Skill Emergence and Transfer in Multi-Agent Environments

Ingmar Kanitscheider*, Bowen Baker*, Todor Markov* and Igor Mordatch
OpenAI
San Francisco, CA
(ingmar,bowen,todor,mordatch)@openai.com

ABSTRACT

A central problem in training artificial agents to perform complex skills is specifying appropriate cost functions whose optimization will lead to the desired behavior. Specifying detailed cost functions is laborious and often inefficient. The training of agents in competitive and cooperative multi-agent environments provides an avenue to circumvent these limitations: By competition and cooperation agents provide to each other a natural curriculum that can lead to the emergence of complicated skills, even if the rewards of the multi-agent game are simple [1].

Here we explore the emergence of complex strategies and skills in a simple hide and seek game simulated in a 3-D physics environment. We show that training using deep reinforcement learning (RL) leads to the emergence of several rounds of strategies, counter-strategies, and skills composed of several sequential behaviors. Our results suggest that training multiple agents in a sufficiently complex environment using large scale RL can lead to open-ended development of behavior. We furthermore show that emergent skills can be extracted and re-used in distinct environments. This skill transfer is both a useful evaluation metric for multi-agent emergence and suggests that multi-agent pre-training might be a useful strategy to generate targeted skills.

ACM Reference Format:
Ingmar Kanitscheider*, Bowen Baker*, Todor Markov* and Igor Mordatch. 2019. Skill Emergence and Transfer in Multi-Agent Environments. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3326794

1 METHODS

To test the emergence of skills, we train a "seeker" agent and a team of one or more "hider" agents to play hide and seek in an environment with movable boxes and ramps that is conducive to the emergence of complex skills. The agent policies are trained using multi-agent deep reinforcement learning. For evaluation, we fine-tune the network policy to a different environment with distinct rewards and compare the performance to a baseline policy that has been trained in the new environment from scratch.

* These authors contributed equally to this paper. Author ordering is random.

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326794

1.1 Environments

The environment is populated with agents, movable rectangular shaped boxes and ramps, and immovable walls. Agents are simulated as spherical objects that can move in a plane and rotate around their z-axis. The interactions of agents and objects are simulated using Mujoco [6], a high-fidelty 3-D physical simulation environment.

Agents observe the 3-dimensional locations and velocities of other agents and objects. Observations are masked if there is no line of sight between agent and object due to occlusions by other objects or walls or if the object lies outside of the agent's observation cone. Agents can move boxes and ramps either by pushing them in front of them or by a specific manipulation action that constrains the relative position of agents and objects to be fixed. In addition, agents can "glue" objects at their current location and prevent other agents from moving them. Translation and rotation actions are discretized in 11 different bins, 5 for each direction and one for no movement, and the gluing and manipulation actions are binary.

In hide-and-seek environments, the seeker receives reward +1.0 for each time step in which they can establish direct line of sight to at least one hider, and -1.0 otherwise. Hiders receive reward +1.0 if neither they themselves nor other hiders are visible by the seeker, and -1.0 otherwise. The shared reward of hiders encourages collaborative strategies.

1.2 Training

Hider and seeker policies are parameterized using a single deep network and are distinguished by a binary observation that is passed to the input of the network. Because there are many entities of the same type in the environment, such as agents, boxes, and ramps, we take inspiration from literature on inductive logical programming [4] and relational reinforcement learning [2] and use a network structure that is *permutation invariant*. Namely, each entity is encoded with an embedding layer that shares weights with all embedding layers for entities of the same type. Entities are then pooled to a fixed dimensional vector followed by an LSTM [3]. To train the agent policies, we use stochastic gradient descent and proximal policy optimization (PPO) [5] for multi-agent reinforcement learning with self-play. In order to prevent training instabilities in multi-agent self-play, we sample 25%-50% of opponent policies uniformly from previous stages of training.

1.3 Fine-Tuning Evaluation

We fine-tune the policies trained in the hide-and-seek task on two types of single-agent evaluation task: The seeker policy is fine-tuned in the "object tracking" environment, in which an agent needs to track an object that is moving at variable speed behind obstructing walls. Tracking is successful whenever the agent is

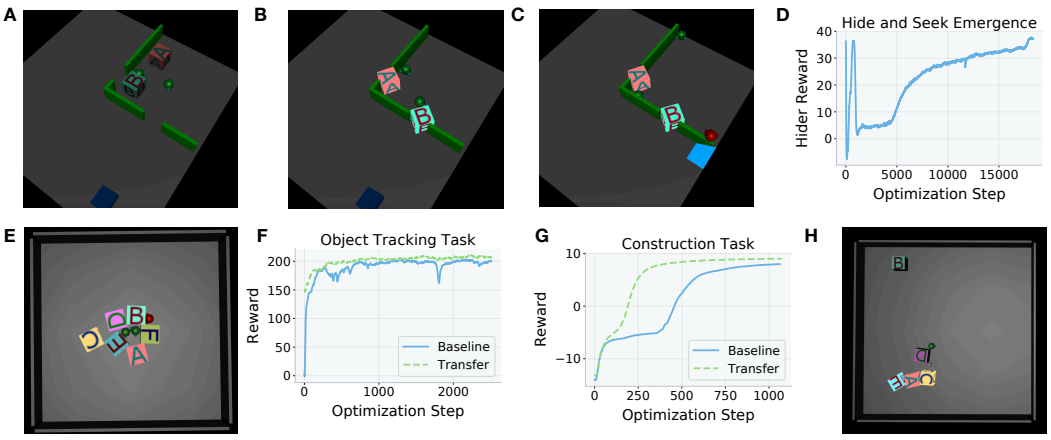

Figure 1: Emergence and transfer results: (A) Green hiders, the red seeker, boxes, and ramps are spawned randomly. (B) Hiders move boxes to the doors in their room and lock them there; glued objects are highlighted. (C) The seeker moves the ramp to the edge of the wall and jumps over. (D) Hider reward over training: Minima and maxima correspond to the emergence of seeker and hider strategies, respectively. (E) Make-shift fort in single room (F) Transfer to object tracking environment (G) Transfer to blueprint construction (H) Example of blueprint construction.

oriented in the direction of the object. The hider policy is fine-tuned in the "blueprint construction" environment, in which 4 boxes need to be moved to specific goal positions (Figure 1H). At each time step, the agent receives a negative reward based on the weighted averaged distance between the boxes and the goal positions. Once all boxes are accurately placed (within a margin of error), the agent wins a large positive reward and the episode ends. In each task, we compare the fine-tuning training curve to the training curve of a baseline policy that is trained from scratch.

2 RESULTS

We first train agents (one seeker and two hiders) in an environment that contains two rooms (one large and one small) that are separated from each other by a wall with two openings. The environment also contain two blocks and a ramp (Figure 1A-C). Boxes and hiders are randomly sampled in the small room, seekers in the large room and the ramp is sampled in either room. The training curve displays the emergence of several rounds of strategies and counter-strategies (Figure 1D): At first, seekers learn to seek efficiently, causing a drop in hider reward. Then, hiders learn to use the boxes to block the openings, thereby preventing the seeker to enter the small room (Figure 1B). Subsequently, the seeker learns moving the ramp next to the wall separating the two rooms and climbing over the wall (Figure 1C). Finally, the hiders learn two different strategies to prevent the seeker from using the ramp: Pulling the ramp inside the small room before blocking the openings and gluing the ramp at a location in the large room far away from the separating wall.

In a second set of experiments we train one seeker and two hiders in a single-room environment with 6 boxes. After successful training hiders learn to position boxes around them to build a make-shift fort, a skill that requires the sequential execution of several distinct behaviors (Figure 1E). We then evaluate the emergent policies on the two transfer tasks: In both object tracking (Figure 1F) and blueprint construction environment (Figure 1G) we observe that fine-tuning leads to higher performance early in training and better asymptotic performance compared to baseline training.

These results are remarkable for several reasons: First, strategies and counter-strategies emerge solely from simple game rules, multi-agent self-play and RL training. While previous work found that multi-agent competition in physical environments can lead to complex skills [1], this work suggests that sufficiently complex environments can give rise to an open-ended evolution of behavior. Second, the transfer results suggest that the skills acquired during emergence training can be efficiently extracted and re-used in distinct environments.

3 CONCLUSIONS

We set out to explore the emergence of complex skills in a multi-agent hide and seek task. We found that multi-agent competition and collaboration using deep RL can lead to the emergence of several rounds of strategies and counter-strategies and skills that require the sequential execution of several distinct actions and that can be transferred to substantially distinct environments. Future experiments will explore to which extent multi-agent RL at large scale in rich environment will lead to an open-ended evolution of ever-more complex behaviors.

REFERENCES

[1] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2017. Emergent Complexity via Multi-Agent Competition. (2017). arXiv:arXiv:1710.03748
[2] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. 2001. Relational Reinforcement Learning. *Machine Learning* 43, 1/2 (2001), 7–52. https://doi.org/10.1023/a:1007694015589
[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
[4] Stephen Muggleton. 1991. Inductive logic programming. *New Generation Computing* 8, 4 (feb 1991), 295–318. https://doi.org/10.1007/bf03037089
[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (2017). arXiv:arXiv:1707.06347
[6] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control.. In *IROS*. IEEE, 5026–5033. http://www.mujoco.org