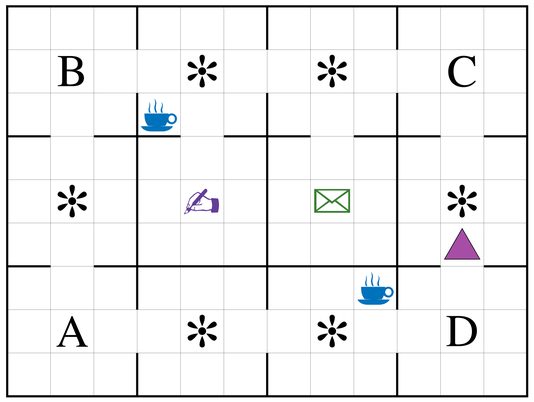
Symbolic Plans as High-Level Instructions for Reinforcement Learning

León Illanes Xi Yan Rodrigo Toro Icarte Sheila A. McIlraith

Department of Computer Science, University of Toronto Vector Institute

Motivation:

Tell an RL agent to solve a specific task.



Symbol	Meaning
▲	Robot
✻	Furniture
☕	Coffee machine
✉	Mail room
✍	Office
A, B, C, D	Marked locations

Task examples:

T1. Deliver mail to the office
T2. Deliver coffee and mail to the office
T3. Visit A, B, C, and D (in any order)

Q: How do we specify a new task?

A: ~~An expert designs a reward function.~~ We specify a goal for an abstract model.

Classical planning:

- Discrete state space
  - Represented by a set of Boolean state properties
- Finite set of deterministic actions
  - Represented by preconditions and effects w.r.t. state properties

In the example:

Propositions: `have-mail/coffee`, `delivered-mail/coffee`, `visited-A/B/C/D`

Actions: `get-mail/coffee`, `deliver-mail/coffee`, `visit-A/B/C/D`

`get-coffee`:	`deliver-coffee`:
pre: (none)	**pre**: have-coffee
eff: have-coffee	**eff**: delivered-coffee, **not** have-coffee

T1. { `delivered-mail` }
T2. { `delivered-mail`, `delivered-coffee` }
T3. { `visited-A`, `visited-B`, `visited-C`, `visited-D` }

We can use plans to guide the RL agent!

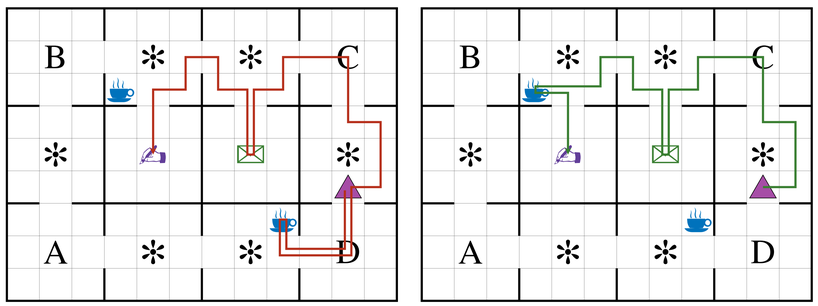
- Plans are sequences of high-level actions
- Modern planners use efficient state-space search algorithms

In the example:

T1. ⟨`get-coffee`, `deliver-coffee`⟩
T2. ⟨`get-coffee`, `get-mail`, `deliver-coffee`, `deliver-mail`⟩
T3. ⟨`go-to-A`, `go-to-B`, `go-to-C`, `go-to-D`⟩

- We need to learn low-level policies for the actions in the plan
- Then, simply execute them one after the other
- This is **hierarchical RL** with a fixed high-level policy
  - Every action maps to an **option**

Blindly following the plans is suboptimal...



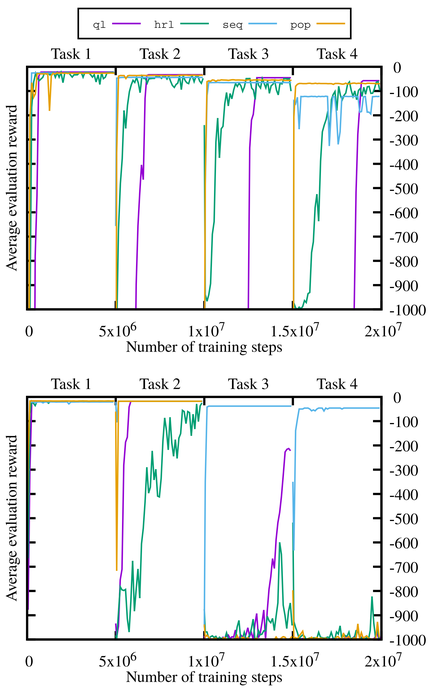
- There are multiple possible plans for the task
- We cannot know which is better until we look at the low-level
- Can we use multiple plans?

Relax the ordering constraints!

- Use **partial-order plans** (POP) instead of sequential ones:
  - A set of actions and a partial-order
  - Well established in the literature
  - We can convert a sequential plan into a POP
- The high-level policy is no longer fixed, but still very restricted
  - The agent now also has to learn how to order these actions

Experiments

We ran experiments in two different environments. In each, we train agents to solve four tasks in increasing order of complexity. The figures below show the rewards obtained by standard q-learning (`ql`), hierarchical RL using the options framework with the options identified in the high-level models (`hrl`), and our approaches using sequential (`seq`) and partial-order (`pop`) plans. Our approaches find good solutions much faster than the alternatives, and effectively transfer learned information from one task to the next.



In additional experiments, we evaluate execution monitoring techniques that may help with incorrect models: