



Discovering Temporal Structure

An Overview of Hierarchical Reinforcement Learning

Martin Klissarov, Akhil Bagaria, Ziyang “Ray” Luo, George Konidaris, Doina Precup, Marlos C. Machado



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”?
 - A “partial order” determined by **abstraction levels**
 - The amount of less relevant details ignored

Motivation: Why HRL?

- Bionics: Intelligence ([animals](#)) reasons and acts across [extended timescales](#).
- Computation: **Standard RL**: states \rightarrow primitive actions
 - Making a decision at **every timestep**
- HRL formalizes the idea of flexibly reasoning over different timescales at multiple levels of [temporal abstraction](#)
- Beneficial in exploring, planning and learning in [open-ended environments](#)
- [MDP](#) vs. [SMDP](#)

$$\left(\begin{array}{l} \text{Per-timestep decision vs. Multiple-timestep decision} \\ P(s' | s, a) \quad \text{VS.} \quad P(s', \tau | s, o) \\ R(s, a) \quad \quad \text{VS.} \quad R(s, a, \tau) \end{array} \right)$$

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

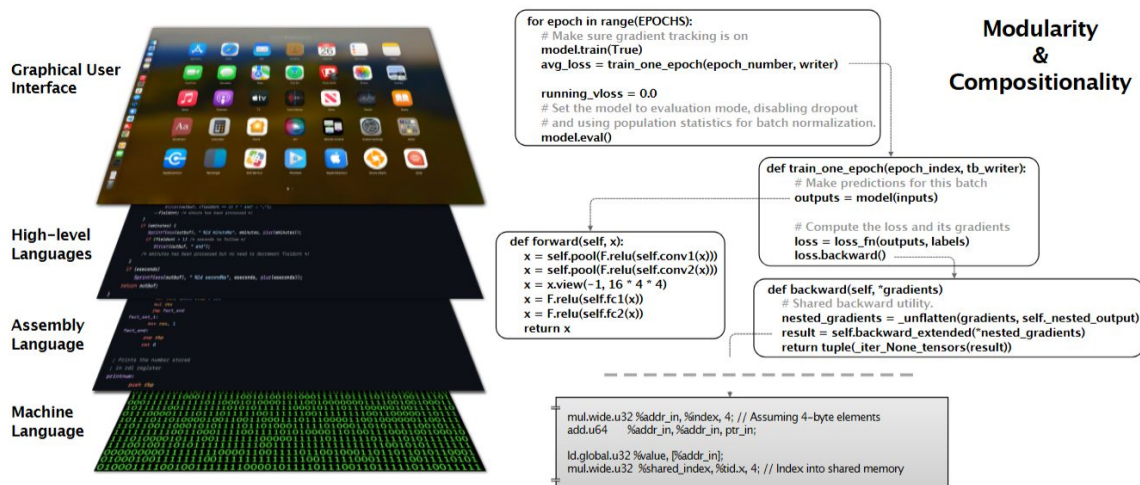
Enter
Nut

Deposit Nut



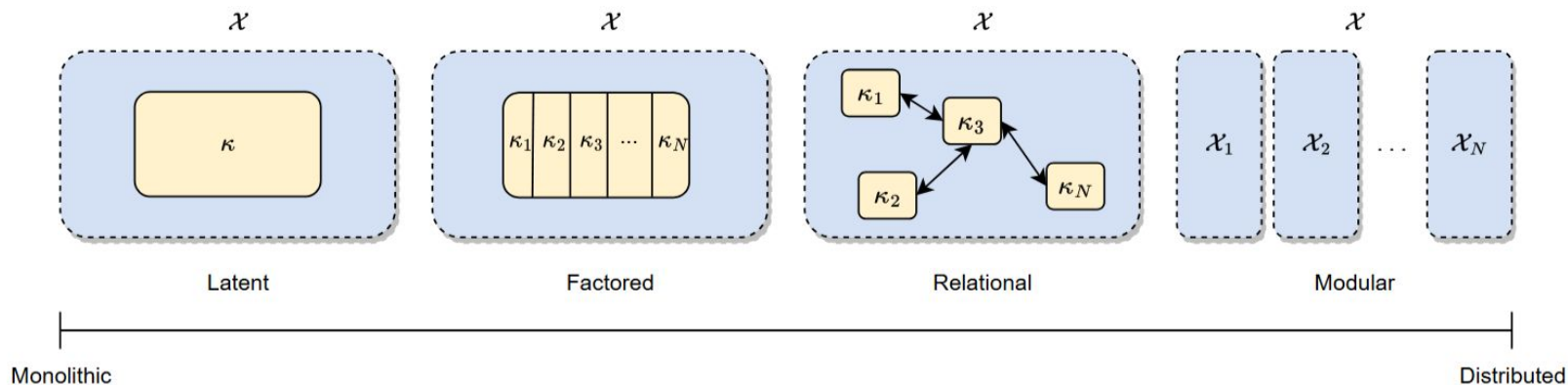
Inductive bias of HRL

- HRL exploits **temporal structure** of the problem
- Complex problem's solution \approx solution of its simpler subproblem with
 - **Compositionality**: complex solution is built from simpler parts
 - **Modularity**: sub-solutions can be used without concerning details



What is “Structure”? ([Mohan et al., 2024](#))

- Side information about **Decomposability**
- **Top-down**: decomposing **problem** into sub-problems (agent-centric)
 - Environment (State/action spaces, Transitions, Rewards)
- **Bottom-up**: decomposing **solution** into sub-solutions (env-centric)
 - Policy, Value functions, Models, Training procedures



But...



what is a good/useful temporal abstraction?

What is the “objective” of HRL?

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

What are good temporal abstractions?

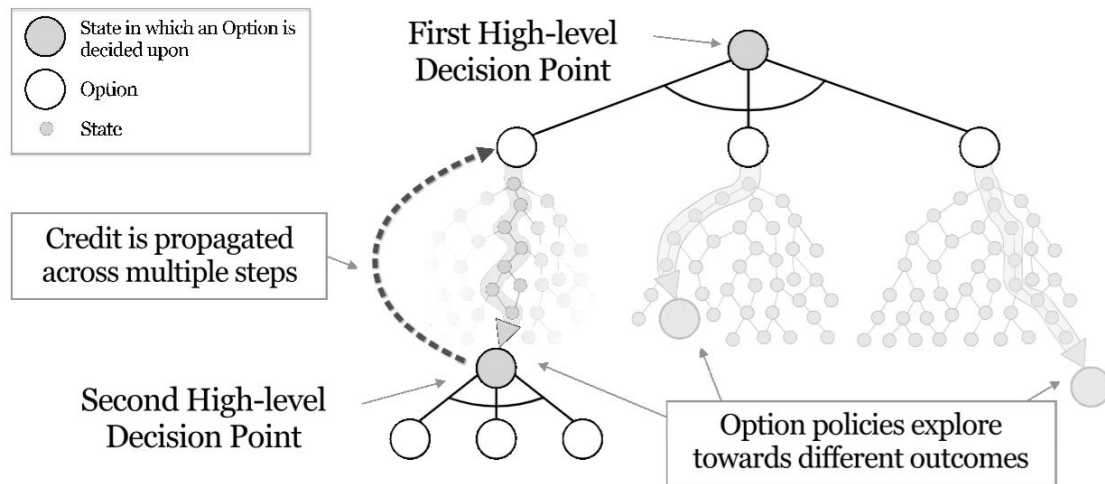
The dual question: what are the **benefits** of useful temporal abstractions?

With good/useful abstraction, we can achieve better:

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

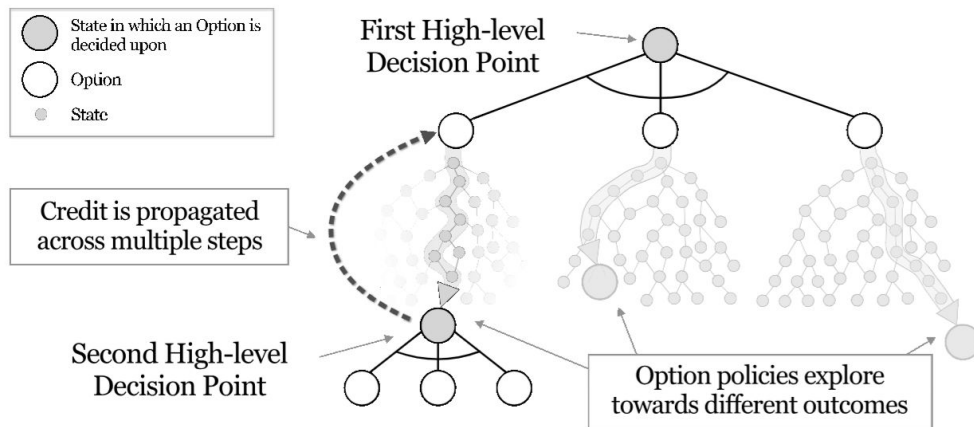
Structured Exploration

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



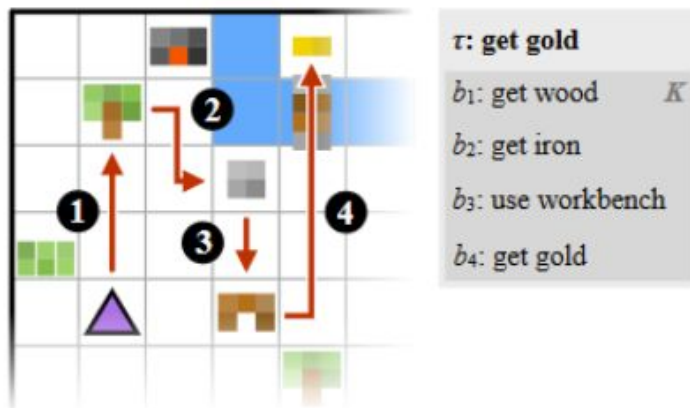
Effective Credit Assignment

- *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
- Identify “critical” decisions (bottlenecks, subgoals)



Transferability (Generalizability)

- Options can be **reused** across different tasks
- Options can **generalize** in similar but novel conditions
- Similar tasks (goals) can use similar 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!

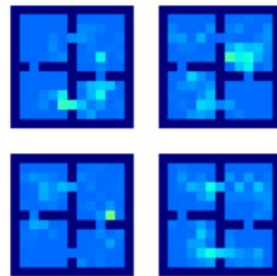
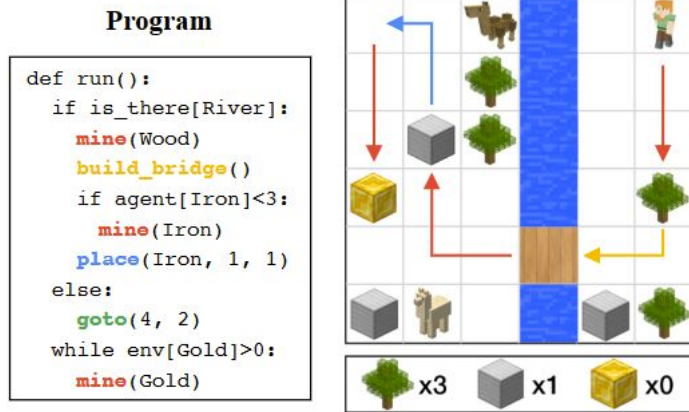
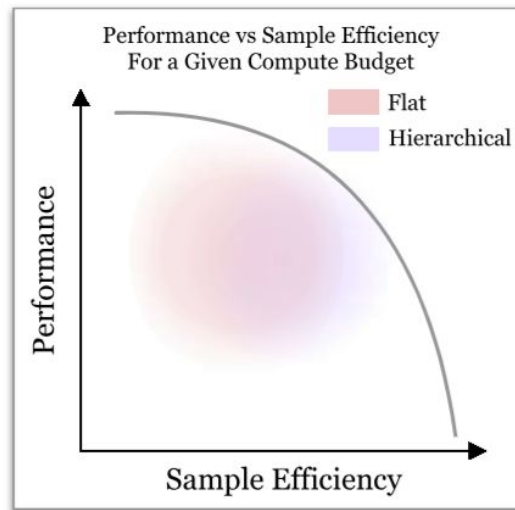
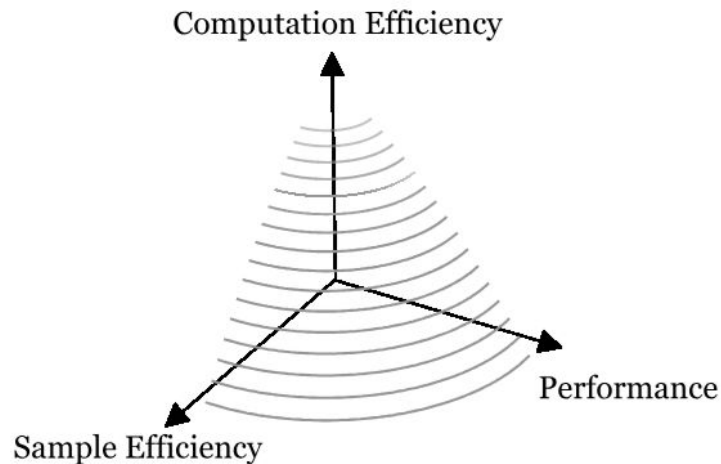


Figure 3: Termination probabilities for the option-critic agent learning with 4 options. The darkest color represents the *walls* in the environment while lighter colors encode higher termination probabilities.



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
- ❏ Related Terminologies
- ❏ Option Discovery Problem
- ❏ Scope of HRL

Option framework, and friends

$$\pi : \mathcal{S} \times \mathcal{O} \rightarrow \Delta(\mathcal{A}) \quad \text{Intra-option policy}$$

defines how the agent acts given state s under option o

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

probability of ending option o upon reaching state s

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

indicates where an option o can start, “affordances”

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

Option reward function

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

High-level policy

Skills are options

- Many papers use the word “skill” but definition identical to option
- Bayesian people’s def: A latent variable, z , that explains observed trajectories $\tau = (s_0, a_0, s_1, \dots, s_T)$

$$p(a_t | s_t, z) \leftrightarrow \pi_o(a_t | s_t)$$

- Some latent variable models explicitly defines task boundaries / termination signal to be inferred

Goal-conditioned RL (GCRL) is HRL

- Intelligence in AI: “Intelligence is the computational part of the ability to achieve **goals** in the world.” ([McCarthy, 1997](#); [Sutton, 2020](#))
- Goal definition (g, r^g, γ_g)
- GCRL: augments the observation with an additional well-defined goal
 - Allow generalization over goal space
 - Actually HRL but with *less focus on discovery* temporal abstractions
 - Assume goals exist and well-defined
- Subgoals induce temporally extended behaviors that achieves them
- Subgoals induce termination functions of options
- **Compositionality**: subgoals can be composed to create solutions
- **Modularity**: solution to subgoals can be reused

HRL = Option Discovery + Effectively Use Options

- Identifying, and utilizing temporal structure
- What is “option **discovery**” in a broader sense?
- My view (conceptually):

$$\mathcal{D} : \text{data} \longrightarrow (\mathcal{Z}^*, \psi^*)$$

Data: online experience / offline trajectories / foundation model

\mathcal{Z}^* : discovered latent codes for options (identifier)

- As high-level actions
- Can be discrete catalog / continuous embedding
- Examples: Goals, program statement, skill embedding

Option constructor. Let the single parameter block $\psi = (\psi_\pi, \psi_\beta, \psi_{\mathcal{I}}, \psi_r)$ collect all weights. The *constructor*¹

$$\mathcal{C}_\psi : \mathcal{Z} \longrightarrow \mathcal{O}, \quad \mathcal{C}_\psi(z) = \left(\pi_\psi(\cdot | \cdot, z), \beta_\psi(\cdot, z), \mathcal{I}_\psi(\cdot, z); r_\psi(\cdot, \cdot | z) \right),$$

Recap

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

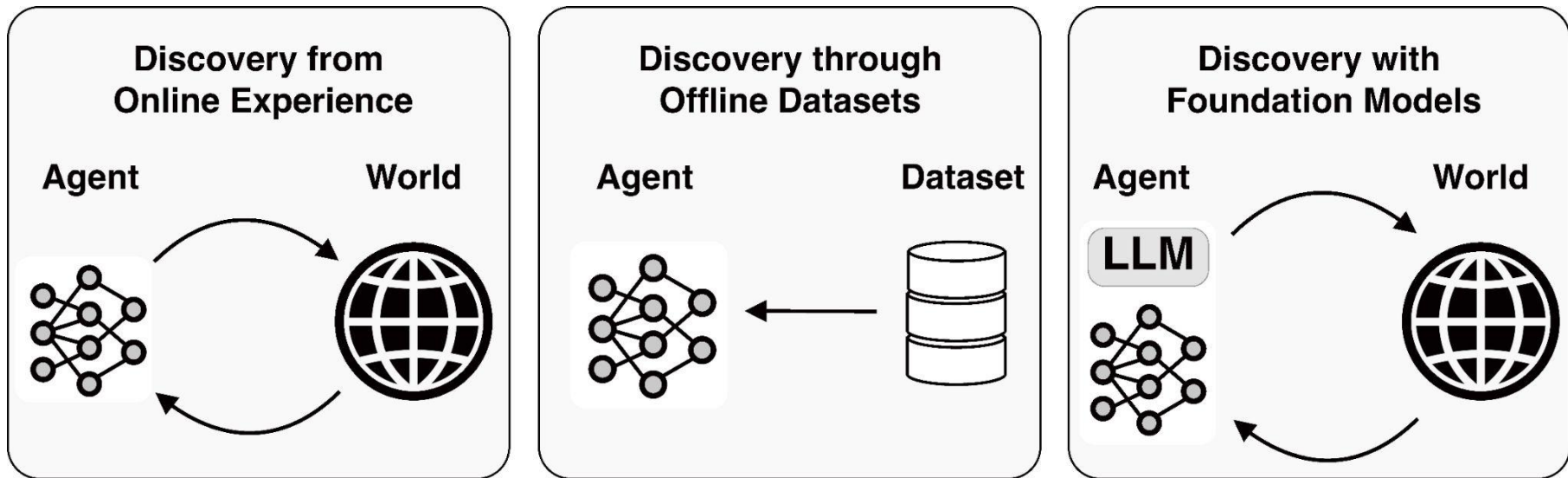


How is Temporal Structure Discovered?

Naturally forms a taxonomy of HRL, based on availability of data, and principles used to discover.

Availability of Data / Prior knowledge

Naturally creates a high-level taxonomy










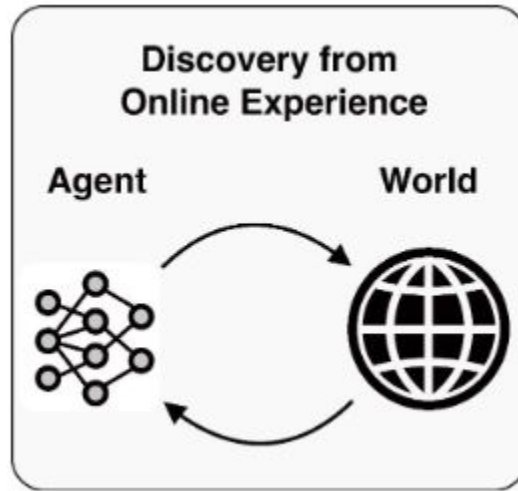
Some principles of online methods can be also adopted into methods with stronger data/prior assumption.

Low-level taxonomy:

Principles of discovery

Proxy objectives

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	
	Hindsight Subgoal Relabeling	..			.
 Foundation Models	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

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

- **Diverse density**

- Bottlenecks: states prevalent in successful but not unsuccessful trajectories

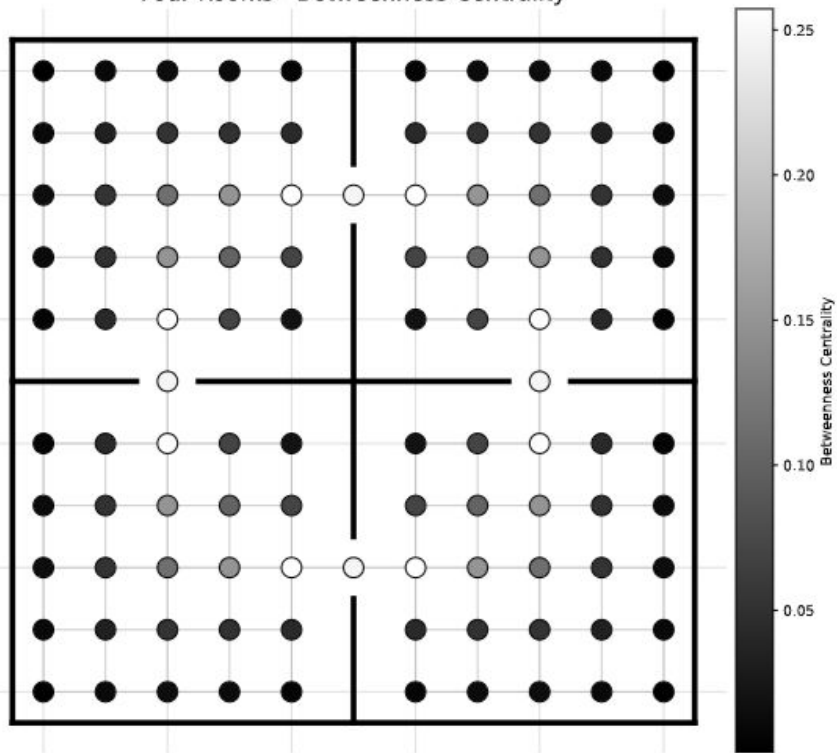
- **Graph partitioning**

- Bottlenecks: states in many paths of the graph (like “bridges”)
- Removing them leads to separating loosely-connected subregions
- As (recursive) *min-cut problem*: finding minimal edges that disconnect source to goal

- **Graph Centrality** (importance of node)

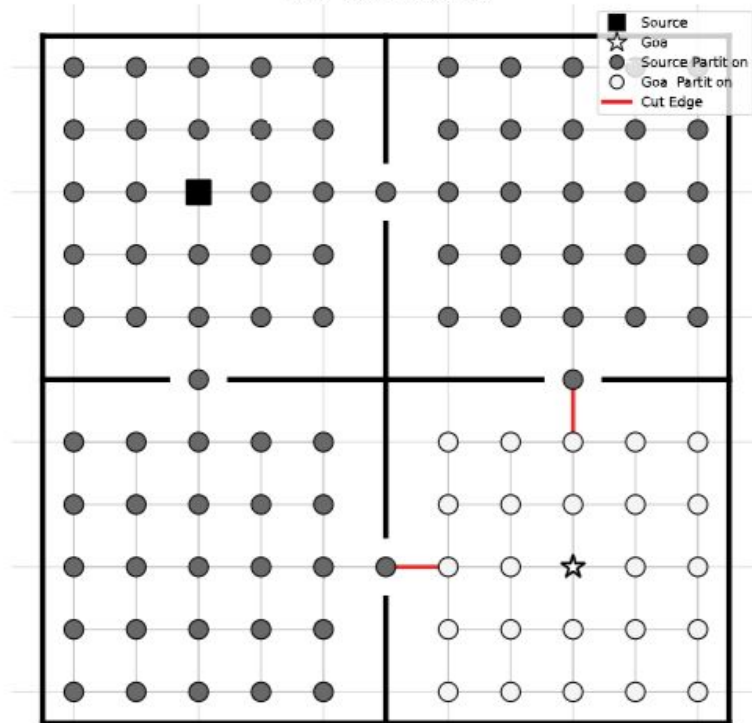
- Betweenness metric: counting how many times a state appears on the shortest path between other nodes

Four Rooms - Betweenness Centrality



(a) Betweenness Centrality

Four Rooms Q-Cut



(b) Q-Cut

Bottleneck Discovery

Discovery from *online* experience

- Top-down decomposition:
 - first know (model) the env, then leverage this info
- Main Benefits
 - Exploration: reaching bottleneck state -> more (hard-to-reach) states accessible
 - CA: jumpy steps, divide into subgraphs
 - Transfer: bottlenecks are often task agnostic
- Limitations
 - Scalability to large, continuous MDPs

Spectral Methods

Discovery from *online* experience

- Still consider the MDP as a graph
- Leveraging **topology** of the state space by analyzing the **eigenstructure** of matrix representations of environment
 - E.g., Graph Laplacian, successor representation

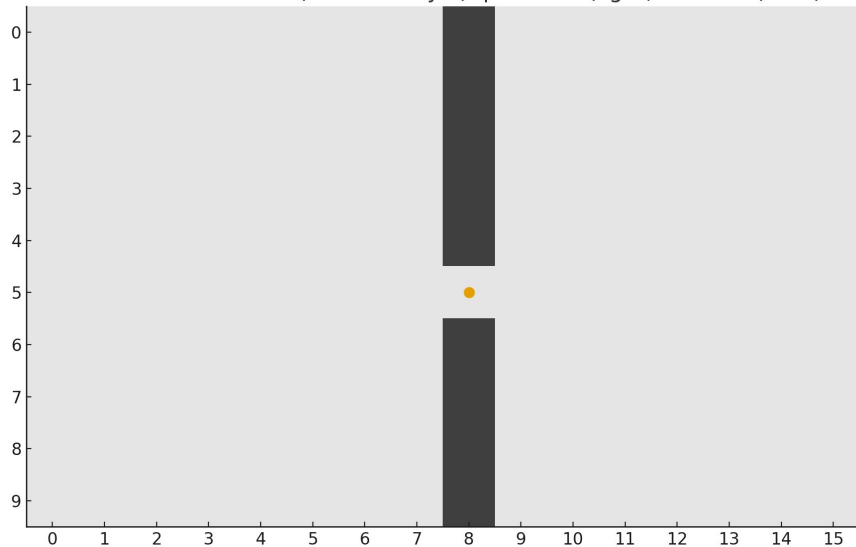
$$\mathcal{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2},$$

$A_{ij} = 1$ if edge $(i, j) \in E$, else 0. (adjacency)

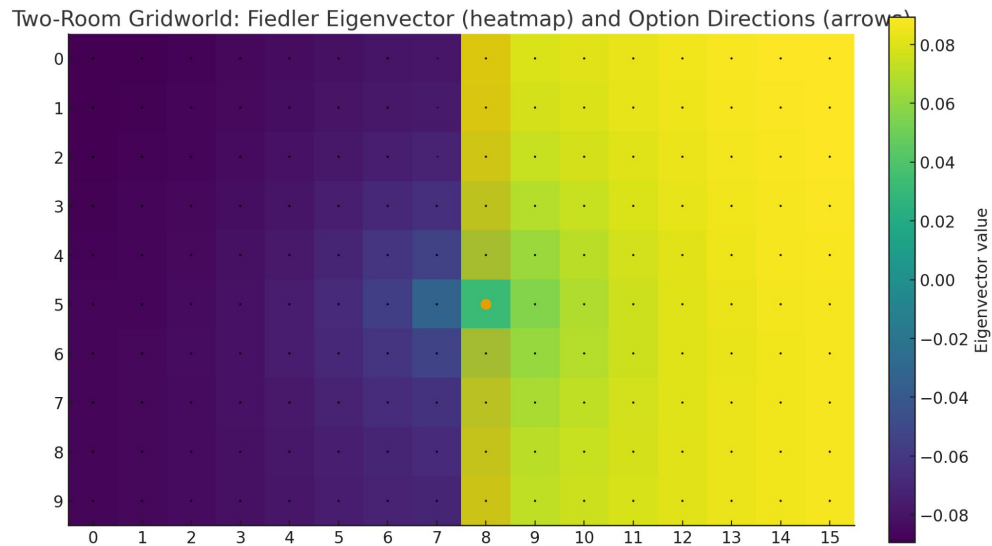
$D = \text{diag}(d_1, \dots, d_n)$ where $d_i = \sum_j A_{ij}$. (degree)

- Eigenvectors of graph Laplacian captures modes of variation
- The 2nd smallest eigenvalue's eigenvector captures the number of connected components in a graph, capturing **bottlenecks**!
- Can be approximated by a learned *eigenfunction* $f(s)$ from experience
- Eigenoptions can be defined by moving in the (\uparrow, \downarrow) direction of $f(s)$

Two-Room Gridworld (MiniGrid-style): passable (light) vs walls (dark)



(second smallest eigenvalue)

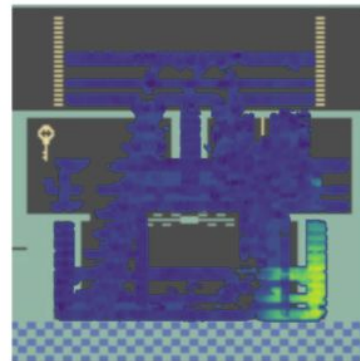
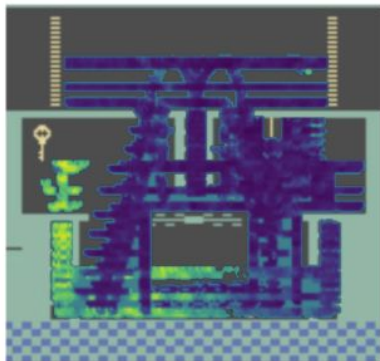
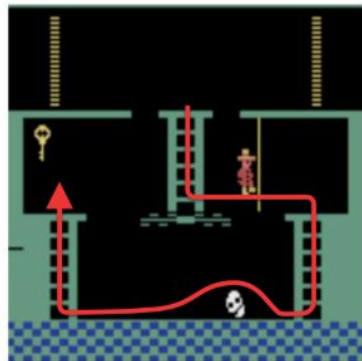


Eigenoption #1

Eigenfunction #1

Eigenoption #2

Eigenfunction #2



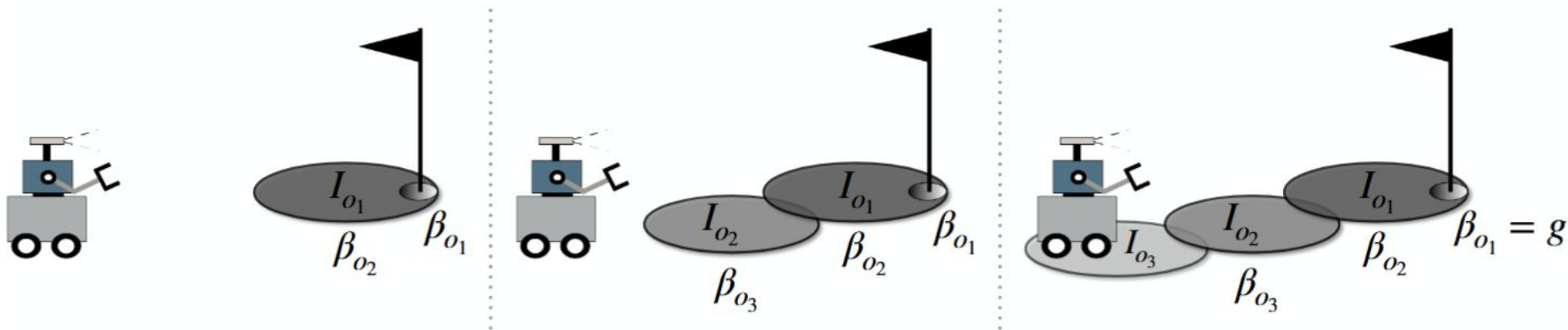
Spectral Methods

Discovery from *online* experience

- Top-down decomposition:
 - first know (model) the env, then leverage this info
- Main Benefits
 - Exploration: explore promising direction
 - Transfer: graph Laplacian is reward-irrelevant
- Limitations
 - Approximation of eigenfunction can be hard, need proxy

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



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

- Bottom-up decomposition:
 - Discover init/termination sets from experiences
- Main Benefits
 - Planning: discovered skills are composable sequentially
 - CA: high-level policy's jumpy transition, guide state distribution to the one that closer to goals
- Limitations
 - Mainly for goal-reaching domains
 - Need successful trajectories

Empowerment Maximization

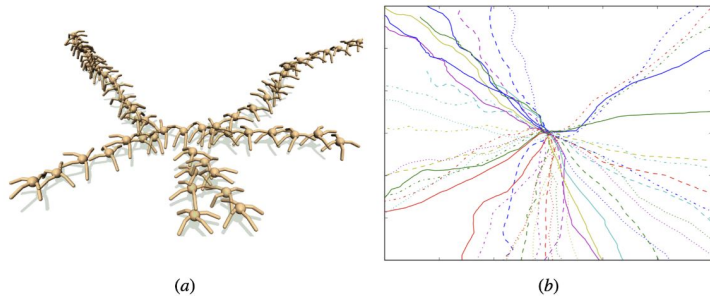
Discovery from *online* experience

- Discover **diverse and controllable** skills by *maximizing the agent's influence (empowerment) over its future observations*
- Empowerment defined as mutual information

$$\mathcal{E}_n(s_t) = \max_{p(\mathbf{a})} I(\mathbf{a}; s_{t+n} | s_t),$$

$$I(\mathbf{a}; s_{t+n} | s_t) = \mathcal{H}(s_{t+n} | s_t) - \mathcal{H}(s_{t+n} | s_t, \mathbf{a})$$

diversity of reachable outcomes unpredictability once action is chosen



$$I(X; Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

- VIC (Gregor et al., 2017): replaces open-loop actions with skills $J_{\text{VIC}} = I(z; s_{t+n} | s_t)$
- DIYAN (Eysenbach et al., 2019): skills should be distinguishable from the current states (not actions)

$$J_{\text{DIYAN}} = I(s; z) + \mathcal{H}(a | s) - I(a; z | s)$$

Empowerment Maximization

Discovery from *online* experience

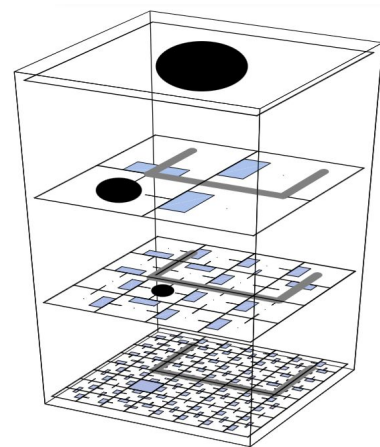
- Bottom-up decomposition:
 - Optimize skill latents that capture the mode of (online) data
 - Skills are learned in an “unsupervised” phase (intrinsic reward)
- Main Benefits
$$r_z(s, a) \triangleq \log q_\phi(z | s) - \log p(z)$$
 - Exploration: diverse skills, and controllable within skills; leading to diverse states and systematic exploration
 - CA: can learn high-level policies over discovered skills
 - Transfer: skill structure is reward-irrelevant
- Limitations
 - Empowerment is hard to estimate and optimize

Via Environment Rewards

Discovery from *online* experience

Feudal methods

- Feudal RL ([Dayan and Hinton, 1993](#)): managers set up a goal, workers achieve the goal
 - Different MDPs in different levels, can be multi-level
 - Information hiding (state abstraction)
 - Reward hiding: managers use environmental reward, workers use intrinsic one
- 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

Discovery from *online* experience

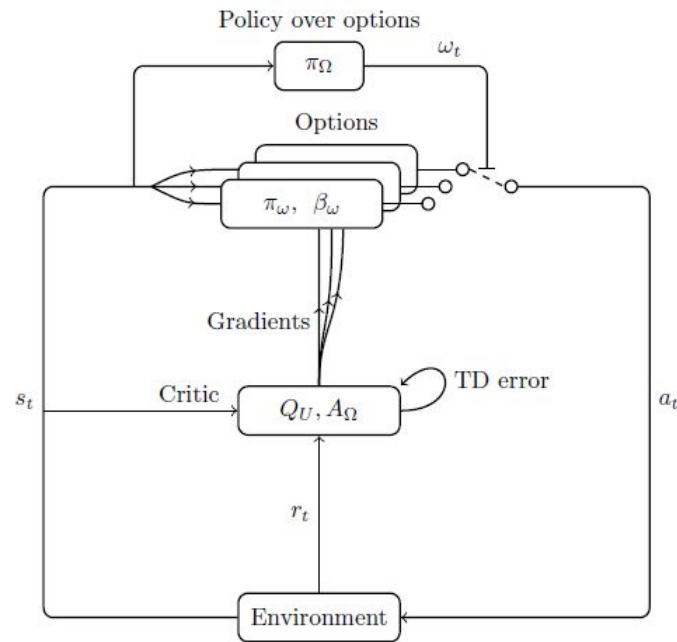
Option-critic ([Bacon et al., 2017](#))

- Frame discovery as *learning*
- Optimize termination function, increase termination prob. if higher value can be achieved switching to other options

$$q_u(s, o, a) = r(s, a) + \gamma \sum_{s'} p(s' | s, a) u_\beta(o, s').$$

$$q_\pi(s, o) = \sum_a \pi(a | s, o) q_u(s, o, a),$$

$$u_\beta(o, s') = (1 - \beta(s', o)) q_\pi(s', o) + \beta(s', o) v_\mu(s'). \quad v_\mu(s) = \sum_o \mu(o|s) q_\pi(s, o).$$



Via Environment Rewards

Discovery from *online* experience

- Bottom-up decomposition:
 - Optimize option to maximize return
- Main Benefits
 - CA, transfer, interpretability: if the problem has a structure, it might emerge, as capturing this structure is rewarded
- Limitations
 - Degeneracy of option
 - One option per action
 - Always using one option
 - Reliance on the informative reward signals

Directly Optimizing Benefits

Discovery from *online* experience

- Discussed works: Formal connections of proxy objectives and agent's capabilities remains unclear
- Methods in this line work on provably optimizing direct metrics on:
 - Planning time
 - Exploration
 - Credit assignment
 - Transfer

Meta Learning

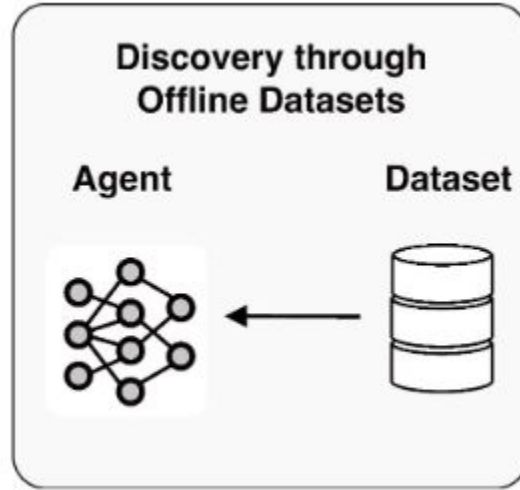
Discovery from *online* experience

- Discover by meta-gradient-based methods
- Usually in meta RL,
 - Outer-loop: learn the meta-parameters (lr, initialization, ...)
 - Inner-loop: learn the policy
- Some parameters in the option framework can be learned in outer-loop, while some in inner-loop
- [Frans et al. \(2018\)](#): high-level policy as meta-parameters
- [Veeriah et al. \(2021\)](#): parameterized subgoals as meta-parameters

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): 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

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

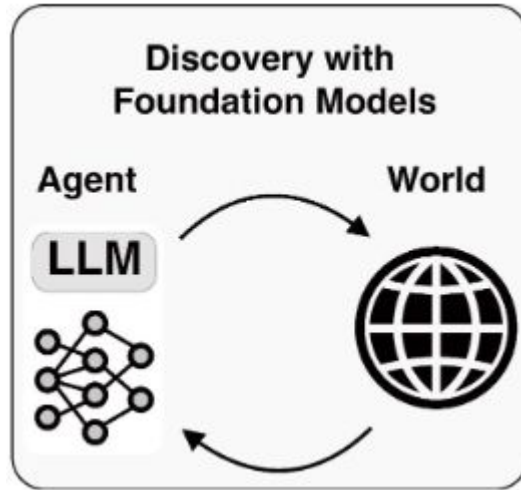
Variational Inference

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

Questions for audience

1. Benefits make sense? Any other fundamental benefits come to your mind? Once we have the options, it can be natural to have all (core) benefits.
2. How to describe discovery? What's the distinction between discovery and learning? Do you think some methods in this paper are not "option discovery" (e.g., works on LLM)?



Blog

Thanks for your attention!

How to Use the Discovered Temporal Structure?

- Deliberate modes over options
 - Call-and-return
 - Redistributed computation: e.g., GPE that allow agents to evaluate multiple options simultaneously