Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Planning Time to Think: Metareasoning for On-Line Planning with Durative Actions

Bence Cserna, Wheeler Ruml
University of New Hampshire
Durham, NH 03824 USA
{bence,ruml} @cs.unh.edu

Jeremy Frank
NASA Ames Research Center
Moffett Field, CA 94035 USA
jeremy.d.frank@nasa.gov

Abstract

When minimizing makespan during off-line planning, the fastest action sequence to reach a particular state is, by definition, preferred. When trying to reach a goal quickly in on-line planning, previous work has inherited that assumption: the faster of two paths that both reach the same state is usually considered to dominate the slower one. In this short paper, we point out that, when planning happens concurrently with execution, selecting a slower action can allow additional time for planning, leading to better plans. We present Slo'RTS, a metareasoning planning algorithm that estimates whether the expected improvement in future decision-making from this increased planning time is enough to make up for the increased duration of the selected action. Using simple benchmarks, we show that Slo'RTS can yield shorter time-to-goal than a conventional planner. This generalizes previous work on metareasoning in on-line planning and highlights the inherent uncertainty present in an on-line setting.

Introduction

Traditionally, planning has been considered from an off-line perspective, in which plan synthesis is completed before plan execution begins. In that setting, if the objective is to minimize plan makespan, then it is clearly advantageous to return a faster plan to achieve the goal. For example, in a heuristic search-based approach to planning, if the planner discovers two alternative plans for achieving the same state, it only needs to retain the faster of the two. In A* search, this corresponds to the usual practice of retaining only the copy of a duplicate state that has the lower g value.

However, many applications of planning demand an on-line approach, in which the objective is to achieve a goal as quickly as possible, and planning takes place concurrently with execution. For example, while the agent is transitioning from state s_1 to state s_2, the planner can decide on the action to execute at s_2. In this way, the agent's choice of trajectory unfolds over time during execution, rather than being completely pre-planned before execution begins. While this may result in a trajectory that is longer than an off-line optimal one, it can result in achieving the goal faster than off-line planning because the planning and execution are concurrent (Kiesel, Burns, and Ruml 2015; Cserna et al. 2016). This setting also models situations in which an agent's goals can be updated during execution, requiring on-line replanning.

The first contribution of this paper is to point out that the on-line setting differs from the off-line one in that it may be advantageous for the planner to select a slower action to execute at s_2 even when a faster one is known to reach the same resulting state s_3. This is because the longer action will give the agent more time to plan before reaching s_3. This might result in a better decision at s_3, allowing the agent to reach a goal sooner. If the decision is substantially better, the difference may even be large enough to offset the delay due to the slower action. Anyone who has slowed down while driving on a highway in order to have more time to study a map before passing a crucial exit is intuitively familiar with this scenario. We generalize this reasoning to cover actions that do not immediately lead to the same state s_3.

The second contribution of this paper is a practical on-line planning algorithm, Slo'RTS (pronounced Slow-are-tee-ess), that takes this observation into account. We work in the paradigm of forward state-space search, using real-time heuristic search algorithms that perform limited lookahead search and then use the lookahead frontier to inform action selection. When operating in a domain that has durative actions, whose execution times can be different, Slo'RTS takes actions' durations into account, estimating the effect on decision-making at future states. In this way, Slo'RTS reasons about its own behavior; in other words, it engages in metareasoning. We implement and test Slo'RTS in some simple gridworld benchmarks, finding that its metareasoning can indeed result in better agent behavior. To our knowledge, this is the first example of a planning algorithm that can dynamically plan to give itself more time to think without assuming that the world is static. More generally, this work is part of a recent resurgence of interest in metareasoning in heuristic search, illustrating how this beautiful idea can yield practical benefits.

Previous Work

We briefly review the real-time heuristic search and metareasoning algorithms that Slo'RTS is based on. Given our objective to minimize time to goal, we will assume that plan cost represents makespan.

Real-Time Search

Local Search Space-Learning Real-Time A* (LSS-LRTA*) is a leading general-purpose real-time search algorithm (Koenig and Sun 2009). Each iteration of LSS-LRTA* operates in two phases: exploration and learning. First, the exploration phase uses an A* search to expand a local search space around the agent until an expansion bound is reached, at which point the agent will commit to the action leading to the best node at the edge of the lookahead frontier. Second, in the learning phase, the heuristic values of expanded states are updated by propagating information backwards from the lookahead frontier using a Dijsktra-like propagation algorithm. The next iteration then uses these updated h values, generating a fresh lookahead search frontier.

Dynamic $\hat{f}$ is a variant of LSS-LRTA* (Kiesel, Burns, and Ruml 2015). It employs an inadmissible heuristic, notated $\hat{h}$. This value is an unbiased estimate of a node's true cost-to-goal, rather than an admissible lower bound. Just as $f(s) = g(s) + h(s)$, we will write $\hat{f} = g(s) + \hat{h}(s)$. During lookahead search, Dynamic $\hat{f}$ sorts the frontier on $\hat{f}$ instead of f, and afterwards, the action selected for execution is the one leading to the frontier node with the best $\hat{f}$ value.

While any $\hat{h}$ could be used, in experiments reported below, we use a version of the standard admissible h that is debiased online using the 'single-step path-based error' method of Thayer, Dionne, and Ruml (2011). During search, the error ϵ in the admissible h is estimated at every expansion by the difference between the f value of the parent node and the f value of its best successor (these will be the same for a perfect h). If $\bar{\epsilon}_s$ is the average error over the nodes along the path from the root to a node s and $d(s)$ is an estimate of the remaining search distance (number of edges along the path) from a state s to the nearest goal, then $\hat{h}(s) = h(s) + \bar{\epsilon}_s \cdot d(s)$.

During the learning phase, if node a inherits its f value from a node b on the search frontier, then Dynamic $\hat{f}$ will update $h(a)$ using the path cost between a and b (the difference in their g values) and b's h value. The remaining error in the estimate of $f(a)$ will then derive from the error in $h(b)$. So for each updated h value, Dynamic $\hat{f}$ also records the d value of the node it was inherited from, and thus $\hat{h}(a) = h(a) + \bar{\epsilon}_b \cdot d(b)$.

Metareasoning Search

Metareasoning On-line Real-time Search (Mo'RTS, pronounced Moe-are-tee-ess) addresses domains that have identity actions, which function as a no-op or idle action by taking time but not changing the state of the world (O'Ceallaigh and Ruml 2015). Mo'RTS uses metareasoning to decide when to execute identity actions. When no identity actions are taken, Mo'RTS works just like dynamic $\hat{f}$. When an identity action is taken, Mo'RTS can preserve its lookahead frontier from its previous search iteration and extend it, allowing deeper lookahead and more accurate decision-making. In an extreme case, Mo'RTS can decide to take so many identity actions that it is able to plan all the way to a goal, thereby imitating the behavior of off-line A* search.

The central decision of Mo'RTS is whether the delay in reaching the goal caused by the duration of an identity action $t_{identity}$ is outweighed by the expected benefit B of additional search:

$$B > t_{identity}. \tag{1}$$

Search will provide benefit if, instead of the current best action α, additional search causes the planner to select some other action β instead. Abusing notation by referring to states by the actions that lead to them, Mo'RTS represents the value of an action α as a belief distribution over possible values, with $p_\alpha(x)$ representing the probability density that α's true f^* value is x. This distribution is assumed to be a Gaussian centered at $\hat{f}(\alpha)$ with variance $\sigma^2 = (\bar{\epsilon}_b \cdot d(b))^2$, where b is the frontier node from which α inherits its value. Just as in Dynamic $\hat{f}$, Mo'RTS allows further search to decrease $d(b)$ and hence sharpen our belief about α.

Because actions are chosen based on their $\hat{f}$ values, to estimate benefit, we need to estimate what α and β's $\hat{f}$ values might be after we have performed more search. This is represented as a Gaussian belief $p'_{\bar{\alpha}}(x)$ centered at $\hat{f}(\alpha)$ with variance

$$\sigma^2_{p'_{\bar{\alpha}}} = \sigma^2_{p_\alpha} \cdot \min(1, \frac{d_s}{d(b)}) \tag{2}$$

where d_s is the distance, in search steps, along the path to a goal that we expect to cover during the extra search. The intuition is that, if no further search were done, the variance of $p'_{\bar{\alpha}}$ is zero since $\hat{f}(\alpha)$ wouldn't change, and if we searched all the way to a goal, we would expect $\hat{f}(\alpha)$ to be distributed like p_α, since that is the definition of p_α. d_s is estimated by dividing the number of expansions that will be done during the search by the expected number of expansions required to make progress along a search path (estimated by the average number of expansions from when a node is generated until it is expanded).

The benefit of search, if the value of the most promising action α were to become x_α and the value of some competing action β were to become x_β, would be

$$b(x_\alpha, x_\beta) = \begin{cases} 0 & \text{if } x_\alpha \leq x_\beta \\ x_\alpha - x_\beta & \text{otherwise} \end{cases} \tag{3}$$

because we would have done α if we had not searched. The expected benefit is the expected value over our estimates of $p'_{\bar{\alpha}}$ and $p'_{\bar{\beta}}$:

$$B = \int_{x_\alpha} p'_{\bar{\alpha}}(x_\alpha) \int_{x_\beta} p'_{\bar{\beta}}(x_\beta) b(x_\alpha, x_\beta) dx_\beta dx_\alpha. \tag{4}$$

Using these estimates, Mo'RTS can decide whether an identity action is worthwhile. However, in domains in which there is no identity action that allows the agent to stop the world and think, Mo'RTS does not apply.

Metareasoning for Durative Actions

Slo'RTS generalizes the ideas behind Mo'RTS. Instead of a special comparison of the expected benefit of search against the time cost of an identity action, we add consideration of time when computing the expected cost of every top-level

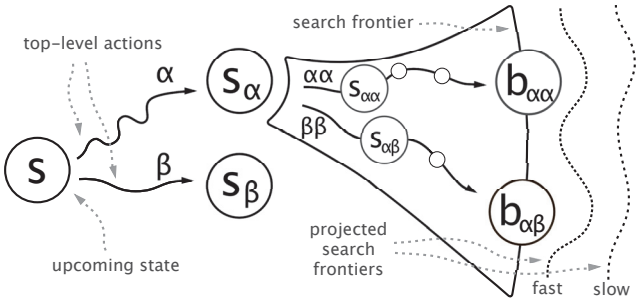


Figure 1: An example search tree considered by Slo'RTS.

action. Figure 1 illustrates a situation in which the agent is currently transitioning to state s and the planner is deciding whether to take action α or β once the agent arrives there. Ordinarily, an action like α would receive a backed up value equal to the minimum of its children, $s_{\alpha\alpha}$ and $s_{\alpha\beta}$, whose values are inherited from frontier nodes $b_{\alpha\alpha}$ and $b_{\alpha\beta}$. But this assumes that we have perfect information about these values and that no further information will be gained. In a real-time search context, this assumption does not hold; there may be insufficient time to fully explore the space, especially early on.

Slo'RTS recognizes that, if we choose to do α at s, by the time we actually reach s_α, we will have refined our beliefs about $s_{\alpha\alpha}$ and $s_{\alpha\beta}$, depending on how long α takes to execute. This leaves open the possibility that, by then, $\alpha\beta$ might be more attractive than $\alpha\alpha$. So when estimating the value of s_α, Slo'RTS computes the expected minimum over what it thinks its beliefs about $\hat{f}(s_{\alpha\alpha})$ and $\hat{f}(s_{\alpha\beta})$ might be after searching during α. In this way, when comparing s_α and s_β, the durations of α and β are taken into account.

More formally, for every child b of every top-level action a, Slo'RTS models its belief about $f^*(s_{ab})$ as a distribution as in Mo'RTS

$$p_{ab} \sim \mathcal{N}(\hat{f}(s_{ab}), (\bar{\epsilon}_{b_{ab}} \cdot d(b_{ab}))^2) \tag{5}$$

and its belief about the location of $\hat{f}(s_{ab})$ after search as

$$p'_{\bar{ab}} \sim \mathcal{N}(\hat{f}(s_{ab}), (\bar{\epsilon}_{b_{ab}} \cdot d(b_{ab}))^2 \cdot \min(1, \frac{d_s}{d(b_{ab})}) \tag{6}$$

Note that d_s will vary depending on the duration of a, causing the belief to be a spike at $\hat{f}(s_{ab})$ if a is very fast and to spread out to mimic p_{ab} with full search to the estimated goal depth. This means that deeper search implies more possibility for the estimate of $\hat{f}(s_{ab})$ to change and for ab to perhaps change its ranking with respect to other actions, possibly becoming the best action at s_a.

To estimate the value of a top-level successor state s_a, we take the expected minimum over the estimates of the beliefs about the children of s_a after planning during a. If the children were α and β, this would be

$$E_a = \int_{x_{a\alpha}} \int_{x_{a\beta}} p'_{\bar{a\alpha}}(x_{a\alpha}) p'_{\bar{a\beta}}(x_{a\beta}) \min(x_{a\alpha}, x_{a\beta}) dx_{a\beta} dx_{a\alpha}. \tag{7}$$

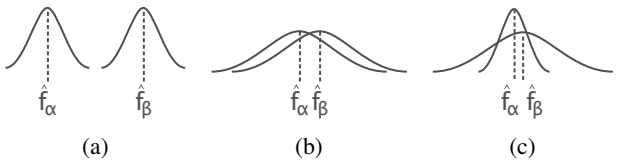


Figure 2: Three scenarios with beliefs about α and β.

This may be lower than $\min(\hat{f}(s_{a\alpha}), \hat{f}(s_{a\beta}))$ if it appears that search might change the values significantly. In the implementation tested below, we integrate numerically, and we limit our consideration to the best two actions under each top-level action. After every search iteration, Slo'RTS chooses the top-level action that has the smallest expected value.

Evaluation

We now turn to a brief analysis of the behavior of Slo'RTS in comparison to conventional on-line planners.

Theorem 1 *Conventional real-time search algorithms always choose the shortest sequence of actions towards any expanded state within the lookahead search space, assuming the heuristic is consistent.*

Proof: Conventional algorithms such as LSS-LRTA* use A* for their lookahead phase. When A* generates a node representing the same state as a previously-generated node, only the one with the lowest g value is retained. This reflects the dynamic programming optimal substructure property that subpaths of shortest paths are themselves shortest paths. Furthermore, if the heuristic is consistent, the g value of any node expanded by A* (and thus within the lookahead space) is optimal. Thus, real-time searches select only shortest paths to expanded nodes in the lookahead space. □

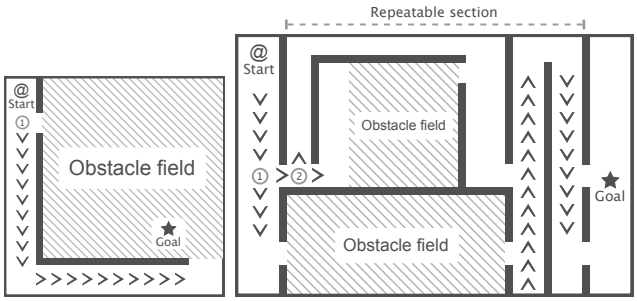
Corollary 1 *When two sequences of actions lead to the same state, conventional real-time search algorithms will always take the faster path over the slower.*

This illustrates the uniqueness of Slo'RTS, which has the ability to select a slower path when it deems it beneficial.

Theorem 2 *If the belief distributions of Slo'RTS are accurate, the algorithm makes the optimal rational decision.*

Proof: The heart of Slo'RTS, equation 7, estimates expected cost. Under the definition of rationality as maximizing expected utility, which, given the objective of Slo'RTS corresponds to minimizing cost, this is the optimal rational decision given the state of knowledge of the planner. □

The added time complexity of Slo'RTS over LSS-LRTA* is $O(b)$ where b is the branching factor. At the end of every planning iteration, when the algorithm makes a decision, it calculates the expected value of taking each top-level action using equation 7. If, as in our implementation, this is limited to two actions per top-level action, then it is constant time for each top-level action. Another approach would be to compare each applicable action against the best, leading to $O(b)$ time per top-level action and $O(b^2)$ overall.



(a) Simple (b) Highway

Figure 3: Schematics of the grid benchmarks.

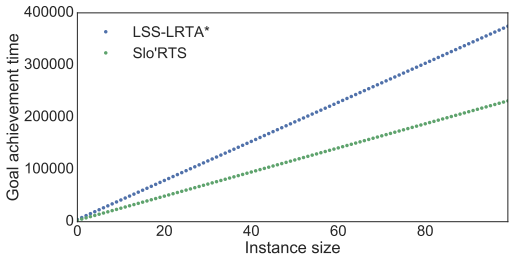


Figure 4: Planner performance on highway instances.

To gain a concrete sense of Slo'RTS' behavior, we first tested the core decision procedure on three classic but completely synthetic scenarios, illustrated in Figure 2. Each scenario involved two top-level actions, one fast and one slow, both leading to the same state at which two actions, α and β, were applicable. In scenario (a), we defined costs in the search space such that Slo'RTS was quite confident about the values of α and β. Additional search was not predicted to have a significant chance of causing β to appear better than α, and Slo'RTS correctly chose the fast action. In scenario (b), the beliefs about α and β were very uncertain, but so similar that it didn't matter much which was chosen. Slo'RTS recognized this and chose fast. In scenario (c), the belief about α is quite certain but β is uncertain and has a chance of leading to a significantly better outcome than α. In this case, Slo'RTS correctly chose the slow action to allow additional search.

We then implemented a Slo'RTS agent in a grid pathfinding domain using the Manhattan distance heuristic and compared it with LSS-LRTA*. In each cardinal direction, both fast and slow actions were available. Figure 3a shows a simple instance with an important early decision marked by 1. The agent has to decide between crossing the obstacle field that looks slightly worse from the outside or going around the L shaped obstacle. The optimal path goes through the obstacle field. Due to the limited lookahead, conventional real-time planners like LSS-LRTA* always take the better looking path and go around the obstacle. Slo'RTS recognizes that the path leading into the obstacle field has potentially better cost and slows down before the decision point. The slow down results in a lookahead that is sufficient to find a better path through the obstacle field.

Lastly, we implemented a class of benchmark problems featuring grid cells with expensive irreversible decisions, shown in Figure 3b. This simulates the example of deciding which exit to take on a highway. The middle section of the map is repeatable, yielding a sequence of decisions. Each time the agents passes through this section, it passes one or two decision points surrounded with irreversible cells, thus the agent cannot turn back to choose another path. Conventional algorithms like LSS-LRTA* always choose fast actions and are thus highly dependent on their lookahead parameter. Slo'RTS correctly identifies the decision points and slows down before reaching them, allowing better decision-making. Since Slo'RTS plans while traveling to the goal, node expansions are a perfect measure of time-to-goal; Figure 4 shows the total time-to-goal (in number of node expansions) as the number of middle segments is increased. Clearly, in these instances the benefits of slowing down to make a better decision outweigh increased execution time.

Discussion

There are several promising ways in which Slo'RTS could be extended. For example, it currently considers only the effects of time on the decisions at the successors of the current state, but this could in principle apply to any future anticipated states of the agent. We also believe that it should be possible to derive the ideal amount of time to spend planning, if the domain allows flexibility in action durations.

There has been extensive prior work on deliberation scheduling, in which decisions must be made about when to plan. Most of this work separates the planner from a meta-reasoning executive that initiate replanning or policy improvement meta-actions (Dean et al. 1993; Musliner, Goldman, and Krebsbach 2003; Krebsbach 2009). Some work computes the value of deliberation off-line (Wu and Durfee 2006). In contrast, Slo'RTS embeds metareasoning in the planner itself, where it has on-line access to detailed information regarding the benefit of planning. Our objective of minimum time-to-goal allows Slo'RTS to directly compare time spent planning to estimated improvement in plan cost.

The decision faced by a multi-armed bandit algorithm is similar to that faced by Slo'RTS, in that it involves a trade-off between exploring to gather more information versus exploiting the estimates that the agent has already gathered. Metareasoning itself has been formulated as an MDP (Lin et al. 2015). Work on control of anytime algorithms is also related, in that one must dynamically decide whether to continue planning (Hansen and Zilberstein 2001; Póczos et al. 2009).

There are many real-time search algorithms and it would be interesting to adapt Slo'RTS to them. daRTAA* attempts to avoid heuristic depressions which cause revisiting the same state multiple times (Hernández and Baier 2012). EDA* provably avoids revisiting states many times, but is limited to undirected domains in which every action can be immediately and exactly undone and is nontrivial to use with lookahead (Sharon, Felner, and Sturtevant 2014).

Conclusion

This paper has noted an issue unique to on-line planning, proposed a principled algorithm for addressing it, and shown that the algorithm can provide benefit in simple benchmarks. While not every domain will see benefit from metareasoning about planning time, it is useful to recognize that planning can concern more than which actions to do, or even whether to think more, but also how to provide the time to do so.

Acknowledgements

We gratefully acknowledge support from NSF (grant 1150068). This work was inspired by discussions with Dylan Hadfield-Menell, starting in the King's College London cafeteria during ICAPS 2016.

References

Cserna, B.; Bogochow, M.; Chambers, S.; Tremblay, M.; Katt, S.; and Ruml, W. 2016. Anytime versus real-time heuristic search for on-line planning. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, 131–132.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1993. Deliberation scheduling for time critical decision making. In *Proceedings of UAI-93*.

Hansen, E. A., and Zilberstein, S. 2001. Monitoring and control of anytime algorithms: a dynamic approach. *Artificial Intelligence* 126:139–157.

Hernández, C., and Baier, J. A. 2012. Avoiding and escaping depressions in real-time heuristic search. *Journal of Artificial Intelligence Research* 43:523–570.

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. 2009. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems* 18(3):313–341.

Krebsbach, K. D. 2009. Deliberation scheduling using gsmdps in stochastic asynchronous domains. *International Journal of Approximate Reasoning* 50(9):1347–1359.

Lin, C. H.; Kolobov, A.; Kamar, E.; and Horvitz, E. 2015. Metareasoning for planning under uncertainty. In *Proceedings of IJCAI-15*.

Musliner, D. J.; Goldman, R. P.; and Krebsbach, K. D. 2003. Deliberation scheduling strategies for adaptive mission planning in real-time environments. In *Proceedings of the Third International Workshop on Self Adaptive Software*.

O'Ceallaigh, D., and Ruml, W. 2015. Metareasoning in real-time heuristic search. In *Proceedings of the Symposium on Combinatorial Search (SoCS-15)*.

Póczos, B.; Abbasi-Yadkori, Y.; Szepesvári, C.; Greiner, R.; and Sturtevant, N. 2009. Learning when to stop thinking and do something! In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML-09)*, 825–832.

Sharon, G.; Felner, A.; and Sturtevant, N. R. 2014. Exponential deepening A* for real-time agent-centered search. In *Proceedings of AAAI-14*.

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)*.

Wu, J., and Durfee, E. H. 2006. Mathematical programming for deliberation scheduling in time-limited domains. In *Proceedings of AAMAS-06*, 874–881.