Metareasoning for Interleaved Planning and Execution

Amihay Elboher, Shahaf S. Shperberg, Solomon E. Shimony
Dept. of Computer Science, Ben-Gurion University
Beer-Sheva, Israel
{amihaye,shperbsh}@post.bgu.ac.il, shimony@cs.bgu.ac.il

Abstract

Agents that plan and act in the real world must deal with the fact that time passes as they are planning. In the presence of tight deadlines, there may be insufficient time to complete the search for a plan before it is time to act. One can gain additional time to search by starting to act before a complete plan is found, incurring the risk of making incorrect action choices. This tradeoff between opportunity and risk, inherent in interleaving planning and execution, is a non-trivial metareasoning problem addressed in this paper.

1 Introduction and Background

The idea of starting to perform actions in the real world (called base-level actions) before completing the search goes back as far as real-time A* (Korf 1990). However, our setting is more flexible, as we do not have a predefined time at which actions must be done. Rather, the agent must reason about when base-level actions should be executed in order to maximize the probability of successful and timely execution. We assume in this work that the world is deterministic, the only uncertainty is at the meta-level, due to uncertainty about how long planning will take and about the time executing the (unknown at search time) resulting plan will take.

This work defines the above tradeoffs as a formal metareasoning problem of decision-making under uncertainty, which we then attempt to solve optimally in the sense of (Russell and Wefald 1991). Doing so for an actual planning or search algorithm is far too complicated. Instead we use an abstract model handling such issues called *Allocating Effort when Actions Expire* $(AE)^2$ (Shperberg et al. 2019), and extend it to allow for action execution before plan completion.

2 Metareasoning Problem Definition

In traditional planning, the plan is fully generated before its first action is executed. In situated temporal planning (Cashmore et al. 2018), actions have deadlines (that can be represented by Timed Initial Literals (TIL) (Cresswell and Coddington 2003; Edelkamp and Hoffmann 2004)) which induce a (possibly unknown) deadline by which planning must conclude. For a partial plan available at a search node i in the planner, a random variable d_i models the unknown deadline by which a potential plan expanded from node i must be generated. This induces a metareasoning problem of deciding which nodes on the open list to expand in order to maximize the chance of finding a plan before its deadline.

The $(AE)^2$ model and its simplified, discrete-time version, $S(AE)^2$ abstract away from the planning problem and assume n independent processes, each attempting to solve the problem under time constraints. We extend $(AE)^2$ to allow base-level action execution in parallel with the search.

As in (Shperberg et al. 2019), we assume a known probabilistic performance profile (Zilberstein and Russell 1996) M_i for each process i, where $M_i(t)$ is the probability of process i completing its computation within processing time t. Each process also has a distribution D_i over deadlines by which time the process must complete its computation.

Our extension of $(AE)^2$, called interleaving planning and execution when actions expire (IPAE for short), assumes each process i has already computed a base-level action sequence H_i, a prefix of the actions in the potential solution being computed by process i. Actions from any action sequence H_i may be executed even before having a complete plan. However, we assume that actions are irreversible, thus any process where the already executed action sequence are not a prefix of its H_i becomes invalid. Upon termination, process i delivers the rest of the plan action sequence, β_i. A valid (timely) execution is one where: (a) process i terminates before starting execution of β_i, and (b) β_i is executed to completion before the overall deadline.

Given distributions of the unknown overall deadline and the length of β_i, it suffices to consider only the time by which process i must terminate and all actions in H_i must be executed: the *induced deadline*. In IPAE we use the random variable D_i to denote the induced deadline distribution.

IPAE is thus defined as follows. Given n processes, each with a (possibly empty) action sequence $H_i \subseteq B$, the set of base-level actions with known durations; a performance profile M_i, and induced deadline distribution D_i, find a policy for allocating computation time to the n processes and legally executing base-level actions from some H_i, such that the probability of executing a timely solution is maximal.

3 Special Case: Known Induced Deadlines

Even in the case of known deadlines, $S(AE)^2$ was shown to be NP-hard (Shperberg et al. 2019) (proven by reduction

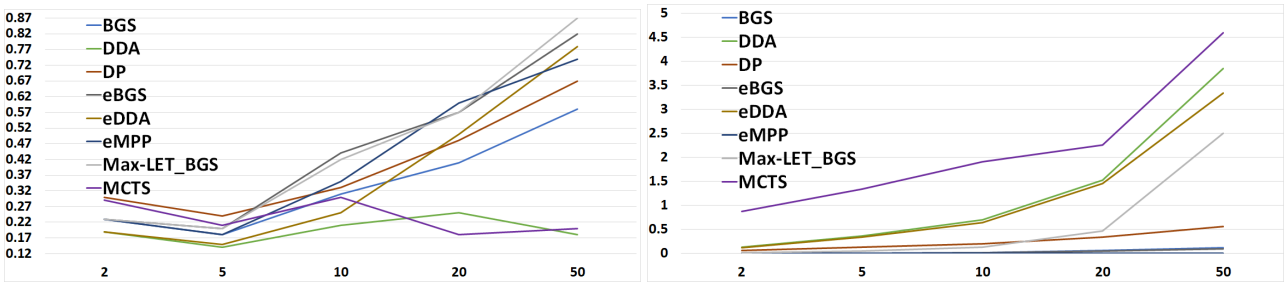

Figure 1: Success Probability (left) and Runtime (right) as a function of number of processes

from the optimization version of knapsack (Garey and Johnson 1979, problem MP9)). This result carries over into IPAE, because any S(AE)^2 problem can be cast as an IPAE problem with all action prefixes H_i being empty. Still, it was shown (Shperberg et al. 2019) that for known deadlines d_i it is sufficient to examine linear contiguous allocation policies sorted by a non-decreasing order of deadlines.

Note that a linear contiguous policy is one where each process i is allocated all its computation time contiguously, and process $i+1$ starts only if process i fails to terminate. Such a policy is fully defined by assigning each process i a start time s_i and a computation duration length l_i. We use the notation of logarithm probability of failure to terminate: $LPF_i(t) = log_2(1 - M_i(t))$. Overall we need to find (s_i, l_i) pairs so as to minimize: $LPF = \sum_{i=1}^{n} LPF_i(l_i)$, subject to the constraint (for all i): $s_i + l_i \leq d_i$, and no overlap, i.e. no j such that $s_i < s_j < s_i + l_i$. The ordering enabled a pseudo-polynomial time dynamic programming (DP) scheme for such S(AE)^2 instances (Shperberg et al. 2021):

Theorem 1. *For known deadlines, DP according to*

$$OPT(t,l) = \max_{0 \leq j \leq d_l - t} (OPT(t+j, l+1) - LPF_l(j)) \qquad (1)$$

finds optimal S(AE)^2 schedules in time polynomial in n, d_n.

For IPAE with known deadlines (IPAEK) linear optimal policies also suffice. But for a DP scheme to work we also need a consistent ordering, and there we falter since in IPAE the ordering depends on the times at which base-level actions are executed. Attempting to alleviate this problem, we define a lazy policy as one where execution of base-level actions is delayed as long as possible. We proved that:

Theorem 2. *In IPAEK there exists an optimal policy that is linear, contiguous, and lazy.*

3.1 Bounded Length Prefixes

Fixing the time when base-level actions are executed we get:

Theorem 3. *Among the set of linear contiguous policies for a specific H_i and with given action execution times, there exists an optimal policy where the processes are allocated in order of non-decreasing effective deadlines.*

The number of such possible settings is exponential in the maximum size of the H_i prefixes. Thus, assuming that this length is bounded by a constant K, we get a pseudo-polynomial time algorithm, by using the DP for $S(AE)^2$ with appropriate effective process deadlines.

3.2 The Equal Slack Case

Denote $SL_i = d_i - dur(H_i)$, the *slack* of process i. If all the SL_i are equal, then for each H_i sequence it is sufficient to consider the actions in H_i to be executed contiguously, starting at SL_i. Now the effective deadline d_i^{eff} of process j equals the time at which the first action $b \in H_i$ which is incompatible with H_j occurs, or d_i otherwise. So need to run the DP only n times, regardless of the length of the H_i.

4 General Algorithms and Empirical Results

IPAE instances were constructed by running A* on 15-puzzle instances, with process i defined for each node i in the open list. H_i is the path to node i, and M_i, D_i based on statistics over heuristics from solving $10,000$ instances.

The following algorithms were examined:

S(AE)^2 algorithms: the *basic greedy scheme* (BGS) (Shperberg et al. 2019), *delay-damage aware* (DDA), and *dynamic programming* (DP) from (Shperberg et al. 2021).

IPAE *demand-execution* adaptation to S(AE)^2 algorithms: eBGS, eDDA. When the S(AE)^2 algorithm allocates time to process i, but a base-level action b is required to make the allocation non-tardy, then execute b first. We have also tested *most promising process* (eMPP), that allocates time to the process most likely to succeed, adapted as above.

A Monte-Carlo tree search (MCTS) algorithm (Browne et al. 2012) using UCT (Kocsis and Szepesvári 2006), based on UCB1 (Auer, Cesa-Bianchi, and Fischer 2002).

Max-LET$_{BGS}$: For each process i fix the actions in H_i at the **L**atest **E**xecution-**T**ime w.r.t the minimal value in the support of D_i. Using the induced effective deadlines we now have a S(AE)^2 instance, solved using BGS.

Preliminary results (Figure 1) indicate that *demand-execution* versions of S(AE)^2 algorithms significantly outperform their basic versions. MCTS demonstrates poor performance. MAX-LET$_{\text{BGS}}$ seems promising. Additional experiments and adaptation to actual online search and planning are the next steps on our research agenda.

Acknowledgements

Supported by BSF Grant #2019730, NSF Grant #2008594, ISF grant #844/17, and the Frankel center for CS at BGU.

References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2): 235–256.

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 *ICAPS*, 39–46. AAAI Press.

Cresswell, S.; and Coddington, A. 2003. Planning with Timed Literals and Deadlines. In *Proceedings of 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, 23–35.

Edelkamp, S.; and Hoffmann, J. 2004. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. Technical Report 195, University of Freiburg.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability, A Guide to the Theory of NP-completeness*, 190. W. H. Freeman and Co.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *ECML*, volume 4212 of *Lecture Notes in Computer Science*, 282–293. Springer.

Korf, R. E. 1990. Real-Time Heuristic Search. *Artif. Intell.* 42(2-3): 189–211.

Russell, S. J.; and Wefald, E. 1991. Principles of Metareasoning. *Artif. Intell.* 49(1-3): 361–395.

Shperberg, S. S.; Coles, A.; Cserna, B.; Karpas, E.; Ruml, W.; and Shimony, S. E. 2019. Allocating Planning Effort When Actions Expire. In *AAAI*, 2371–2378. AAAI Press.

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

Zilberstein, S.; and Russell, S. J. 1996. Optimal Composition of Real-Time Systems. *Artif. Intell.* 82(1-2): 181–213.