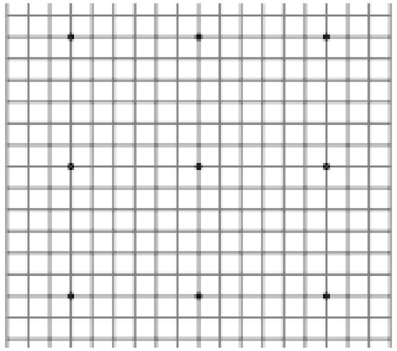
A Research on the Method of Automatic Curriculum Learning for Reinforcement Learning

Weizhe Chen

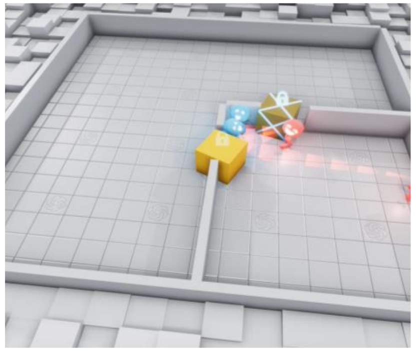
Motivation

- Reinforcement Learning has succussed in multiple domains



Go

StarCraft



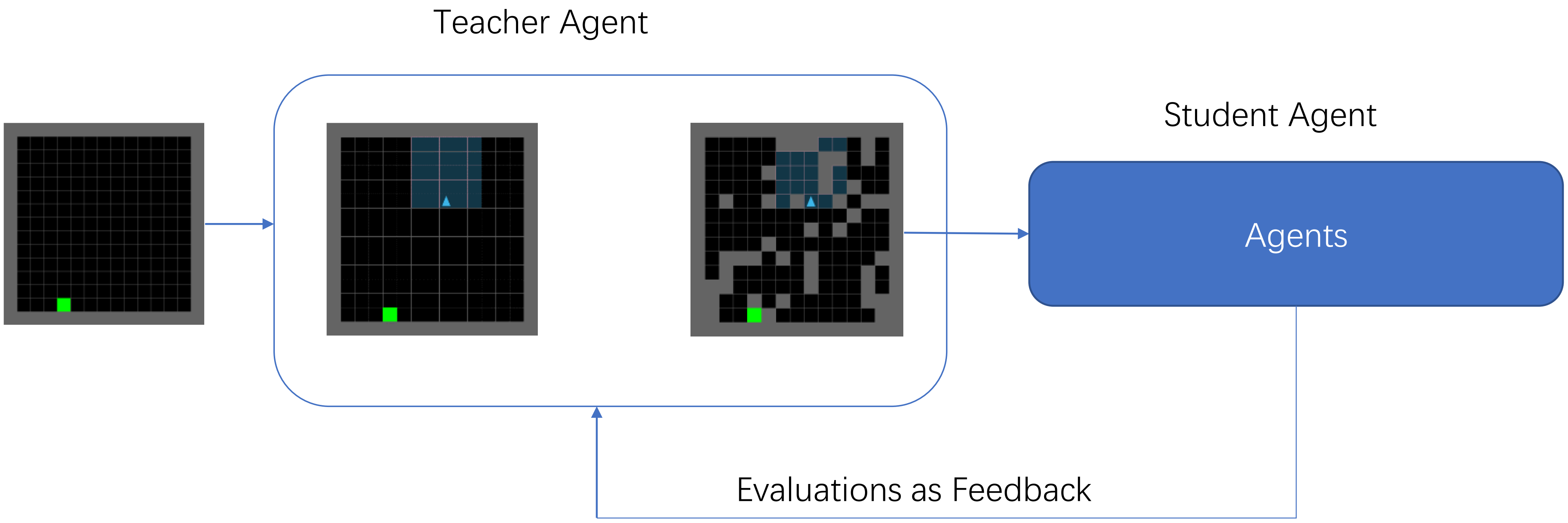
Hide and Seek

- Curriculum learning is one of the most important auxiliary tool behind
  - Turn a hard enough problem into a series of sub-task and learn them sequentially

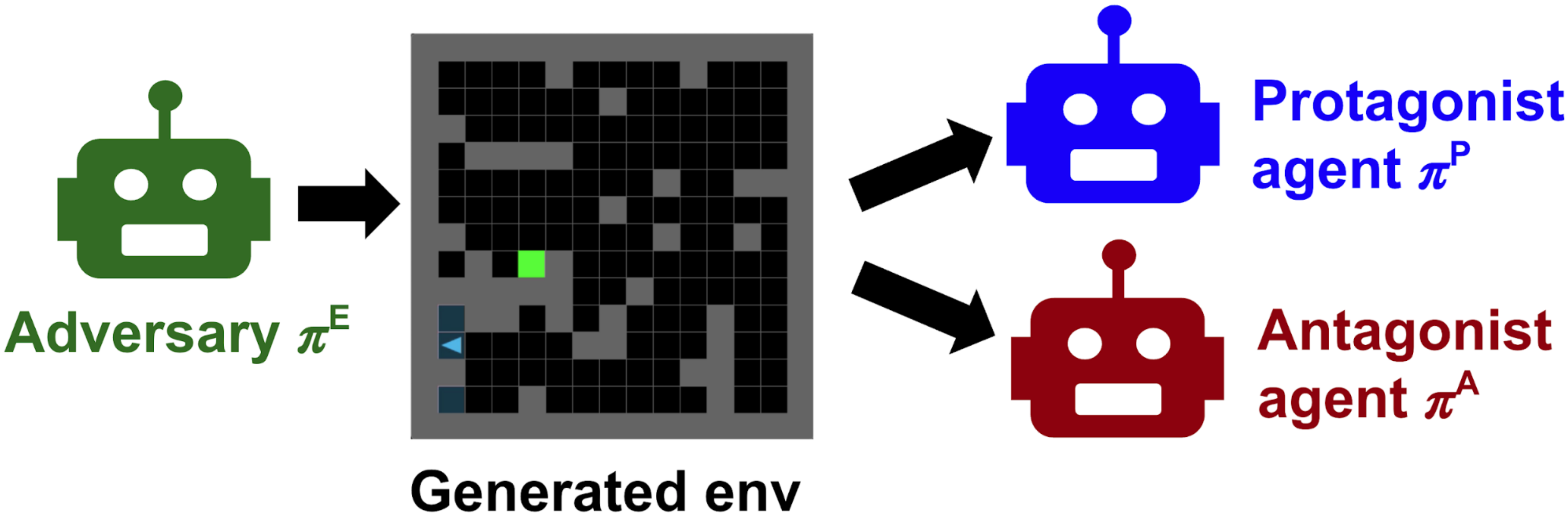
The computing power is huge

- Dota2: 50k CPU, 500 GPU
- 2 vs 2 football: 4096+128 CPU, 16 TPU, 50 days

Unsupervised Environment Design



PAIRED

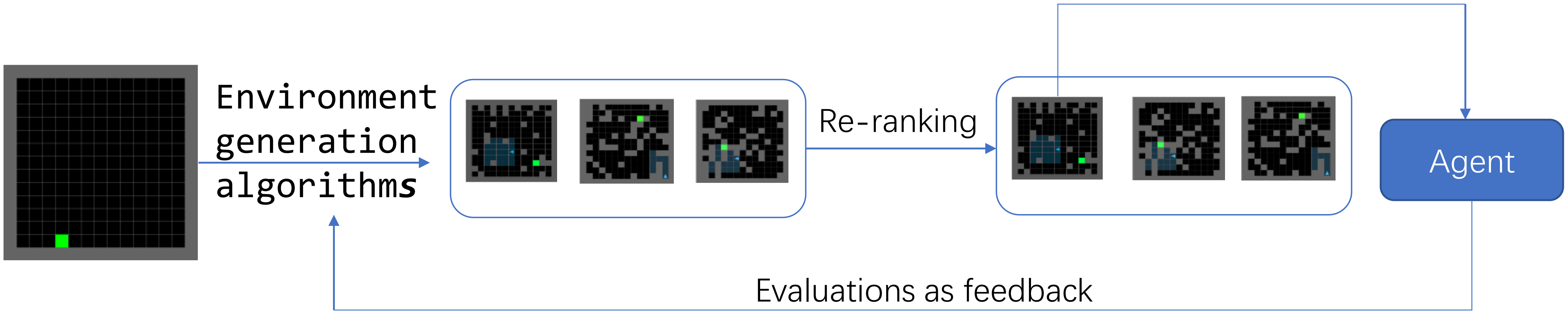


[1]. Dennis M, Jaques N, Vinitsky E, et al. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design[J]. Neurips, 2020.

PAIRED

- Using the additional agent to calculate the regret value.
  - $U = reward(Antagonist) - reward(Protagonist)$
  - $Reward'(Antagonist) = U$
  - $Reward'(Protagonist) = -U$
  - $Reward(Environment) = U$
  - Making the environment to be easier for the agent to learn since the difference of a good policy and the reward of a bad policy is now larger

Problem Definition



Problem Definition

- A special Re-ranking Problem:
  - Re-ranking: one step in the recommendation system to rank the items so that the recommendation can consider more than just the predicted score but also diversity
  - Environment: Item
  - Agent: User

Markov Metric

- Defined as the difference between states
- Normal Requirement: $|V(s) - V(t)| \leq d(s,t)$
- well-setup metrics are hard to calculate!
- In this paper, we define $d(s,t) = |V(s) - V(t)|$
  - Can use the trained value function

SOPED

- From intuition, it is easier for agent to learn if the current environment is not changing too much from the previous one
- We re-rank the potential environments by the last environment we used in training
- Similarity Ordered Population-based Environment Design (SOPED)

Randomly initialize $\pi^P, \pi^A, \pi^\Lambda$;

while *Not converged* **do**

  Generate the environment parameters using the teacher agents population:$\theta_i \sim \Theta_i \sim \pi_i^\Lambda$;

  Calculate the initial observation $o_i = I^{\theta_i}(s_i)$;

  Calculate the distance with the latest trained environment $M_{\theta^{r-1}}$, and choosing the corresponding θ_i as the environment parameters used in this round of trainig θ^r;

  Create the training environment M_θ according to θ_r;

  Sample training trajectories $\tau^A$and τ^P in M_θ, and calculate $U(\pi^A)$ 和 $U(\pi^P)$;

  Calculate $Regret = U(\pi^A) - U(\pi^P)$;

  Train policy π^P with reward $-Regret$;

  Train policies π^Aandπ^Λ with reward $Regret$;

end

return $\pi^P, \pi^A, \pi^\Lambda$

SDOPED

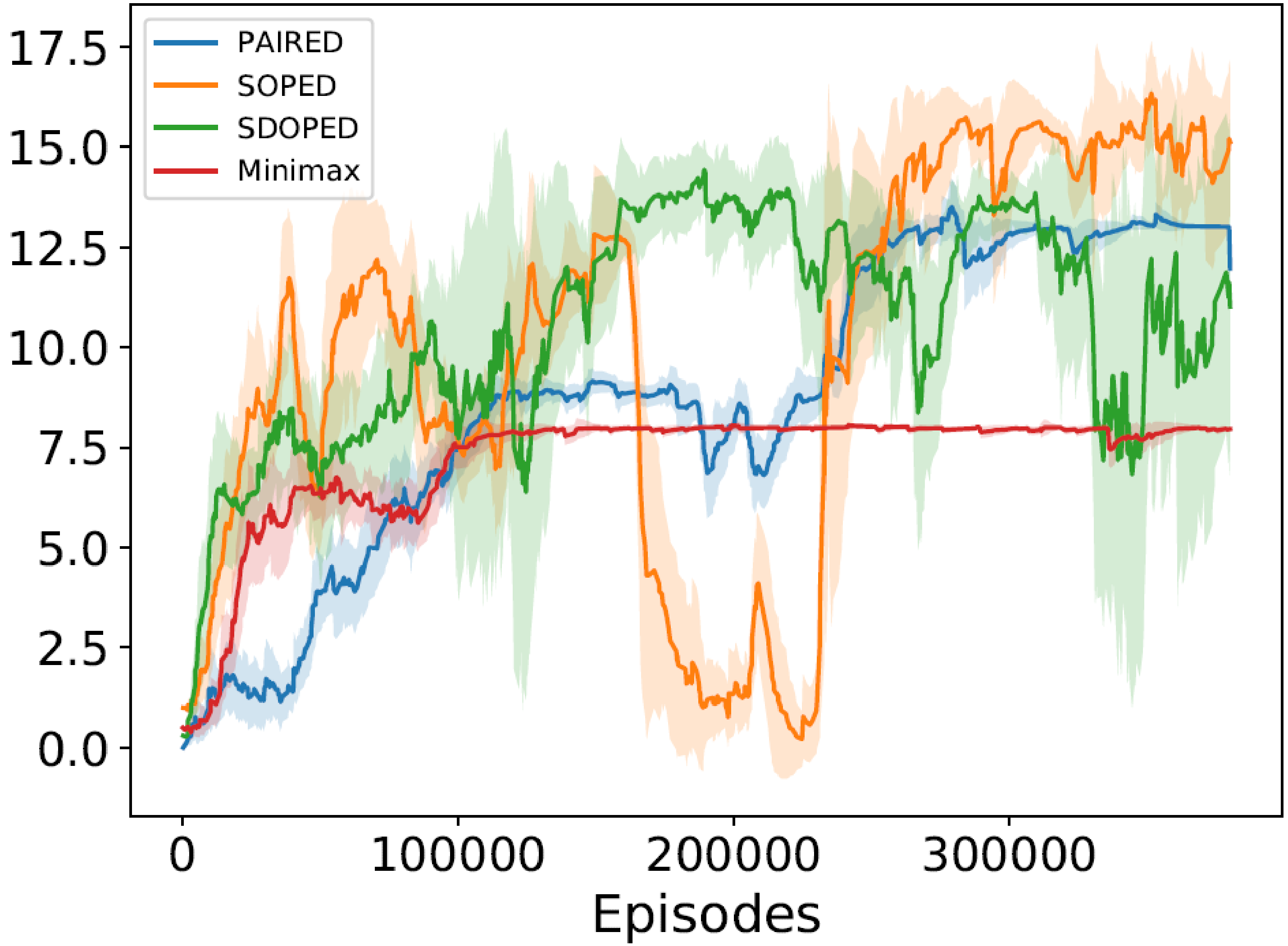
- On the other hand, SOPED will more likely make the high0level generator to stuck in a local optimal rather than a global optimal
- We need to consider diversity during our training
- Determinantal Point Process (DPP) is used
  - Given the similarity of each pair of items and the score of the items, DPP can re-rank the sequence so that it both consider the score and the diversity
  - Similarity $s = e^{-0.5d}$, where d is the distance
- Similarity and Diversity Ordered Population-based Environment Design, SDOPED

SDOPED

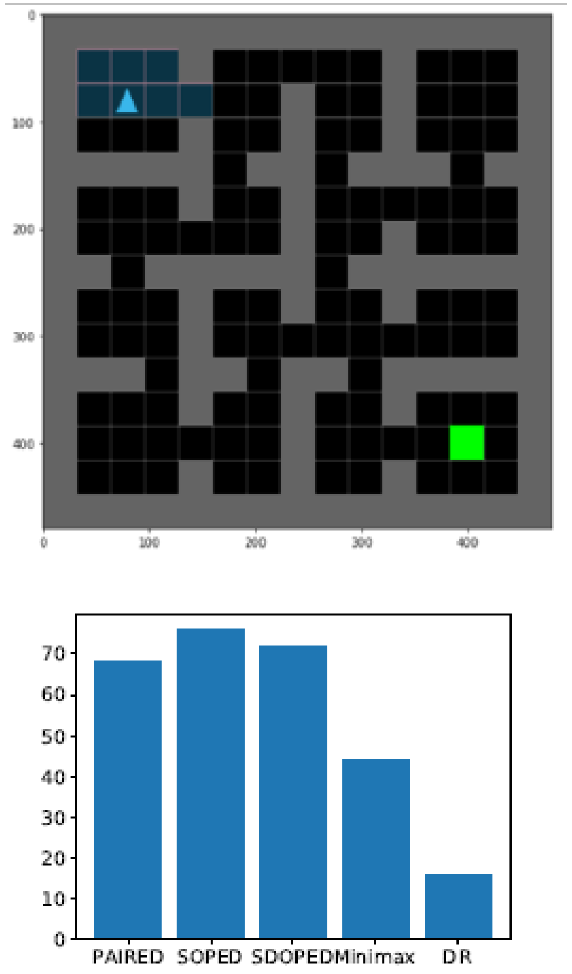
Randomly initialize $\pi^P, \pi^A, \pi^\Lambda$;
while *Not converged* **do**
    Generate the environment parameters using the teacher agents population:$\theta_i \sim \Theta_i \sim \pi_i^\Lambda$;
    Calculate the initial observation $o_i = I^{\theta_i}(s_i)$;
    Calculate the distance with the latest trained environment $M_{\theta^{r-1}}$ as the score of the item;
    Calculating the similarity of each pair of environments as S;
    Calculate the Kernel Matrix $L = Diag(r)\dot{S}\dot{D}iag(r)$ $\Theta_{small} = DPP(L)$;
    Uniformly sample $\theta \in \Theta_{small}$;
    Create actual environment M_θ according to θ_r;
    Sample training trajectories $\tau^A$and τ^P in M_θ, and calculate $U(\pi^A)$ 和 $U(\pi^P)$;
    Calculate $Regret = U(\pi^A) - U(\pi^P)$;
    Train policy π^P with reward $-Regret$;
    Train policies π^Aandπ^Λ with reward $Regret$;
end
return $\pi^P, \pi^A, \pi^\Lambda$

Experiment

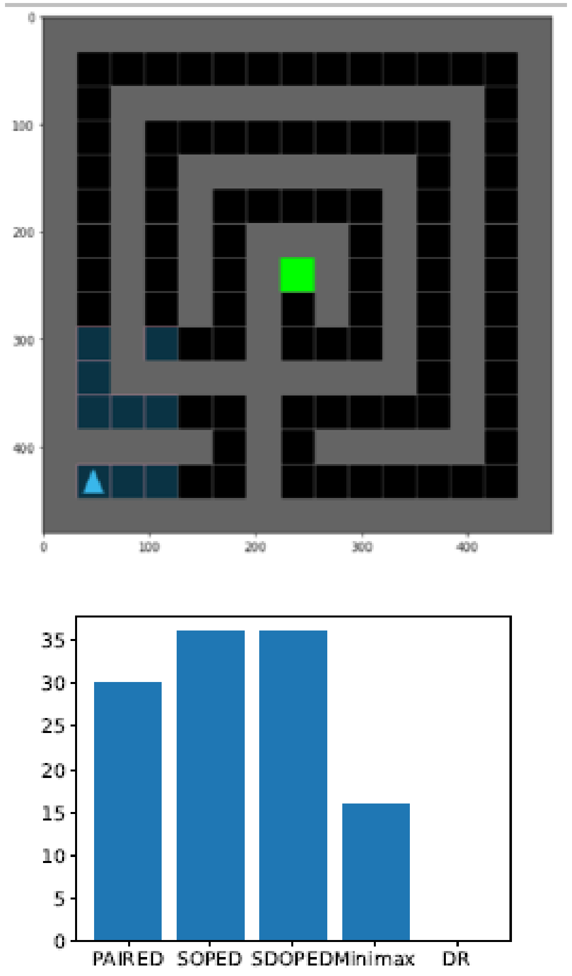
- Maze Design
  - Solved Path Length:



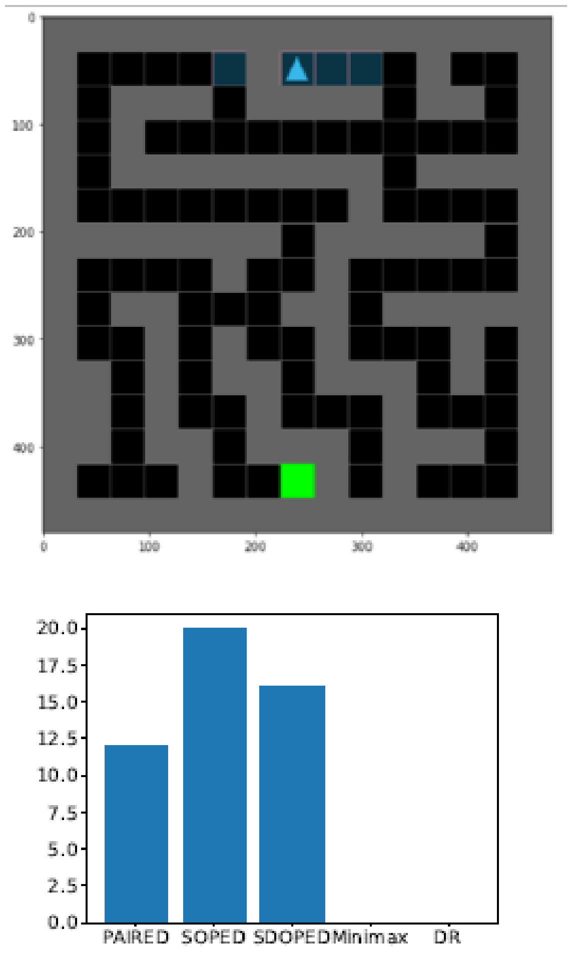
Experiment



(a). 16 Room



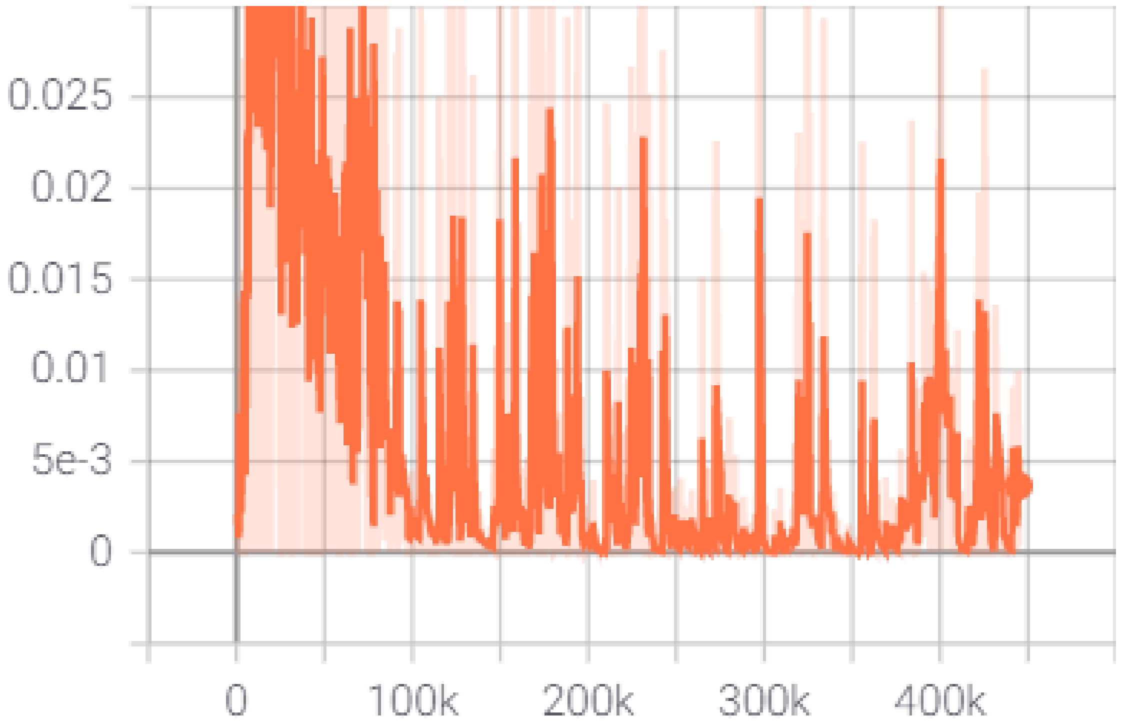
(b). Labyrinth



(c). Maze

Experiment

- Value loss:



Takeaway

- Using environment design, we can improve the training efficiency of the agents, and make the final result more robust
- The value predictor of the high-level environment generator can be easily trained, but the environment generator is very hard to train
- Even if the generator seems to have converge, it still could suddenly completely change its output