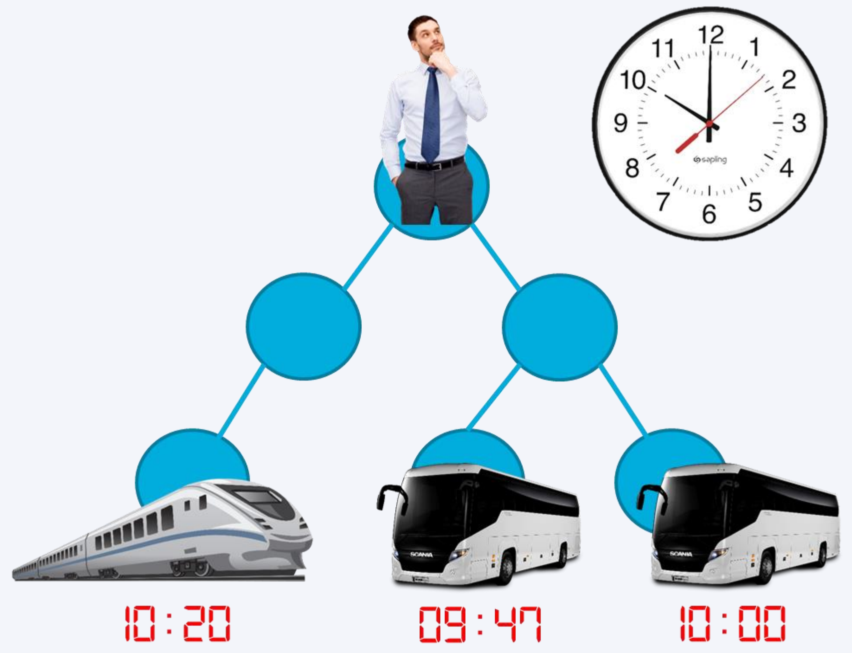
Situated Temporal Planning Using Deadline-aware Metareasoning

Shahaf S. Shperberg[1], Andrew Coles[2], Erez Karpas[3], Wheeler Ruml[4], and Solomon Eyal Shimony[1]

[1]Ben-Gurion University, Israel; [2]King's College London, UK; [3]Technion, Israel; [3]University of New Hampshire, USA

1 Situated Temporal Planning [Cashmore et al. 2018]



Example: planning a route:

- 'take 10:00 bus' action expires at 10:00 subtree of plans becomes invalid; consider only if sufficient time to complete plan
- exploring 'take 9:47 bus' action can invalidate 10:00 action; searching under multiple nodes means less time for each
- a plan expiration time and cost are uncertain until the plan is complete but completion effort also uncertain

which plans to explore?

2 Allocating Effort when Actions Expire (AE2) [Shperberg et al. 2019]

n partial plans/nodes/processes to share CPU time. Given for each process i:

- **effort CDF:** $M_i(t)$ = probability that i requires CPU time ≤ t
- **success probability:** P_i = probability that i finds a solution (without considering time found)
- **deadline CDF:** $D_i(t)$ = probability that i expires before t (clock wall time). Not certain until solution is complete

Find a **schedule** for the processes that **maximizes probability** of finding a solution that is **still valid** when found.

AE2 is NP-Hard. The $(AE)^2$ can be modeled as an MDP, but the **state space is exponential in n.**

Known deadlines:

Linear contiguous policies (LCP): (1, 1, 1, 3, 3, 2, 2, 2, ...) No feedback and allocation for all process are contiguously

With known deadlines, there exists an LCP that is an optimal solution.

3 Known deadlines: pseudo-polynomial algorithm

- $LPF_i(t_0, t_u)$: the log probability of failure for the process to find a timely solution within t_u processing time units, starting at time t_0
- -LPF is equivalent to utility

DP algorithm:

1. Sort processes in a non-decreasing order of deadlines
2. Compute the optimal utility of scheduling processes **l** thought **n** starting from time **t**:

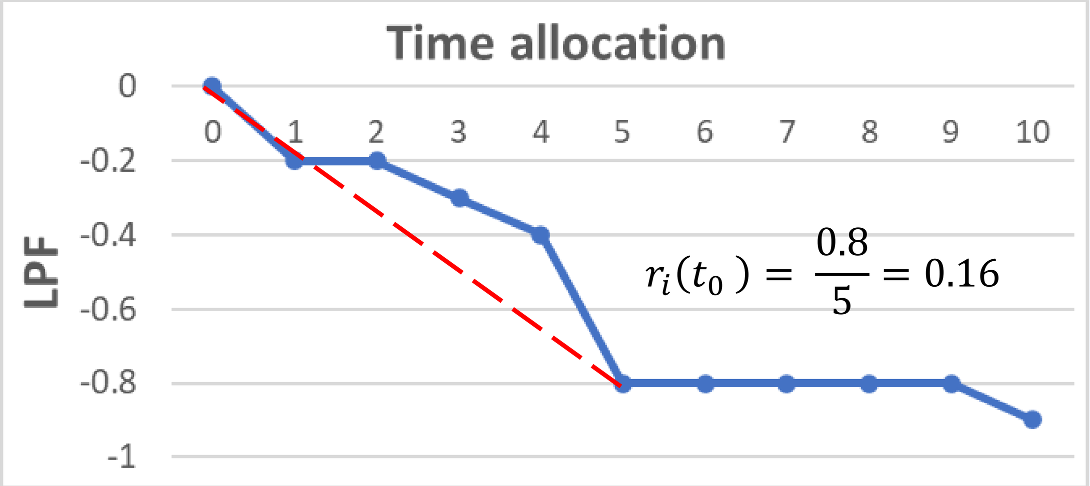
$$OPT(t,l) = \max_{0 \le j \le d_l - t} \left(OPT(t+j, l+1) - LPF_l(t,j) \right)$$

3. Return $OPT(0,1)$ as the utility of the optimal policy.

- runs in time polynomial in **n** and $\max_i d_i$
- Usable when a solution needs to be found before a known (common to all processes) timeout, e.g., Algorithm Portfolio

4 Unknown deadlines: greedy schemes

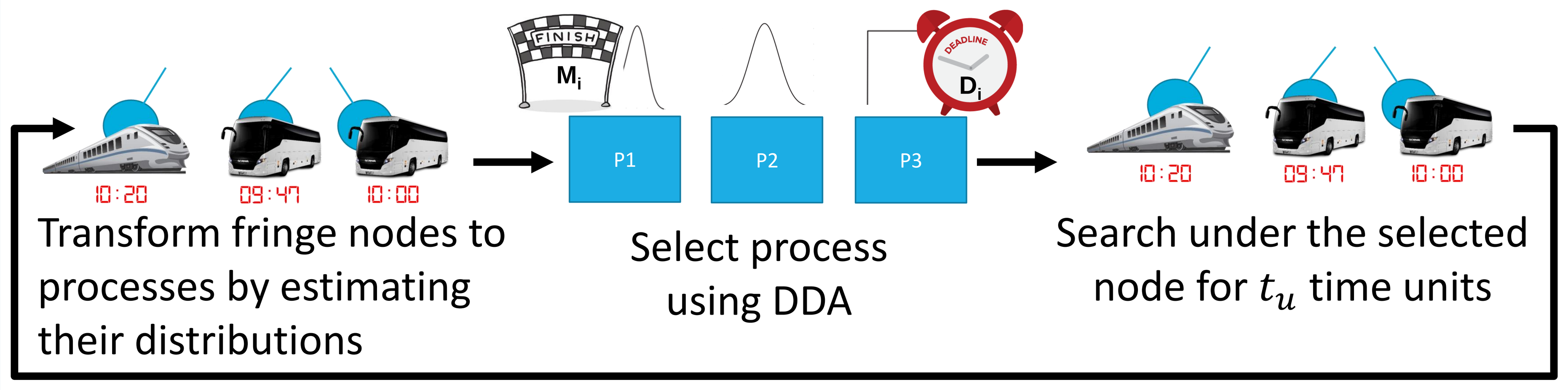
- $r_i(t_0)$: the best rate of improvement per time unit (slope) for process i:



- **P-Greedy** [Shperberg et al. 2019]: Allocate t_u time units to the process that maximizes: $Q_i(t_0) = r_i(t_0) + \frac{\alpha}{E(D_i)}$
- **Delay-Damage Aware (DDA)**: Allocate t_u time units to the process that maximizes: $Q_i(t_0) = r_i(t_0) - \gamma \cdot r_i(t_0 + t_u)$

DDA uses a more methodological way to decide which process is damaged the most by delay

5 Search -> Metareasoning -> Search



Estimating the distributions:

- OPTIC uses a temporal relaxed planning graph (TRPG) (Coles et al. 2010) to estimate **distance-to-go** $d(s)$ and **latest start time** $lst(s)$ for every state.
- $d(s)$ is used for estimating distribution of remaining search time (M_i)
- $lst(s)$ is used for estimating distribution for deadlines (D_i)

6 Empirical Evaluation - results

Domain	baseline		DDA		DDA (dom tuned)	
airport	19.0	(19–19)	20.0	(20–20)	20.5	(19–21)
pw-nt	4.0	(3–4)	4.0	(3–5)	3.9	(3–5)
rcll 1	37.7	(37–40)	73.7	(53–92)	83.9	(59–99)
rcll 2	1.0	(1–1)	4.0	(2–23)	2.7	(0–13)
sat cmplx	5.0	(5–5)	5.0	(5–5)	3.8	(2–5)
sat tw	5.0	(5–5)	5.0	(5–5)	3.6	(3–5)
trucks	6.0	(6–6)	6.9	(6–9)	5.7	(5–8)
turtlebot	14.0	(14–14)	12.5	(10–13)	13.0	(13–13)
umts-flaw	4.1	(4–5)	5.1	(5–6)	5.0	(5–5)
umts	48.0	(48–48)	45.5	(42–49)	45.7	(44–49)
TOTAL	143.8	(142–147)	181.7	(151–227)	187.7	(153–223)

7 Conclusion

Considering planning time is **hard**, but ignoring it is not the solution!

Supported by BSF Grant No. 2019730, NSF Grant No. 2008594, ISF grant #844/17, and by the Frankel center for CS at BGU. This Project is also funded by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 821988 (ADE).