NEW YORK UNIVERSITY

Facebook AI Research

Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play

Sainbayar Sukhbaatar

New York University

Joint work with:

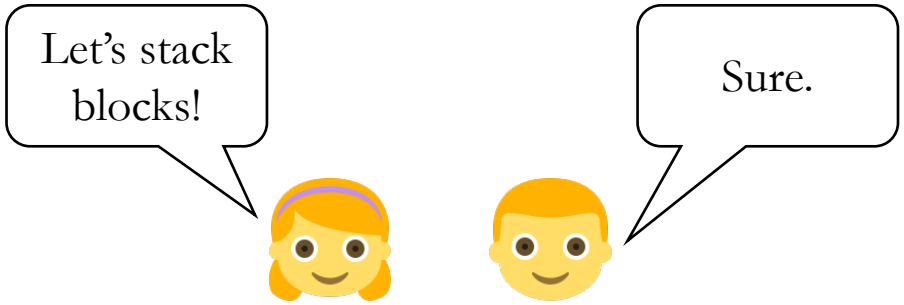
Arthur Szlam
Facebook
AI Research

Rob Fergus
Facebook
AI Research

Motivation

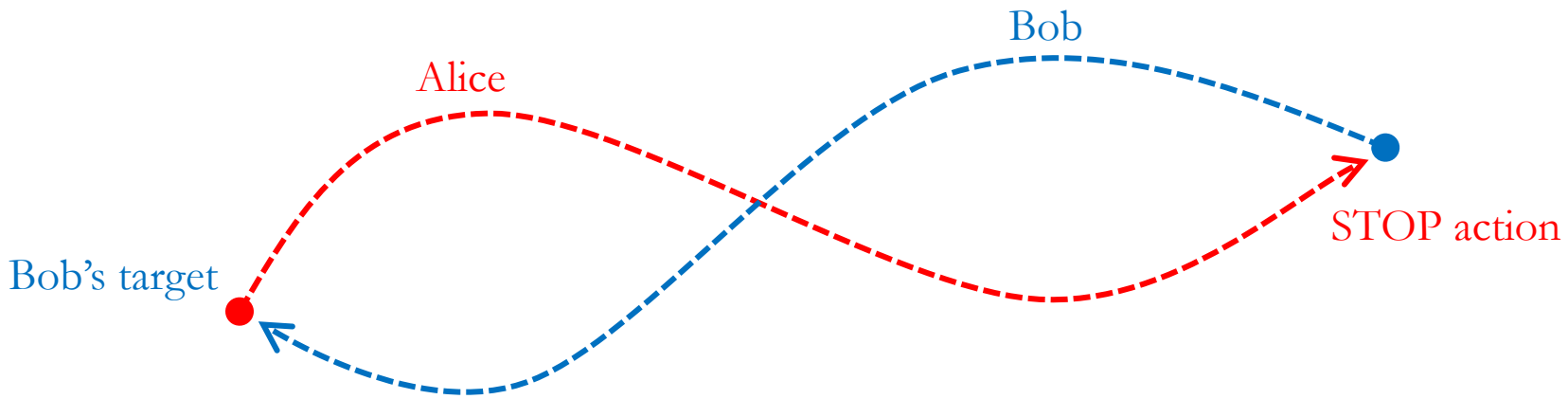
- Reinforcement Learning (RL) typically requires a **huge** number of episodes
- Often **supervision** signal (i.e. reward) is **expensive** to obtain
- Can we learn about environment in **unsupervised** way?
- Assuming interaction with the environment is cheap

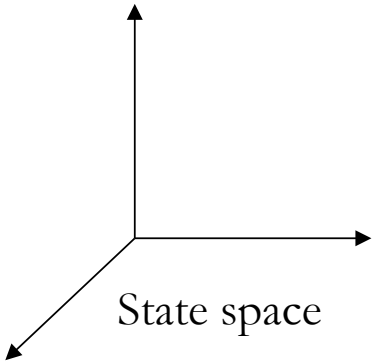
Approach



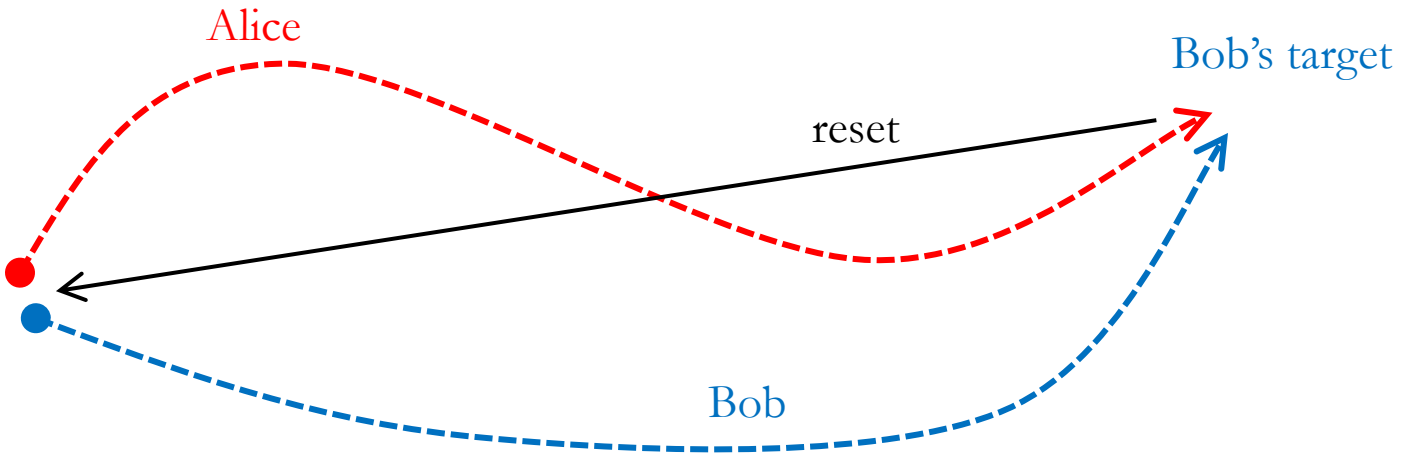
- Agent plays a **game** where it challenges itself
- **Single** physical agent, but **two** separate minds:
  - Alice's job is to **propose** a task
  - Bob's job is to **complete** that task
- Alice propose a task by actually **doing** it
- We consider two classes of environments:
  1. Actions are reversible within same time → **reverse** self-play
  2. Reset to the initial state is allowed → **repeat** self-play
- Jointly train with **self-play** and **target task**

Reverse self-play





Repeat self-play



- Internal reward during self-play

$$R_b = -t_b$$

Time spent

$$R_a = \max(0, t_b - t_a)$$

Intuition: make Bob fail with less effort

- Policy of Alice and Bob
  - Self-play: $a_{\text{Alice}} = f_A(s_t, s_0)$ $a_{\text{Bob}} = f_B(s'_t, s'_0)$
  - Target task: $a_{\text{Target}} = f_B(s''_t, e)$

target state

task description

function SELFPLAYEPISODE(REVERSE/REPEAT, t_{MAX})

$t_a \leftarrow 0$

$s_0 \leftarrow$ env.observe()

$s_{\text{Target}} = s_0$

while True **do**

Alice's turn

$t_a \leftarrow t_a + 1$

$s_{t_a} \leftarrow$ env.observe()

$a_{t_a} \leftarrow$ policy.Alice(s_{t_a})

if $a^{t_a} =$ STOP **then**

$s_{\text{Target}} = s_{t_a}$

env.reset()

break

env.act(a^{t_a})

$t_b \leftarrow 0$

while True **do**

Bob's turn

$t_b \leftarrow t_b + 1$

$s'_{t_b} \leftarrow$ env.observe()

$a'_{t_b} \leftarrow$ policy.Bob(s'_{t_b})

env.act(a'_{t_b})

if $s'_{t_b} = s_{\text{Target}}$ **or** $t_b > t_{\text{Max}}$ **then**

break

$R_a = \max(0, t_b - t_a)$

$R_b = -t_b$

policy.Alice.update(R_a)

policy.Bob.update(R_b)

return

Self-play equilibrium & Universal Bob

- Under some strong assumptions (tabular policies, finite state, etc.), Bob must learn all possible tasks
- Let's assume the self-play is converged to a Nash equilibrium (can't gain anything fixing other's policy)
- If Bob fails on a certain task, then Alice would propose that task to increase her reward
- Then Bob must've seen this task and learnt it to increase his reward

Related works

- **Self-play:** checkers (Samuel, 1959), backgammon (Tesauro, 1995), and Go, (Silver et al., 2016), and RoboSoccer 213 (Riedmiller et al., 2009)

- **GANs** (Goodfellow et al., 2014): dialogue generation (Li et al., 2017), variational auto-encoders (Mescheder et al., 2017)

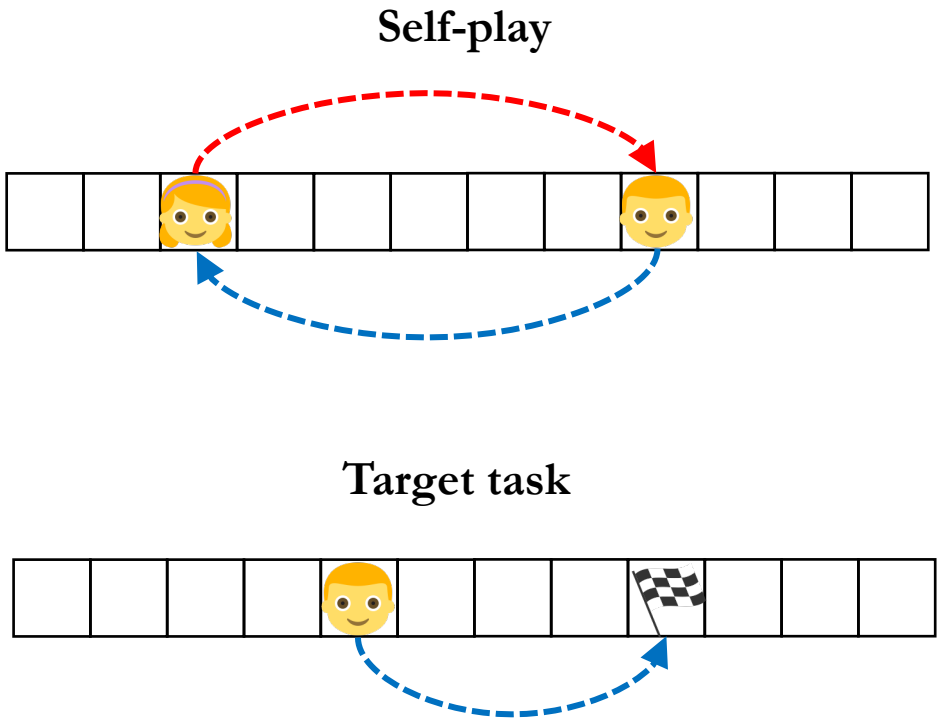
- **Intrinsic motivation** (Barto, 2013; Singh et al., 2004; Klyubin et al., 2005; Schmidhuber, 1991): curiosity-driven exploration (Schmidhuber, 1991; Bellemare et al., 2016; Strehl & Littman, 2008; Lopes et al., 2012; Tang et al., 2016)

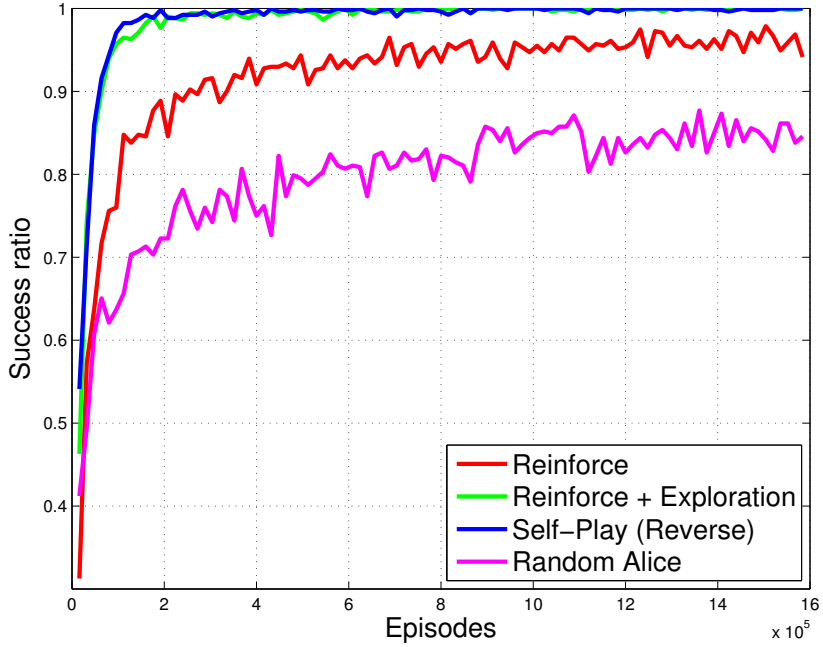
Experiments

- Use **Reinforce** algorithm with learnt baseline and entropy regularization
- 2-layer NN model for Alice and Bob (separate)
- Train on 20% target task + 80% self-play
- Discrete and continuous environments

Simple Long hallway

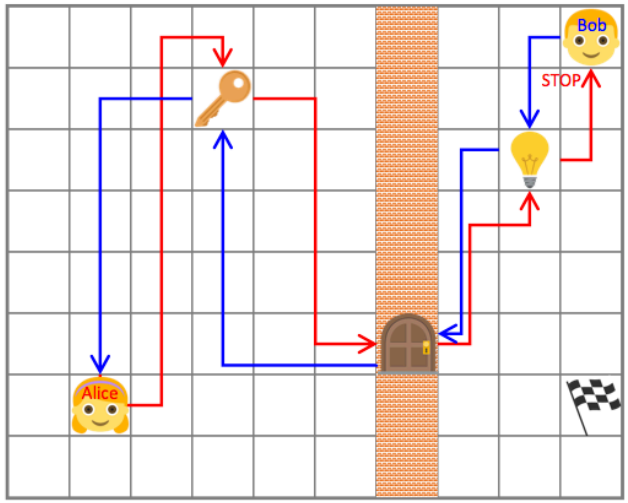
- Learn to navigate in a long corridor
- Simple tabular policies

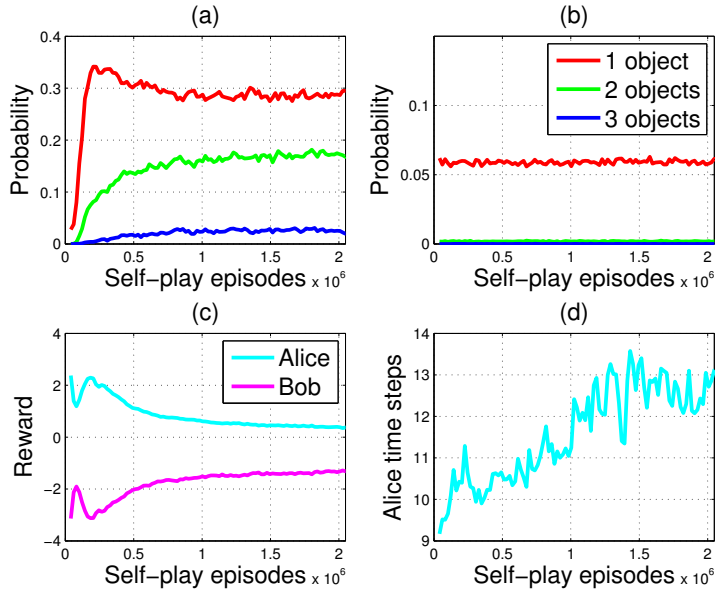


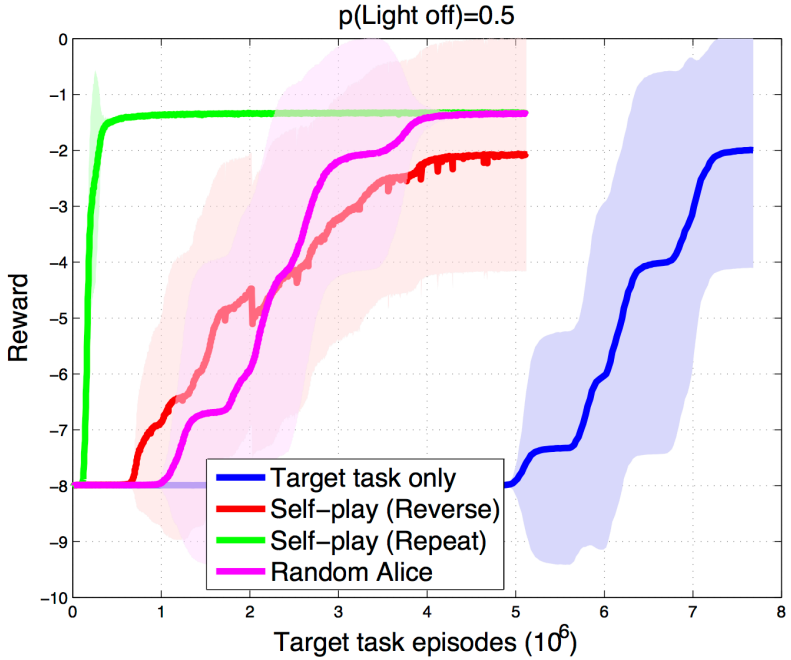


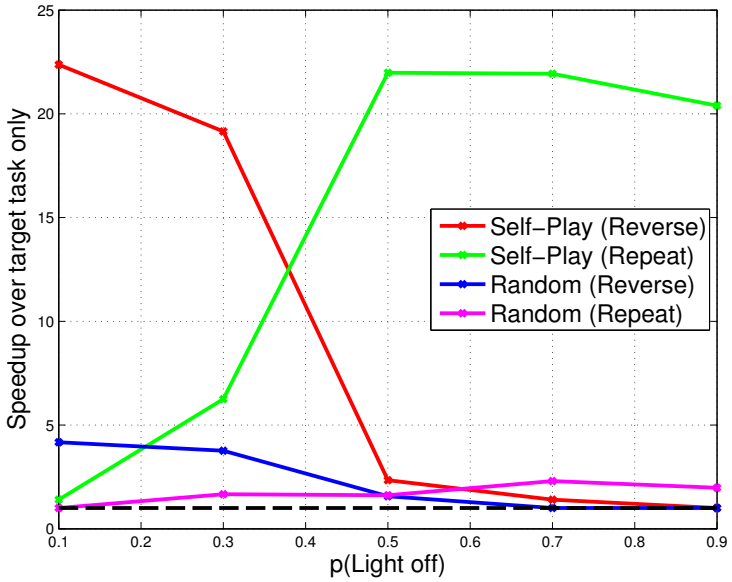
MazeBase: LightKey task

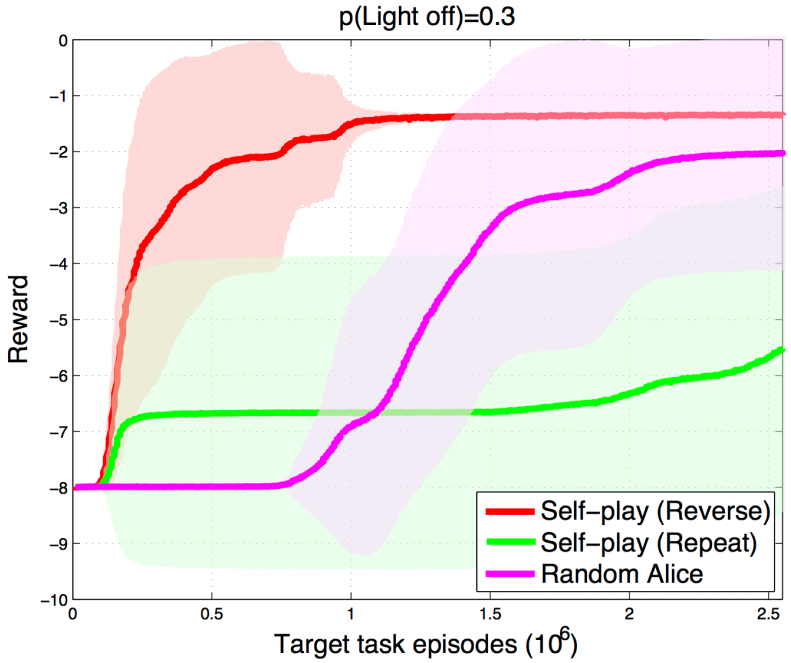
- Small 2D grid separated into two rooms by a wall
- Toggle the key to lock/unlock the door
  - Can't go through a locked door
- Toggle the switch to turn on/off the light
  - Only the switch is visible in dark
- The grid is procedurally generated
- Target task is to reach the goal flag in the opposite room when light is off and door is locked.



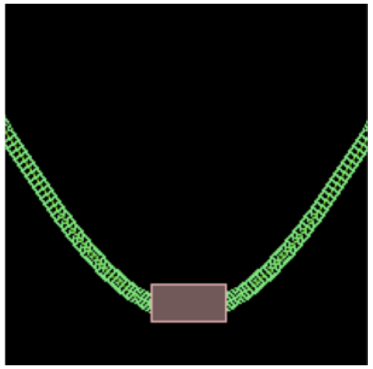

(a)
Probability
Self−play episodes x 10^6
(b)
1 object
2 objects
3 objects
Probability
Self−play episodes x 10^6
(c)
Alice
Bob
Reward
Self−play episodes x 10^6
(d)
Alice time steps
Self−play episodes x 10^6


p(Light off)=0.5
Reward
Target task only
Self−play (Reverse)
Self−play (Repeat)
Random Alice
Target task episodes (10^6)

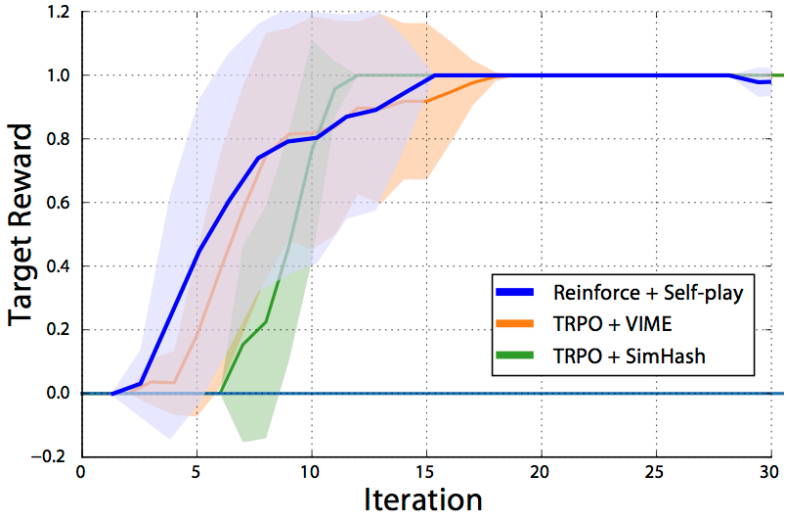

Speedup over target task only
Self−Play (Reverse)
Self−Play (Repeat)
Random (Reverse)
Random (Repeat)
p(Light off)


p(Light off)=0.3
Reward
Self−play (Reverse)
Self−play (Repeat)
Random Alice
Target task episodes (10^6)

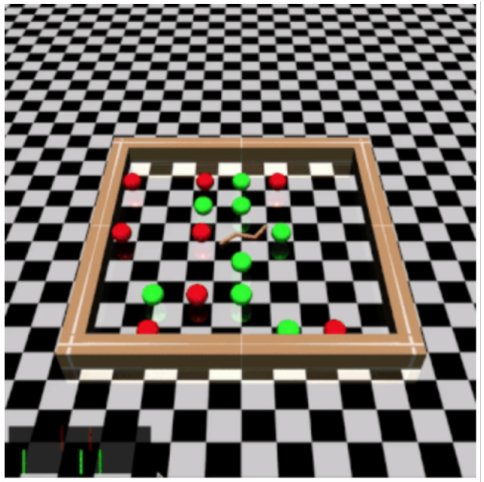
RLLab: Mountain Car



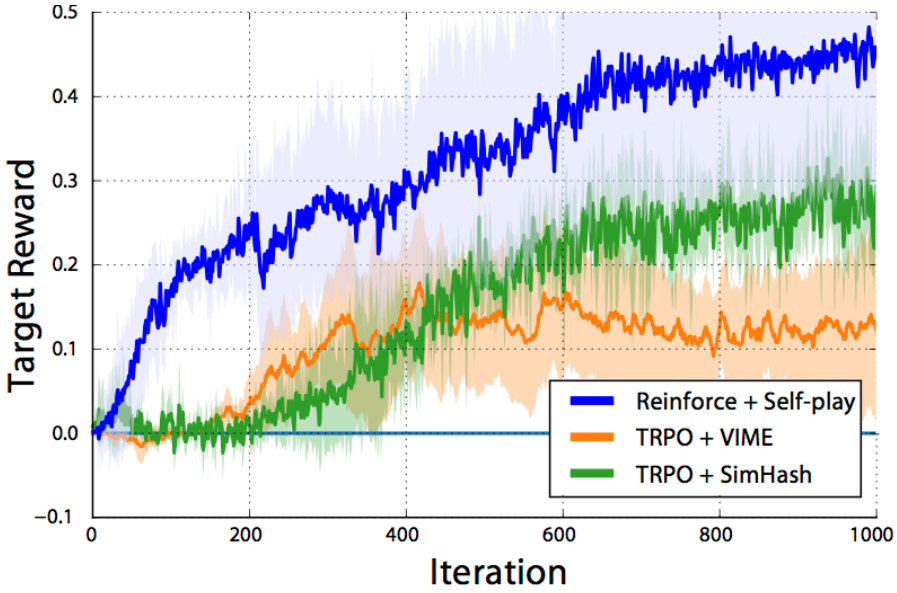
- Control a car stuck in 1D valley
  - Need to build momentum by reversing
- Sparse reward
  - +1 reward only if it reaches the left hill top
- Hard task because random exploration fails
- Asymmetric environment → repeat self-play
- As good as other exploration methods

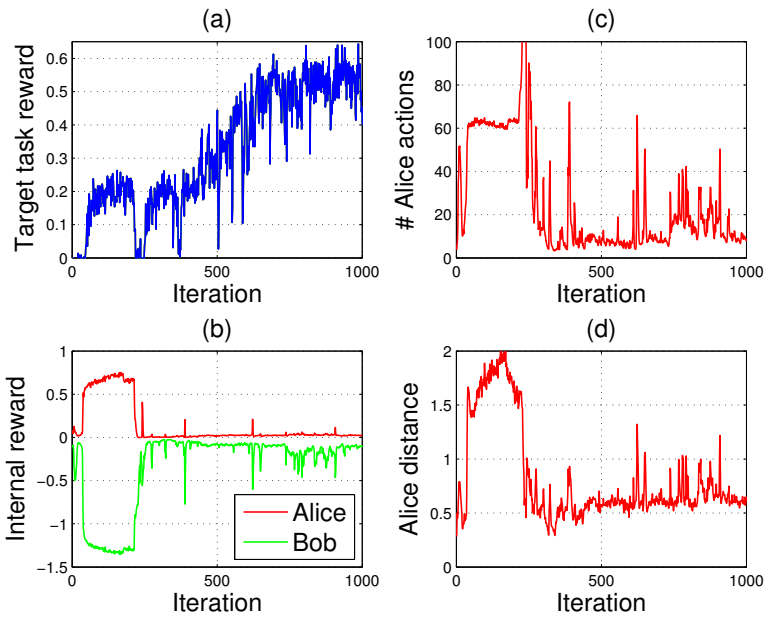


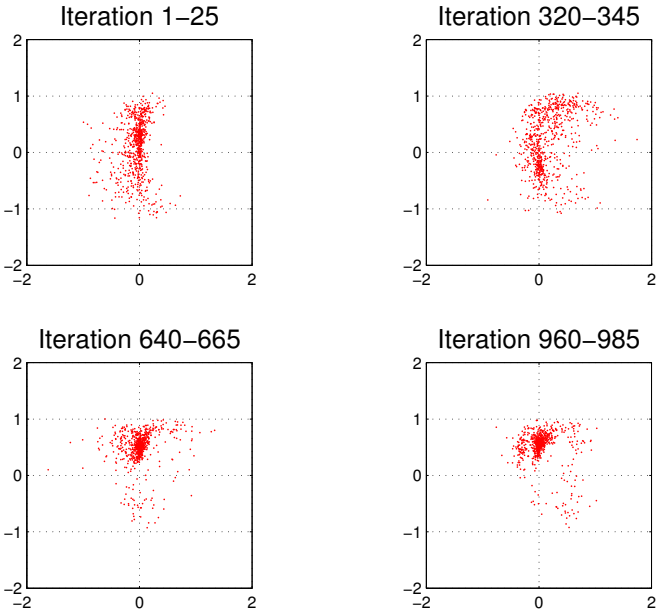
RLLab: Swimmer Gather

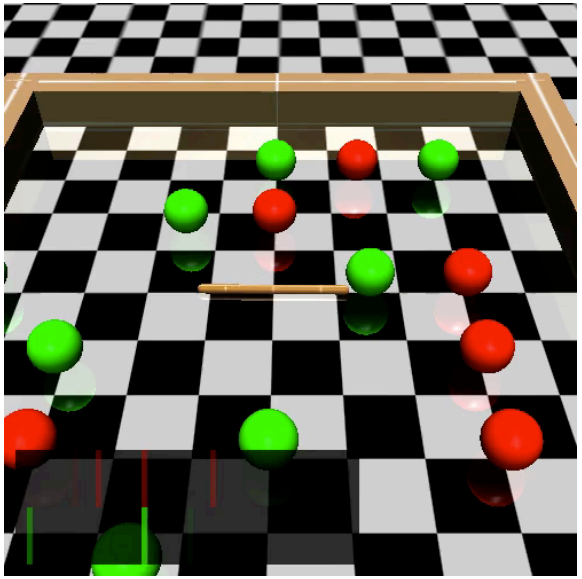


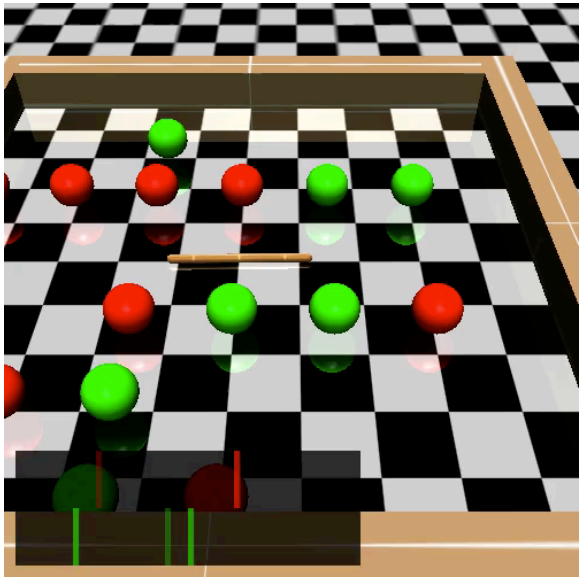
- Control a worm with two flexible joints, swimming in a 2D viscous fluid
- Reward +1 for eating green apples and -1 for touching red bombs
- Reverse self-play even though the environment is not strictly symmetric
- No apples or bombs during self-play
- Use only location when deciding Bob's success

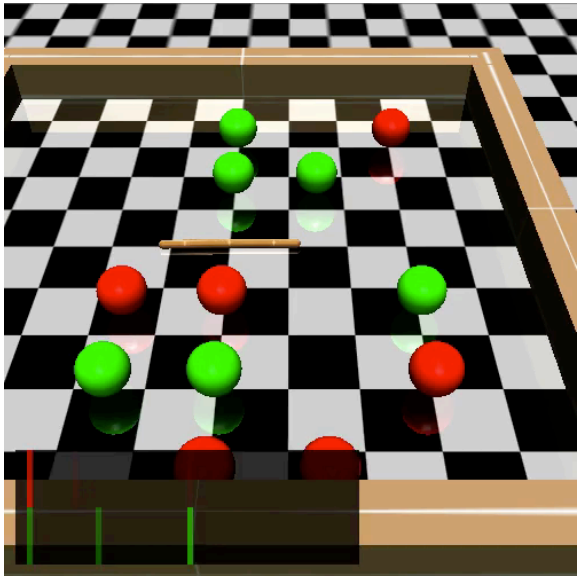

Target Reward
Iteration
Reinforce + Self-play
TRPO + VIME
TRPO + SimHash

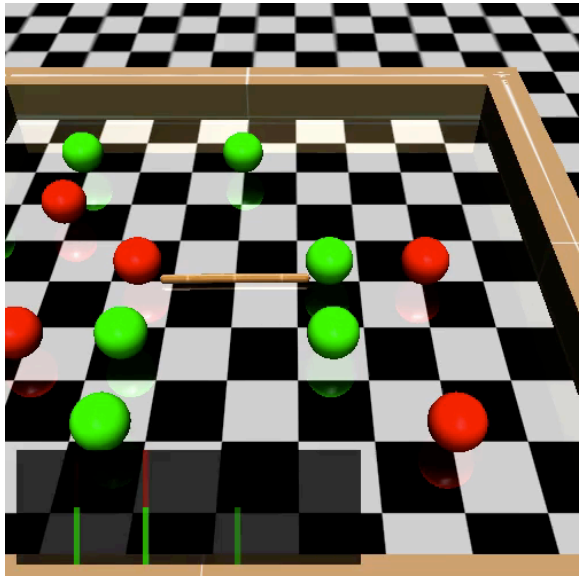

(a)
Target task reward
Iteration
(c)
Alice actions
Iteration
(b)
Internal reward
Alice
Bob
Iteration
(d)
Alice distance
Iteration


Iteration 1–25
Iteration 320–345
Iteration 640–665
Iteration 960–985









Discussion

https://arxiv.org/abs/1703.05407

- Simple methods that works with discrete and continuous environments
- Meta–exploration for Alice
  - We want Alice to propose diverse set of tasks
  - But Alice focuses on the single best task
  - Multiple Alices?
- Future works:
  - Alice explicitly mark the target state
  - Alice propose task by communication without doing it
  - Alice propose a hypothesis and Bob test it