南京航空航天大學

Nanjing University of Aeronautics and Astronautics

Hierarchical Reinforcement Learning with Advantage-Based Auxiliary Rewards

Siyuan Li*
IIIS, Tsinghua University
sy-li17@mails.tsinghua.edu.cn

Rui Wang*
Tsinghua University
rui1@stanford.edu

Minxue Tang
Tsinghua University
tangmx16@mails.tsinghua.edu.cn

Chongjie Zhang
IIIS, Tsinghua University
chongjie@tsinghua.edu.cn

NeurIPS 2019

introduce

Hierarchical Reinforcement Learning (HRL) is a promising approach to solving long-horizon problems with sparse and delayed rewards

two major categories of HRL :

1. subgoal-based methods : the high-level policy specifies subgoals for low-level skills to learn

2. selector methods : high-level policy to select pre-trained skills in downstream tasks, and each selected skill is executed for a fixed number of steps

HAAR Learning Framework

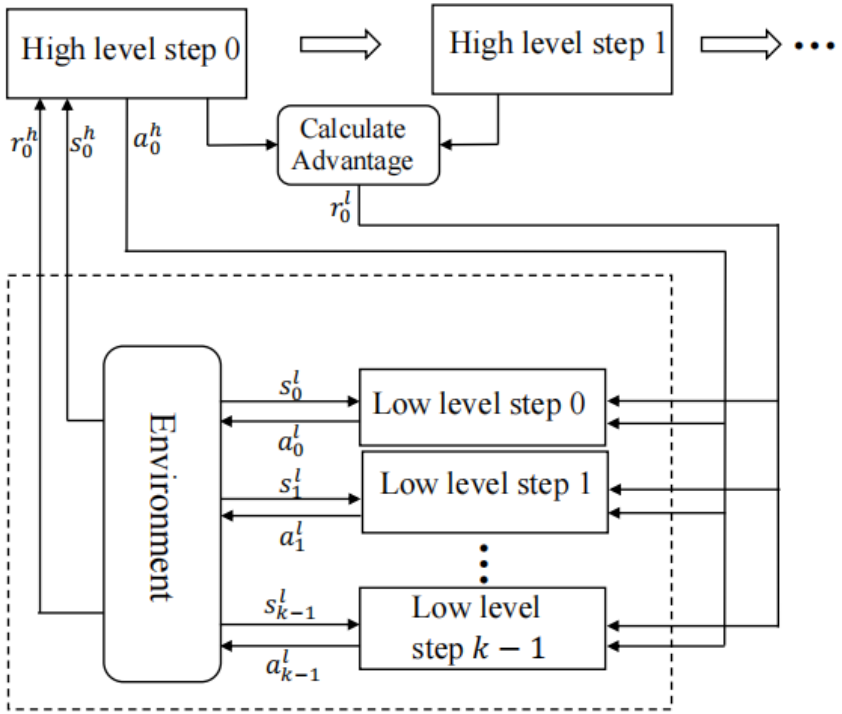


Figure 1: A schematic illustration of HAAR carried out in one high-level step. Within high-level step 0, a total of k low-level steps are taken. Then the process continues to high-level step 1 and everything in the dashed box is repeated.

Algorithm

Algorithm 1 HAAR algorithm

1: Pre-train low-level skills π_l.
2: Initialize high-level policy π_h with a random policy.
3: Initialize the initial skill length k_1 and the shortest skill length k_s.
4: **for** i $\in \{1, ..., N\}$ **do**
5: Collect experiences following the scheme in Figure 1, under π_h and π_l for T low-level steps.
6: Modify low-level experience with auxiliary reward r_t^l defined in Equation (1).
7: Optimize π_h with the high-level experience of the i-th iteration.
8: Optimize π_l with the modified low-level experience of the i-th iteration.
9: $k_{i+1} = max(f(k_i), k_s)$.
10: **end for**
11: **return** π_h, π_l.

improvement:

In most previous works of skill learning and hierarchical learning [13, 22, 8], the skill length k is fixed. When k is too small, the horizon for the high-level policy will be long and the agent explores slowly. When k is too large, the high-level policy becomes inflexible, hence a non-optimal policy. To balance exploration and optimality, we develop a skill length annealing method (Line 9). The skill length k_i is annealed with the iteration number i as $k_i = k_1 e^{-\tau i}$, where k_1 is the initial skill length and τ is the annealing temperature. We define a shortest length k_s such that when $k_i < k_s$, we set k_i to k_s to prevent the skills from collapsing into a single action.

Advantage Function-Based Auxiliary Reward

high-level advantage : $A_h(s^h, a^h) = \mathbb{E}_{s^h_{t+k} \sim (\pi_h, \pi_l)}[r^h_t + \gamma_h V_h(s^h_{t+k}) | a^h_t = a^h, s^h_t = s^h] - V_h(s^h)$

To encourage the selected low-level skills to reach states with greater values, we set the estimated high-level advantage function as our auxiliary rewards to the low-level skills.

$$R_l^{s_t^h, a_t^h}(s_t^l .. s_{t+k-1}^l) = A_h(s_t^h, a_t^h),$$

where $R_l^{s_t^h, a_t^h}(s_t^l .. s_{t+k-1}^l)$ denotes the sum of k-step auxiliary rewards under the high-level state-action pair (s_t^h, a_t^h). For simplicity, We do a one-step estimation of the advantage function in Equation (1). As the low-level skill is task-agnostic and do not distinguish between high-level states, we split the total auxiliary reward evenly among each low-level step, i.e., we have

$$\underset{t \le i < t+k}{r_i^l} = \frac{1}{k} A_h(s_t^h, a_t^h) \tag{1}$$

$$= \frac{r_t^h + \gamma_h V_h(s_{t+k}^h) - V_h(s_t^h)}{k}. \tag{2}$$

Interpretation

An intuitive interpretation of this auxiliary reward function is that, when the temporally-extended execution of skills quickly backs up the sparse environment rewards to high-level states, we can utilize the high-level value functions to guide the learning of low-level skills.

Interpretation

An intuitive interpretation of this auxiliary reward function is that, when the temporally-extended execution of skills quickly backs up the sparse environment rewards to high-level states, we can utilize the high-level value functions to guide the learning of low-level skills.

Proof : Monotonic Improvement of Joint Policy

We define the expected start state value as our objective function to maximize (for convenience, we use π in place of π_{joint})

$$\eta(\pi) = \eta(\pi_h, \pi_l) = \mathbb{E}_{(s_t^h, a_t^h)\sim(\pi_h,\pi_l)}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} r_h(s_t^h, a_t^h)\right]. \tag{3}$$

assertion : the optimization of the high level policy π_h with fixed low level policy π_l leads to improvement in the joint policy.

Namely, when π_l is fixed, maximizing $\eta_h(\pi_h)$ is equivalent to maximizing $\eta(\pi)$.

Now we consider the update for the low-level policy. We can write the objective of the new joint policy $\tilde{\pi}$ in terms of its advantage over π as (proved in Lemma 1 in the appendix)

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{(s_t^h, a_t^h)\sim\tilde{\pi}}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_h(s_t^h, a_t^h)\right]. \tag{4}$$

Since $\eta(\pi)$ is independent of $\eta(\tilde{\pi})$, we can express the optimization of the joint policy as

$$\max_{\tilde{\pi}} \eta(\tilde{\pi}) = \max_{\tilde{\pi}} \mathbb{E}_{(s_t^h, a_t^h)\sim\tilde{\pi}}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_h(s_t^h, a_t^h)\right]. \tag{5}$$

Proof : Monotonic Improvement of Joint Policy

Let $\tilde{\pi}_l$ denote a new low-level policy. In the episodic case, the optimization algorithm for $\tilde{\pi}_l$ tries to maximize

$$\eta_l(\tilde{\pi}_l) = \mathbb{E}_{s_0^l \sim \rho_0^l}[V_l(s_0^l)] = \mathbb{E}_{(s_t^l, a_t^l) \sim (\tilde{\pi}_l, \pi_h)} \left[\sum_{t=0,1,2,\ldots} \gamma_l^t r_l(s_t^l, a_t^l) \right]. \tag{6}$$

Recall our definition of low-level reward in Equation (1) and substitute it into Equation (6), we have (detailed derivation can be found in Lemma 2 in the appendix)

$$\eta_l(\tilde{\pi}_l) = \mathbb{E}_{s_0^l}[V_l(s_0^l)] \approx \frac{1-\gamma_l^k}{1-\gamma_l} \mathbb{E}_{(s_t^h, a_t^h) \sim (\tilde{\pi}_l, \pi_h)} \left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_h(s_t^h, a_t^h) \right]. \tag{7}$$

Proof : Monotonic Improvement of Joint Policy

Proof A.1 *For an objective function defined as*

$$\begin{aligned}\eta(\pi) =& \mathbb{E}_{s_0^h}[V_h(s_0^h)] \\ =& \mathbb{E}_{s_0^h, a_0^h, \ldots \sim \pi}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} r_h(s_t^h)\right]\end{aligned} \tag{9}$$

Changing π to $\tilde{\pi}$, we have

$$\begin{aligned}&\mathbb{E}_{s_0^h, a_0^h, \ldots \sim \tilde{\pi}}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_h(s_t^h, a_t^h)\right] \\ =& \mathbb{E}_{s_0^h, a_0^h, \ldots \sim \tilde{\pi}}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} (r_h(s_t^h) + \gamma_h V_h(s_{t+k}^h) - V_h(s_t^h))\right] \\ =& \mathbb{E}_{s_0^h, a_0^h, \ldots \sim \tilde{\pi}}\left[-V_h(s_0^h) + \sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} r_h(s_t^h)\right] \\ =& -\eta(\pi) + \eta(\tilde{\pi})\end{aligned} \tag{10}$$

Rearranging two sides of the equation, we obtain

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{(s_t^h, a_t^h) \sim \tilde{\pi}}\left[\sum_{t=0,k,2k,\ldots} \gamma_h^{t/k} A_h(s_t^h, a_t^h)\right] \tag{11}$$

Proof : Monotonic Improvement of Joint Policy

Lemma 2 *With the auxiliary reward of* $r_{\substack{i \\ t\le i<t+k}}^{l} = \frac{1}{k}A_h(s_t^h, a_t^h)$*, the low level policy* $\tilde{\pi}_l$*'s expected start value can be written as*

$$\mathbb{E}_{s_0^l}[V_l(s_0^l)] \approx \frac{1-\gamma_l^k}{1-\gamma_l}\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\gamma_h^{t/k}A_h(s_t^h,a_t^h)\right] \tag{12}$$

Where τ_h *is the trajectory of high-level steps* $= s_0^h, a_0^h, s_k^h, a_k^h, \ldots$*. The approximation is correct under the condition that the fixed high level policy* π_h*, the low level discount factor* γ_l *as well as the high level discount factor* γ_h *are both close to* 1*, and the skill length* k *is not extremely large.*

Proof A.2 *We define* τ *to be the trajectory sampled with* $(\tilde{\pi}_l, \pi_h)$*.*

$$\tau = s_0^h, a_0^h, s_1^l, a_1^l, \ldots, s_{nk}^h, a_{nk}^h, s_{nk+1}^l, a_{nk+1}^l \cdots$$

The high-level trajectory is defined as

$$\tau_h = s_0^h, a_0^h, s_k^h, a_k^h, \ldots, s_{nk}^h, a_{nk}^h, \ldots$$

The k-step low level trajectory within a single step of (s_t^h, a_t^h) *is defined as*

$$\tau_l(t) = s_t^l, a_t^l, \ldots s_{t+k-1}^l, a_{t+k-1}^l$$

Now we can write our low level policy's expected start value as

$$\mathbb{E}_{s_0^l}[V_l(s_0^l)] = \mathbb{E}_{\tau\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,1,2,\ldots}\gamma_l^t r_l(s_t^l,a_t^l)\right] \tag{13}$$

$$=\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\mathbb{E}_{\tau_l(t)\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{i=0}^{k-1}\gamma_l^{t+i}A_h(s_t^h,a_t^h)\right]\right] \tag{14}$$

$$=\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\sum_{i=0}^{k-1}\gamma_l^{t+i}A_h(s_t^h,a_t^h)\right] \tag{15}$$

$$=\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\gamma_l^t\frac{1-\gamma_l^k}{1-\gamma_l}A_h(s_t^h,a_t^h)\right] \tag{16}$$

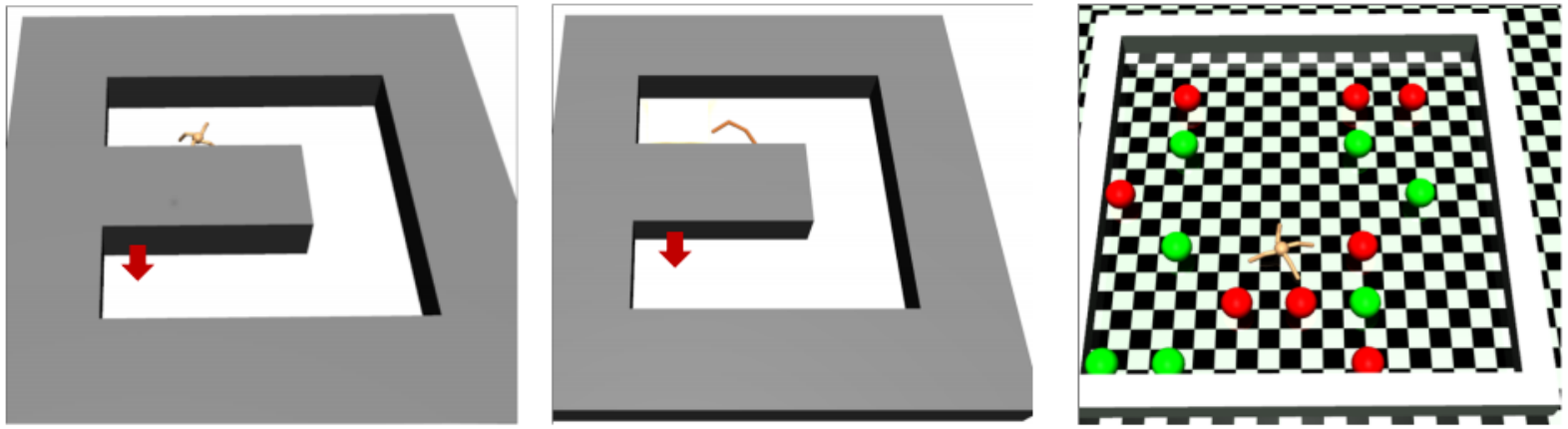
$$=\frac{1-\gamma_l^k}{1-\gamma_l}\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\gamma_l^t A_h(s_t^h,a_t^h)\right] \tag{17}$$

When γ_l *and* γ_h *are both close to* 1 *and* k *is not exceedingly large, we can approximate* γ_l *with* $\gamma_h^{1/k}$ *like below*

$$\begin{aligned}&\frac{1-\gamma_l^k}{1-\gamma_l}\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\gamma_l^t A_h(s_t^h,a_t^h)\right]\\ \approx&\frac{1-\gamma_l^k}{1-\gamma_l}\mathbb{E}_{\tau_h\sim(\tilde{\pi}_l,\pi_h)}\left[\sum_{t=0,k,2k,\ldots}\gamma_h^{t/k}A_h(s_t^h,a_t^h)\right]\end{aligned} \tag{18}$$

Experiments

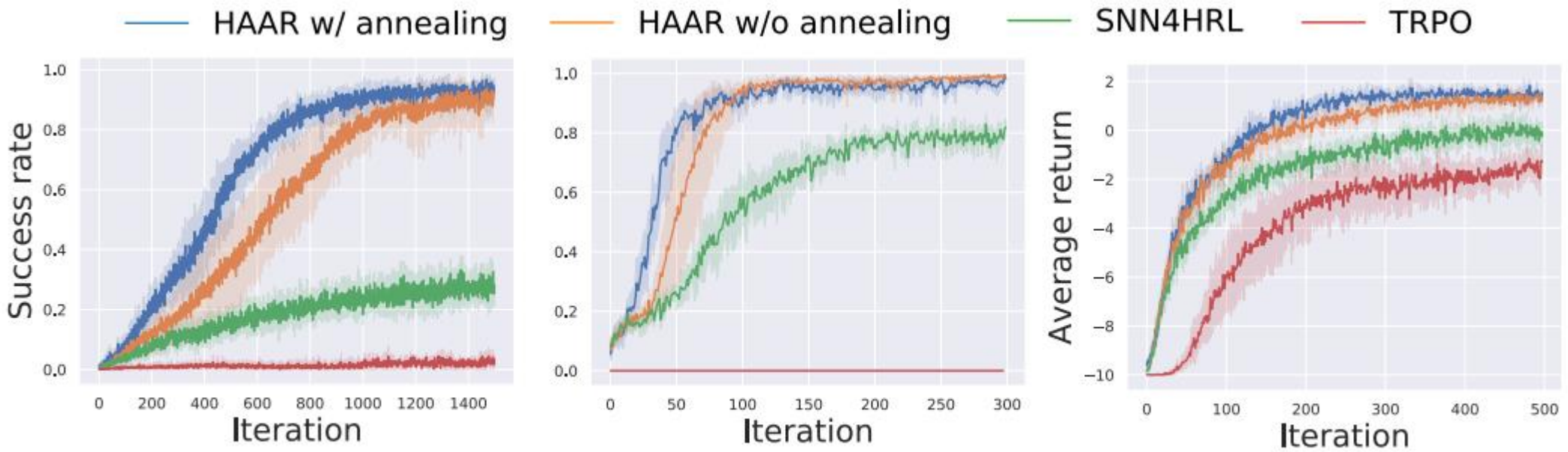
- Ant Maze: The ant is rewarded for reaching the specified position in a maze shown in Figure 2(a). We randomize the start position of the ant to acquire even sampling of states.
- Swimmer Maze: The swimmer is rewarded for reaching the goal position in a maze shown in Figure 2(b).
- Ant Gather: The ant is rewarded for collecting the food distributed in a finite area while punished for touching the bombs, as shown in Figure 2(c).



(a) (b) (c)

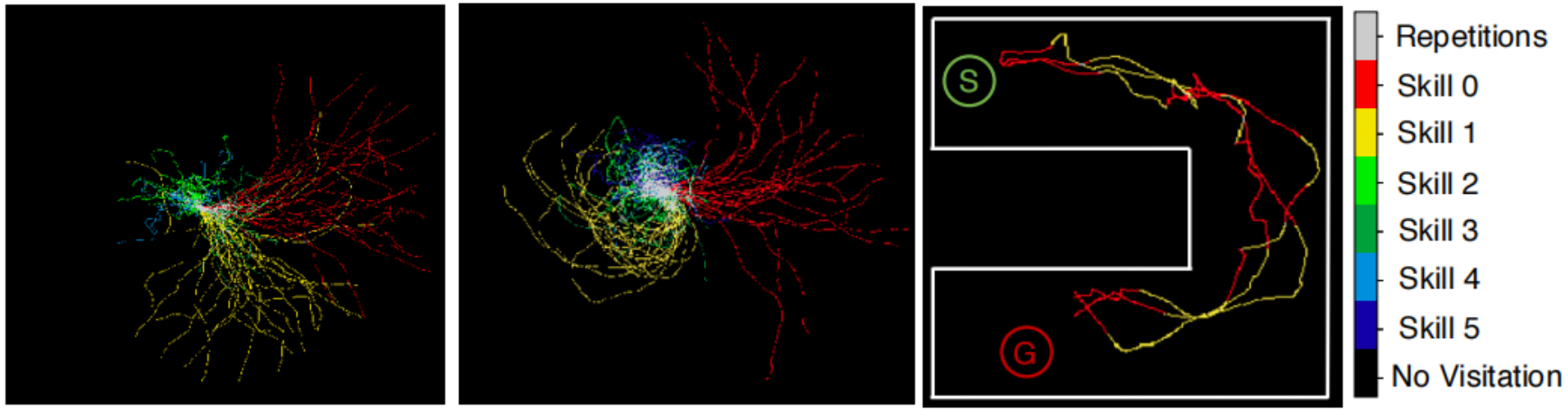
Figure 2: A collection of environments that we use. (a) Ant in maze (b) Swimmer in maze (c) Ant in gathering task.

Results and Comparisons



(a) Ant Maze (b) Swimmer Maze (c) Ant Gather

Visualization of Skills and Trajectories



(a) (b) (c)

Figure 4: (a) Visitation plot of initial low-level skills of the ant. (b) Low-level skills after training with auxiliary rewards in Ant Maze. (c) Sample trajectories of the ant after training with HAAR in Ant Maze.

In Figure 4, (a) and (b) demonstrate a batch of experiences collected with low-level skills before and after training, respectively. The ant is always initialized at the center and uses a single skill to walk for an arbitrary length of time. Comparing (b) with (a), we note that the ant learns to turn right (Skill 1 in yellow) and go forward (Skill 0 in red) and well utilizes these two skills in the Maze task in (c), where it tries to go to (G) from (S). We offer analysis for other experiments in Appendix C.

Transfer of Policies

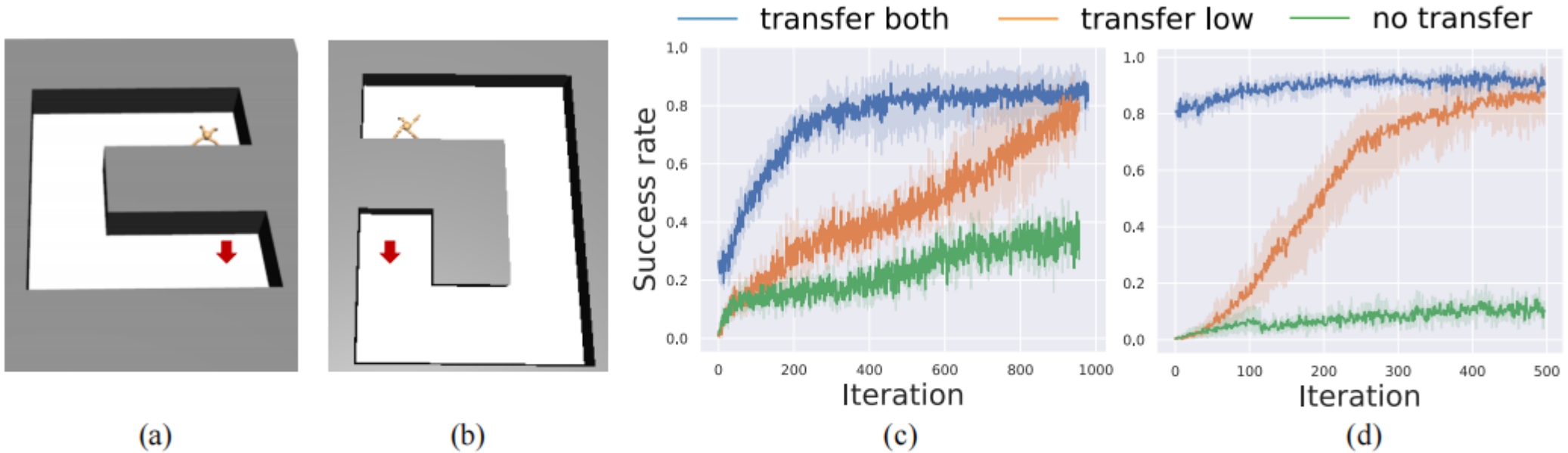


Figure 5: (a) and (b) are tasks to test the transferability of learned policies. (c) and (d) are the corresponding learning curves of transferring both high and low-level policies, transferring only low-level policy, and not transferring (the raw form of HAAR).

南京航空航天大學

Nanjing University of Aeronautics and Astronautics

HIerarchical Reinforcement learning with Off-policy correction

Ofir Nachum
Google Brain
ofirnachum@google.com

Shixiang Gu*
Google Brain
shanegu@google.com

Honglak Lee
Google Brain
honglak@google.com

Sergey Levine†
Google Brain
slevine@google.com

NIPS 2018

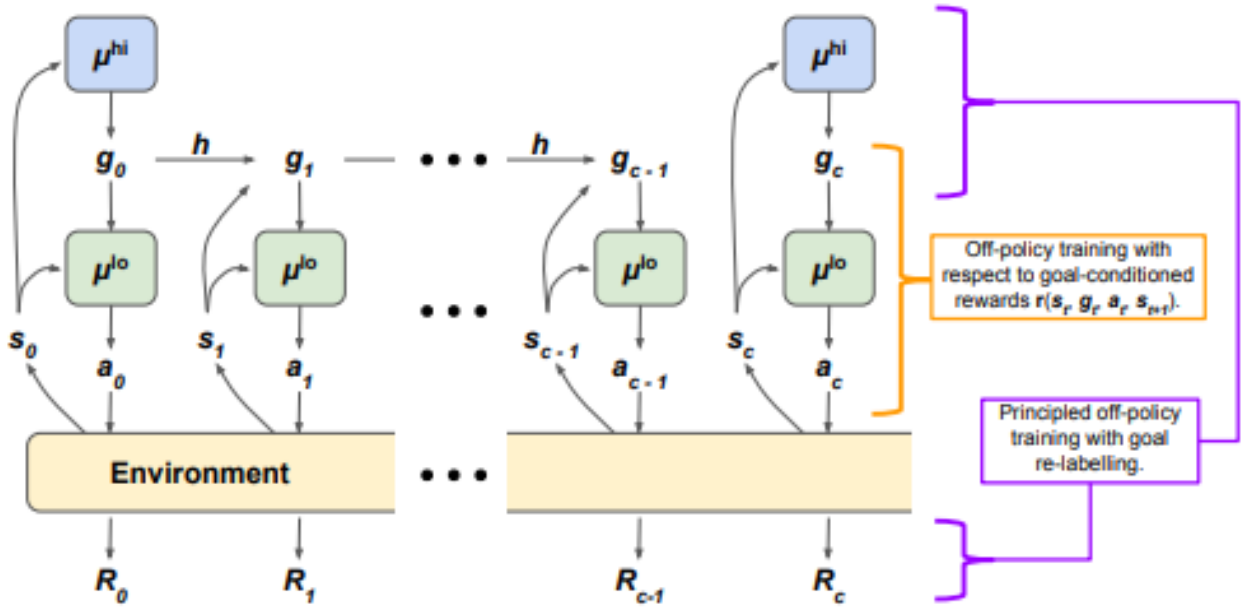
Universal Value Function Approximators (UVFA)

$$r_g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

$$\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$$

$$Q^{\pi}(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]$$

Framwork



1. Collect experience $s_t, g_t, a_t, R_t, \ldots$.
2. Train μ^{lo} with experience transitions $(s_t, g_t, a_t, r_t, s_{t+1}, g_{t+1})$ using g_t as additional state observation and reward given by goal-conditioned function $r_t = r(s_t, g_t, a_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||_2$.
3. Train μ^{hi} on temporally-extended experience $(s_t, \tilde{g}_t, \sum R_{t:t+c-1}, s_{t+c})$, where $\tilde{g}_t$ is re-labelled high-level action to maximize probability of past low-level actions $a_{t:t+c-1}$.
4. Repeat.

the goal transition model h is defined as

$$h(s_t, g_t, s_{t+1}) = s_t + g_t - s_{t+1}.$$

Off-Policy Corrections for Higher-Level Training

the changing behavior of the lower-level policy creates a non-stationary problem for the higher-level policy, and old off-policy experience may exhibit different transitions conditioned on the same goals.

method : replace g_t in state-action-reward transitions $(s_t, g_t, \sum R_{t:t+c-1}, s_{t+c})$

high-level action $\tilde{g}_t$ chosen to maximize the probability $\mu^{lo}(a_{t:t+c-1}|s_{t:t+c-1}, \tilde{g}_{t:t+c-1})$.

Most RL algorithms will use random action-space exploration to select actions, which means that the behavior policy (even for deterministic algorithms such as DDPG [27]) is stochastic and the log probability $\log \mu^{lo}(a_{t:t+c-1}|s_{t:t+c-1}, \tilde{g}_{t:t+c-1})$ may be computed as

$$\log \mu^{lo}(a_{t:t+c-1}|s_{t:t+c-1}, \tilde{g}_{t:t+c-1}) \propto -\frac{1}{2} \sum_{i=t}^{t+c-1} ||a_i - \mu^{lo}(s_i, \tilde{g}_i)||_2^2 + \text{const.} \quad (5)$$

南京航空航天大學

Nanjing University of Aeronautics and Astronautics

LEARNING MULTI-LEVEL HIERARCHIES WITH HINDSIGHT

Andrew Levy
Department of Computer Science
Brown University
Providence, RI, USA
andrew_levy2@brown.edu

George Konidaris
Department of Computer Science
Brown University
Providence, RI, USA
gdk@cs.brown.edu

Robert Platt
College of Computer and Information Science
Northeastern University
Boston, MA, USA
rplatt@ccs.neu.edu

Kate Saenko
Department of Computer Science
Boston University
Boston, MA, USA
saenko@bu.edu

ICLR 2019

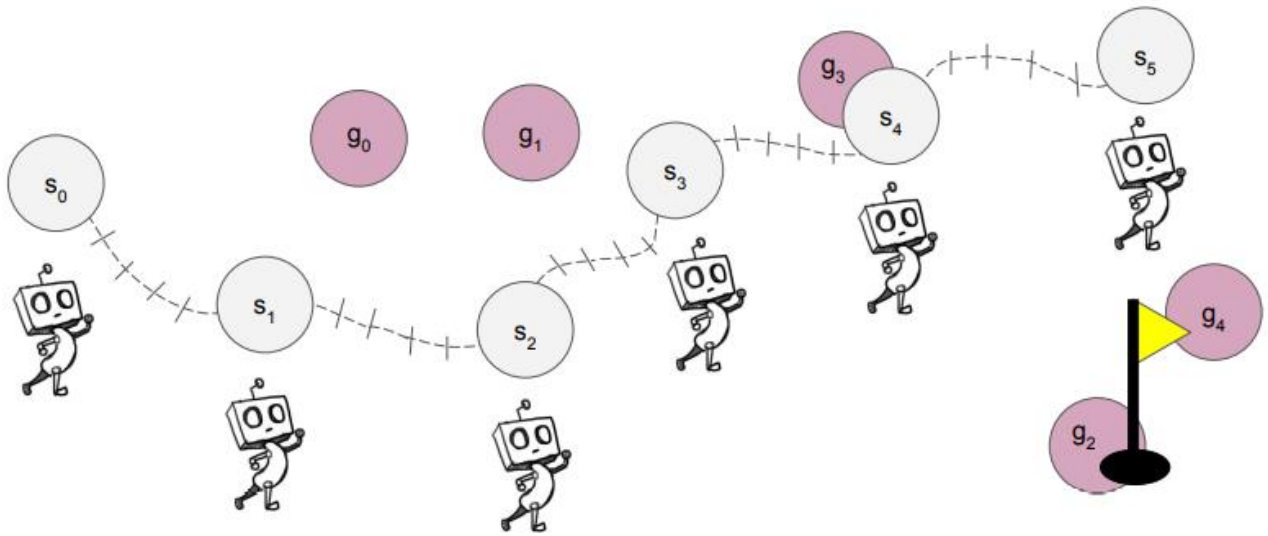
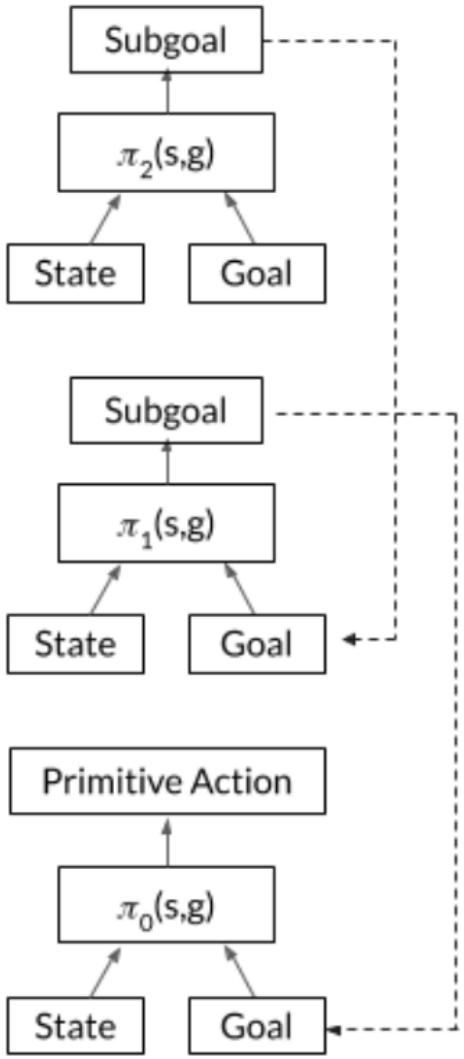


Figure 2: An example episode trajectory for a simple toy example. The tic marks along the trajectory show the next states for the robot after each primitive action is executed. The pink circles show the original subgoal actions. The gray circles show the subgoal states reached in hindsight after at most H actions by the low-level policy.

There are two causes of non-stationary transition functions in our framework that will need to be overcome in order to learn multiple policies in parallel. One cause of non-stationary transition functions is updates to lower level policies. That is, whenever π_i changes, the transition function at levels above i, $T_{j|\Pi_{j-1}}, j > i$, can change. The second cause is exploring lower level policies. Because all levels have a deterministic policy in our algorithm, all levels will need to explore with some behavior policy π_{i_b} that is different than the policy it is learning π_i.

nested policy



Hindsight action transitions

Hindsight action transitions have two key components

The first is that the subgoal state achieved in hindsight is used as the action component in the transition, not the originally proposed subgoal state

The second key component of the hindsight action transition is the reward function used at all subgoal levels. requements :
1 . incentivize short paths to the goal
2 . independent of the path taken at lower levels

[initial state=s0,action=s1,reward=-1,next state=s1,goal=yellow flag,discount rate=gamma]

[initial state=s1,action=s2,reward=-1,next state=s2,goal=yellow flag,discount rate=gamma]

Hindsight goal transitions

Enable each level to learn more effectively in sparse reward tasks by extending the idea of Hindsight Experience Replay.

1. high level set goal
2. low level execut H steps, get H transitions
3. choose hindsight goal’ from H transitions
4. reset the goal of H transitions to goal’ and modify their reward at the same time

Subgoal testing transitions

enable a level to understand whether a subgoal state can be achieved by the current set of lower level policies.
if subgoal a_i is not achieved in at most H actions by level i − 1, level i will be penalized with a low reward, penalty.

[initial state = s2, action = g2, reward = -5, next state = s3, goal = Yellow Flag, discount rate = 0]

Experiments

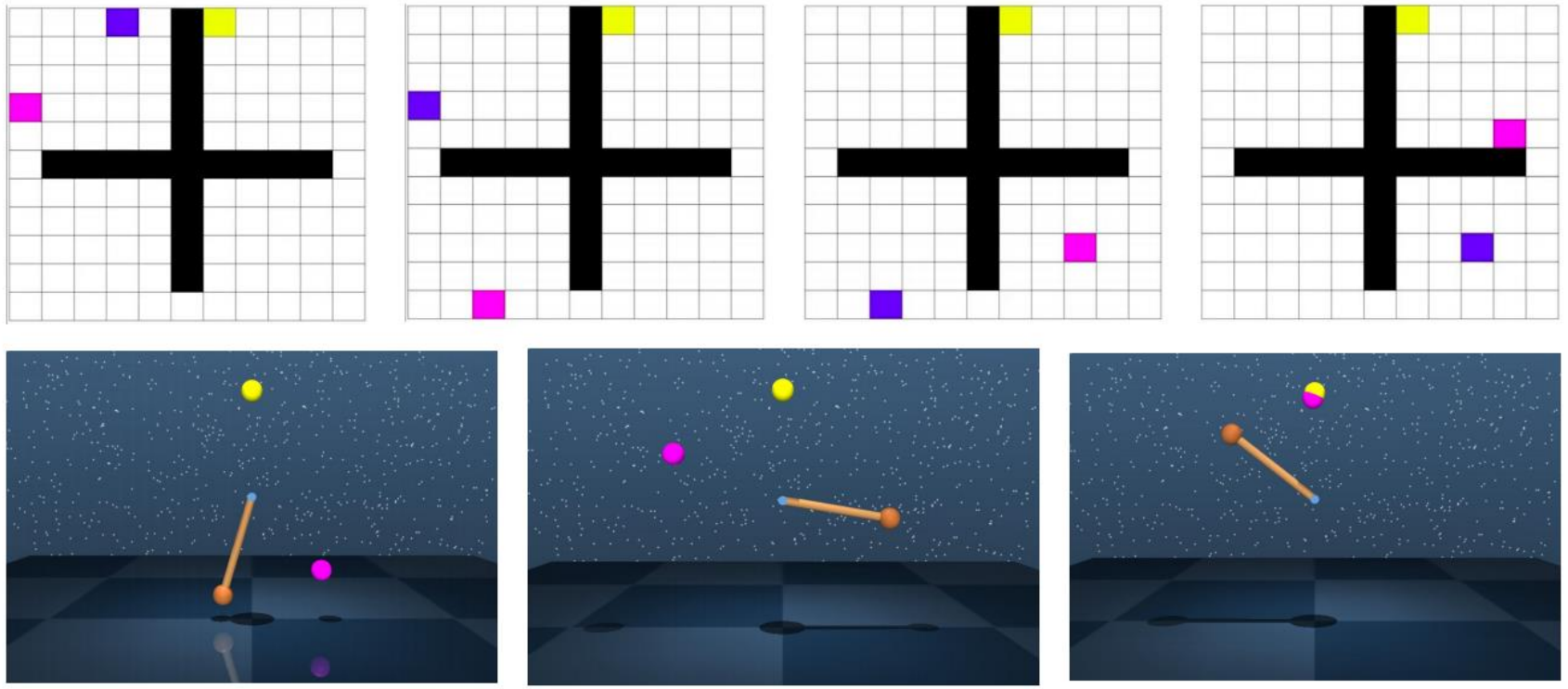


Figure 3: Episode sequences from the four rooms (top) and inverted pendulum tasks (bottom). In the four rooms task, the k=2 level agent is the blue square; the goal is the yellow square; the learned subgoal is the purple square. In the inverted pendulum task, the goal is the yellow sphere and the subgoal is the purple sphere.

Experiments

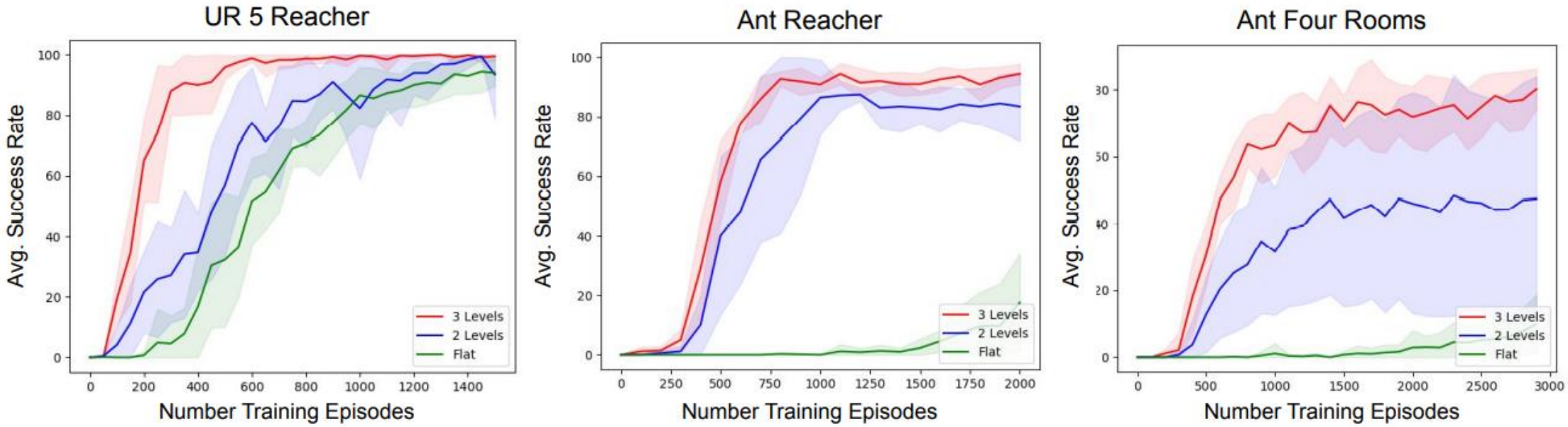


Figure 4: Average success rates for 3-level (red), 2-level agent (blue), and flat (green) agents in each task. The error bars show 1 standard deviation.