Switching Successor Measures
for
Hierarchical Zero-Shot Reinforcement Learning

Stefan Stojanovic & Alexandre Proutiere

KTH Royal Institute of Technology, Stockholm, Sweden

RLBRew Workshop @ RLC 2026

Reinforcement Learning Beyond Rewards

Towards Scalable General-Purpose Agents

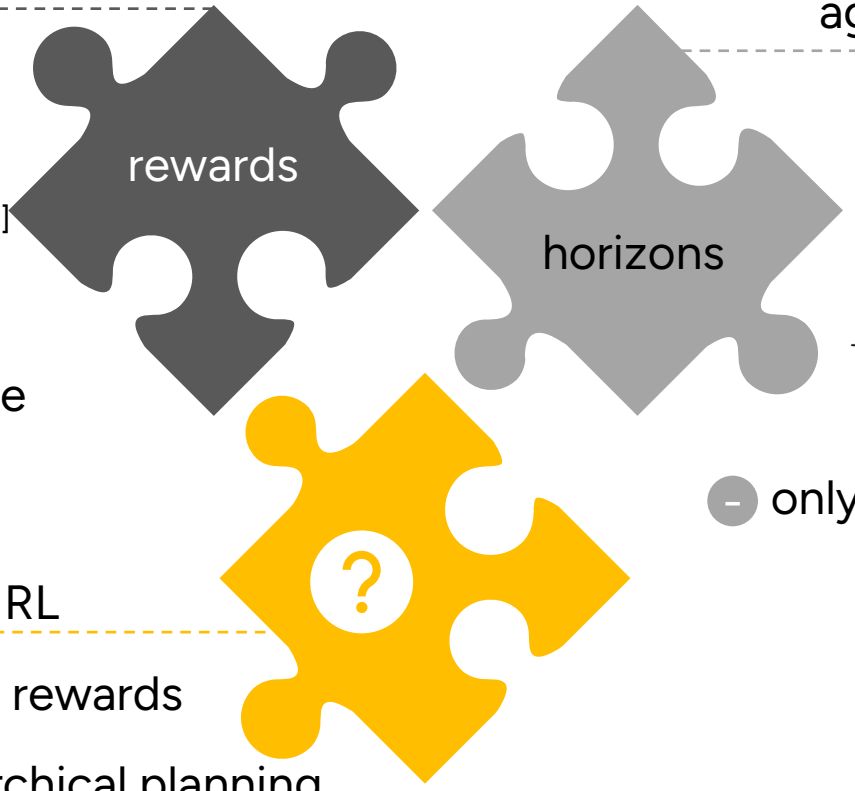
How can we train agents that maximize ***any*** reward?

successor representations decouple dynamics from rewards

e.g., FB model of successor measures[1]

$$M^{\pi_z}_{s,a}(s') \approx F(s,a,z)^\top B(s')$$

- cannot hierarchically decompose long-horizon tasks



How can we train ***hierarchical*** agents to solve long-horizon tasks?

decompose long-horizon tasks into subtasks

e.g., HIQL for goal-conditioned RL[2]

$$V(s;g) \implies \pi_h(w|s,g), \pi_\ell(a|s,w)$$

- only single-goal tasks can be decomposed
- fixed subgoal horizon

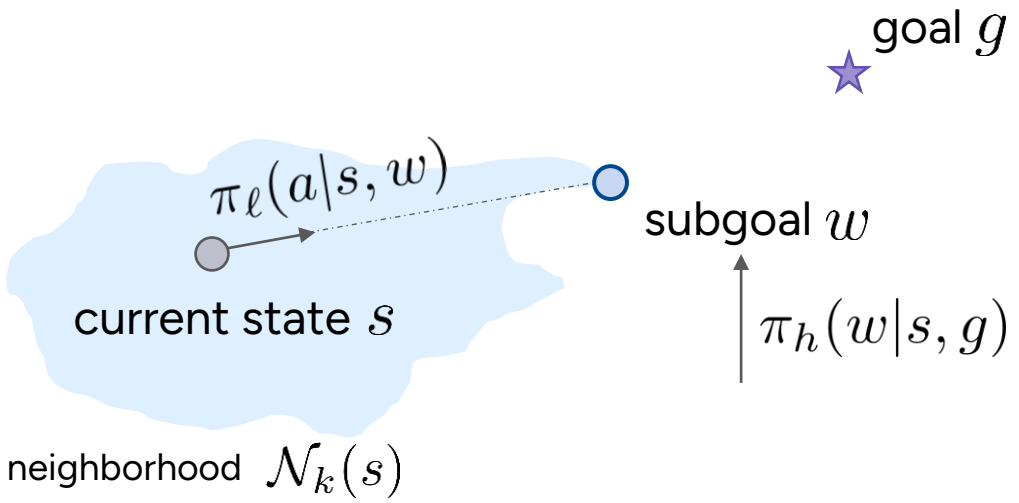
Hierarchical zero-shot RL

- generalizes to arbitrary rewards
- enables adaptive hierarchical planning

[1] Ahmed Touati, Jérémy Rapin, Yann Ollivier. "Does zero-shot reinforcement learning exist?" *ICLR* (2023)
[2] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, Sergey Levine. "HIQL: Offline goal-conditioned RL with latent states as actions." *NeurIPS* (2023)

Policy Switching: Conceptual Framework

- HIQL decomposes goals into easier-to-reach subgoals
  - subgoals are reachable within *k* steps (hyperparameter)
  - select the subgoal with the highest value $V(w; g)$
- Can we generalize this beyond fixed-horizon goal reaching?
  - Two-stage hierarchical execution for general tasks
  - No fixed *k*-step horizon



Stage 1: **Follow subgoal policy**
collect local rewards and navigate towards the intermediate subgoal

Hitting time switch
automatically switch policies when the subgoal is reached

Stage 2: **Follow task policy**
act efficiently to maximize the global task reward

- Replan at every timestep – the selected subgoal need not be reached

Policy Switching: Mathematical Framework

- Recall definition of successor measures: $M_s^\pi(s') = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s'\} | s_0 = s\right]$
- Why hitting time policy switching?
  - Non-Markovian hitting time successor measure maps directly to Markovian successor measures
- Main result: Successor measure decomposition

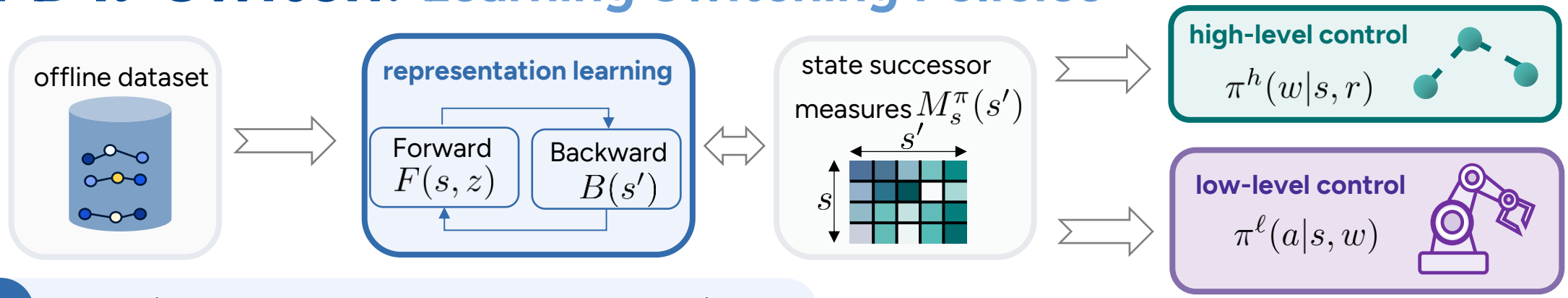
$$M_s^{\pi_w \to \pi}(s') = M_s^{\pi_w}(s') + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}\left(M_w^{\pi}(s') - M_w^{\pi_w}(s')\right)$$

successor measure of switching policy	subgoal policy term	switch at hitting time	post-switching state distribution	pre-switching correction term

- Corollary for advantage fcn: $A_s^{\pi_w \to \pi}(r) = V^{\pi_w}(s; r) + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}\left(V^{\pi}(w; r) - V^{\pi_w}(w; r)\right) - V^{\pi}(s; r)$
- High-level policy = maximization over subgoals of this expression
- Compare with HIQL that effectively maximizes: $M_s^{\pi_\beta}(w) V^{\pi}(w; r)$
- Related to jumpy world models[3]: Appendix A.3

[3] Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Marc G Bellemare, Alessandro Lazaric, and Ahmed Touati. "Compositional planning with jumpy world model" *ICML* (2026)

FB π-Switch: Learning Switching Policies



Step 1: Learning state-successor representations

$$M_s^{\pi_w \to \pi}(s') = M_s^{\pi_w}(s') + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}\left(M_w^{\pi}(s') - M_w^{\pi_w}(s')\right)$$

- How do we learn $M_s^\pi(s')$ for all states and policies?
  - standard FB: learns $M_{s,a}^\pi(s')$ jointly with policy
  - decouple representation and policy learning, following the HIQL approach
  - action-free FB model: still meaningful since $V^{\pi_z}(s;r) = \sum_{s'} M_s^{\pi_z}(s')r(s') \approx F(s,z)^\top \overbrace{\sum_{s'} B(s')r(s')}^{z_r}$
  - IQL-based representation objective[4] (similar to ICVF[5]):

$$\mathcal{L}_{\text{Rep}}(F,B) = \mathbb{E}_{(s_t,s_{t+1}),s',z}\left|\tau - \mathbf{1}\{\text{TD}(F(s_t,z)^\top z) < 0\}\right|\text{TD}(F(s_t,z)^\top B(s'))^2$$

[4] Iya Kostrikov, Ashvin Nair, and Sergey Levine. "Offline reinforcement learning with implicit Q-learning". *ICLR* (2022)
[5] Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. "Reinforcement learning from passive data via latent intention". *ICML* (2023)

FB π-Switch: Learning Switching Policies

Step 2: Learning high-level policy

- Estimate the switching advantage function using the learned representations:

$$A_{\mathrm{FB}}(s, w, z) := F(s, z_w)^\top z + \frac{F(s, z_w)^\top z_w}{F(w, z_w)^\top z_w} \Big(F(w, z) - F(w, z_w) \Big)^\top z - F(s, z)^\top z$$

- We use AWR-based loss[6] to favor subgoals with high switching advantage:

$$\mathcal{L}_{\mathrm{Plan}}(\pi^h) = -\mathbb{E}_{s,w\sim\mathcal{D},z\sim\mathcal{Z}} \Big[\exp\Big(\beta A_{\mathrm{FB}}(s, w, z)\Big) \log \pi^h(z_w \mid s, z) \Big].$$

Step 3: Learning low-level policy

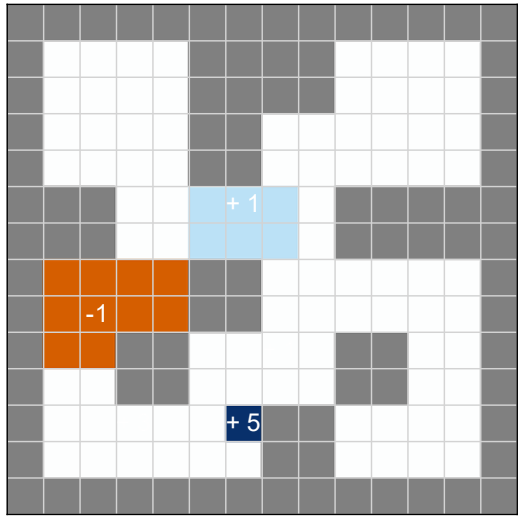
- AWR-based loss maximizing the one-step value improvement:

$$\mathcal{L}_{\mathrm{Act}}(\pi^\ell) = -\mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D} \\ z\sim\mathcal{Z}}} \Big[\exp\Big(\beta(F(s_{t+1}, z)^\top z - F(s_t, z)^\top z)\Big) \log \pi^\ell(a_t \mid s_t, z) \Big].$$

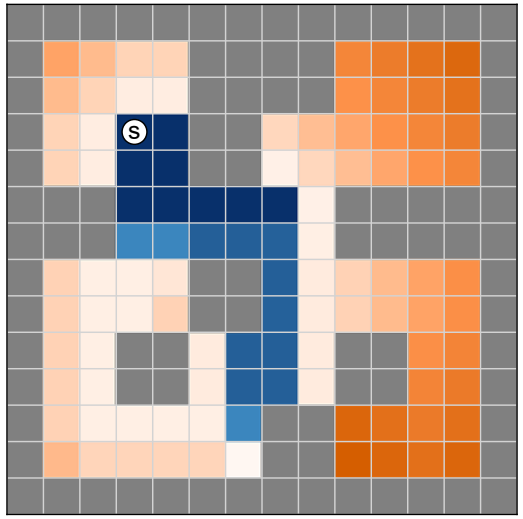
[6] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning". arXiv (2019)

Experiments: Switching Framework for Discrete Mazes

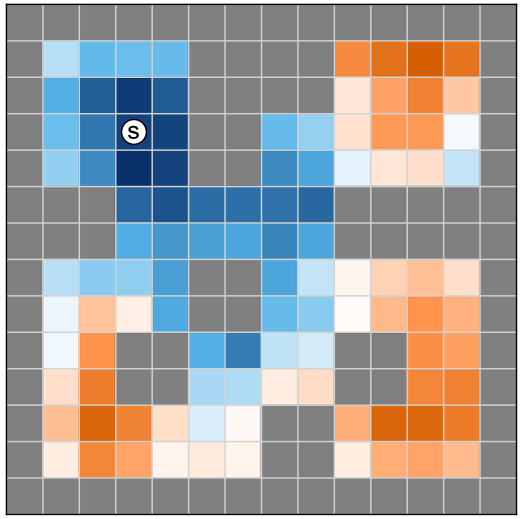
$r(s)$



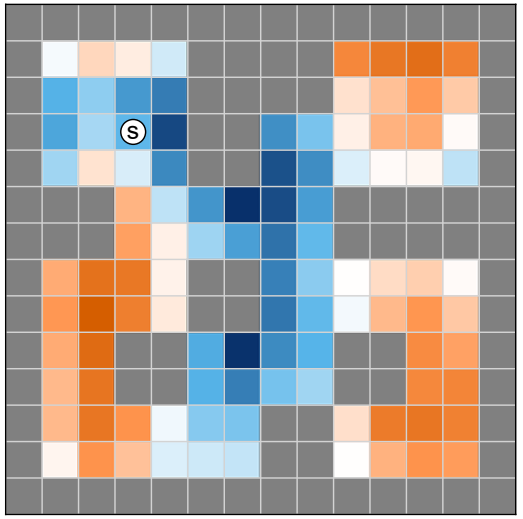
ground truth



learned estimate

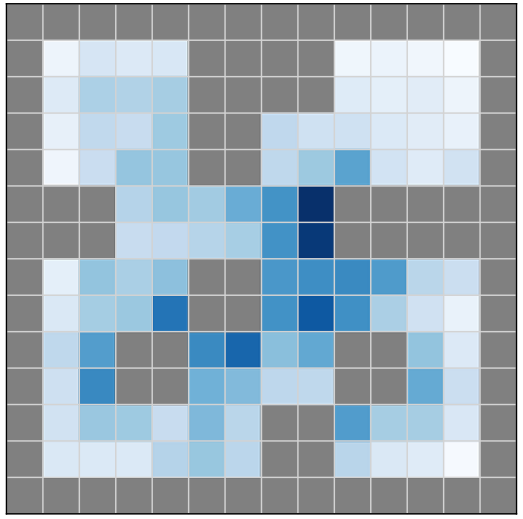


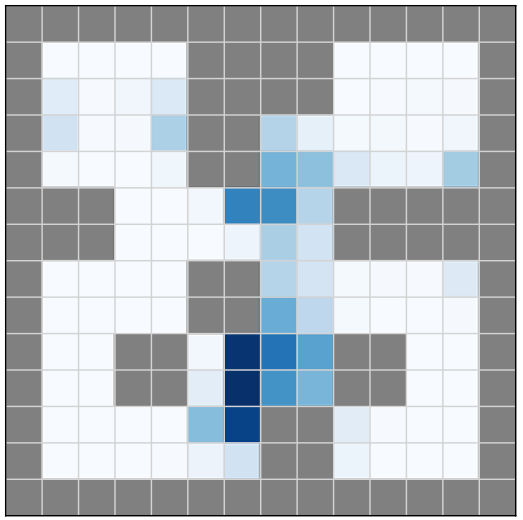
removing pre-switching correction term



$A_s^{\pi_w \to \pi^\star}(r)$

- Switching advantage: highlights states along the optimal path
- Pre-switch correction: tricky to estimate — in environments with lazy states, it reduces to an indicator function
- Empirical finding: skipping the correction term gives much better performance, even in simple discrete mazes

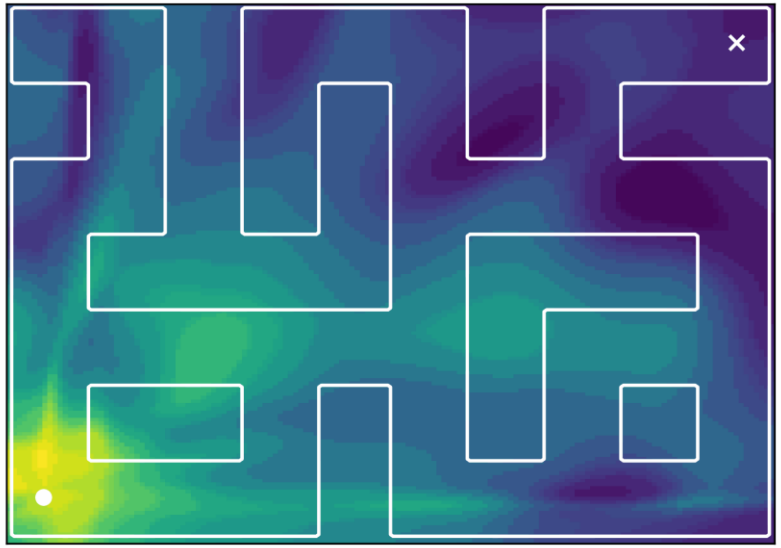




$\pi^h(w|s,r)$

Experiments: Switching Framework for Goal-Conditioned RL

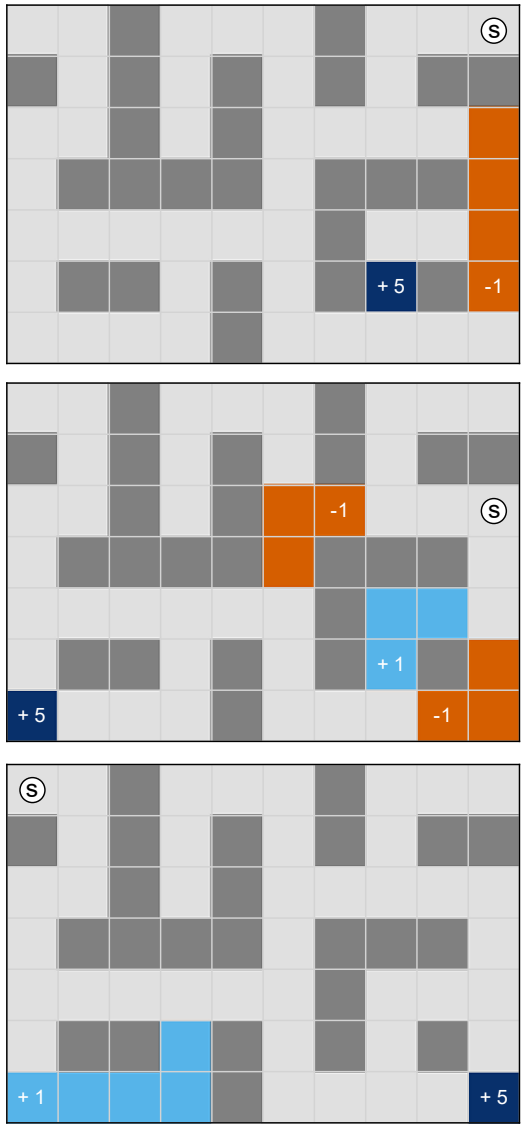
- Experiments in AntMaze environments (Medium, Large, Giant, Teleport)
  - Why Antmaze? Interpretable, yet challenging, high-dimensional
  - Strong FB baselines available
  - Key assumption: the underlying FB model must learn successor measures sufficiently well in the first place



AntMaze	HIQL		ICVF	One-Step FB	FB		FB π-Switch	
	vanilla	$-\pi^h$			vanilla	$+\pi^h$	vanilla	$-\pi^h$
Medium	93 ± 1	73 ± 3	46 ± 37	35 ± 7	47 ± 1	50 ± 12	87 ± 6	70 ± 4
Large	73 ± 6	23 ± 4	28 ± 3	23 ± 18	21 ± 5	20 ± 5	66 ± 9	37 ± 7
Giant	5 ± 3	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	1 ± 1	1 ± 1
Teleport	42 ± 4	32 ± 6	23 ± 6	8 ± 2	25 ± 7	13 ± 7	40 ± 6	29 ± 4

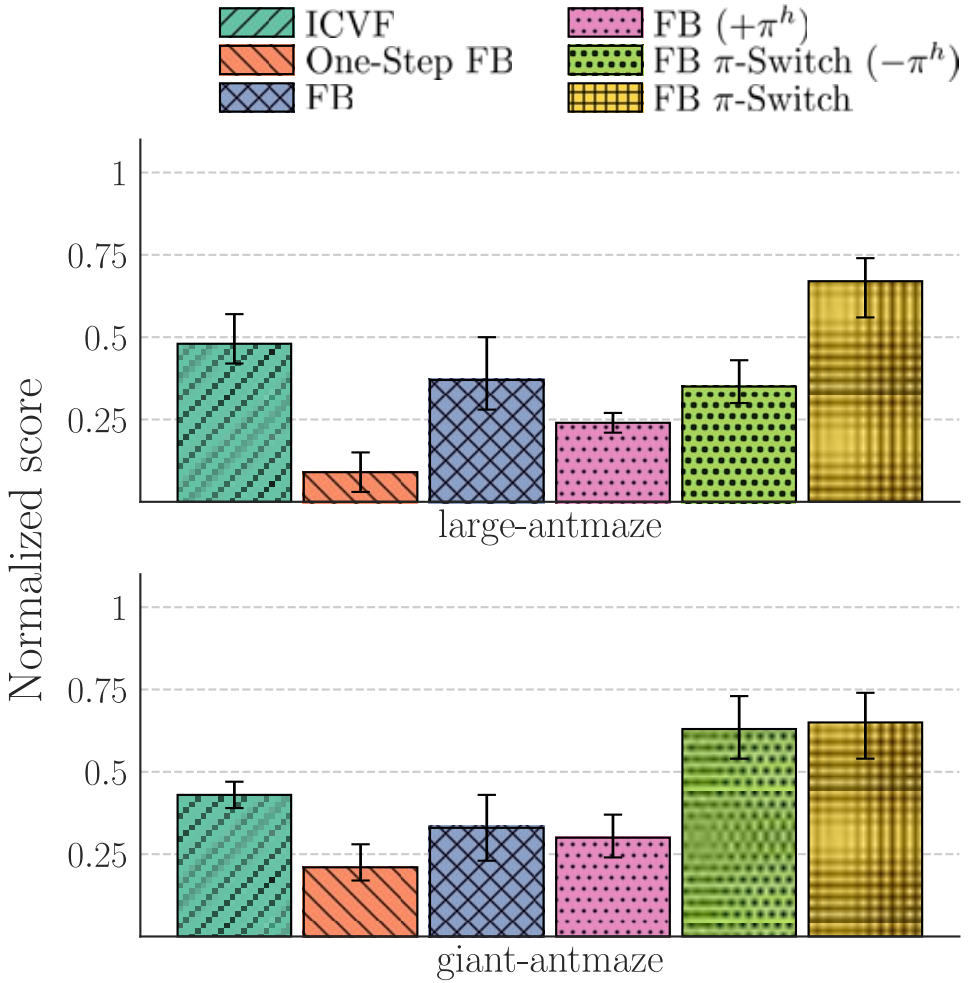
Experiments: Switching Framework for General Tasks

reward examples



- 5 hand-crafted (interpretability!) reward scenarios per environment
- Hierarchies do not help every task – Medium Antmaze is too simple to benefit
- The underlying base model is crucial
- Why doesn't standard FB perform better?
  - Lkely due to poor low-level policies.
- Why is the improvement smaller on Giant environment?
  - The underlying base FB models are weak; better training could unlock larger gains.

more



Take-aways

- **Novel hierarchical framework:**
  Hierarchical policies emerge from a *single* successor representation.
- **Beyond navigation:**
  Applies to general reward functions, not only goal-reaching.
- **Adaptive switching:**
  No fixed planning horizon – policies switch at the subgoal hitting time.
- **Strong perfromance:**
  Competitive with HIQL and stronger than non-hierarchical baselines.

Open questions

- **When does hierarchy help?**
  What class of tasks benefits from hierarchical control?
- **What is the high-level policy objective learning in practice?**
  What objective is maximized when using the learned representations?
- **Is policy switching the *right* abstraction?**
  Are there better conceptual frameworks for hierarchical control?

Project webpage

https://stestokth.github.io/switching-successors/