Metareasoning for Interleaved Planning and Execution

Amihay Elboher, Shahaf S. Shperberg, Solomon E. Shimony and Wheeler Ruml
Dept. of Computer Science, Ben-Gurion University Beer-Sheva, Israel, University of New Hampshire, USA

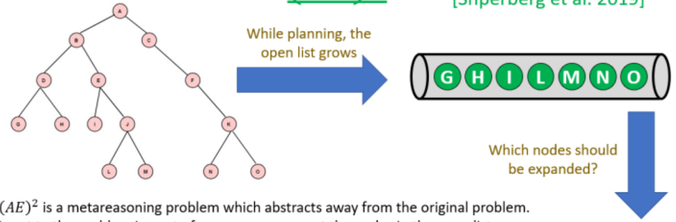
1 Situated Temporal Planning

The action of “taking the train of 9:15” expires at 9:15.
Therefore, if your plan contains this train travel, you should finish the planning such that you will be in the station before 9:15.

In situated temporal planning, the goal is to find a plan such that when finishing the planning process, the plan is still executable.

2 $(AE)^2$

[Shperberg et al. 2019]



The $(AE)^2$ is a metareasoning problem which abstracts away from the original problem.
The input to the problem is a set of processes represent the nodes in the open list, where each process has:

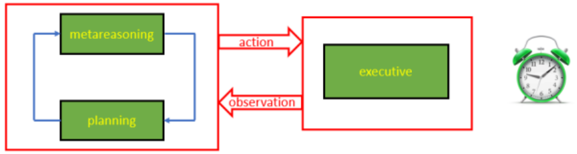
1. Deadline random variable;
2. Time allocation needed for this process to terminate computation.

Both are random variables from a known distribution.

Optional output:
Allocate 10 time units to process H, then 4 to N, and at last 6 to L.

The goal is to find a time allocation for the processes in order to maximize the probability of finding an executable plan.

3 $IPAE$ – Interleave Planning And Execution

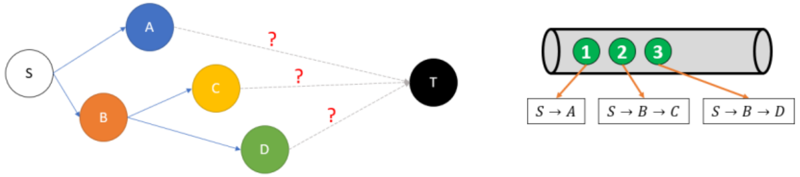


The $IPAE$ problem generalizes $(AE)^2$ by letting actions to be executed even before a complete plan was found.

Any process in $IPAE$ has, in addition to the deadline and allocation time random variables, a partial plan that was already been executed. The partial plan of each process corresponds to the sequence of expansions that created the node this process represents.

The goal is to find time allocation between the processes, actions to execute and their execution times.

4 Example: finding a route



You want to find a route from some location S to another location T, and you have already computed 3 partial plans: 1. $S \to A$; 2. $S \to B \to C$; 3. $S \to B \to D$.
These are your options:
1. Stay is S to gather more information, taking the risk of not finding a complete plan on time;
2. Committing to the action $S \to A$, which saves the time of the travel from S to A but invalidates partial plans 2-3.
3. Committing to the action $S \to B$, which has a symmetrical effect.

5 Reductive and Adaptive Approaches

Reduction

Determine a sequence of actions to executes and set the execution time for each action.

For each *action a that was set to time t*, each process p that does not contain a as the next action in its partial plan, is invalidated at the latest on time t.
To simulate the invalidation by the execution of a, we change the deadline random variable of p such that the accumulated probability on time t will be 1.

Any other process is not invalidated; hence we do not change its deadline.

This constitutes a reduction from $IPAE$ to $(AE)^2$, since the whole information can be derived from the deadlines and allocation time random variables.
Note again that it assumes determining of *action-time* sequence.

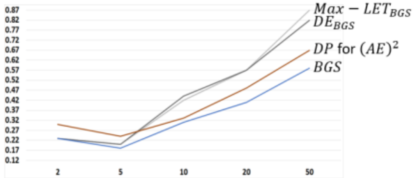
Adaptation

Consider an $(AE)^2$ algorithm α.
This algorithm α tells us which process p should be allocated computation time unit next.

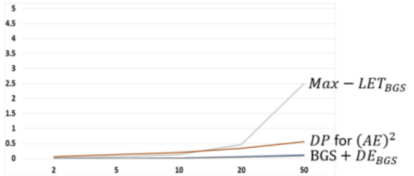
To adopt α to $IPAE$, we check before the allocation whether p demands execution of some action a.
If this is the case, we first committing to a, and just then allocate the computation time unit to p.

We denote this adaptation by DE_α.

6 Preliminary Empirical Results



Probability of success as a function of #processes



Runtime as a function of #processes

The $Max - LET_{BGS}$ algorithm is based on the reductive approach. For each partial plan $H_i = (a_{i_1} \ldots a_{i_k})$ it set the time of each action a_{i_j} to the *Latest-Execution-Time* possible, and choosing the sequence that maximizes the probability of success when using basic greedy scheme (BGS) [Shperberg et al. 2019] as the $(AE)^2$ algorithm.

The DP is a dynamic programming algorithm for $(AE)^2$ introduced by [Shperberg et al. 2021]. The DP and BGS were not adopted to $IPAE$.

The $Max - LET_{BGS}$ and DE_{BGS} algorithms seem more promising then DP and BGS, though the results are somewhat unstable and require further examination.

It is always a good idea to *think*, but sometimes it is time to **act**!

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