even in such cases, this delay may be useful as it can be used to do additional search before physically reaching a possible trap (Cserna, Ruml, and Frank 2017). It thus is a non-trivial issue when we might wish to un-commit actions, even in such a seemingly simple case.

We might also wish to un-commit actions when we observe that the predicted f-costs resulting from deeper search are much worse than those initially projected. In such cases where actual action execution has not reached such an unexpectedly bad state, it may be better to un-commit actions and expand nodes that seemed worse and pruned by the commitments earlier on. Again, deciding when to un-commit actions is a non-trivial issue.

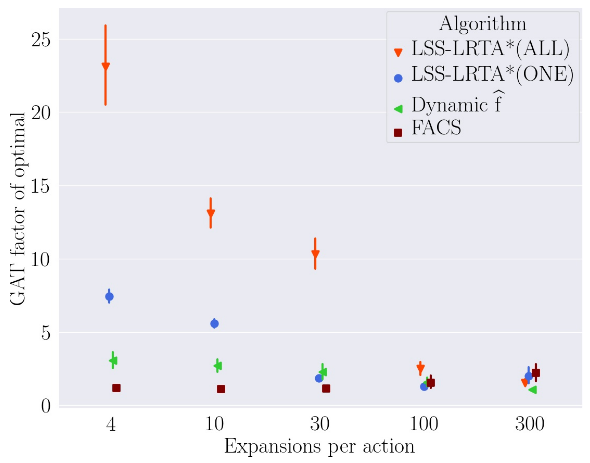


Figure 5: GAT with tar pits at both ends.

Re-examination of the set of assumptions about the search process is also needed, especially if we use a completely different search component in the online setting. Of special interest is the effect of using envelope search (Björnsson, Bulitko, and Sturtevant 2009; Gall, Cserna, and Ruml 2020) or Monte-Carlo tree search (MCTS) (Browne et al. 2012; Schulte and Keller 2014) on both the metareasoning decisions and on actual performance.

Although we had success with assuming a Gaussian form for heuristic value uncertainty, one could certainly explore using training data to define these distributions, along the lines of the work by (Fickert et al. 2020).

Last but not least are low-hanging fruit relating to additional experimentation with parameters of the scheme developed in this work. One issue is varying the frequency at which we perform metareasoning independently from the expansion rate. Is it better to perform metareasoning after each expansion (possibly more precise but a large overhead), once per real-world action, or only after the search phase finishes, as done in the empirical evaluation in this paper?

Summary

This paper introduced FACS, a basic metareasoning scheme for action commitment geared at focusing a real-time search in order to get additional time to search farther ahead in the search tree. Due to numerous assumptions and decisions needed in order to simplify the analysis that might have been done differently, this is preliminary work that has considerable room for expansion. Nevertheless, favorable empirical results in contrived grid pathfinding scenarios show that this approach has promise and could lead to a principled treatment of one of the most fundamental issues on online planning and execution.

Acknowledgements

This research was supported by grant 2019730 from the United States-Israel Binational Science Foundation (BSF) and grant 2008594 from the United States National Science Foundation (NSF).

References

Björnsson, Y.; Bulitko, V.; and Sturtevant, N. R. 2009. TBA*: Time-Bounded A*. In *Proceedings of IJCAI*, 431–436.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1): 1–43.

Cashmore, M.; Coles, A.; Cserna, B.; Karpas, E.; Magazzeni, D.; and Ruml, W. 2018. Temporal planning while the clock ticks. In *Proceedings of ICAPS*.

Cserna, B.; Ruml, W.; and Frank, J. 2017. Planning time to think: Metareasoning for on-line planning with durative actions. In *Proceedings of ICAPS*.

Fickert, M.; Gu, T.; Staut, L.; Ruml, W.; Hoffmann, J.; and Petrik, M. 2020. Beliefs We Can Believe In: Replacing Assumptions with Data in Real-Time Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9827–9834.

Gall, K. C.; Cserna, B.; and Ruml, W. 2020. Envelope-Based Approaches to Real-Time Heuristic Search. In *Proceedings of AAAI*, 2351–2358. AAAI Press.

Hernández, C.; Baier, J. A.; Uras, T.; and Koenig, S. 2012. Time-bounded adaptive A*. In *Proceedings of AAMAS*, 997–1006.

Kiesel, S.; Burns, E.; and Ruml, W. 2015. Achieving goals quickly using real-time search: experimental results in video games. *Journal of Artificial Intelligence Research*, 54: 123–158.

Koenig, S.; and Sun, X. 2008. Comparing real-time and incremental heuristic search for real-time situated agents. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(3): 313—341.

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

Nadarajah, S.; and Kotz, S. 2008. Exact Distribution of the Max/Min of Two Gaussian Random Variables. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(2): 210–212.

O'Ceallaigh, D.; and Ruml, W. 2015. Metareasoning for Concurrent Planning and Execution. *ICAPS Workshop on Planning and Robotics (PlanRob-15)*, 86.

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

Schulte, T.; and Keller, T. 2014. Balancing exploration and exploitation in classical planning. In *Proceedings of SoCS*.

Shperberg, S. S.; Coles, A.; Karpas, E.; Ruml, W.; and Shimony, S. E. 2021. Situated Temporal Planning Using Deadline-aware Metareasoning. In *Proceedings of ICAPS*, 340–348.

Thayer, J. T.; Dionne, A.; and Ruml, W. 2011. Learning Inadmissible Heuristics During Search. In *Proceedings of the Twenty-first International Conference on Automated Planning and Scheduling (ICAPS-11)*.