Matthew E. Taylor
(Matt)

Postdoc positions available

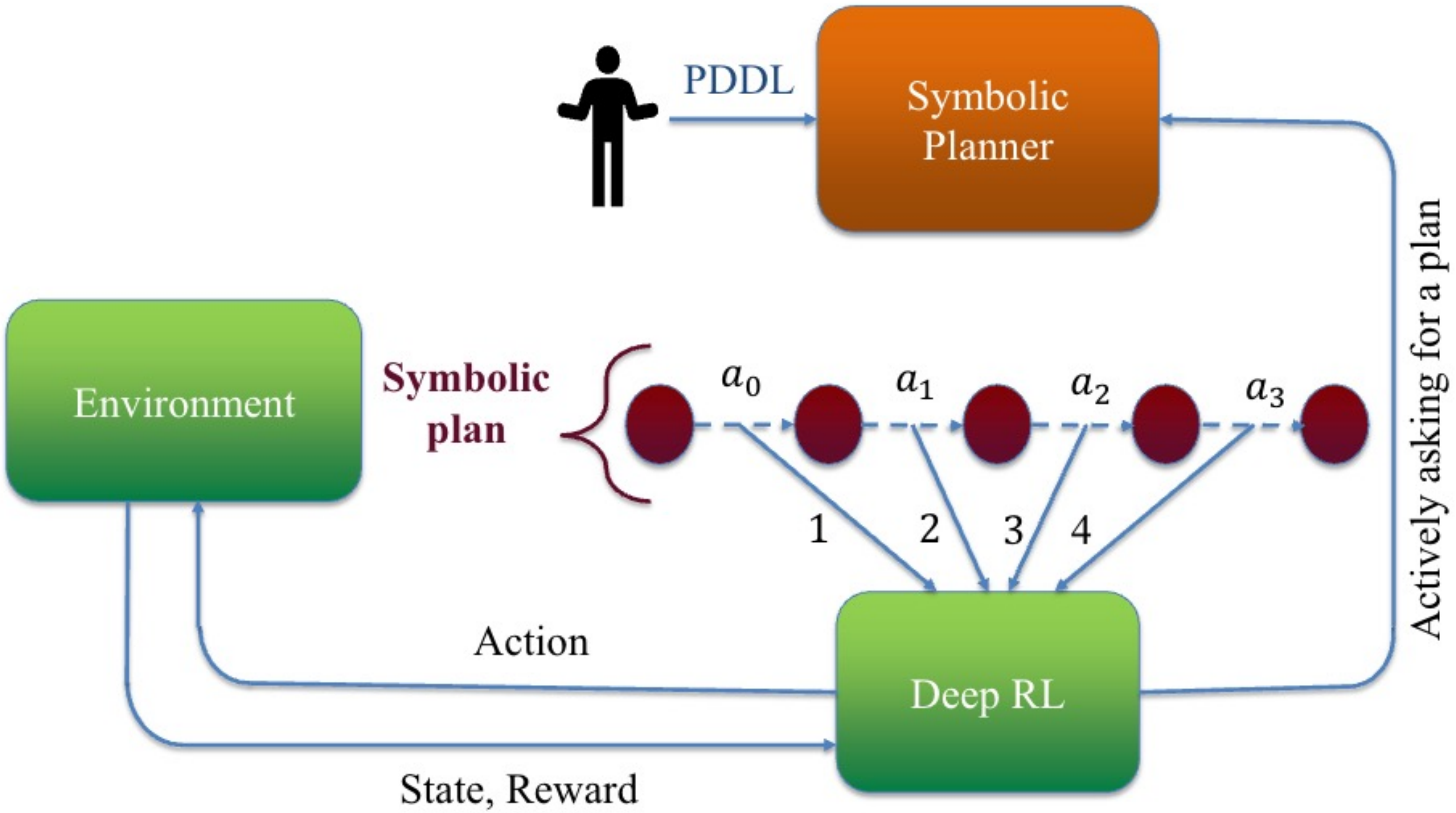
University of Alberta: Intelligent Robot Learning Lab (irll.ca)
AI Redefined: Research Director (AI-R.com)

Canada CIFAR AI Chair, Amii

Work in Progress: Using Symbolic Planning with Deep RL to Improve Learning

Tianpei Yang[1], Srijita Das[1], Christabel Wayllace[2], Matthew D. Taylor[1]

Methodology



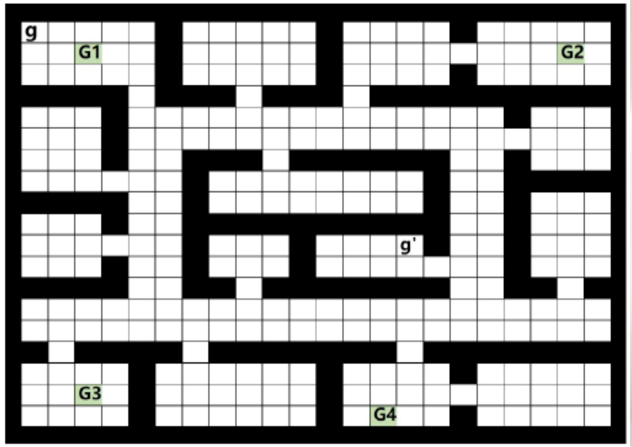
The Intelligent Robot Learning Laboratory

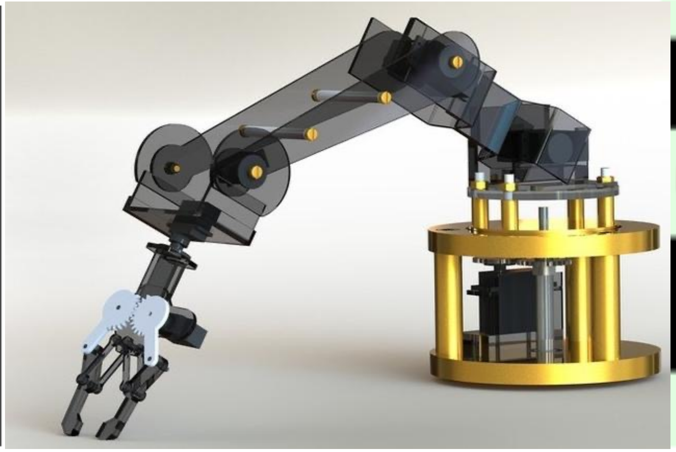
PADDLE: Logic Program Guided Policy Reuse in Deep Reinforcement Learning

Hao Zhang, Tianpei Yang, Yan Zheng, Jianye Hao and Matthew E. Taylor

Introduction

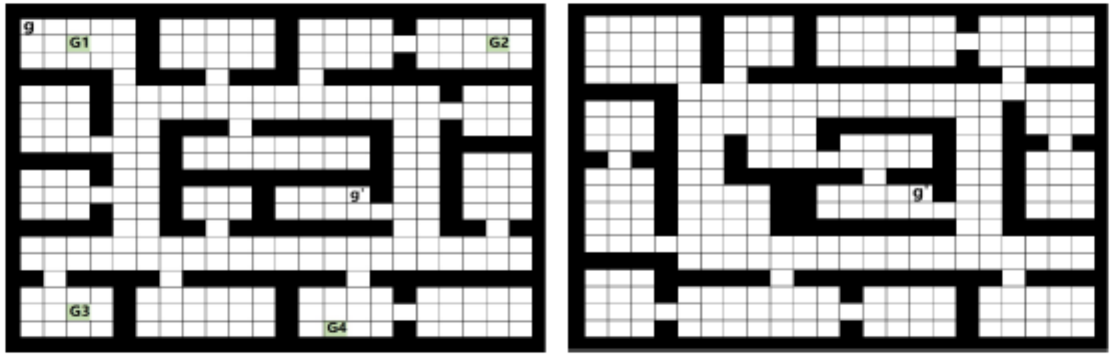
- Deep Reinforcement Learning (DRL) is faced with **sample inefficiency**, learning from scratch is difficult
- Transfer learning can leverage prior knowledge from past learned tasks
- Source task (domain) → Target task (domain)

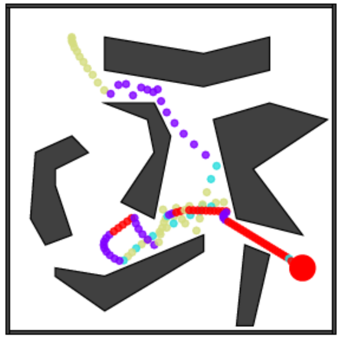


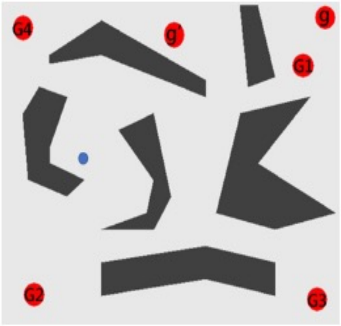


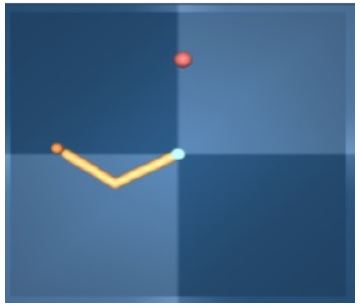
Introduction

- Select suitable source policies to reuse
  - Measure the similarity between state spaces/MDPs – *Hard to generalize to complex domains*
  - Measure the average performance of each source policy
  - Measure the average performance of partial of each source policy









[1] REPAINT: Knowledge Transfer in Deep Reinforcement Learning. ICML 2021.
[2] Efficient deep reinforcement learning via adaptive policy transfer. IJCAI 2020.
[3] Transfer learning for reinforcement learning domains: A survey. JMLR 2009.

The Intelligent Robot Learning Laboratory

Motivation

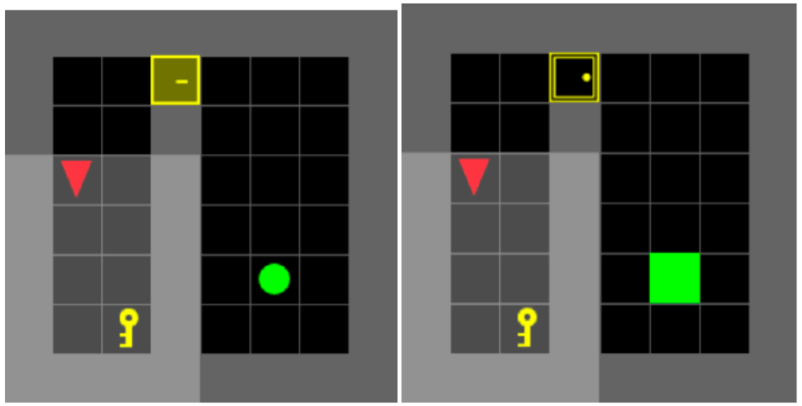
- Previous works learn policies/Q-tables → don't reveal the logic [1-4]
- Inductive Logic Programming can leverage neural symbolic learning to generate logic programs automatically
  - Rely on human expert definitions
  - Cannot be extended to continuous action spaces [5]

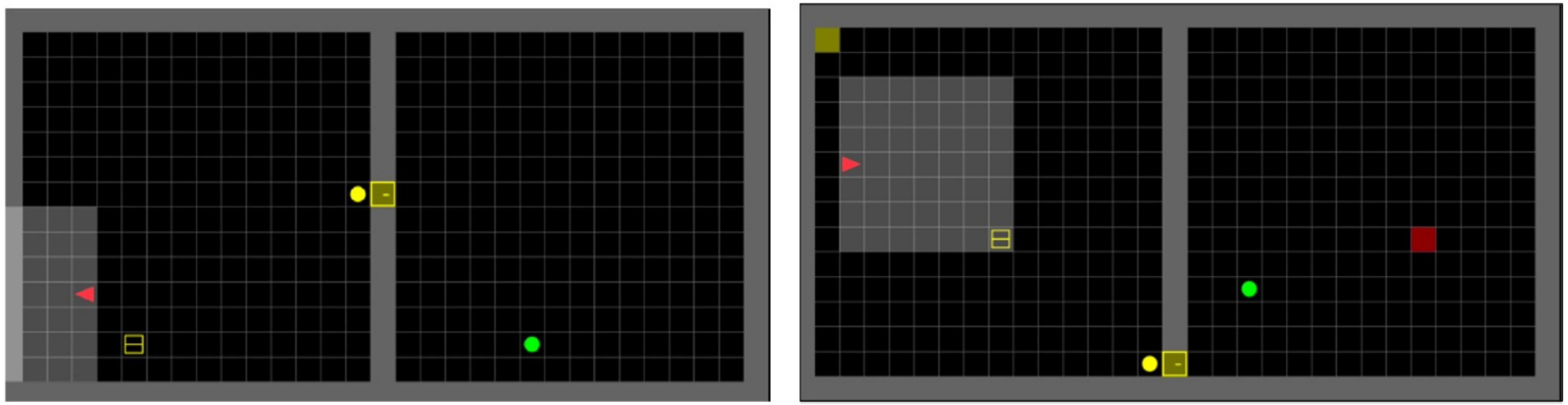
[1] REPAINT: Knowledge Transfer in Deep Reinforcement Learning. ICML 2021.
[2] Efficient deep reinforcement learning via adaptive policy transfer. IJCAI 2020.
[3] An Optimal Online Method of Selecting Source Policies for Reinforcement Learning. AAAI 2018.
[4] Probabilistic policy reuse in a reinforcement learning agent. AAMAS 2006.
[5] Neural logic reinforcement learning. ICML 2019.

Motivation

- Performance or Learning Progress might not be good indicator

DoorBall OpenGoal

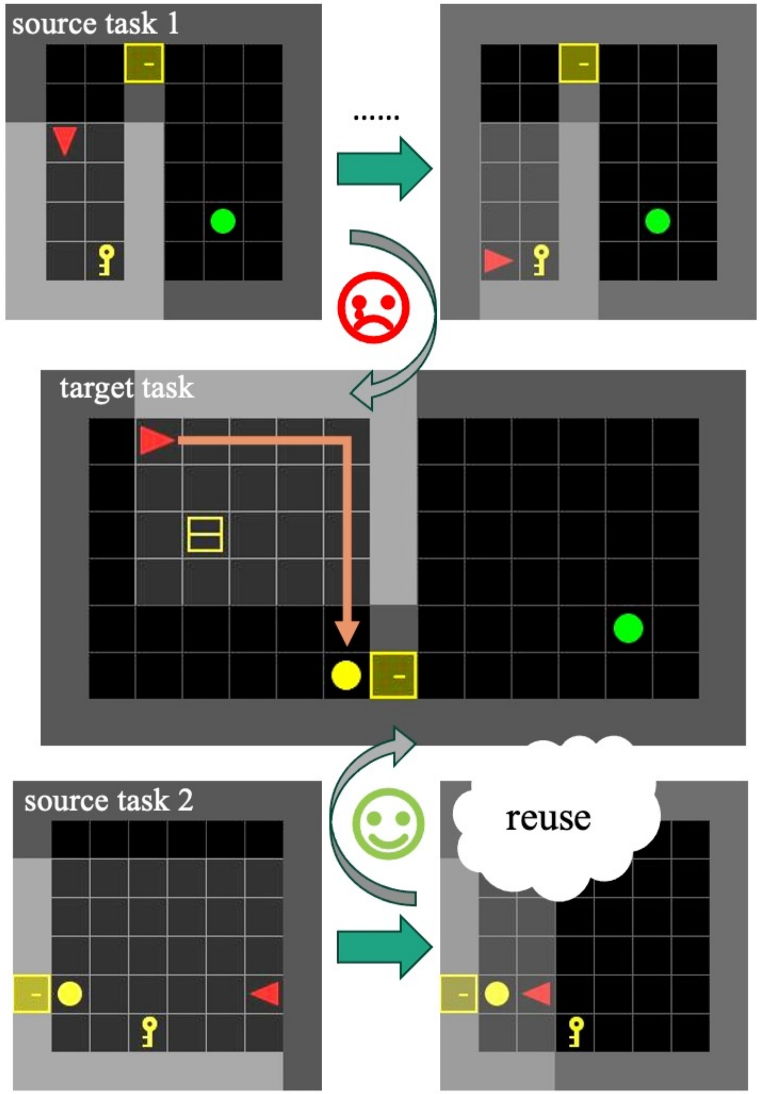




(a) BlockedBoxUnlockPickup (b) BlockedBoxPlaceGoal

Key question: how to find logic similarities behind different tasks?

Different Source Tasks at Different Times



[gt_goal, **gt_key**, gt_door]: −¬has(X), is_agent(X), ¬is_open (Y), is_door(Y)
[**pick**, drop, toggle]: −at(X), is_key(X), ¬have (Y), is_agent(Y)
[gt_goal, gt_key, **gt_door**]: −has_key(X), is_agent(X), ¬is_open (Y), is_door(Y)
[pick, drop, **toggle**]: −at(X), is_door(X), has_key (Y), is_agent(Y)
[**gt_goal**, gt_key, gt_door]: −has_key(X), is_agent(X), is_open (Y), is_door(Y)
[pick, **drop**, toggle]: −at(X), is_goal(X), has_key (Y), is_agent(Y)
[**gt_goal**, gt_key, gt_door]: −¬has(X), is_agent(X), is_open (Y), is_door(Y)
[**pick**, drop, toggle]: : −at(X), is_*goal*(X), ¬have (Y), is_agent(Y)

State: ¬has(X) , is_agent(X),is_blocked (Y), is_door(Y), ¬has_key(Z) , is_env(Z)
Goal: gt_goal, gt_key, gt_door, **gt_blockage**, gt_box

State: at(X), is_blockage(X), ¬have (Y), is_agent(Y)
Goal: **pick**, drop, toggle

[**gt_blockage**, gt_key, gt_door]: −¬has(X), is_agent(X), is_blocked (Y), is_door(Y)
[**pick**, drop, toggle]: −at(X), is_blockage(X), ¬have (Y), is_agent(Y)
[gt_blockage, **gt_key**, gt_door]: −has_blockage(X), is_agent(X), ¬is_open (Y), is_door(Y)
[pick, **drop**, toggle]: −at(X), is_door(X), has_blockage (Y), is_agent(Y)
[gt_blockage, **gt_key**, gt_door]: −¬has(X), is_agent(X), ¬is_open (Y), is_door(Y)
[**pick**, drop, toggle]: −at(X), is_key(X), ¬have(Y), is_agent(Y)
[gt_blockage, gt_key, **gt_door**]: −has_key(X), is_agent(X), ¬is_open (Y), is_door(Y)
[pick, drop, **toggle**]: : −at(X), is_door(X), has_key (Y), is_agent(Y)

Methodology

- ProgrAm guiDeD poLicy rEuse (PADDLE)
  - Hybrid Decision Model
    - Learn the primitive policy as well as the logic
  - Dual Similarity Measurement
    - Measure the similarity and performance between source and target
  - Policy Reuse Module
    - Learn the target policy by reusing the suitable source policy

The Intelligent Robot Learning Laboratory

PADDLE – Hybrid Decision Model

- Integrate ILP with DRL in a hierarchical manner
  - Decompose the problem into several sub-problems
  - High-level: Using ILP to learn logic rules [1-2]
  - Low-level: Using DRL to solve each sub-problems
- ILP
  - Objective: learn a set of first-order logic rules (clauses)
  - Each of which is composed of: Head atoms and Body atoms

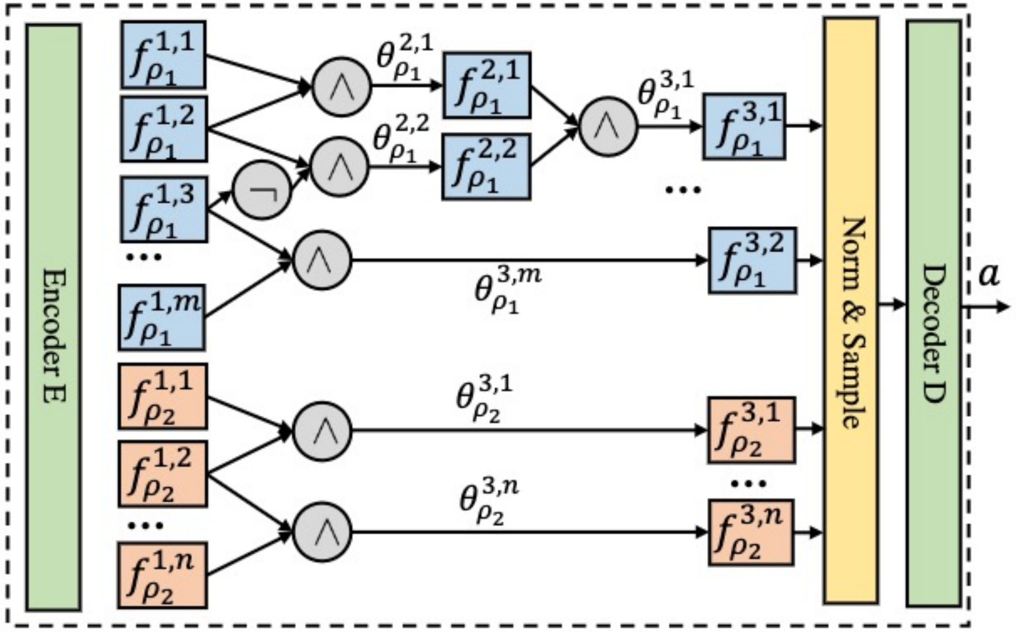
$$\mathbf{pick}\,():-\mathbf{at}(\mathbf{X}),\mathbf{is\_blockage}(\mathbf{X}),\neg\mathbf{have}\,(\mathbf{Y}),\mathbf{is\_agent}(\mathbf{Y})$$

[1] GALOIS: Boosting Deep Reinforcement Learning via Generalizable Logic Synthesis. NeurIPS 2022.
[2] Neural logic reinforcement learning. ICML 2019.

PADDLE – Hybrid Decision Model

➢ DILP

- Perform the deduction of the atoms using weights θ_p associated with possible clauses
- Learn the weight θ_p via policy gradient algorithms



(0.98) gt_blockage ():-¬has(X), is_agent(X), is_blocked (Y), is_door(Y)
(0.96) pick ():-at(X), is_blockage(X), ¬have (Y), is_agent(Y)
(1.00) gt_door ():-has_blockage(X), is_agent(X), ¬is_open (Y), is_door(Y)
(0.98) drop ():-at(X), is_door(X), has_blockage (Y), is_agent(Y)
(1.00) gt_key ():-¬has(X), is_agent(X), ¬is_open (Y), is_door(Y)
(1.00) pick ():-at(X), is_key(X), ¬have(Y), is_agnet(Y)
(0.95) gt_door ():-has_key(X), is_agent(X), ¬is_open (Y), is_door(Y)
(1.00) toggle ():-at(X), is_door(X), has_key (Y), is_agent(Y)

[1] GALOIS: Boosting Deep Reinforcement Learning via Generalizable Logic Synthesis. NeurIPS 2022.
[2] Neural logic reinforcement learning. ICML 2019.

PADDLE – Dual Similarity Measurement

Logic Similarity Function

Algorithm 1: Coincidence Degree Function Ψ

For clause in D_t**:**
  IF body atom of clause never appeared: new $c_t^{len(c_t)+1}$ **add clause**
  ELSE: c_t^k **add clause (**c_t^k **contains clauses corresponding to body atoms)**
For $c_{\mathcal{T}}^i$ **in** $c_{\mathcal{T}}$
  For j=1 in n
    max_similarity, max_c **= 0, [ ]**
    For $c_{\mathcal{S}_j}^k$ **in** $c_{\mathcal{S}_j}$
      $\psi(c_{\mathcal{T}}^i, c_{\mathcal{S}_j}^k) = \gamma_{\textbf{head}} * \cap_{head(c_{\mathcal{T}}^i),head(c_{\mathcal{S}_j}^k)} + \gamma_{\textbf{body}} * \cap_{body(c_{\mathcal{T}}^i),body(c_{\mathcal{S}_j}^k)}$
      IF $psi(c_{\mathcal{T}}^i, c_{\mathcal{S}_j}^k)$ **> max_similarity: max_similarity, max_**c **=** $psi(c_{\mathcal{T}}^i, c_{\mathcal{S}_j}^k)$, $[c_{\mathcal{S}_j}^k]$
      IF: $psi(c_{\mathcal{T}}^i, c_{\mathcal{S}_j}^k)$ **= max_similarity: max_**c **add** $c_{\mathcal{S}_j}^k$
  $\Psi(c_{\mathcal{T}}^i)$ **add (max_similarity, max_**c**)**

Performance Function

$$\Phi(c_k^j) = G(c_k^j) + \gamma_c H(\pi_{c_k^j}), k \in [\mathcal{S}_1, ..., \mathcal{S}_n, \mathcal{T}], j \in [1, n_k].$$

$$\gamma_c = \gamma min(0, clip(-G(c_k^j, \epsilon)))$$

- ➢ Dual Similarity

$$\Lambda(c_k^j, c_{\mathcal{T}}^i) = \psi(c_k^j, c_{\mathcal{T}}^i) + \Phi(c_k^j), k \in [\mathcal{S}_1, ..., \mathcal{S}_n, \mathcal{T}], j \in [1, n_k], i \in [1 : n_{\mathcal{T}}]$$

The Intelligent Robot Learning Laboratory

PADDLE - Policy Reuse Module

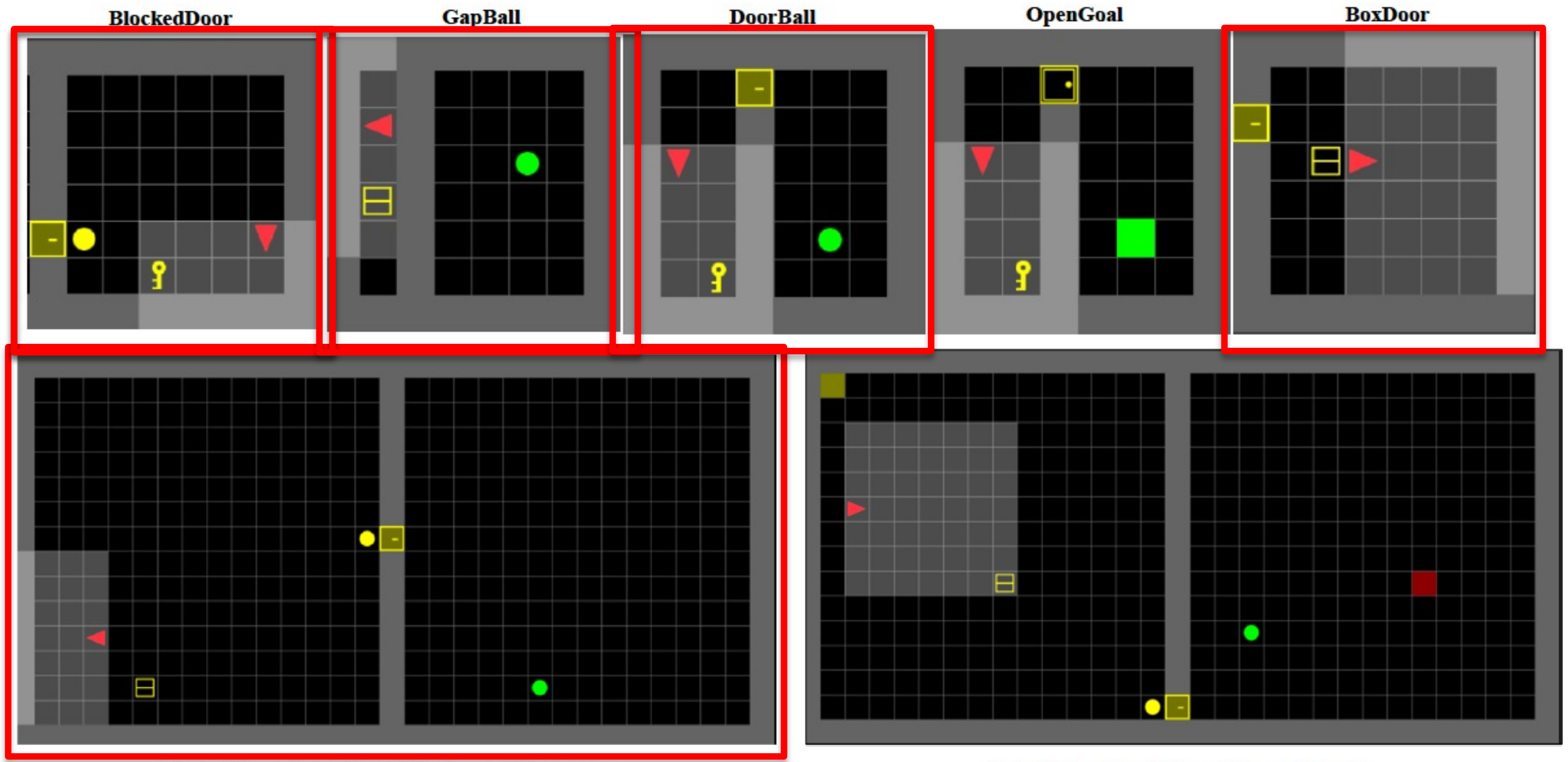
- A set of pre-trained source policies $\{\pi_1, \pi_2, \cdots \pi_n\}$, target policy π_T
- Select the policy with the highest Dual Similarity

Algorithm 2: PADDLE

Require: Source policies $\Pi = \{\pi_1, \ldots, \pi_n\}$**, hyper-parameters** $\gamma_{\text{head}}, \gamma_{\text{body}}$
Initialize target policy $\pi_{\mathcal{T}}$ **and performance function** Φ **(initialize with a constant)**
Get Coincidence Degree Function Ψ
For each episode do:
  When select sub-goal from high-level module do:
    Obtain the $c^c_{\mathcal{T}}$ **according to the clause activated**
    similarity_list, source_c_list = $\Psi(c^c_{\mathcal{T}})$
    For i=1 to n do:
      W(π_i)**=**$\Lambda(c^{max}_{S_i}, c^c_{\mathcal{T}})$
    W$(\pi_{\mathcal{T}})$**=**$\Lambda(c^c_{\mathcal{T}}, c^c_{\mathcal{T}})$
    π_g**=**$argmax_{\pi \in \Pi}$**W**(π)
    replace the corresponding module, Collect samples $S = (s, a, s', r)$ **using** π_g
    For the sample set T_g **obtained after each** π_g**:**
      $G(c_g)$=$average(sum(R(T_g)))$
  Update $\pi_{\mathcal{T}}$ **using** $S = (s, a, s', r)$

Experiments

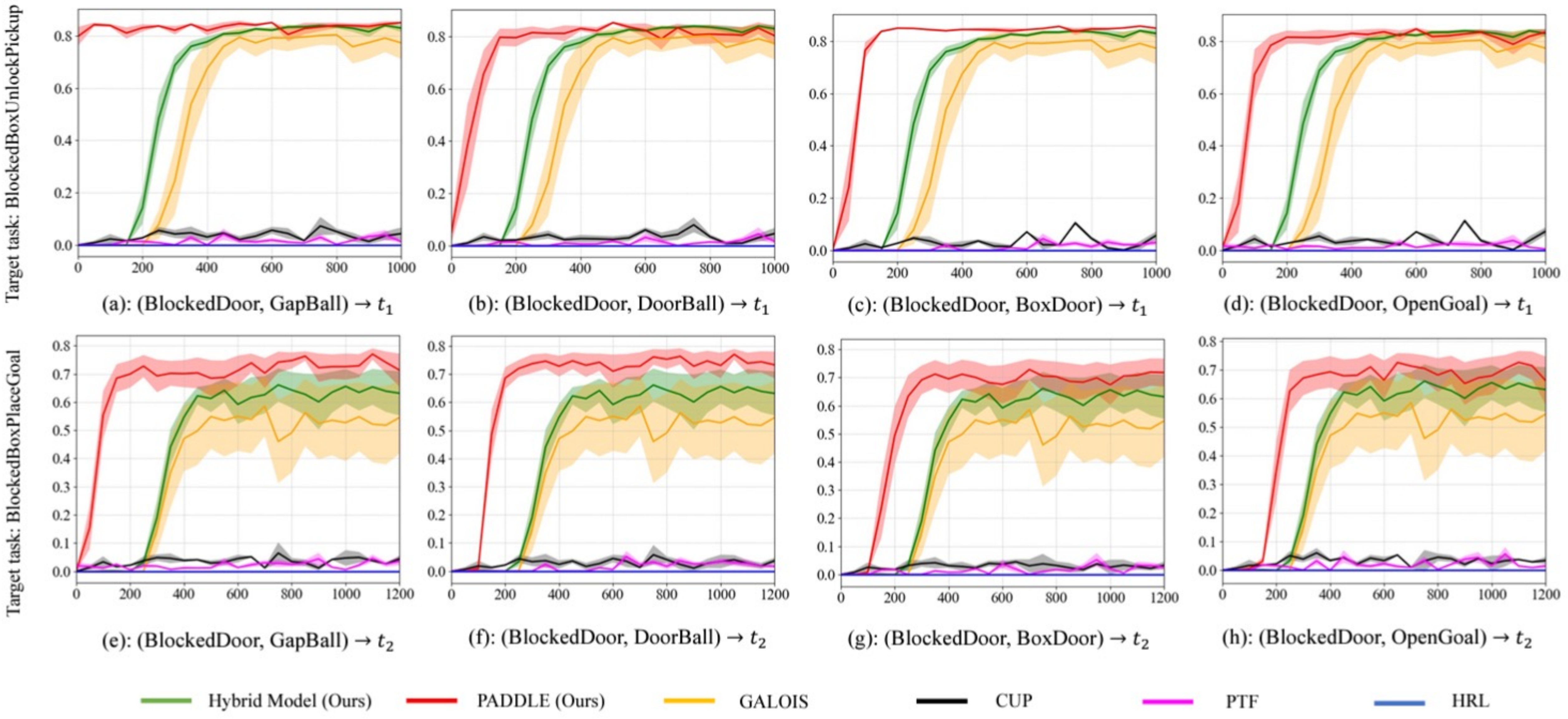
- Environments
  - Minigrid
  - Pointmaze
- Baselines
  - PPO [1]
  - CUP [2]
  - PTF [3]
  - HDQN [4] (modified to HPPO)



(a) BlockedBoxUnlockPickup (b) BlockedBoxPlaceGoal

[1] Proximal Policy Optimization. ArXiv:1707.06347.
[2] CUP: Critic-Guided Policy Reuse. NeurIPS 2022.
[3] Efficient deep reinforcement learning via adaptive policy transfer. IJCAI 2020.
[4] Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. NIPS 2016.

Results

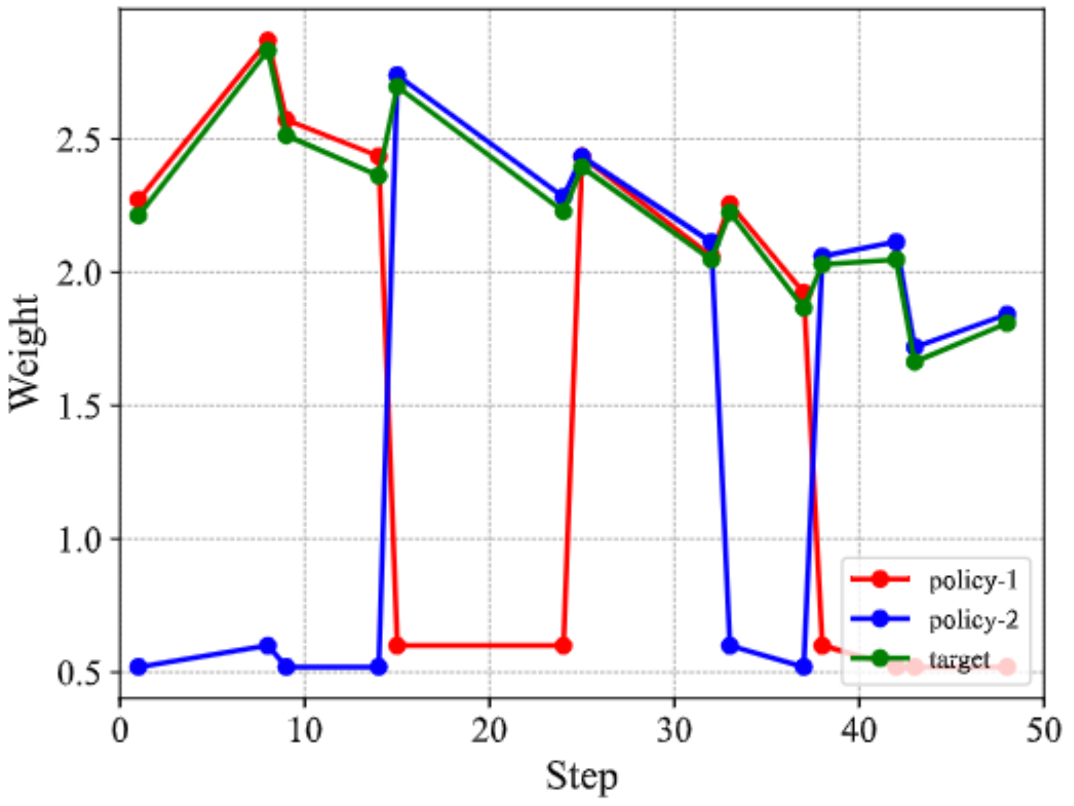


Target task: BlockedBoxUnlockPickup
Target task: BlockedBoxPlaceGoal
(a): (BlockedDoor, GapBall) → t_1
(b): (BlockedDoor, DoorBall) → t_1
(c): (BlockedDoor, BoxDoor) → t_1
(d): (BlockedDoor, OpenGoal) → t_1
(e): (BlockedDoor, GapBall) → t_2
(f): (BlockedDoor, DoorBall) → t_2
(g): (BlockedDoor, BoxDoor) → t_2
(h): (BlockedDoor, OpenGoal) → t_2
Hybrid Model (Ours)
PADDLE (Ours)
GALOIS
CUP
PTF
HRL

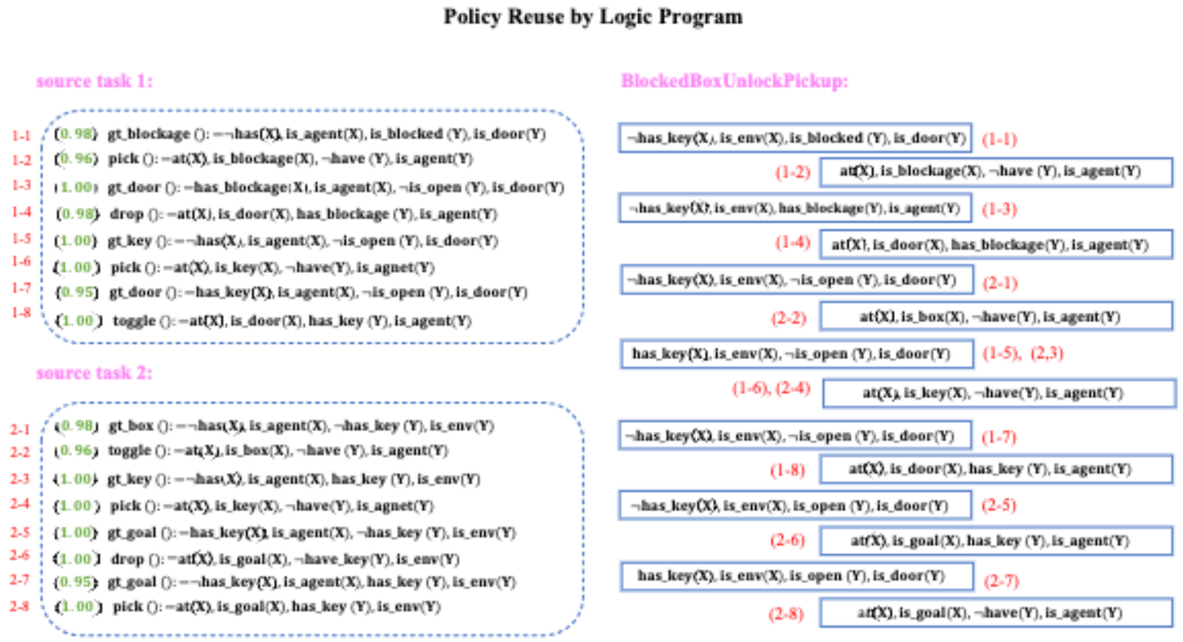
The Intelligent Robot Learning Laboratory

Ablations

- Reuse process visualization



(a)



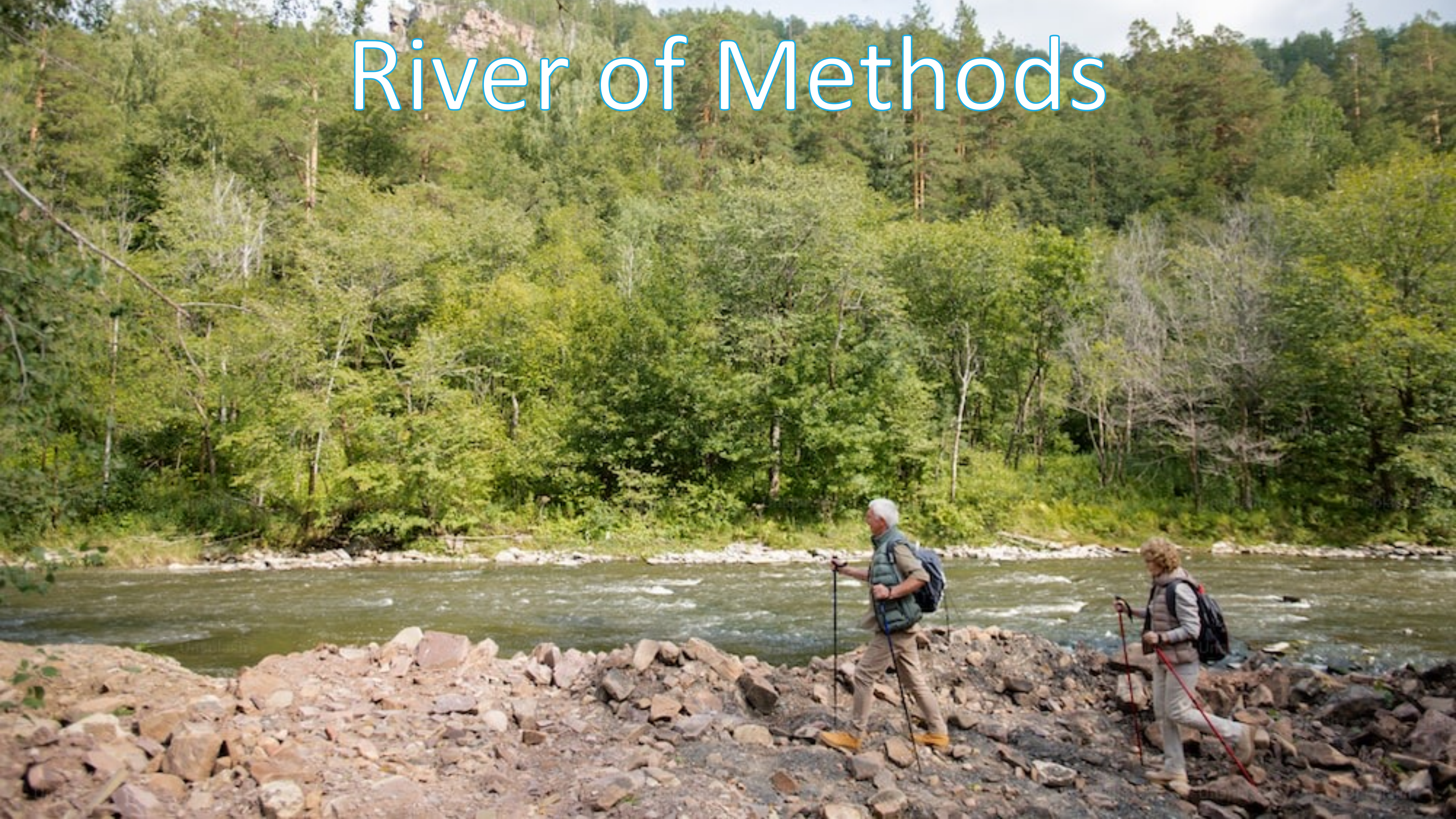
(b)

Conclusions

- PADDLE: measure the logic similarities between tasks for transfer
- Is the dual similarity measurement helpful? Yes!
  - Does the logic similarity help to improve the policy reuse performance? Yes!
  - Does PADDLE learn the optimal performance with non-optimal policies? Yes!
- In the future:
  - Non-expert humans generate clauses?
  - Learn clauses from scratch?
  - Correct if not perfect?


Forest of Research

River of Methods



PADDLE to Reuse Past Knowledge