COMPUTER SCIENCE & ENGINEERING
TEXAS A&M UNIVERSITY

Data Analytics at Texas A&M (DATA Lab)

Rank the Episodes: A Simple Approach for Exploration in Procedurally-Generated Environments

Daochen Zha, Wenye Ma, Lei Yuan, Xia Hu, Ji Liu

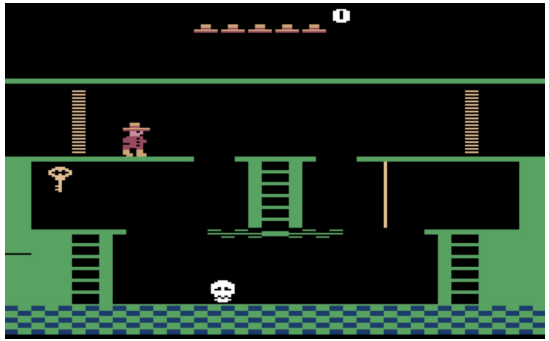
Department of Computer Science and Engineering, Texas A&M University
AI Platform, Kwai Inc.

Email: daochen.zha@tamu.edu

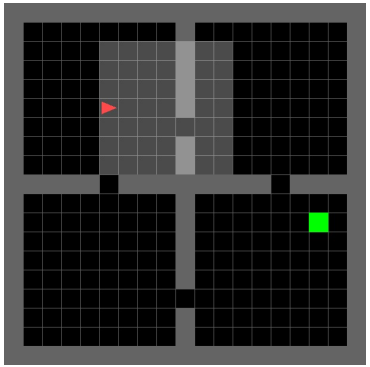
Code: https://github.com/daochenzha/rapid

Background: Exploration is an Open Challenge for RL

- *What is exploration?* ***The ability of the agents to discover novel states in the environments.***
- *Why we need exploration?* ***Learning a good policy with reinforcement learning relies on the ability of discovering the novel states in the first place (the rewards can be sparse).***
- *How we can encourage exploration?* ***The most popular method for encouraging exploration is to give intrinsic rewards based on uncertainty, e.g., count-based exploration [1], curiosity driven exploration [2], etc.***



Montezuma's Revenge

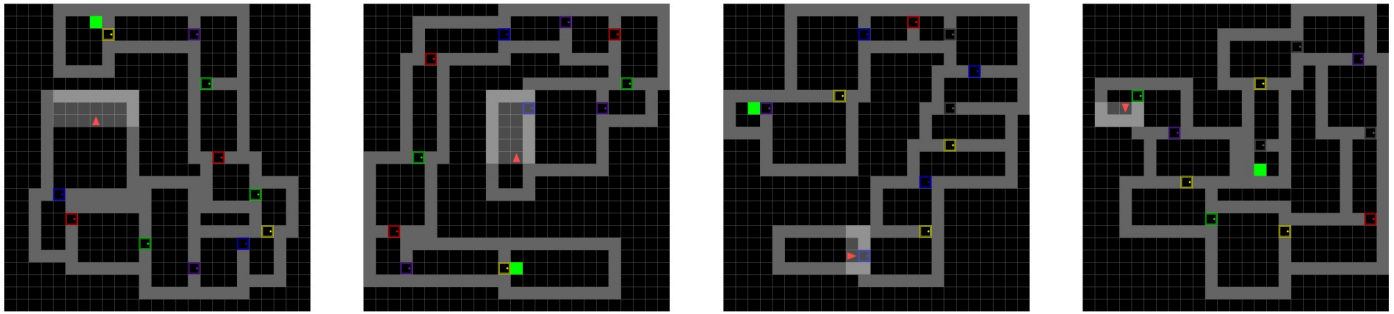


4-room

[1] Bellemare, Marc G., et al. Unifying count-based exploration and intrinsic motivation. NeurIPS 2016.
[2] Pathak, Deepak, et al. Curiosity-driven exploration by self-supervised prediction. ICML 2017.

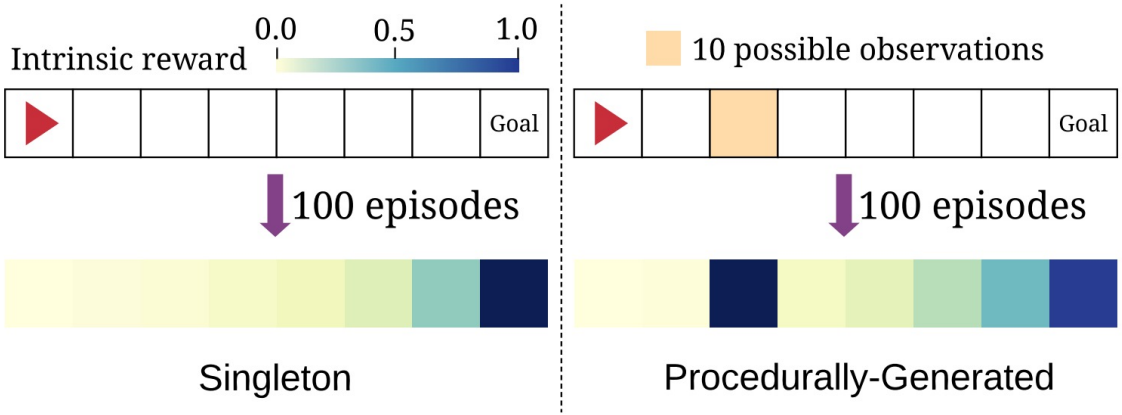
Motivation: Exploration Needs to Generalize

- *Overfitting Issue:* ***While Deep RL has made impressive progress in recent years, most of the previous studies use the same singleton environment for training and testing. Recent studies show that the agent trained in this way is susceptible to overfitting and may fail to generalize to even a slightly different environment [1][2].***
- *Procedurally-Generated Environments:* ***To address this issue, some procedurally-generated environments are proposed, where a different environment is generated in each episode.***



[1] Rajeswaran, Aravind, et al. Towards Generalization and Simplicity in Continuous Control. NeurIPS 2017.
[2] Zhang, Amy, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. arXiv 2018.

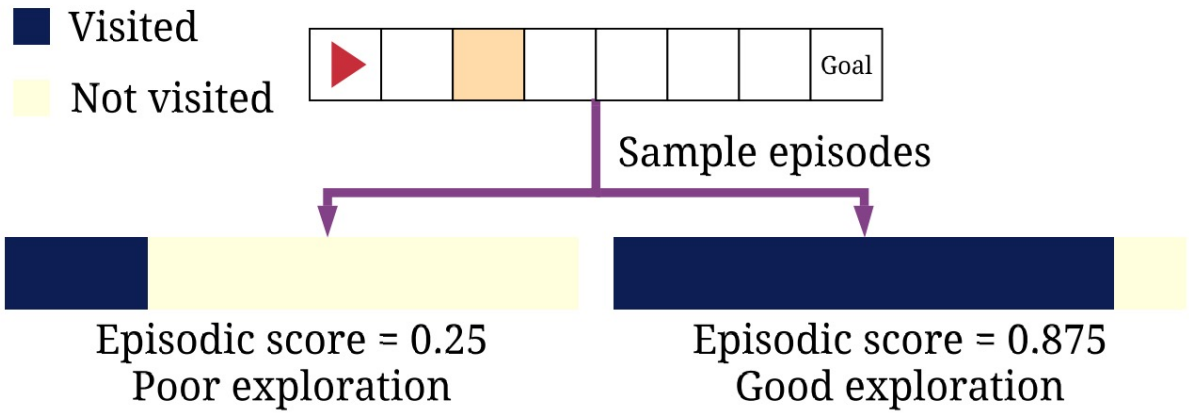
Motivation: Can Intrinsic Rewards Generalize?



- *1D Maze as a Motivating Examples:* ***The agent (red) needs to move right in each step to reach a goal.***
- *Count-Based Exploration in Procedurally-Generated Environments:* ***It is less effective because visiting a novel state does not necessarily mean a good exploration behavior***

Motivation: Episode-Level Exploration

- ***Why Humans can Generalize?*** ***Because humans often view the agent in episode-level rather than state-level. For example, one can easily tell whether an agent has well explored an environment by looking into the coverage rate of the current episode, even if the current environment is totally different from the previous ones.***



A New Exploration Strategy that Can Generalize

- *Episode-Level Exploration: We propose Exploration via Ranking the Episodes (RAPID)*

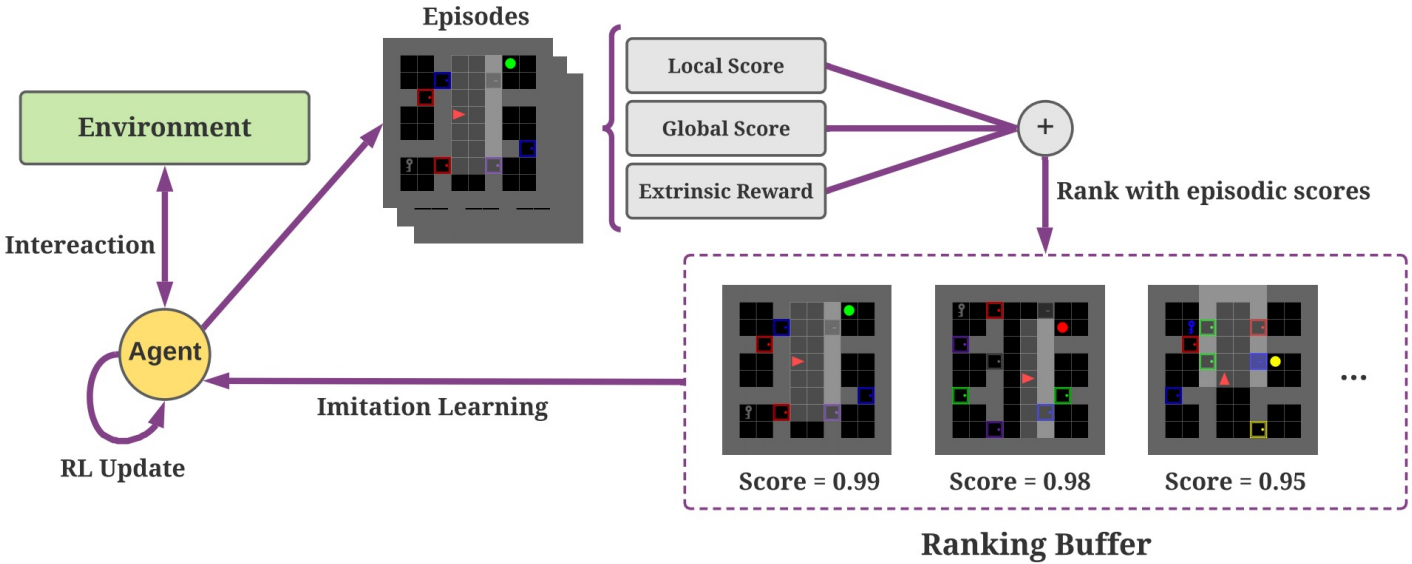


Figure 2: An overview of RAPID. The past episodes are assigned episodic exploration scores based on the local view, the global view, and the extrinsic reward. Those highly scored episodes are stored in a small ranking buffer. The agent is then encouraged to reproduce the past good exploration behaviors, i.e., the episodes in the buffer, with imitation learning.

Episodic Exploration Score

- ***Local Score:*** ***A local view of an episode. The number of district states in an episode divided by the total number of states in this episode.***

$$S_{\text{local}} = \frac{N_{\text{distinct}}}{N_{\text{total}}}$$

- ***Global Score:*** ***A score that also considers previous episodes. The mean count-based score of the current episode.***

$$S_{\text{global}} = \frac{1}{N_{\text{total}}} \sum_{s} \frac{1}{\sqrt{N(s)}}$$

- ***Episodic Exploration Score:*** ***We further consider the extrinsic rewards. The final score is the weighted sum of the three scores.***

$$S = w_0 S_{\text{ext}} + w_1 S_{\text{local}} + w_2 S_{\text{global}}$$

Episode-Level Exploration via Ranking

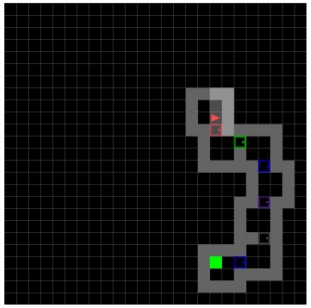
Algorithm 1 Exploration via Ranking the Episodes (RAPID)

Input: Training steps S, buffer size D, RL rollout steps T
Initialize the policy π_θ, replay buffer $\mathcal{D}$
for iteration = 1, 2, ... until convergence **do**
  Execute π_θ for T timesteps
  Update π_θ with RL objective (also update value functions if any)
  for each generated episode τ **do**
    Compute episodic exploration score S_τ based on Eq. (4)
    Give score S_τ to all the state-action pairs in τ and store them to the buffer
    Rank the state-action pairs in $\mathcal{D}$ based on their exploration scores
    if $\mathcal{D}.length > D$ **then**
      Discard the state-action pairs with low scores so that $\mathcal{D}.length = D$
    end if
    for step = 1, 2, .., S **do**
      Sample a batch from $\mathcal{D}$ and train π_θ using the data in $\mathcal{D}$ with behavior cloning
    end for
  end for
end for

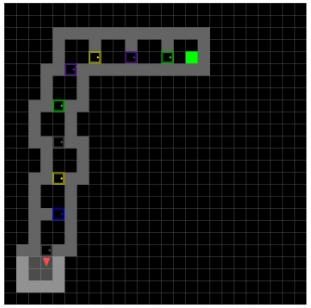
Ranking

Imitation Learning

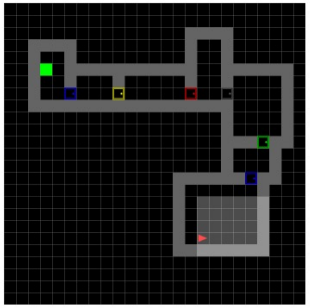
Results on MiniGrid Benchmarks



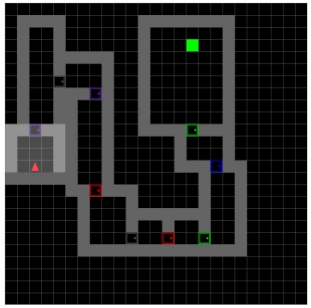
(a) MultiRoom-N7-S4



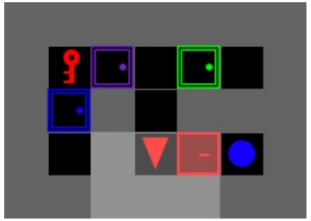
(b) MultiRoom-N10-S4



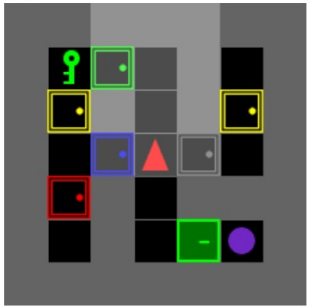
(c) MultiRoom-N7-S8



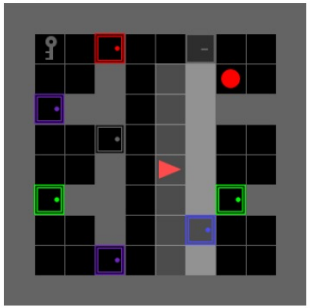
(d) MultiRoom-N10-S10



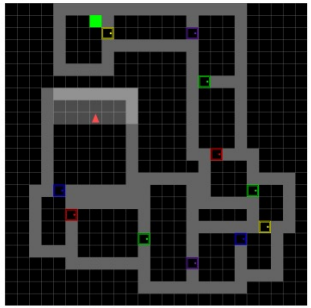
(e) KeyCorridor-S3-R2



(f) KeyCorridor-S3-R3

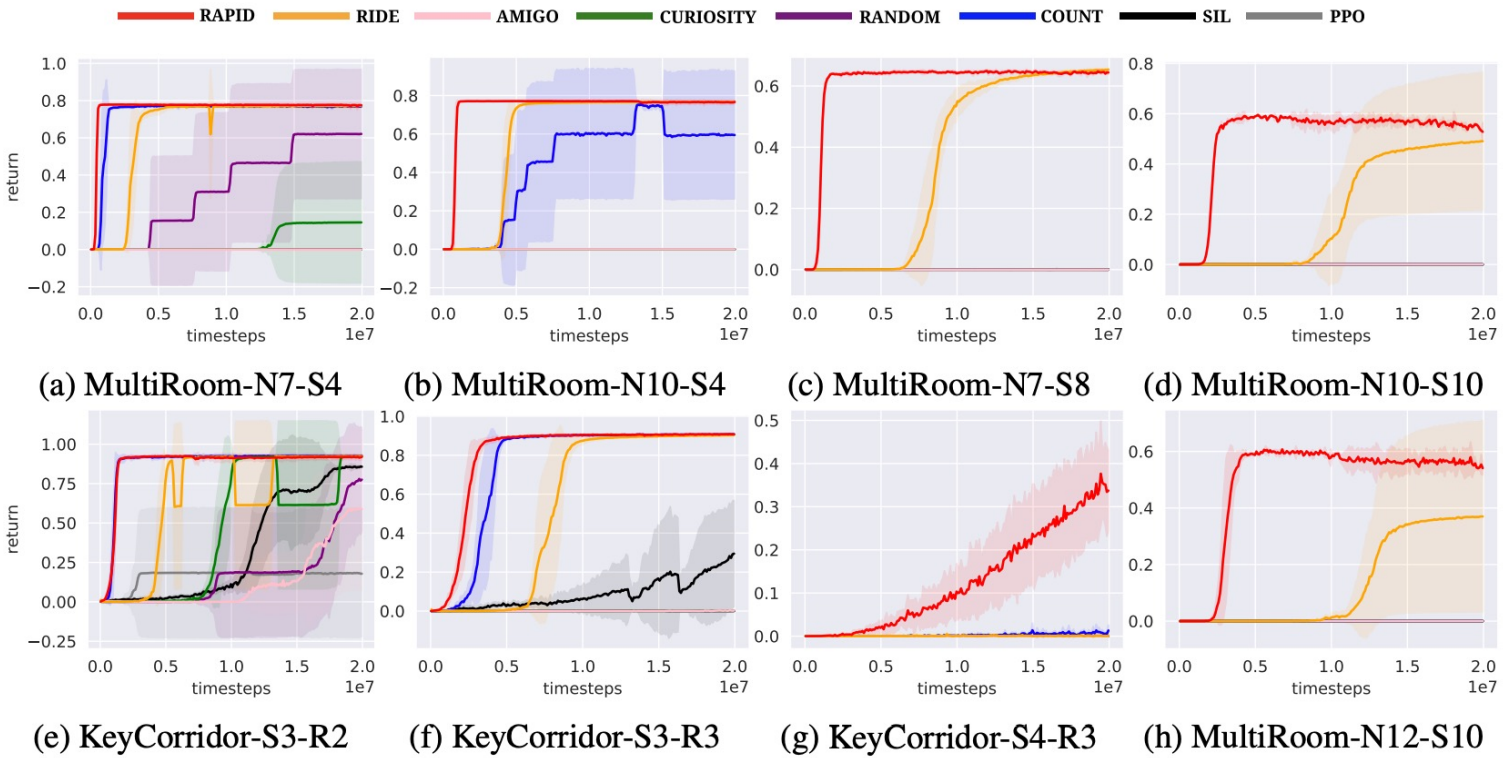


(g) KeyCorridor-S4-R3



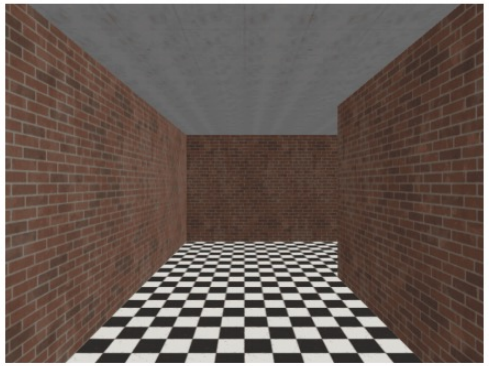
(h) MultiRoom-N12-S10

Results on MiniGrid Benchmarks

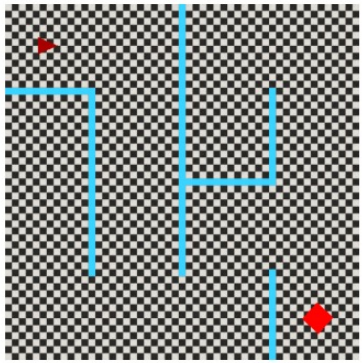


- ***Observation 1:*** ***RAPID is fast, accelerating the convergence with up to 10X.***
- ***Observation 2:*** ***RAPID is effective. Only one that works in KeyCorridor-S4-R3.***

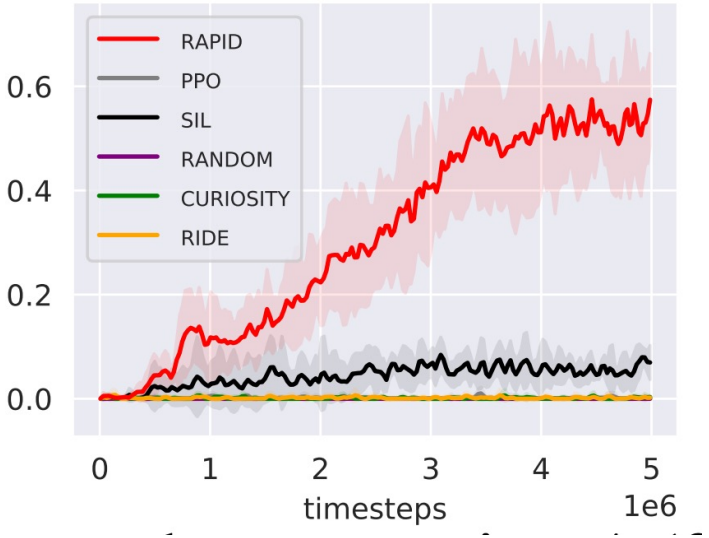
Results on 3D Naviation Task



(c) Maze



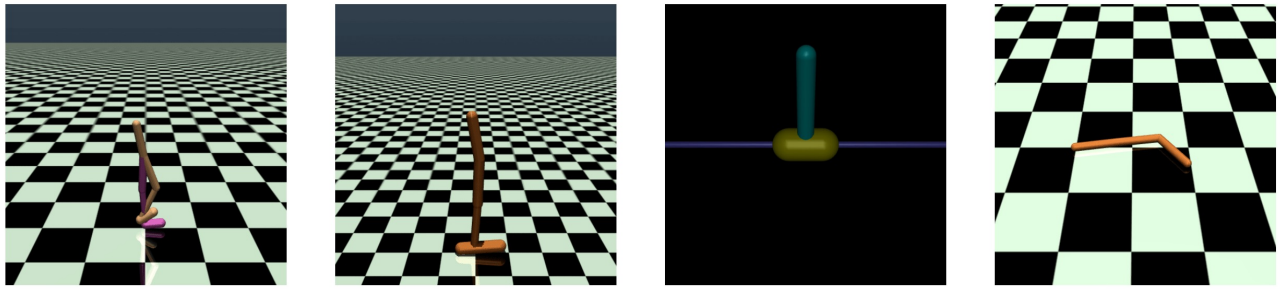
(d) Maze (top view)



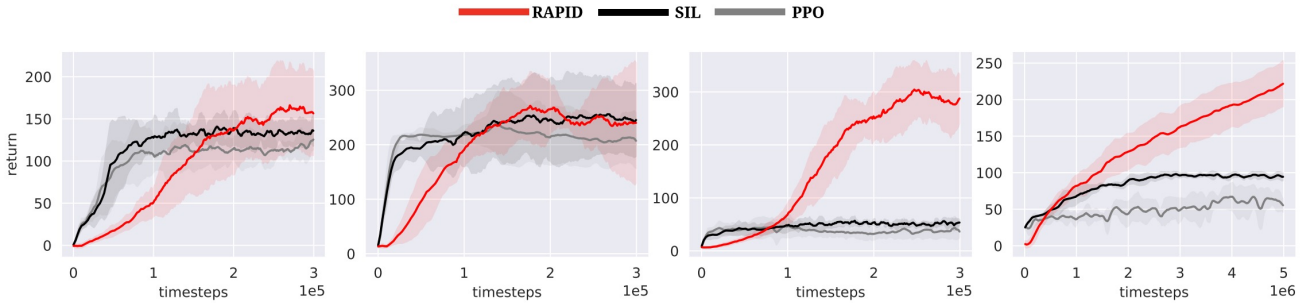
- *Observation:* ***RAPID also works with raw images as inputs.***

Results on Sparse MujoCo Tasks

- *Setting:* ***The rewards are delayed until the last timestep of an episode.***
- *Observation:* ***RAPID also works with continuous action spaces.***



(a) Walker2d (b) Hopper (c) InvertedPendulum (d) Swimmer



(a) Walker2d (b) Hopper (c) InvertedPendulum (d) Swimmer

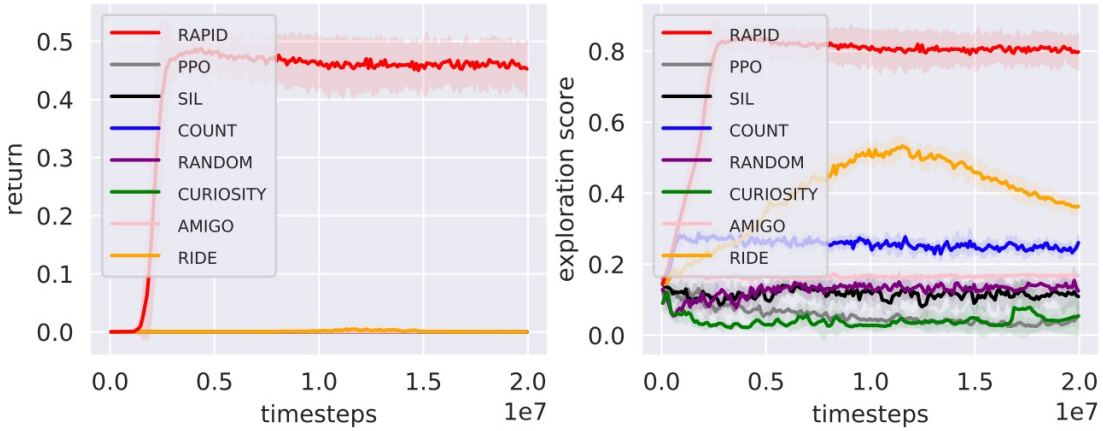
Ablation Study

- *Observation 1:* ***All the three scores and the ranking buffer helps.***
- *Observation 2:* ***Local score plays a key role (most of previous work only considers global view).***

	RAPID	w/o local	w/o global	w/o reward	w/o buffer	w/o ranking
MR-N7S4	$\mathbf{0.787 \pm 0.001}$	$\mathbf{0.787 \pm 0.000}$	$\mathbf{0.787 \pm 0.001}$	0.781 ± 0.002	0.000 ± 0.000	0.002 ± 0.002
MR-N10S4	$\mathbf{0.778 \pm 0.000}$	$\mathbf{0.778 \pm 0.000}$	$\mathbf{0.778 \pm 0.001}$	0.775 ± 0.002	0.000 ± 0.000	0.000 ± 0.000
MR-N7S8	$\mathbf{0.678 \pm 0.001}$	0.677 ± 0.002	0.677 ± 0.002	0.652 ± 0.004	0.000 ± 0.000	0.000 ± 0.000
MR-N10S10	$\mathbf{0.632 \pm 0.001}$	0.238 ± 0.288	0.630 ± 0.002	0.604 ± 0.010	0.000 ± 0.000	0.000 ± 0.000
MR-N12S10	$\mathbf{0.644 \pm 0.001}$	0.001 ± 0.001	0.633 ± 0.005	0.613 ± 0.007	0.000 ± 0.000	0.000 ± 0.000
KC-S3R2	$\mathbf{0.934 \pm 0.004}$	0.933 ± 0.002	$\mathbf{0.934 \pm 0.000}$	0.929 ± 0.003	0.018 ± 0.008	0.527 ± 0.380
KC-S3R3	$\mathbf{0.922 \pm 0.001}$	0.885 ± 0.022	0.912 ± 0.003	0.903 ± 0.002	0.012 ± 0.007	0.013 ± 0.006
KC-S4R3	$\mathbf{0.473 \pm 0.087}$	0.150 ± 0.095	0.244 ± 0.144	0.035 ± 0.035	0.000 ± 0.000	0.001 ± 0.001

Table 1: Performance of RAPID and the ablations on MiniGrid environments. The mean maximum returns and standard deviations are reported. MR stands for MultiRoom; KC stands for KeyCorridor. Learning curves are in Appendix C.

Pure Exploration on MultiRoom-N12-S10



(c) Pure exploration (d) Local exploration score

- *Observation 1:* ***RAPID works well even without extrinsic rewards.***
- *Observation 2:* ***RIDE is indirectly optimizing local exploration score.***

Takeaways

Some insights:

(1) Episode-level score could better generalize in procedurally-generated environments.

(2) Episode-level exploration also works well in singleton setting (e.g., MuJoCo) and continuous state/action spaces.

(3) Ranking and imitating good exploration behaviors is very effective (a good exploration behavior could be leveraged more than once)

Our contributions:

(1) We propose a practical algorithm, named RAPID, for efficient exploration.

(2) Unlike traditional state-level intrinsic rewards, we approach sample efficiency for hard-exploration problems in episode-level.

(3) Our code is made publicly available.

Code: https://github.com/daochenzha/rapid