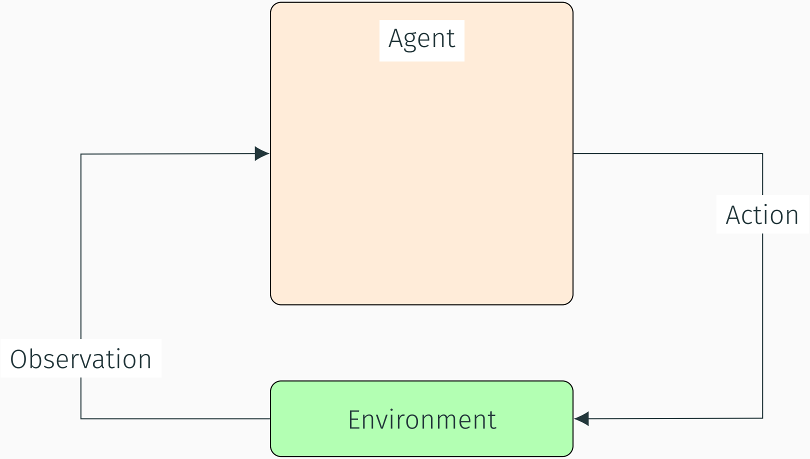
Planning and Acting While the Clock Ticks

Andrew Coles,[1] Erez Karpas,[2] Andrey Lavrinenko,[3]
Wheeler Ruml,[4] Solomon Eyal Shimony,[3] Shahaf Shperberg[3]

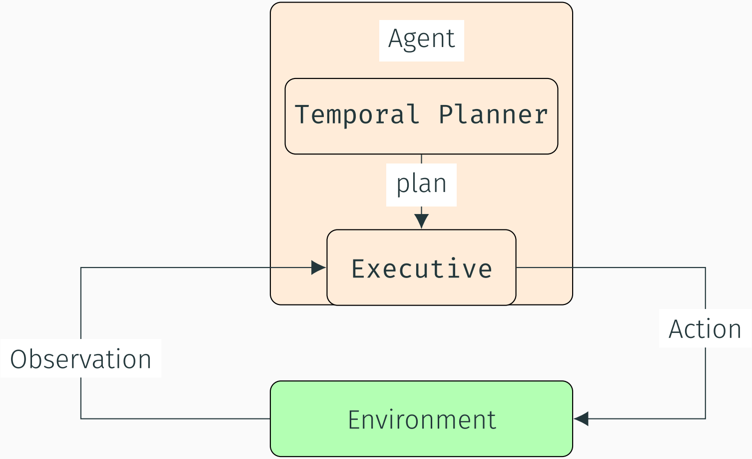
[1]King's College London
[2]Technion
[3]Ben-Gurion University
[4]University of New Hampshire

Motivation

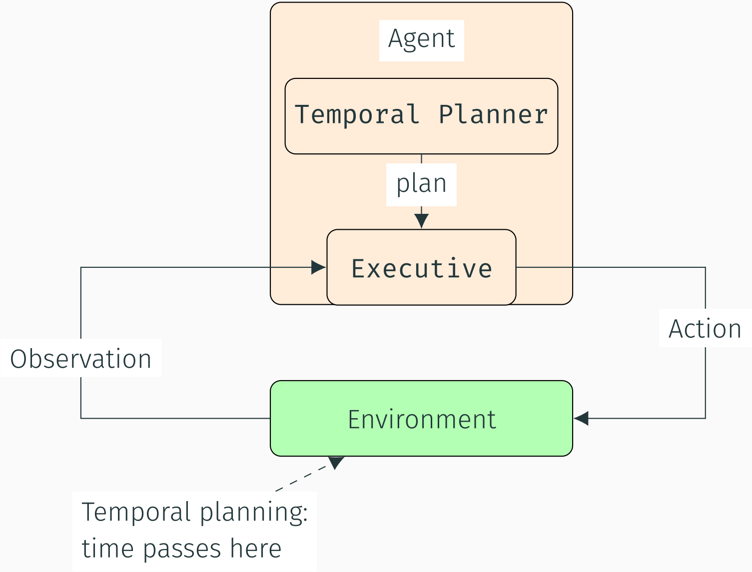
Planning



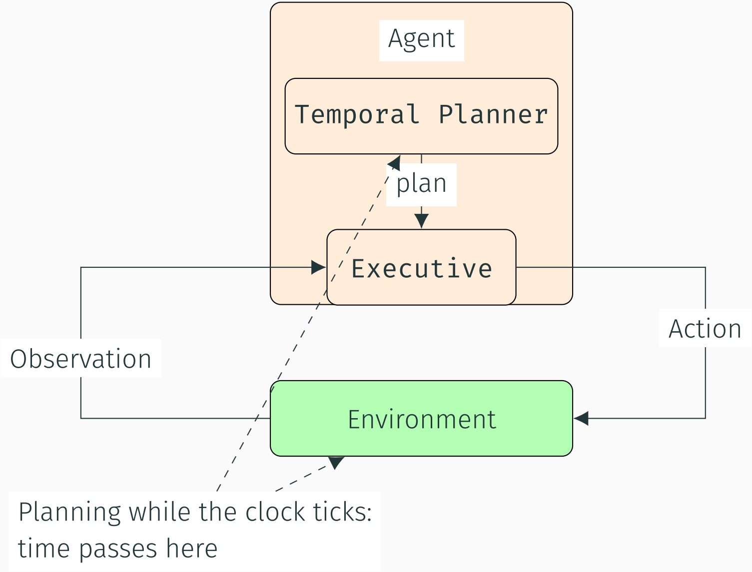
Temporal Planning



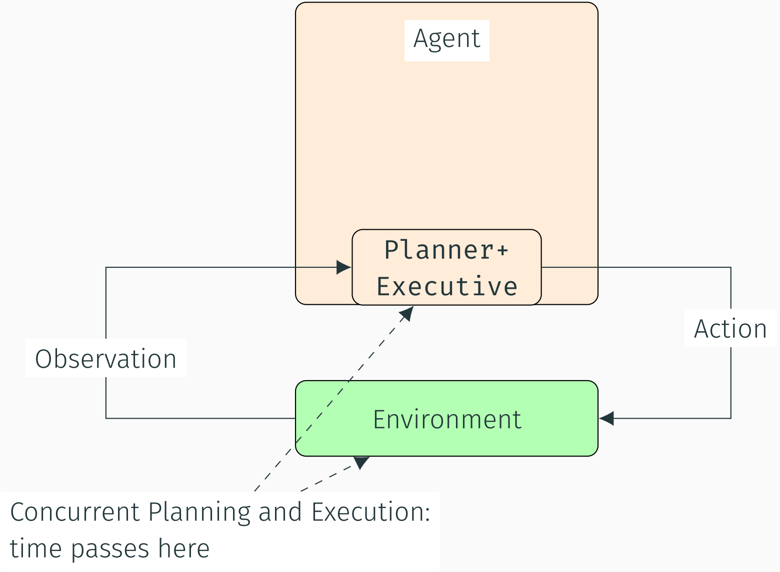
Temporal Planning



Temporal Planning while the Clock Ticks



Temporal Planning while the Clock Ticks



Why does this matter? Temporal Constraints!

- If we do not have any external temporal constraints (e.g., deadlines), then planning time will never affect plan feasibility
- But what if we have 10 seconds to achieve our goal?
  - 10 seconds from planning start time
  - Then if we take t seconds to plan, the plan's makespan must be less than $10 - t$ seconds

Action Commitment

- In situated temporal planning, the planner must complete planning before the first action is executed
- This guarantees that any plan that is returned will be correct
- It does not guarantee that we will reach our goal on time

Action Commitment

- In situated temporal planning, the planner must complete planning before the first action is executed
- This guarantees that any plan that is returned will be correct
- It does not guarantee that we will reach our goal on time

- Example: consider an autonomous vehicle planning a long drive, when a large truck starts backing up towards it

Action Commitment

- In situated temporal planning, the planner must complete planning before the first action is executed
- This guarantees that any plan that is returned will be correct
- It does not guarantee that we will reach our goal on time

- Example: consider an autonomous vehicle planning a long drive, when a large truck starts backing up towards it
- Maybe the vehicle should start driving forward, even if it does not have a complete plan

Problem Statement

Formal Problem Statement

- A planning problem is given by a tuple $\langle F, A, I, G \rangle$
  - F is a set of Boolean facts that describe the state of the world
  - A is a set of durative actions
  - $I \subseteq F$ is the initial state
  - $G \subseteq F$ is the goal
- A plan π is a set of tuples $\langle a, t, d \rangle$, where:
  - $a \in A$ is an action
  - $t \in \mathbb{R}^{0+}$ is its start time
  - $d \in \mathbb{R}^{0+}$ is its duration
- A plan π is valid if applying each action at its designated start time achieves the goal G

Formal Problem Statement

- A situated planning problem is given by a tuple $\langle F, A, I, G \rangle$
  - F is a set of Boolean facts that describe the state of the world
  - A is a set of durative actions
  - $I \subseteq F$ is the initial state
  - $G \subseteq F$ is the goal
- A plan π is a set of tuples $\langle a, t, d \rangle$, where:
  - $a \in A$ is an action
  - $t \in \mathbb{R}^{0+}$ is its start time
  - $d \in \mathbb{R}^{0+}$ is its duration
- A plan π is valid if applying each action at its designated start time achieves the goal G
  and if planning took x time, then $t \geq x$ for every $\langle a, t, d \rangle \in \pi$

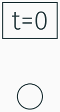
Formal Problem Statement

- A concurrent planning and execution problem is given by a tuple $\langle F, A, I, G \rangle$
  - F is a set of Boolean facts that describe the state of the world
  - A is a set of durative actions
  - $I \subseteq F$ is the initial state
  - $G \subseteq F$ is the goal
- A plan π is a set of tuples $\langle a, t, d \rangle$, where:
  - $a \in A$ is an action
  - $t \in \mathbb{R}^{0+}$ is its start time
  - $d \in \mathbb{R}^{0+}$ is its duration
- A plan π is valid if applying each action at its designated start time achieves the goal G

  and if $\langle a, t, d \rangle \in \pi$ is output at time x then $t \geq x$

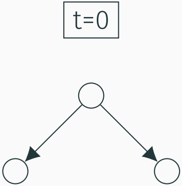
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans



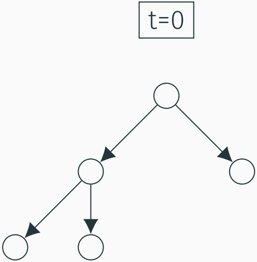
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans

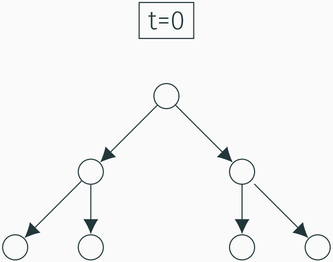


Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans

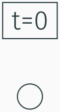


- Planning can be solved by searching a tree of possible plans



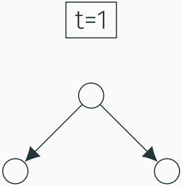
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search



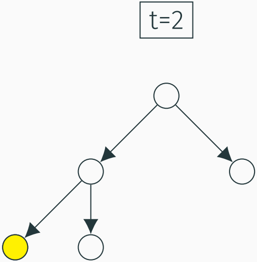
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search



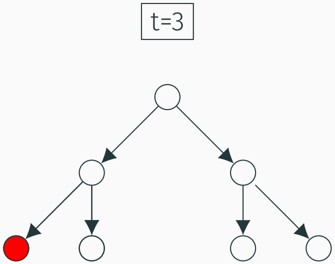
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search



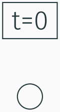
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search



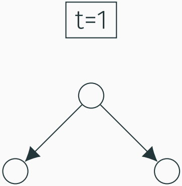
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search
- In concurrent planning and execution, can commit to an action and prune the rest of the tree



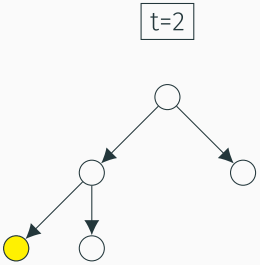
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search
- In concurrent planning and execution, can commit to an action and prune the rest of the tree



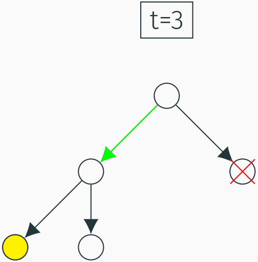
Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search
- In concurrent planning and execution, can commit to an action and prune the rest of the tree



Searching While the Clock Ticks

- Planning can be solved by searching a tree of possible plans
- In situated planning, nodes can expire during search
- In concurrent planning and execution, can commit to an action and prune the rest of the tree



Concurrent Planning and Execution

Components for Concurrent Planning and Execution

- Search
- Heuristic
- Temporal Reasoning
- Metareasoning

Components for Concurrent Planning and Execution

- Search
  - Same search space as POPF – search over sequences of start/end of durative actions
- Heuristic
- Temporal Reasoning
- Metareasoning

Components for Concurrent Planning and Execution

- Search
    - Same search space as POPF – search over sequences of start/end of durative actions
- Heuristic
    - Temporal Relaxed Planning Graph
- Temporal Reasoning
- Metareasoning

Components for Concurrent Planning and Execution

- Search
    - Same search space as POPF – search over sequences of start/end of durative actions
- Heuristic
    - Temporal Relaxed Planning Graph
- Temporal Reasoning
    - Modified version of the Simple Temporal Network of POPF to account for when planning started and the current time
- Metareasoning

Components for Concurrent Planning and Execution

- Search
  - Same search space as POPF – search over sequences of start/end of durative actions
- Heuristic
  - Temporal Relaxed Planning Graph
- Temporal Reasoning
  - Modified version of the Simple Temporal Network of POPF to account for when planning started and the current time
- Metareasoning
  - The focus of the rest of this talk

Metareasoning for Concurrent Planning and Execution

The Metareasoning Problem

- Metareasoning deals with choosing computational actions to optimize some objective
    - Maximizing the probability of timely goal achievement
- The meta-level problem can be described as a POMDP with
    - State: the state of the search tree
    - Actions:
        - Expand state in the search tree (also in situated planning)
        - Execute action (only in concurrent planning and execution)
    - Is harder to solve than the original problem

Practical Metareasoning

- For situated planning we developed a greedy decision rule called Delay-Damage Aware (DDA)
- DDA is based on two distributions for each search node:
  - D_i – distribution on deadline
  - M_i – distribution on remaining search time
- Distributions are estimated based on observations collected during search

- Also developed an abstract metareasoning model called CoPE which uses DDA for abstract concurrent planning and execution

Practical? Metareasoning

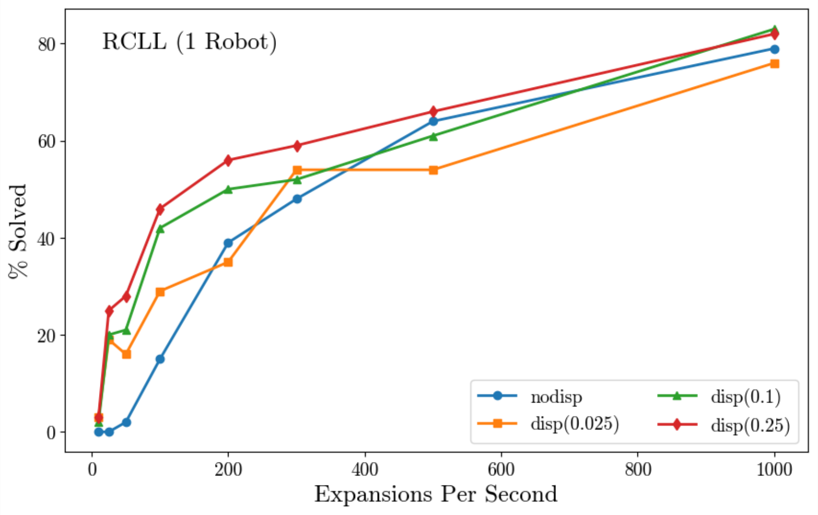
- For situated planning we developed a greedy decision rule called Delay-Damage Aware (DDA)
- DDA is based on two distributions for each search node:
  - D_i – distribution on deadline
  - M_i – distribution on remaining search time
- Distributions are estimated based on observations collected during search

- Also developed an abstract metareasoning model called CoPE which uses DDA for abstract concurrent planning and execution
- Plugging this into the planner resulted in terrible performance

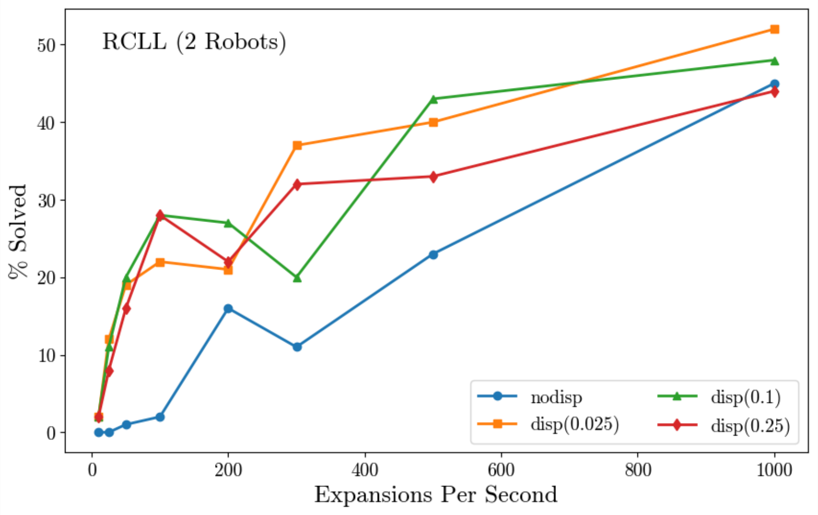
Metareasoning for CoPE: Measurements

- The reason our CoPE metareasoning did badly is because it assumed the estimated of M_i and D_i were accurate
- It then committed to executing an action, which is an irreversible decision
    - Unlike expanding a node, which only wastes a little time
- Our solution here: introducing measurements
    - We must take into account the fact that our distribution estimates D_i and M_i are inaccurate
    - We have the option to expand nodes to gain more information (probing)
        - Rough idea: if an action looks like it should be executed now, but we did not expand enough nodes under it, focus search in the subtree rooted at that action

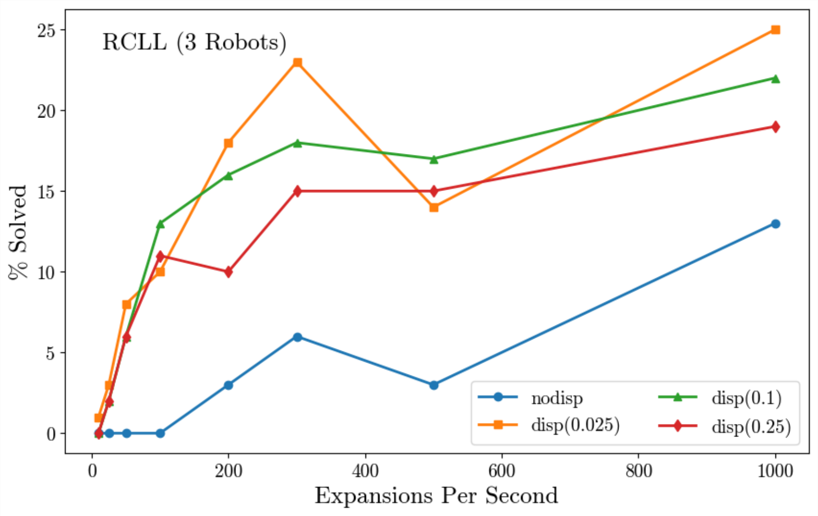
Empirical Results - RCLL 1



Empirical Results - RCLL 2



Empirical Results - RCLL 3



Thank You

"Time flies like an arrow; fruit flies like a banana." (Anthony Oettinger)