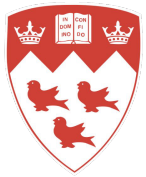


McGill UNIVERSITY

Blog on HRL

Hierarchical Reinforcement Learning

Ray Luo

WITH SPECIAL THANKS TO

Doina Precup, Xujie Si, Martin Klissarov, Akhil Bagaria, George Konidaris, Marlos C. Machado, Tianwei Ni, Xutong Zhao, Nishanth Anand Vermgal, Zijing Wu, Zihan Wang, Yivan Zhang, Isabeau Premont-schwarz, David Abel, Padideh Nouri, Jonathan Concalo Carr

What is “Hierarchical” RL?

- An arrangement of **items** that are represented as being "above", "below", or "at the same **level** as" one another. (Wikipedia)
- What are **items** in RL?
  - **Actions/policy** (control; temporal):
    - managers propose high-level goals, workers take concrete low-level acts
  - States (perception; spacial):
    - manager knows less details than workers
- How to decide “above” / “below”?
  - Determined by **abstraction levels**
  - The amount of details ignored

Blog on HRL

Temporal abstractions (option/skill/subgoal)

Desiderata of HRL

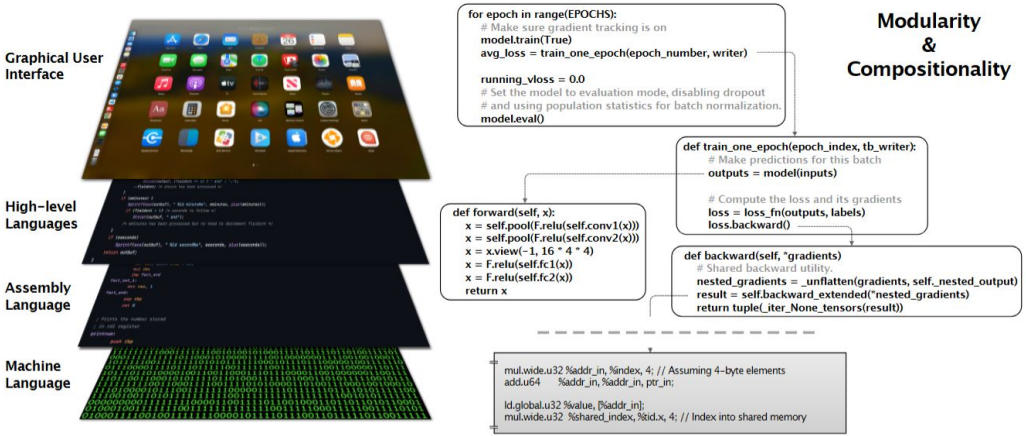
- HRL exploits temporal **structure** that organize knowledge across time
  - **Procedural** knowledge (policies, skills / **goal**-driven behavior)
  - **Predictive** knowledge (value functions / models)

Building knowledge within RL

HRL exploits temporal **structure** that organize knowledge across time

Knowledge should be:

- Expressive: can represent many things (objects, goals, high-level strategies)
- Learnable: from streaming experience, ideally without supervision
- Composable: fast planning with assembling existing pieces
- Abstract: managing complexity and omit useless details



Summary: **Why** HRL?

- Intelligence (animals) reasons and acts across extended timescales.
- Growing knowledge in open-ended environments.

➢ May manage those knowledge with temporal abstractions!

- **Standard RL**: states -> primitive actions
  - Making a decision at **every timestep**
- **Hierarchical RL**: formalizing the idea of flexibly reasoning over different timescales at multiple levels of temporal abstraction

How and When does temporal abstraction help in RL?

And…
What behaviors/skills/options/structures/abstractions do we desire?

What can good temporal abstractions achieve?

The duel question: what are the benefits of useful temporal abstractions?

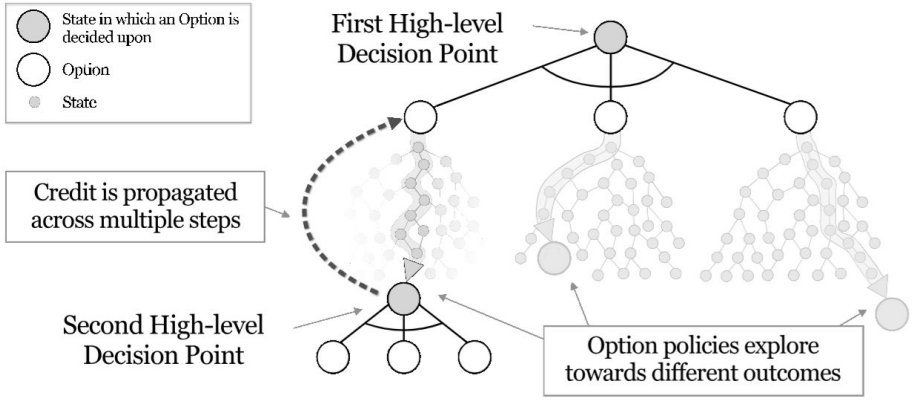
With good/useful abstraction, we can achieve better:

- Exploration
- Credit Assignment
- Transferability (Generalizability)
  - Interpretability

} Faster, sample efficient learning and planning

Structured Exploration

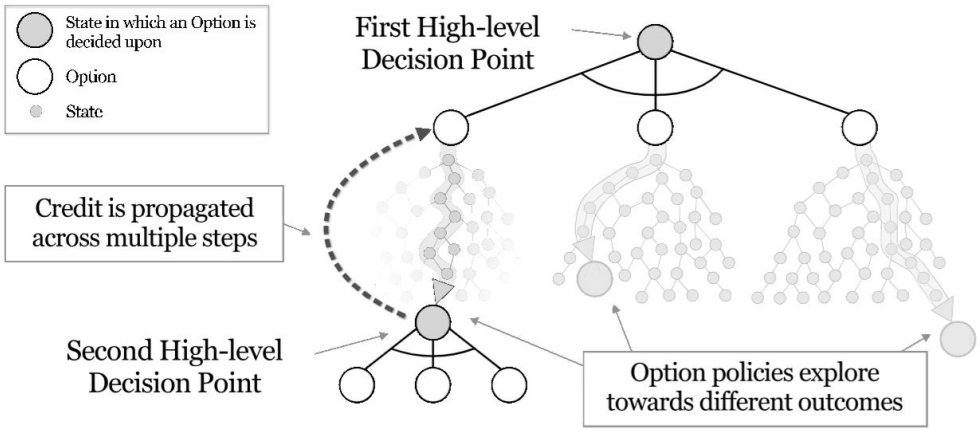
- Explore for pursuing subgoals, in different meaningful directions
- Explore within more abstract state/action spaces
- Explore over extended time, not just single action

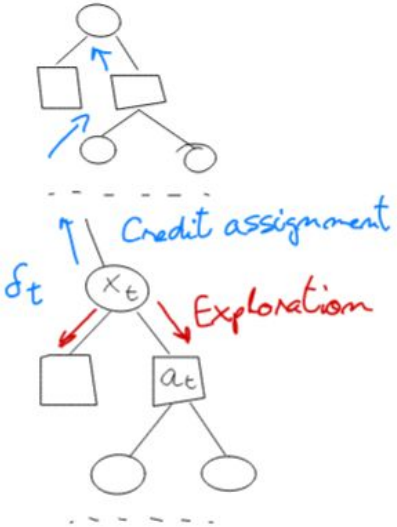


← Back

Effective Credit Assignment

- ➢ *CA problem*: determining which actions (or decisions) in a trajectory are responsible for later rewards
- Allow learning signals (TD-errors/gradients) to be assigned at higher levels of abstraction
- Identify “critical” decisions (bottlenecks, subgoals)

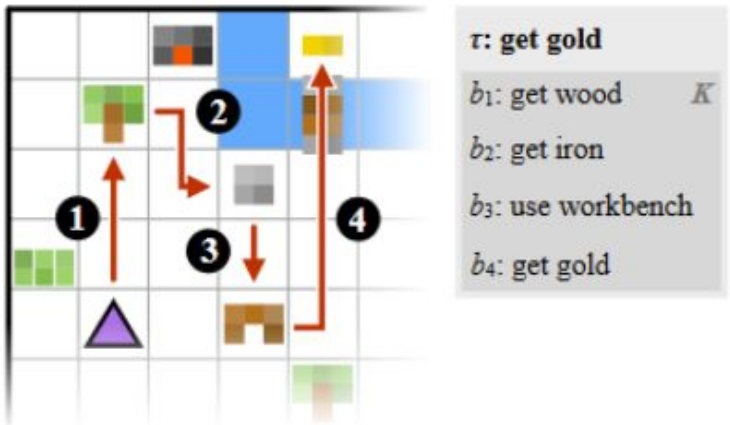




← Back

Transferability and Generalizability

- Options can be reused across different tasks
- Options can generalize in similar but novel conditions
  - Similar tasks (goals) require similar/same options to complete



← Back

Interpretability

- HRL provides an **interface** via a more abstract formulation
- More abstract representations are (sometimes) easier for humans to understand
  - As the system is smaller
  - Mind over-abstraction!

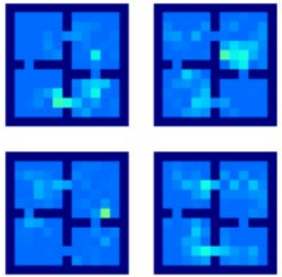
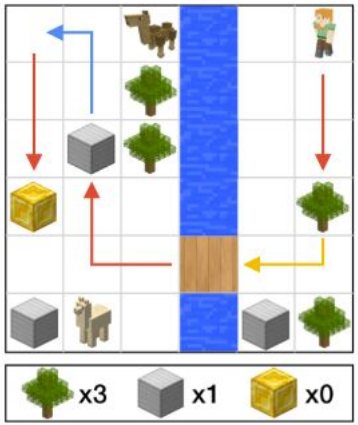


Figure 3: Termination probabilities for the option-critic agent learning with 4 options. The darkest color represents the *walls* in the environment while lighter colors encode higher termination probabilities.

Program

```
def run():
  if is_there[River]:
    mine(Wood)
    build_bridge()
    if agent[Iron]<3:
      mine(Iron)
    place(Iron, 1, 1)
  else:
    goto(4, 2)
  while env[Gold]>0:
    mine(Gold)
```

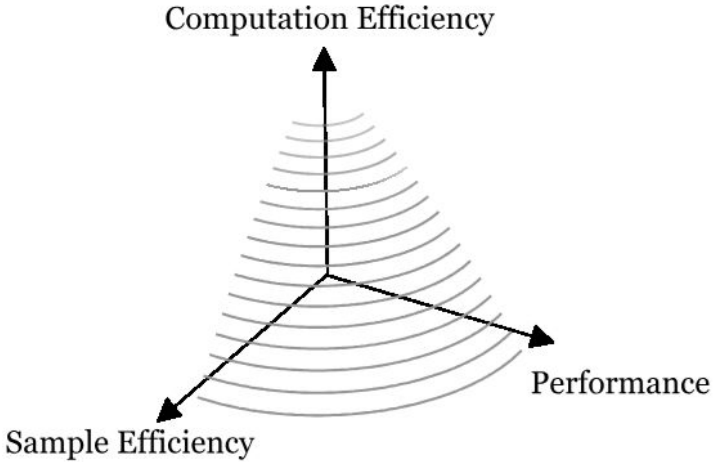


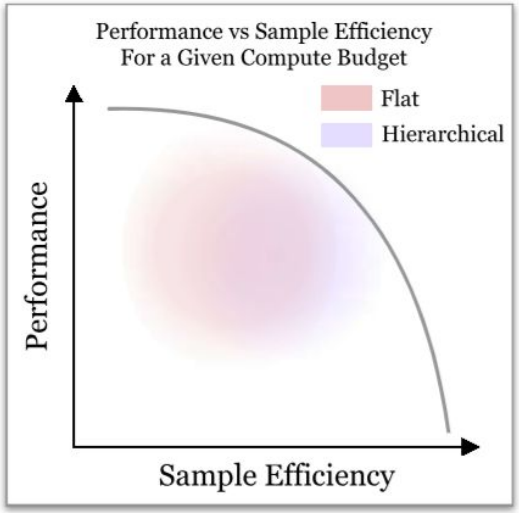
Program as a (high-level) policy

← Back

HRL aims for desirable trade-offs

- No free lunch! (Wolpert and Macready, 1997)
- Flat RL can also achieve optimal policies
  - But it may take forever!
- HRL agents face a trade-off between **performance, sample efficiency, and computation efficiency**
- Another hidden axis: Sometimes, some existing knowledge is easy to get (offline dataset, foundation models, …)





Formalism

- Option, SMDP
- Alternative Terminologies
- Option Discovery Problem
- Scope of HRL

Formalize the term “knowledge” and “temporal abstraction/structure”

Option framework, and friends

$\pi : \mathcal{S} \times \mathcal{O} \to \Delta(\mathcal{A})$ **Intra**-option policy

defines how the agent acts given state *s* under option *o*

$\beta : \mathcal{S} \times \mathcal{O} \to [0, 1]$ Termination function

probability of ending option *o* upon reaching state *s*

$\mathcal{I} : \mathcal{S} \times \mathcal{O} \to [0, 1]$ Initiation function (set)

indicates where an option *o* can start, “affordances”

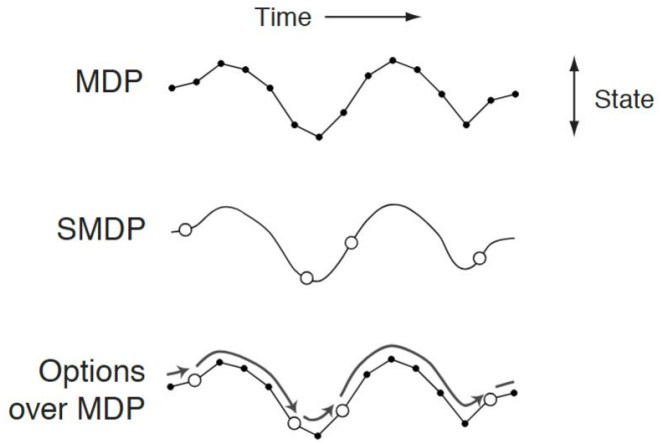
$r^o : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$

Option reward function

$\mu : \mathcal{S} \to \Delta(\mathcal{O} \cup \mathcal{A})$

High-level policy

Decision-making with Options



MDP vs. Semi-MDP (Sutton, et al., 1998)

Per-timestep vs. Multiple-timestep decision

$P(s' \mid s, a)$ vs. $P(s', \tau \mid s, o)$

$R(s, a)$ vs. $R(s, a, \tau)$

r.v. of # timestep induced by an option