



McGill
UNIVERSITY



Blog on HRL

Hierarchical Reinforcement Learning

Ray Luo

WITH SPECIAL THANKS TO

*Doina Precup, Xujiu Si, Martin Klissarov, Akhil Bagaria, George Konidaris, Marlos C. Machado, Tianwei Ni,
Xutong Zhao, Nishanth Anand Vermgal, Zijiang 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

HIGH-LEVEL GOAL: Gather Nuts for Winter



SUB-GOAL 1:

Climb Tree



Grip Bark



Grip Bark Move Branch Ascend

SUB-GOAL 2:

Collect Acorns



Identify Nut



Move Paws Pick Up Nut Put in Pouch

SUB-GOAL 3:

Store in Drey



Find Drey



Find Drey Enter Nut Deposit Nut

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)

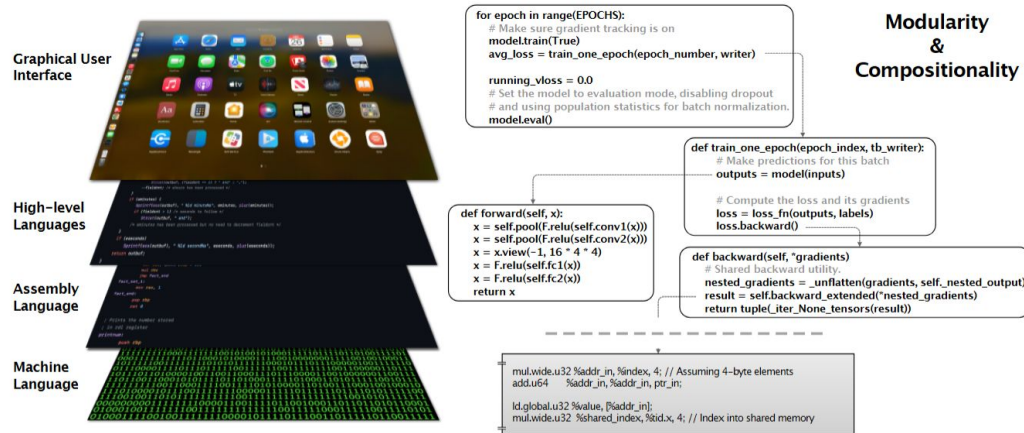


Building knowledge within RL

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?

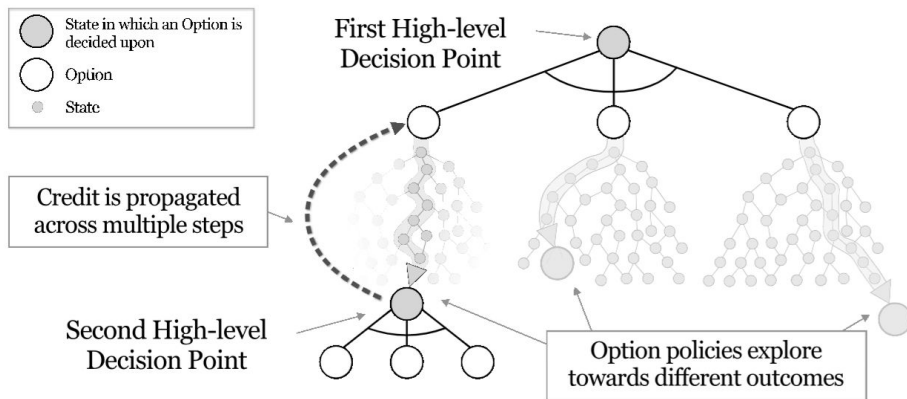
The dual question: what are the **benefits** of useful temporal abstractions?

With good/useful abstraction, we can achieve better:

- Exploration
 - Credit Assignment
 - Transferability (Generalizability)
 - Interpretability
- } Faster, sample efficient learning and planning

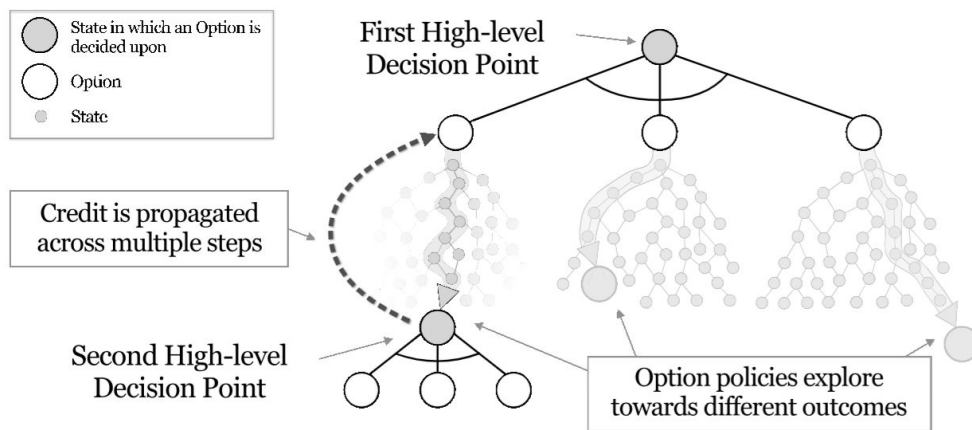
Structured Exploration

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



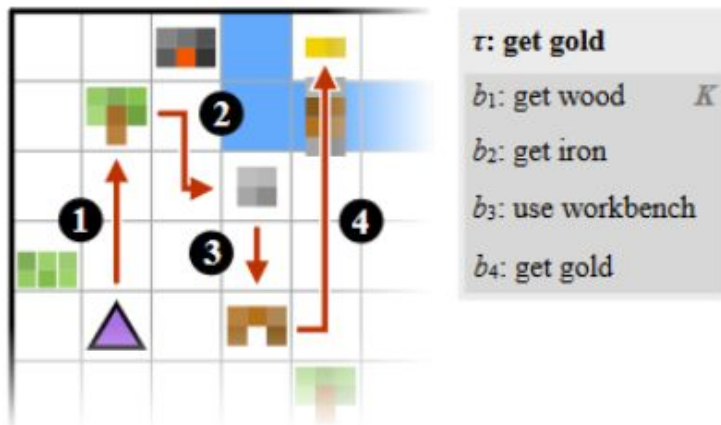
Effective Credit Assignment

- *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, and a flexible timescale
 - Identify “critical” decisions (bottlenecks, subgoals)



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



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!

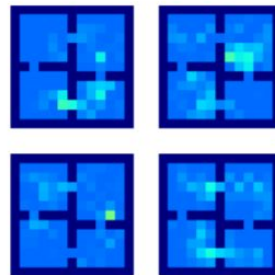
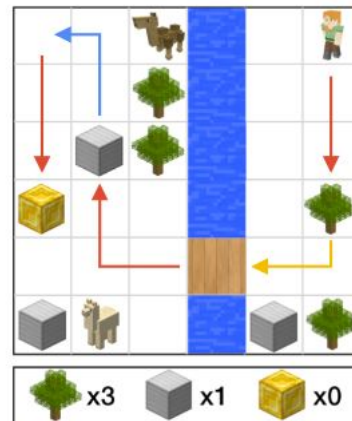


Figure 3: Termination probabilities for the option-critic agent learning with 4 options. The darkest color represents the walls in the environment while lighter colors encode higher termination probabilities.

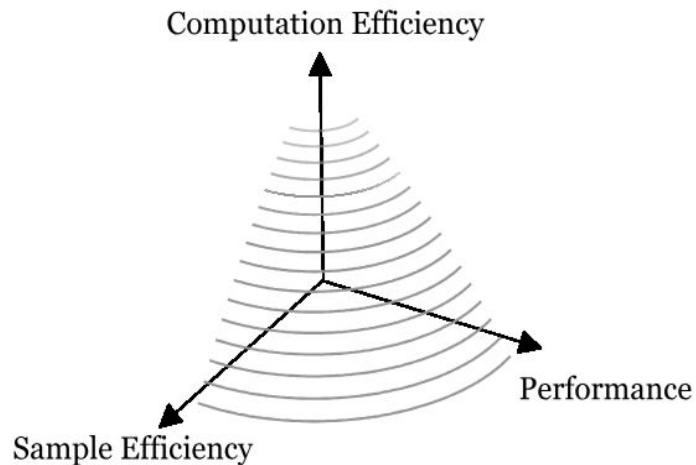
Program

```
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)
```



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**
 - When abstraction exist...
 - When abstraction are yet to be discovered...





Formalism

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

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

Option framework,

$\pi : \mathcal{S} \times \mathcal{O} \rightarrow \Delta(\mathcal{A})$ **Intra**-option (low-level) policy
defines how the agent acts given state s under option o

$\beta : \mathcal{S} \times \mathcal{O} \rightarrow [0, 1]$ Termination function
probability of ending option o upon reaching state s

$\mathcal{I} : \mathcal{S} \times \mathcal{O} \rightarrow \{0, 1\}$ Initiation set
indicates where an option o can start, “affordances”

and friends

$$z \in \mathcal{Z}$$

Option identifier

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

Option reward function

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

High-level policy

Squirrel Example (call-and-return)



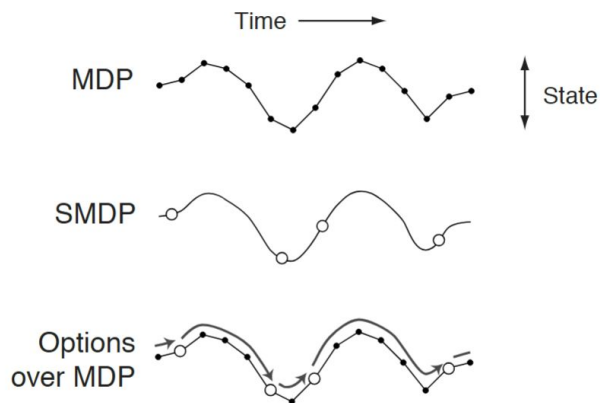
“Store in Drey” option

Initiation: Having acorns in cheek pouches

Policy: Carry the acorns, and find a drey

Termination: Acorns are in the drey

Semi-MDP, and options



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

Per-timestep vs. Multiple-timestep decision

$P(s' | s, a)$ VS. $P(s', \tau | s, o)$ \rightarrow # timestep

- In SMDP, after one decision, environment will evolve many timesteps
- MDP + Options = SMDP ([Sutton et al., 1999](#))
 - This works in call-and-return model: high-level decision is an option; duration is controlled by the termination function
 - Which means: an SMDP can be **induced** by an MDP and a set of options

Between Goals and Options

- Intelligence in AI: “Intelligence is the computational part of the ability to achieve **goals** in the world.” ([McCarthy, 1997](#); [Sutton, 2020](#))
- Goals can be generalized to some reward (reward hypothesis, [David Silver et al. 2021](#))
- Goal-condition RL: augments the observation with an additional well-defined goal
 - Allow generalization over goal space
 - UVFA ([Schaul et al, 2015](#)): value depend on state and goal, $V(s, g)$
 - Goal-condition RL is HRL!
- Subgoals induce temporally extended behaviors that achieves them
- Subgoals induce **termination** functions of options
- To achieve a subgoal, a precondition might be achieved (**initiation**)

Temporally abstract predictive knowledge

General Value Functions

- Given a cumulant function c , state-dependent continuation function γ and policy π , the General Value Function $v_{\pi, \gamma, c}$ is defined as:

$$v_{\pi, c, \gamma}(s) = \mathbf{E} \left[\sum_{k=t}^{\infty} c(S_k, A_k, S_{k+1}) \prod_{i=t+1}^k \gamma(S_i) \mid S_t = s, A_{t:\infty} \sim \pi \right]$$

- *Cumulant* c can output a vector (even a matrix)
- *Continuation function* γ maps states to $[0,1]$ (further generalizations are possible)
- Cf. Horde architecture (Sutton et al, 2011); Adam White's thesis; inspiration from Pandemonium architecture
- Special case: policy is optimal wrt c, γ , $v_{c, \gamma}^*$ - Universal Value Function approximation (UVFA) (Schaul et al, 2015)

Temporally abstract predictive knowledge

Option Models

1. *Expected reward* $r_\omega(s)$: the expected return during ω 's execution from state s
 2. *Transition model* $P_\omega(s'|s)$: specifies *where* the agent will end up after the option/program execution and *when* termination will happen
- “Jumpy” planning is better for temporal credit assignment, accurate value estimation

HRL = Option Discovery + Effectively Use Options

- HRL is to identify and utilize temporal structure
- We can assume either:
 - (a) Options pre-exist, and easy to get
 - (b) Options are not (or partly) given, but natural temporal structure of the problem exists

Two key problems in HRL:

- Option **discovery**: find good option (triples), assuming (b)
- Effectively **use** options: given options, how to better predict/control/plan

Recap

- HRL solves control problem with temporal abstractions, in different timescales
- HRL leverages the inductive bias of compositionality and modularity
- HRL finds good temporal abstractions that benefit:
 - Exploration
 - Credit assignment
 - Transferability
 - Interpretability
- HRL eagers to find a desirable trade-off between performance, sample efficiency, and computation efficiency
- Procedural knowledge: Option
- Predictive knowledge: GVFs, option models
- HRL comes down to
 - Discover options
 - Effectively use those options



How options are Discovered?

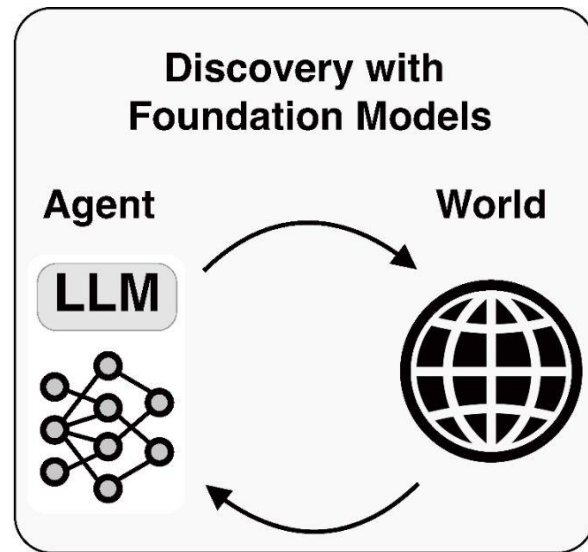
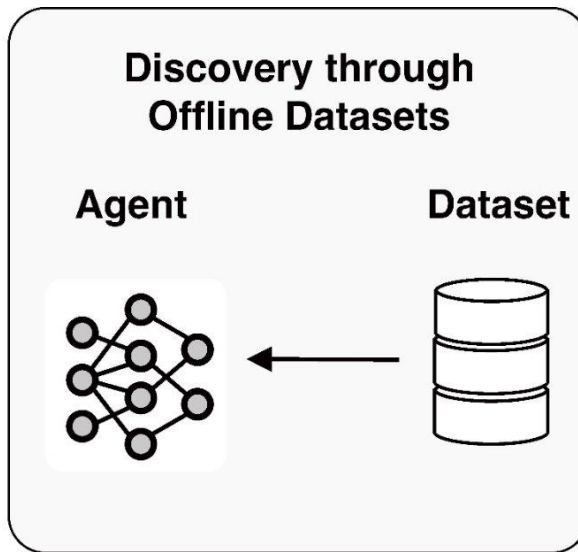
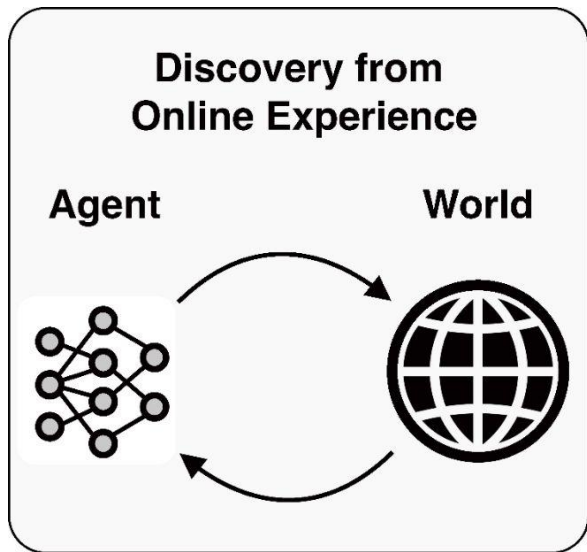
Where are the options from?

Option Discovery Problem

- Option discovery is to construct a set of options. (with the [benefits](#))
- Such construction is usually combinatorially challenging, need a lot of samples and time if do it from raw interaction!
- So, we need additional *assumptions* that helps to find options.
- Here we list some common assumptions:
- **Assumption 1:** Each option is driven by some associated objective, e.g., option reward
 - E.g., achieving subgoal, aligning to some existing behavior
- **Assumption 2:** The option set is structured
 - E.g., option embedding / program code as an identifier, which enables generalization
 - Or, fix or constrain a subset of the option triple (e.g., $I(s)=1$, option initiates at any state)
- **Assumption 3:** Some data/model pre-exist

Availability of Data / Prior knowledge

Naturally creates a high-level taxonomy










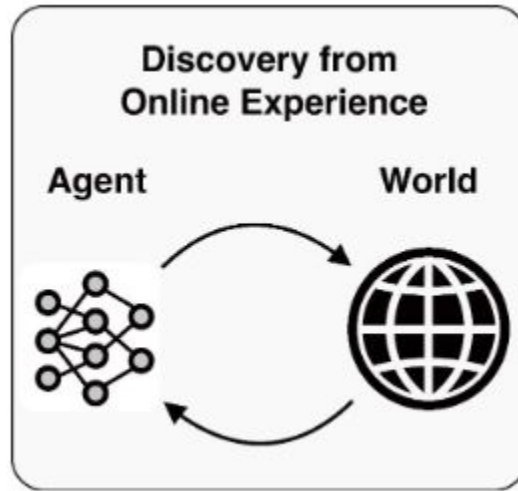
Some principles of online methods can be also adopted into methods with stronger data/prior assumption.

Low-level taxonomy:

Principles of discovery

See the [survey paper](#)
for details of each
methods

Category	Methods	 Credit Assign.	 Explor.	 Transf.	 Interpr.
 Online	Bottleneck Discovery
	Spectral Methods
	Skill Chaining
	Empowerment Maximization
	Via Environment Reward
	Optimizing HRL Benefits Directly
	Meta-Learning
	Curriculum Learning
 Offline	Intrinsic Motivation
	Variational Inference
 Foundation Models	Hindsight Subgoal Relabeling
	Embedding Similarity
	Providing Feedback
	Reward as Code
	Direct Policy Modeling



Discovery from *Online* Experience

Bottleneck Discovery

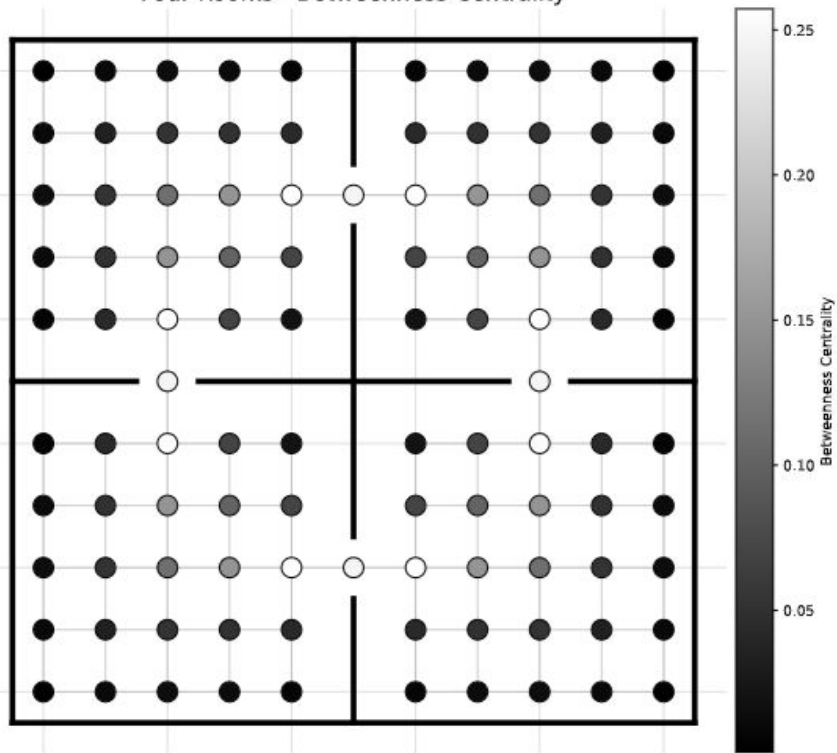
Discovery from *online* experience

- Environments (MDPs) as graphs
 - Node: state
 - Edge: single-step reachability ($p>0$)
 - Bottleneck states as goals

$$DD(s) = \prod_{\tau \in \mathcal{T}^+} P(s \in \tau) \prod_{\tau \in \mathcal{T}^-} (1 - P(s \in \tau)),$$

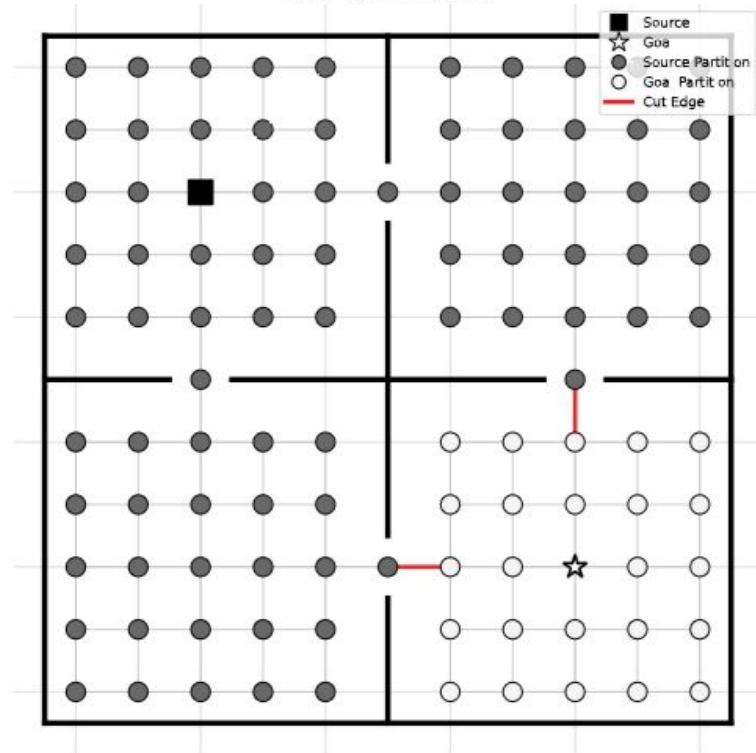
- **Frequency (e.g., diverse density)**
 - Diverse density: bottlenecks are states prevalent in successful but not unsuccessful trajectories
 - Betweenness centrality: bottlenecks are states prevalent in the shortest paths
- **Graph partitioning / State Graph Analysis**
 - Bottlenecks: states in many paths of the graph (like “bridges”)
 - Removing them leads to separating loosely-connected subregions
 - Graph-laplacian-based methods: approximate graph topology by an eigenfunction
- Information-theoretic ideas

Four Rooms - Betweenness Centrality



(a) Betweenness Centrality

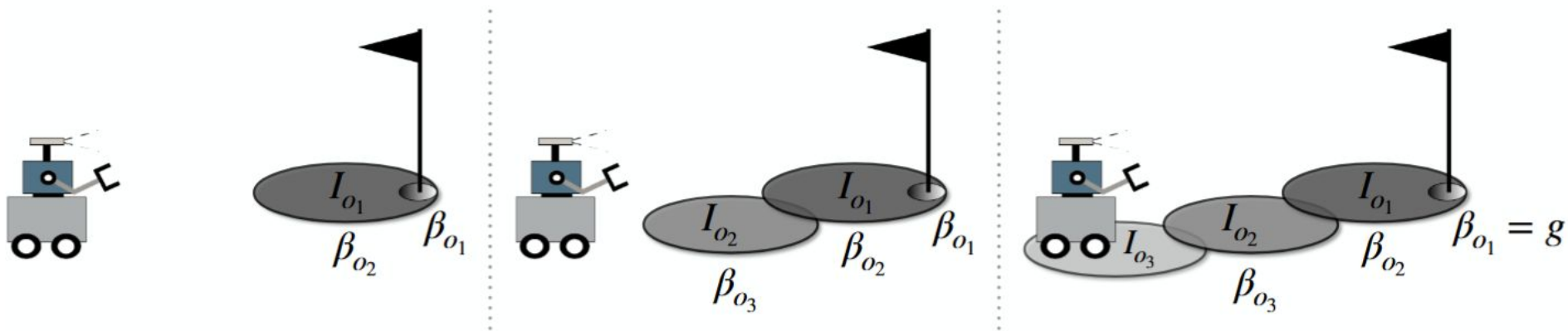
Four Rooms Q-Cut



(b) Q-Cut

Skill Chaining (Sequentially Composable Options) Discovery from *online* experience

- Assume: having some successful trajectories
- Construct options backward from the final goal (I, π)
- Goals of later options are initiation sets of earlier ones
- Stop until the initial states are reached

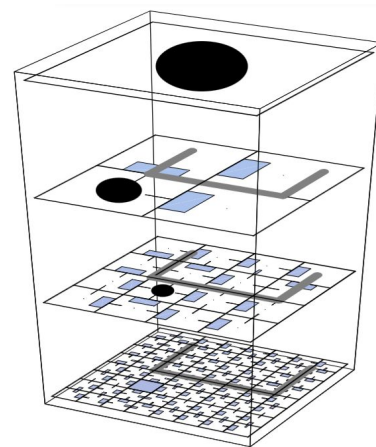


Via Environment Rewards

Discovery from *online* experience

Feudal methods

- Feudal RL ([Dayan and Hinton, 1993](#)): managers set up a **subgoal**, workers achieve that subgoal
 - Different MDPs in different levels, can be multi-level
 - Information hiding (state abstraction)
 - Reward hiding: managers use environmental reward, workers use intrinsic one (provided by the goal specification)
- FuN ([Vezhnevets et al., 2017](#)): a bi-level deep RL instantiation
 - Manager: outputs a “direction” vector as subgoal
 - Worker: intrinsic reward encourages state transition towards goal
 - Termination: Fixed k timesteps

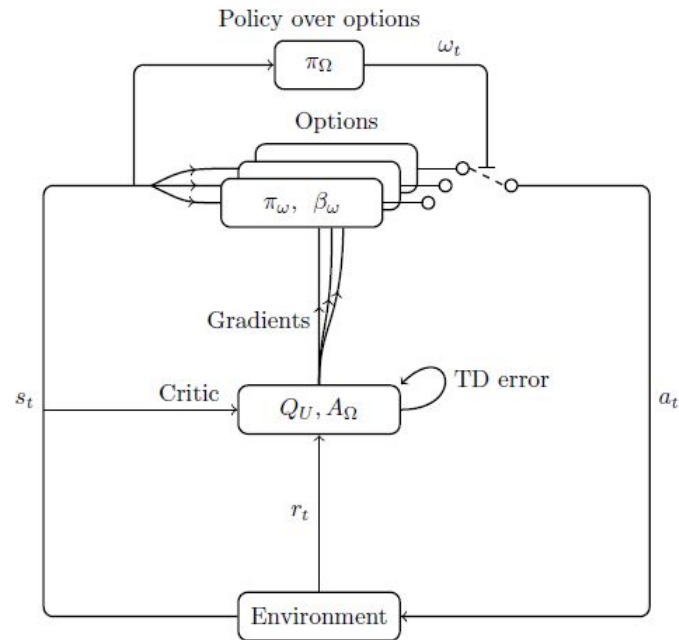


Via Environment Rewards

Option-critic ([Bacon et al., 2017](#))

- Frame discovery as *learning*
- Given the # of options
- Optimize termination function
 - Increase termination prob. if higher value can be achieved switching to other options
 - Driven by some value function related to the termination signal
- Meanwhile, optimize the option policy / high-level policy in the same loop

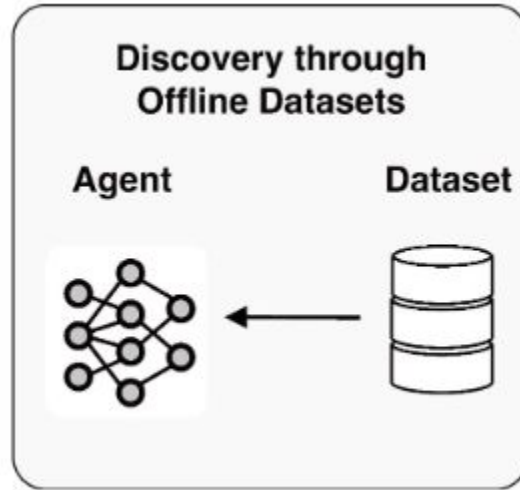
Discovery from *online* experience



Curriculum Learning

Discovery from *online* experience

- Master challenging tasks by progressing through a sequence of tasks with *increasing complexity*
- HRL perspective: how should the high-level policy choose next goal?
- Explicit curriculum
 - optimize goal selection so that updates lead to a progress of performance
- Implicit curriculum (HER, [Andrychowicz et al., 2018](#))
 - relabeling experiences with alternative goals present in the trajectory and learns a goal-conditioned policy
 - naturally creates a curriculum with increasing difficulty



Discovery via *Offline* Dataset

$$\mathcal{D} = \{\tau_i\}_{i=1}^N \text{ where } \tau_i = (s_t^i, a_t^i, r_t^i)_{t=1}^T :$$

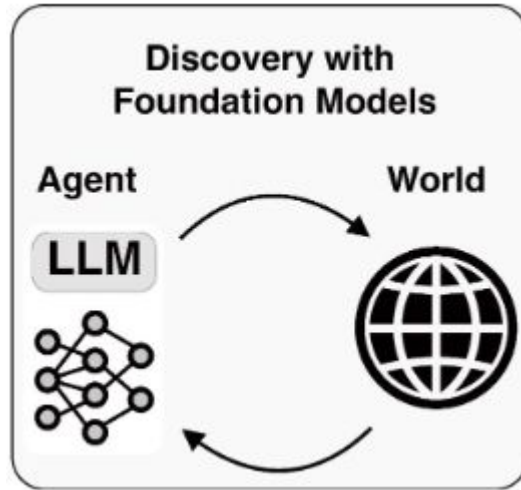
Variational Inference

Discovery from *offline* dataset

- Skills as latent variables (skill embedding + boundary) inferred from *unlabeled* trajectories via reconstruction objectives, often using VAE
- Extension: incorporating *minimum description length* (MDL) principle, encouraging using concise skills sequences to explain the data

Hindsight Subgoal Relabeling

- Identify and relabel subgoals within the dataset of trajectories
- Subgoals can be important waypoints, or just $s_{\{t+k\}}$ (as goal of s_t)
- Heuristics to select subgoals depend on methods



Discovery with *Foundation Models*

Embedding Similarity

Discovery with *foundation models*

- Define goal-conditioned option reward based on similarity of pretrained embeddings of goals and observations

Providing Feedback

- LLMs may tell you:
 - Successfully achieves the goal?
 - Preference over states/trajectories?
- Those signals can be distilled into (option) reward models

Reward as Code

Discovery with *foundation models*

- Generate executable code to compute reward function programmatically
- Task/env/goal info -> LLM -> program(s)/automata

Directly Modeling the Policy

- Generate (high-level or low-level) action sequence conditioned on natural language goal and state information

How to Use the Discovered Temporal Structure?

- Execution modes over options
 - **Call-and-return** (most common)
 - Execute one option until termination
 - One option per timestep
 - **Redistributed computation**: e.g., GPE that allow agents to evaluate multiple options simultaneously
 - **Generalized policy evaluation (GPE)**: compute the value of a policy π on a set of reward functions \mathcal{R}
 - **Generalized policy improvement (GPI)**: given a set of policies Π and a reward function r , compute a new policy such that:

$$Q_r^{\pi'}(s, a) \geq \sup_{\pi \in \Pi} Q_r^{\pi}(s, a), \forall s \in \mathcal{S} \forall a \in \mathcal{A}$$



Blog

Thanks for your attention!