Arena: A General Evaluation Platform and Building Toolkit for Multi-Agent Intelligence

Yuhang Song,[1] Andrzej Wojcicki,[3] Thomas Lukasiewicz,[1] Jianyi Wang,[4] Abi Aryan,[5] Zhenghua Xu,[1,2]* Mai Xu,[4] Zihan Ding,[6] Lianlong Wu[1]

[1]Department of Computer Science, University of Oxford, United Kingdom
[2]State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, China
[3]Lighthouse, [4]School of Electronic and Information Engineering, Beihang University, China
[5]University of California, Los Angeles, United States, [6]Imperial College London, United Kingdom
{yuhang.song, thomas.lukasiewicz}@cs.ox.ac.uk, andrzej@wojcicki.xyz, zhenghua.xu@hebut.edu.cn

Abstract

Learning agents that are not only capable of taking tests, but also innovating is becoming a hot topic in AI. One of the most promising paths towards this vision is multi-agent learning, where agents act as the environment for each other, and improving each agent means proposing new problems for others. However, existing evaluation platforms are either not compatible with multi-agent settings, or limited to a specific game. That is, there is not yet a general evaluation platform for research on multi-agent intelligence. To this end, we introduce Arena, a general evaluation platform for multi-agent intelligence with 35 games of diverse logics and representations. Furthermore, multi-agent intelligence is still at the stage where many problems remain unexplored. Therefore, we provide a building toolkit for researchers to easily invent and build novel multi-agent problems from the provided game set based on a GUI-configurable social tree and five basic multi-agent reward schemes. Finally, we provide Python implementations of five state-of-the-art deep multi-agent reinforcement learning baselines. Along with the baseline implementations, we release a set of 100 best agents/teams that we can train with different training schemes for each game, as the base for evaluating agents with population performance. As such, the research community can perform comparisons under a stable and uniform standard. All the implementations and accompanied tutorials have been open-sourced for the community at https://sites.google.com/view/arena-unity/.

Introduction

Modern learning algorithms are more of outstanding test-takers, but less of innovators, i.e., the ceiling of an agent's intelligence may be limited by the complexity of its environment (Leibo et al. 2019). Thus, the emergence of innovation is becoming a hot topic for AI. One of the most promising paths towards such a vision is learning via social interaction, i.e., multi-agent learning. In multi-agent learning, how the agents should beat the opponents or collaborate with each other is not defined or limited by the creator of the environment, e.g., the inventor of the ancient Go never defines what strategies are good. However, enormous and sophisticated strategies are invented while a population of human players/artificial agents evolves by improving themselves over the others, i.e., each agent is acting as an environment for the others and improving itself means proposing new problems for the others.

To study a new class of intelligence, general evaluation platforms with diverse games are milestones that push forward the research to the next levels. For example, ALE (Bellemare et al. 2013), Mujoco (Todorov, Erez, and Tassa 2012), and DM-Suite (Tassa et al. 2018) are the most spread general evaluation platforms that greatly accelerate the research in general reinforcement learning. However, there is no such general evaluation platform for multi-agent intelligence. Although some platforms support multi-agent settings (Wydmuch, Kempka, and Jaśkowski 2018; Vinyals et al. 2017), they are not general evaluation platforms, i.e., built for specific games. Thus, in this paper, we propose the first general evaluation platform for multi-agent intelligence, called *Arena*, containing 35 multi-agent games in total, with diverse logics and representations; see Fig. 1.

Apart from training and evaluation, multi-agent intelligence research is still at a stage where many problems remain undiscovered or unexplored. Thus, the second contribution of Arena is a building toolkit for multi-agent intelligence, enabling the easy creation of different multi-agent scenarios. For example, in the sample game in Fig. 2 (a), after defining the basic behavior of the agent (i.e., moving and turning) and the "alive" state of the agent (i.e., it stays on the playground), it can be extended to different multi-agent scenarios with minimal effort. For example, (1) five players fight each other until only one agent is left alive (see Fig. 2 (b)), or (2) 5×2 players form 2 teams, and each agent fights for its own team until all players in a team are dead (see Fig. 2 (c)), or (3) multiple players form multiple teams in hierarchies, where the collaboration and competition relationships between the teams are customized (see Fig. 3).

Thus, Arena is not just a research platform for the evaluation with a fixed set of games, but also a building toolkit for researchers to invent and build novel multi-agent problems.

To achieve the above vision of building a toolkit for multi-agent intelligence, (1) we provide a GUI-configurable tree that defines the social structure of agents, called *social tree*;

*Corresponding author: Zhenghua Xu.

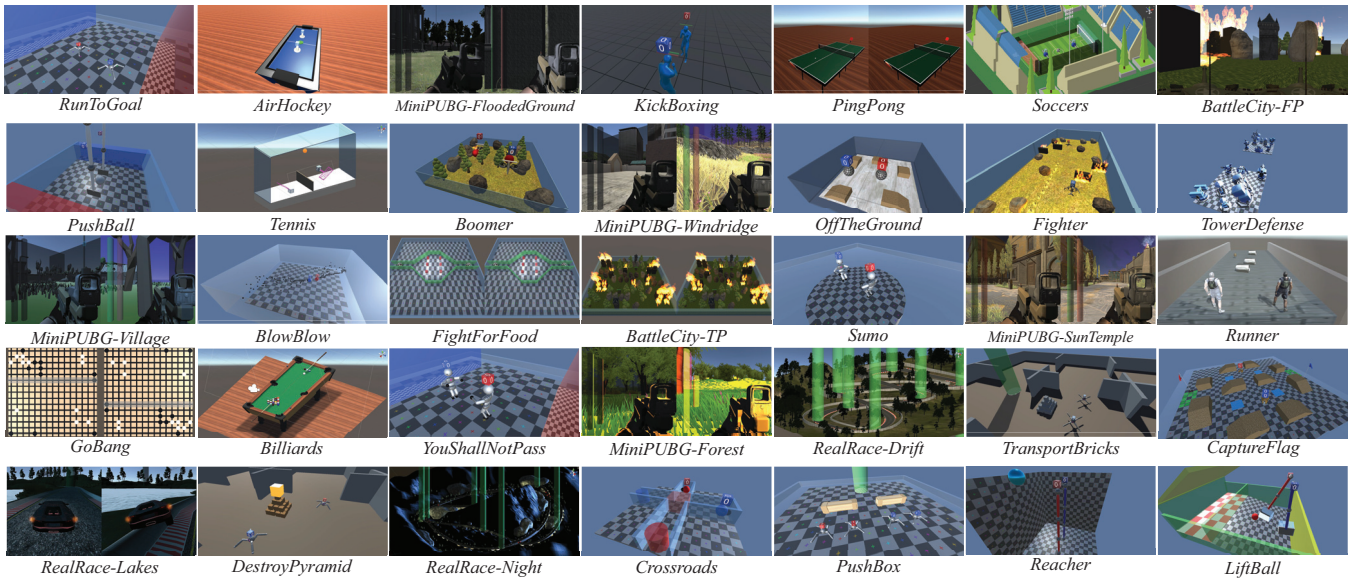


Figure 1: Game set of Arena.

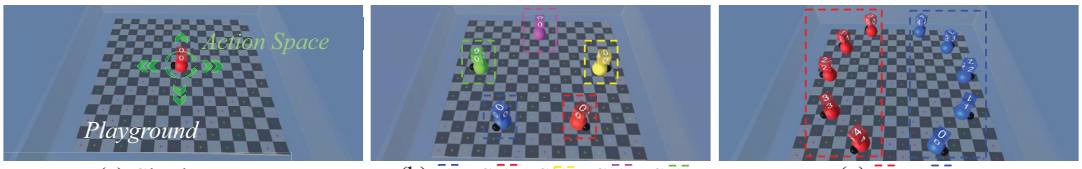
and (2) based on the social tree, we propose 5 basic multi-agent reward schemes (BMaRSs), which define different social paradigms at each node in the social tree. Specifically, each BMaRS is a restriction applied to the reward function, so it corresponds to a batch of reward functions that can lead to a specific social paradigm. For each BMaRS, Arena provides multiple ready-to-use reward functions, simplifying the construction of games with complex social relationships. Furthermore, if the agent is controlling each joint of a robot, it has long been a burden for researchers that low-level intelligence (such as the basic skill of moving) must first be built, before they can study high-level multi-agent intelligence (Heess et al. 2017). Thus, Arena provides many ready-to-use dense reward functions in each BMaRS that handle such low-level intelligence. Additionally, Arena also offers a verification option for customized reward functions, so the researchers can make sure that the programmed reward functions lie in one of the BMaRSs that produces a specific social paradigm. Thus, with the above efforts towards a building toolkit for multi-agent intelligence and the provided set of 35 games for a general evaluation platform, one can easily customize a set of games of a new social paradigm to study a yet unexplored problem.

Finally, we provide Python implementations of several state-of-the-art deep multi-agent reinforcement learning baselines, which can be used as starting points for the development of novel multi-agent algorithms, as well as the validation of new environments. Along with the baseline implementations, we also release a set of 100 best agents/teams that we can train with different training schemes for each game, as the base for evaluating agents with population performance (Balduzzi et al. 2019; 2018). So, the research community can perform comparisons under a stable and uniform standard.

To summarize, this paper's contributions are as follows: (1) a general evaluation platform for multi-agent intelligence with a set of diverse games, most of which are new to the community or still stand as a challenge for state-of-the-art algorithms, (2) a building toolkit for multi-agent games, enabling the easy creation of new social paradigms based on GUI-configurable social trees and BMaRSs, (3) the baseline implementations of 5 state-of-the-art multi-agent algorithms for both competitive and collaborative settings, and (4) sets of benchmark agents/teams for the community to conduct stable and uniform population evaluation (Balduzzi et al. 2018). Code for games, building toolkit, and baselines, as well as all corresponding tutorials have been released online at https://sites.google.com/view/arena-unity/.

The Platform

State-of-the-Art Engine. The engine behind Arena is the world-leading game engine *Unity* (Juliani et al. 2018), which provides Arena with several desirable features on rendering, physics, customizability, and community. There are also other choices of popular engines. Some platforms contain a wide set of diverse games (Bellemare et al. 2013; Nichol et al. 2018; Perez-Liebana et al. 2016; OpenAI 2016). However, they are designed mostly for single-agent scenarios and are extremely hard to customize (adding multiple players or creating new games), since the games are provided as compiled binary ROMs. Other downsides of these choices include deterministic environments, unrealistic rendering, and unrealistic physics. Other platforms (Todorov, Erez, and Tassa 2012; Tassa et al. 2018) are, in nature, more physics engines than game engines, which lack a visual editor for easily creating customized games, and cannot handle more "game-like" features, such as instantiating and destroying objects in real-time during the simulation. The rest of the platforms are limited in the sense that they are built for specific tasks, such as for first-person shoot-



(a) Single agent (b) 1VS1VS1VS1VS1 (c) 5VS5

Figure 2: Game examples of the extensible multi-agent platform.

ing (Wydmuch, Kempka, and Jaśkowski 2018), Real-Time Strategy (RTS) (Tian et al. 2017), vision understanding (Qiu et al. 2017), in-door scene understanding (Handa et al. 2016; Brodeur et al. 2017; Savva et al. 2017; Chang et al. 2017; Puig et al. 2018; Gao et al. 2019), surviving (Suarez et al. 2019), and interaction (Wu et al. 2018; Savva et al. 2019; Kolve et al. 2017), or specific games, such as Starcraft (Vinyals et al. 2017) and Dota2 (OpenAI 2018). Thus, creating a general evaluation platform on these engines is not a reasonable choice. DeepMind Lab (Beattie et al. 2016), Psychlab (Leibo et al. 2018), and Malmo (Johnson et al. 2016) are more appropriate choices when building a customizable general evaluation platform. However, the main drawbacks of the above engines are tied to their dated nature. The rendering system of these engines are either low-polygon pixelated (Malmo, based on Minecraft) or outdated (DeepMind Lab and Psychlab, based on Quake III). The physics systems of these engines are either rudimentary (Malmo), or have a gap (Juliani et al. 2018) to the physical world (DeepMind Lab and Psychlab). Besides, they are all incompatible with a visual editor, making it quite cumbersome to build customized scenarios.

To summarize, built on Unity, Arena has the following advantages over other platforms: (1) realistic rendering, so that features, such as complex lighting, textures, and shaders, are fully handled by the background engine and easily produced in a customized game, (2) realistic physics, so that enough and realistic stochasticity is introduced in the game and transferring a policy learned within a simulator to the real world is easier, (3) user-friendly visual editor, so that building new multi-agent scenarios in Arena is easy, and (4) a large and active development community, so that creating new games is easy with millions of off-the-shelf assets.

Game Sets Towards General Intelligence. The first contribution of Arena is to provide a set of multi-agent games with diverse game logics and representations, so that it may push forward the research of general multi-agent intelligence. Specifically, Arena provides: (1) 27 new games that are not yet studied in the community, (2) 8 games, of which the basic logics are inspired by other research, but equipped with realistic rendering effects, physics engine, and all features described in the following two paragraphs, such as extensibility to other social paradigms, and (3) interface to the popular stand-alone domain StarCraft. The game set is shown in Fig. 1. For more detailed information, see Tables 2-7 in the extended paper (Song et al. 2019).

Building Toolkit for Multi-Agent Environments. As the second contribution, we provide a building toolkit for multi-agent environments: we provide (1) a GUI-configurable social tree that defines how agents are grouped together with each other, and (2) 5 basic multi-agent reward schemes (BMaRSs) applied on each node in the social tree, so that different social relationships can be easily built and verified, and low-level intelligence (like motor skills) can be handled.

Other Features. Learning to communicate is an important research area in multi-agent intelligence (Das et al. 2017; Mordatch and Abbeel 2018). Thus, Arena provides a broadcast board at each node of the social tree (accessible for any agent as a child of the node), which enables the study of learning communication at each level. Also, global states may be used in research for different purposes (Lowe et al. 2017; Gupta, Egorov, and Kochenderfer 2017; Foerster et al. 2017; 2018). Thus, Arena provides the option to broadcast it to all agents. Besides, a top-down view of the global game is often appreciated for visualizing population behavior (Johnson et al. 2016; Wydmuch, Kempka, and Jaśkowski 2018; Jaderberg et al. 2018; Liu et al. 2019). Thus, Arena by default enables this option. Finally, there is a necessity for competitive agents to evaluate against human players, and also a research trend for collaborative agents to team up with human players. Thus, Arena provides a gaming interface for humans, so that a human player can take the place of any agent in the game.

Basic Multi-Agent Reward Schemes and Social Trees

Preliminaries. We consider a Markov game as defined in (Littman 1994), consisting of multiple agents $x \in \mathcal{X}$, a finite global state space $\mathcal{S}$, a finite action space $\mathcal{A}_x$ for each agent x, and a bounded-step reward space $r_{x,t} \in \mathbb{R}$ for each agent x. The environment consists of a transition function $g : \mathcal{S} \times \times\{\mathcal{A}_x : x \in \mathcal{X}\} \to \mathcal{S}$, which is a stochastic function $s_{t+1} \sim g\big(s_t, (a_{x,t})_{x\in\mathcal{X}}\big)$, a reward function for each agent $f_x : \mathcal{S} \times \times\{\mathcal{A}_x : x \in \mathcal{X}\} \to \mathbb{R}$, which is a deterministic function $r_{x,t+1} = f_x\big(s_t, (a_{x,t})_{x\in\mathcal{X}}\big)$, a joint reward function $f = (f_x)_{x\in\mathcal{X}}$, and episode reward $R_x^f = \sum_{t=1}^{T} r_{x,t}$ for each agent x under the joint reward function f. For the agent, we consider that it observes $s_{x,t} \in \mathcal{S}_x$, where $\mathcal{S}_x$ consists of a part of the information from the global state space $\mathcal{S}$. Thus, we have a policy $\pi_x : \mathcal{S}_x \to \mathcal{A}_x$, which is a stochastic function $a_{x,t} \sim \pi_x(s_{x,t})$. Besides, we consider that the agent x can take a policy π_x from a set of policies Π_x and assume

that the random seed of all sampling operations is k, which is sampled from the whole seed space $\mathcal{K}$.

We investigate the effect of $\{x : x \in \mathcal{X}\}$ and $\{\pi_x : \pi_x \in \Pi_x\}$ on $\{R_x^f : x \in \mathcal{X}\}$. By applying different restrictions on the effect, we have different BMaRSs, each one of which is a set of joint reward functions $\mathcal{F} = \{f : \cdot\}$ that produce a similar effect on the population $\mathcal{X}$. The term reward scheme first appears in (Tampuu et al. 2017) as a tabular, which is applied to a special case of Pong. While we define it in a general form and show that many examples are special cases within this general form.

In a non-sequential setting (normal-form game), the reward scheme serves a similar purpose as the payoff matrix (Myerson 2013), which is also represented as a tabular. See Lemma 2 in the extended paper (Song et al. 2019) for how the payoff matrix is aligned with BMaRSs. In the following, we define 5 different BMaRSs. Along defining these BMaRSs, we also describe the ready-to-use reward functions f within these BMaRSs, which is provided by Arena as a dropdown list.

Non-learnable BMaRSs ($\mathcal{F}^{NL}$) are a set of joint reward functions f as follows:

$$\mathcal{F}^{NL} = \big\{f : \forall k \in \mathcal{K}, \forall x \in \mathcal{X}, \forall \pi_x \in \Pi_x, \partial R_x^f / \partial \pi_x = \mathbf{0}\big\}, \tag{1}$$

where $\mathbf{0}$ is a zero matrix of the same size and shape as the parameter space that defines π_x. Intuitively, $\mathcal{F}^{NL}$ means that R_x^f for any agent $x \in \mathcal{X}$ cannot be optimized by improving its policy π_x.

Isolated BMaRSs ($\mathcal{F}^{IS}$) are a set of joint reward functions f as follows:

$$\mathcal{F}^{IS} = \big\{f : f \notin \mathcal{F}^{NL} \text{ and } \forall k \in \mathcal{K}, \forall x \in \mathcal{X}, \forall x' \in \mathcal{X}\backslash\{x\}, \forall \pi_x \in \Pi_x, \forall \pi_{x'} \in \Pi_{x'}, \frac{\partial R_x^f}{\partial \pi_{x'}} = \mathbf{0}\big\}, \tag{2}$$

Intuitively, $\mathcal{F}^{IS}$ means that the episode reward R_x^f received by any agent $x \in \mathcal{X}$ is not related to any policy $\pi_{x'}$ taken by any other agent $x' \in \mathcal{X}\backslash\{x\}$.

Reward functions f_x in f of $\mathcal{F}^{IS}$ are often called *internal reward functions* in other multi-agent approaches (Hendtlass 2004; Jaderberg et al. 2018; Bansal et al. 2018), meaning that apart from the reward functions applied at a population level (such as win/loss), which are too sparse to learn, there are also reward functions directing the learning process towards receiving the population-level rewards, but are more frequently available, i.e., more dense (Heess et al. 2017). $\mathcal{F}^{IS}$ is especially practical if the agent is a robot requiring continuous control of applying force on each of its joints, which means basic motor skills (such as moving) need to be learned before generating population-level intelligence. Thus, we provide f in $\mathcal{F}^{IS}$ of energy cost, punishment of applying a big force, encouragement of keeping a steady velocity, and moving distance towards target.

Competitive BMaRSs ($\mathcal{F}^{CP}$), inspired by (Cai and Daskalakis 2011), are defined as

$$\mathcal{F}^{CP} = \big\{f : f \notin \mathcal{F}^{NL} \cup \mathcal{F}^{IS} \text{ and } \forall k \in \mathcal{K}, \forall x \in \mathcal{X}, \forall \pi_x \in \Pi_x, \forall \pi_{x'} \in \Pi_{x'}, \frac{\partial \int_{x' \in \mathcal{X}} R_{x'}^f dx'}{\partial \pi_x} = \mathbf{0}\big\}, \tag{3}$$

which intuitively means that for any agent $x \in \mathcal{X}$, taking any possible policy $\pi_x \in \Pi_x$, the sum of the episode reward of all agents will not change. If the episode length is 1, it expresses a classic multi-player zero-sum game (Cai and Daskalakis 2011). Useful examples of f within $\mathcal{F}^{CP}$ are: (1) agents fight for a limited amount of resources that are always exhausted at the end of the episode, and the agent is rewarded for the amount of resources that it gained, and (2) fight till death, and the reward is given based on the order of death (the reward can also be based on the reversed order, so that the one departing the game first receives the highest reward, such as in some poker games, the one who first discards all cards wins). *Rock, Paper, and Scissors* in normal-form game (Myerson 2013) and *Cyclic Game* in (Balduzzi et al. 2019) are both special cases of $\mathcal{F}^{CP}$; see Lemmas 2 and 3 in the extended paper (Song et al. 2019).

Collaborative BMaRSs ($\mathcal{F}^{CL}$), inspired by (Cai and Daskalakis 2011), are defined as

$$\mathcal{F}^{CL} = \big\{f : f \notin \mathcal{F}^{NL} \cup \mathcal{F}^{IS} \text{ and } \forall k \in \mathcal{K}, \forall x \in \mathcal{X}, \forall x' \in \mathcal{X}\backslash\{x\}, \forall \pi_x \in \Pi_x, \forall \pi_{x'} \in \Pi_{x'}, \frac{\partial R_{x'}^f}{\partial R_x^f} \geqslant 0\big\}, \tag{4}$$

which, intuitively, means that there is no conflict of interest ($\partial R_{x'}^f / \partial R_x^f < 0$) for any pair of agents (x', x). Besides, since $f \notin \mathcal{F}^{NL} \cup \mathcal{F}^{IS}$, there is at least one pair of agents (x, x') that makes $\partial R_{x'}^f / \partial R_x^f > 0$. This indicates that this pair of agents shares a common interest, so that improving R_x^f for agent x means improving $R_{x'}^f$ for agent x'. The most common example of f within $\mathcal{F}^{CL}$ is that f_x for all $x \in \mathcal{X}$ is identical, such as the moving distance of an object that can be pushed forward by the joint effort of multiple agents, or the alive duration of the population (as long as there is at least one agent alive in the population, the population is alive). Thus, we provide f in $\mathcal{F}^{CL}$: living time of the team (both positive and negative, since some games require the team to survive as long as possible, while other games require the team to depart as early as possible, such as poker).

Competitive and Collaborative Mixed BMaRSs ($\mathcal{F}^{CC}$) are defined as a catch-all for any other than the above four ones. First, the term $\partial \int_{x' \in \mathcal{X}} R_{x'}^f dx' / \partial \pi_x = \mathbf{0}$ in (3) can be written as $\int_{x' \in \mathcal{X}} \partial R_{x'}^f / \partial R_x^f dx' = 0$ (see Lemma 1 in extended paper (Song et al. 2019), which makes an alternative (3). Considering $\mathcal{F}^{CP}$ in this alternative (3) and $\mathcal{F}^{CL}$ in (4), an intuitive explanation of $\mathcal{F}^{CC}$ is that there exist circumstances when $\partial R_{x'}^f / \partial R_x^f < 0$, meaning that the agents are competitive at this point. But the derivative of total interest $\int_{x' \in \mathcal{X}} \partial R_{x'}^f / \partial R_x^f dx'$ is not always 0; thus, the total interest can be maximized with specific policies, meaning that the agents are collaborative at this point.

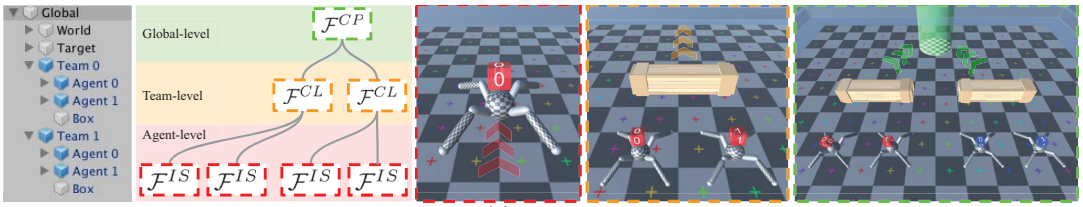


Figure 3: An example of a social tree and BMaRSs applied on it.

Apart from providing several practical f in each BMaRS, we also provide a verification option for each BMaRS, meaning that one can customize an f and use this verification option to make sure that the programmed f lies in a specific BMaRS. The implementation of verification option can be found in Section 1 in the extended paper (Song et al. 2019).

The Social Tree. The BMaRSs defined above apply to an agent group of all sizes. To define more complex and structured social paradigms, we use a tree structure (*social tree*) to organize the agents and apply BMaRSs on each node of the tree. We illustrate this by an example. The GUI interface in Fig. 3 (a) defines a tree structure in Fig. 3 (b), representing a population of 4 agents.

The tree structure can be easily reconfigured by dragging, duplicating, or deleting nodes in the GUI interface in Fig. 3 (a). In this example, each agent has an agent-level BMaRS. The agent is a robot ant, so that the agent-level BMaRSs are $\mathcal{F}^{IS}$, specifically, the option of *ant-motion* that directs the learning towards basic motion skills such as moving forward, as shown in Fig. 3 (c). Each two agents form a *team* (which is a set of agents or teams), the two agents have team-level BMaRSs. In this example, the two robot ants collaborate with each other to push a box forward, as shown in Fig. 3 (d). Thus, the team-level BMaRSs are $\mathcal{F}^{CL}$, specifically, the moving distance of the box. On the two teams, we have global-level BMaRSs. In this example, the two teams are set to have a match regarding which team pushes its box to the target point first, as shown in Fig. 3 (e). Thus, the global-level BMaRSs are $\mathcal{F}^{CP}$, specifically, the ranking of the box reaching the target. The final reward function applied to each agent is a weighted sum of the above three BMaRSs at three levels. One can imagine defining a social tree of more than three levels, where small teams form bigger teams, and BMaRSs are defined at each node to give more complex and structured social problems. After defining the social tree and applying BMaRSs on each node, the environment is ready for use with an abstraction layer handling everything else, such as assigning viewports to each agent in the window, applying the team color, displaying the agent ID, and generating a top-down view.

The Learning Agents

The Baselines. We provide Python implementations of several state-of-the-art baselines that can be used as starting points for the development of novel multi-agent algorithms, as well as for the validation of new environments. Specifically, we first implement a fully decentralized system, where each agent is a self-contained PPO (Schulman et al. 2017), with independent critic, actor, and optimizer. We also implement two state-of-the-art methods based on self-play in (OpenAI 2018) (SP) and population-based training in (Jaderberg et al. 2018; DeepMind 2019) (PB). For collaborative agents, we implement two state of the arts: centralized critic (Lowe et al. 2017) (CC) and centralized critic with a counterfactual baseline (Foerster et al. 2018) (CF).

The Evaluation Metric. It is recently raising attention that evaluating an agent against a single-agent or hand-coded bot is unstable and misleading (Balduzzi et al. 2018). Thus, the population performance is introduced to evaluate an agent's (or an agent group's) performance among a base population. To enable population evaluation, we release 100 best agents, which we can train with different training schemes for each game as the base population. One can call the provided function to get the ranking of an agent among the base population, or get the averaged ranking of a population among the base population. Moreover, we provide a human ranking among the base population, which provides an indication of human-level intelligence in the game. We will accept the submission of agents from the community as well as keep implementing algorithms introduced in the future, so that the base population will be upgraded, as the level of research in multi-agent intelligence advances.

Experiments

Experiments are conducted from three aspects. First, we evaluate our game set from the perspective of stochasticity, realistic rendering, and simulation speed, all of which are presented in the extended paper (Song et al. 2019) due to page limit. Other advantages from the Unity engine have been verified by (Juliani et al. 2018). Second, we evaluate our design of the extensible multi-agent building toolkit with a case study, showing that by applying different social trees and BMaRSs, different population-level strategies can be learned. Third, we report the experimental results of 5 baselines that we implemented and show that by using the provided population performance evaluation metric, the training progress can be visualized in a less noisy and more analyzable way.

Case Study of Social Tree and BMaRSs. We use the game *Crossroads* from Arena to study the effectiveness of

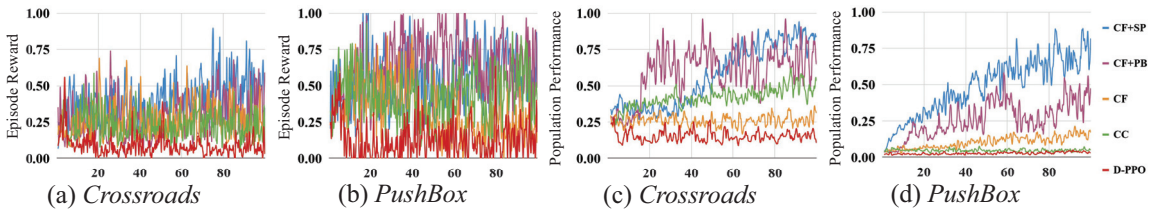


Figure 4: Visualizing training progress over episode reward (a,b) and population performance (c,d) of different baselines: D-PPO (Decentralized Proximal Policy Optimization), SP (Self-Play), PB (Population-Based training), CC (Centralized Critic), and CF (Counterfactual Baseline).

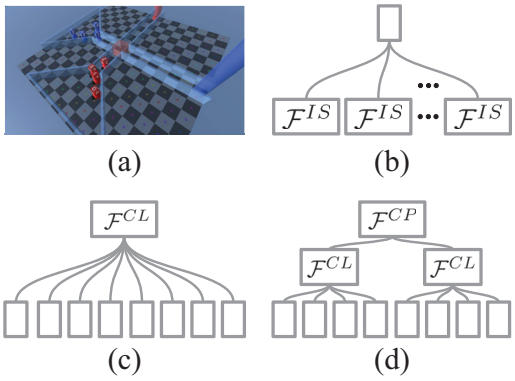


Figure 5: Case Study of Social Tree and BMaRSs.

the proposed social tree and BMaRSs via designing different social paradigms. Specifically, in the game *Crossroads* shown in Fig. 5 (a), the agent can move and turn, the final goal of the agent is to reach the target on the other side of the crossroad. By defining different social trees and applying different BMaRSs, as shown in Fig. 5 (b) to (d), the agents learn different strategies. In Fig. 5 (b), *isolated* BMaRSs ($\mathcal{F}^{IS}$) are applied to all agents, i.e., each agent minimizes the time that it takes to reach the target. The result shows that the learned agents simply rush forward, and they easily crash with each other at the center of the crossroad, producing a traffic jam. In Fig. 5 (c), *collaborative* BMaRSs ($\mathcal{F}^{CL}$) are applied to the parent node of all agents, i.e., all agents are rewarded with the time that the last one of them takes to reach the target. The result shows that the agents learn to wait for each other to go across the crossroad, so that they can all get across as efficiently as possible. In Fig. 5 (d), *collaborative* BMaRSs ($\mathcal{F}^{CL}$) are applied on the parent node of every 4 agents (which form a team), and *competitive* BMaRSs ($\mathcal{F}^{CP}$) are applied on the parent node of the two teams. Specifically speaking, each two agents in the same team are rewarded with the same reward, and the reward is 1 for the team that gets all of its agents to the target first, 0 for the other team. The results show that each team learns to block the road of the other team with one agent, so that the other agents in the team can get across undisturbed. Then, the agent that blocks the road leaves for the target, after all its teammates have reached the target.

Baselines and Evaluation Metric. We compare 5 baselines on two games: (1) *Crossroads* in Fig. 5 (a) with the BMaRS settings of Fig. 5 (d) and (2) *PushBox* in Fig. 3 (e) with the BMaRS settings of Fig. 3 (b). The BMaRS settings of both games contain competitive as well as collaborative social relationships, i.e., multiple agents form collaborative teams, and teams compete with each other. Thus, we investigate SP and PB baselines at the level of teams competing with each other, as well as investigate CC and CF baselines at the level of agents collaborating with each other in a team. As can be seen, the curve of episode reward shown in Fig. 4 (a) and (b) is extremely noisy, as the environment is non-stationary with the strategy of other collaborators and/or competitors evolving during the training. However, in Fig. 4 (c) and (d), which is the curve of ranking in the released base population, i.e., population performance, all methods are comparable with clear performance gaps.

Related Work

Surveys of multi-agent intelligence research can be found in (Hernandez-Leal, Kartal, and Taylor 2018). Different ideas have been explored on competitive and collaborative multi-agent settings.

Collaborative Settings. The simplest way to deploy multi-agent collaborative systems is to make each agent have a completely independent learning process (fully decentralized) (Matignon, Laurent, and Le Fort-Piat 2012). However, collaborative behaviors are hardly observed under such fully decentralized setting; thus, a fully centralized system is utilized in (Peng et al. 2017), where the policy has access to the global state and is shared by all agents. However, it is impractical, since the global state is mostly unavailable in practice, and the system does not support extending the number of agents. Thus, centralized training and decentralized execution are gaining attention (Kraemer and Banerjee 2016). For multi-agent systems, this idea is mostly explored under actor-critic algorithms (Foerster et al. 2018). Other ideas include using a joint action-value function, (Lauer and Riedmiller 2004) addressing the variance problem by a large batch size (Bansal et al. 2018), and learning grounded cooperative communication protocols between agents (Foerster et al. 2016).

Competitive Settings. Competitive multi-agent intelligence originally comes from computational game theory

(Bowling et al. 2015). Later on, deep multi-agent reinforcement learning (D-MARL) is preferred, due to its scalability, and as it achieves notable advances on two-player games, such as Poker and Go (Moravčík et al. 2017; Silver et al. 2017). Later, D-MARL was applied to more diverse problems, such as high-dimensional video games (OpenAI 2018; DeepMind 2019) and those involving physics control (Bansal et al. 2018). When solving more practical problems, many issues have been raised, such as ensuring diversity amongst agents (Marivate 2015), avoiding overfitting to the policy of the opponents (Lanctot et al. 2017). Many ideas address such issues (Kleiman-Weiner et al. 2016). Following on D-MARL, a very promising recent direction is self-play (Tesauro 1995). Fictitious self-play (Heinrich and Silver 2016) first shows promising performance on the competitive game Leduc Poker. However, as the stability and parallelizability are improving with the invention of new reinforcement learning algorithms, state-of-the-art approaches adopt a simpler form of self-play (OpenAI 2018), which produces a superior-human intelligence on large video games, like Dota2. Another promising recent idea is population-based training, as adopted in StarCraft (DeepMind 2019).

Summary and Outlook

This paper has introduced the first general evaluation platform for multi-agent intelligence research. Besides, with the efforts on a building toolkit of multi-agent environments, the platform also allows for easily building new multi-agent problems. Additionally, with the released implementations of several state-of-the-art baselines, researchers can start their adventure instantly. Finally, by releasing a base population, the community can conduct comparisons under a stable and uniform evaluation metric.

Acknowledgments. This work was supported by the China Scholarship Council under the State Scholarship Fund, by the Graduate Travel and Special Project Grants from the Somerville College of the University of Oxford, by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, by the AXA Reseach Fund, by the National Natural Science Foundation of China under the grants 61906063, 61876013, and 61922009, by the Natural Science Foundation of Tianjin under the grant 19JCQNJC00400, by the "100 Talents Plan" of Hebei Province, and by the Yuanguang Scholar Fund of Hebei University of Technology.

References

Balduzzi, D.; Tuyls, K.; Perolat, J.; and Graepel, T. 2018. Re-evaluating evaluation. In *NIPS*.

Balduzzi, D.; Garnelo, M.; Bachrach, Y.; Czarnecki, W. M.; Perolat, J.; Jaderberg, M.; and Graepel, T. 2019. Open-ended learning in symmetric zero-sum games. *arXiv:1901.08106*.

Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2018. Emergent complexity via multi-agent competition. In *ICLR*.

Beattie, C.; Leibo, J. Z.; Teplyashin, D.; Ward, T.; Wainwright, M.; Küttler, H.; Lefrancq, A.; Green, S.; Valdés, V.; Sadik, A.; et al. 2016. Deepmind lab. *arXiv:1612.03801*.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The Arcade learning environment: An evaluation platform for general agents. *JAIR*.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science*.

Brodeur, S.; Perez, E.; Anand, A.; Golemo, F.; Celotti, L.; Strub, F.; Rouat, J.; Larochelle, H.; and Courville, A. 2017. Home: A household multimodal environment. *arXiv:1711.11017*.

Cai, Y., and Daskalakis, C. 2011. On minmax theorems for multiplayer games. In *SIAM*.

Chang, A.; Dai, A.; Funkhouser, T.; Halber, M.; Niessner, M.; Savva, M.; Song, S.; Zeng, A.; and Zhang, Y. 2017. Matterport3D: Learning from RGB-D data in indoor environments. *arXiv:1709.06158*.

Das, A.; Kottur, S.; Moura, J. M.; Lee, S.; and Batra, D. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*.

DeepMind. 2019. AlphaStar: Mastering the real-time strategy game StarCraft II. In *DeepMind*.

Foerster, J.; Assael, I. A.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P. H.; Kohli, P.; and Whiteson, S. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *ICML*.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *AAAI*.

Gao, X.; Gong, R.; Shu, T.; Xie, X.; Wang, S.; and Zhu, S.-C. 2019. VRKitchen: An interactive 3D environment for learning real life cooking tasks. In *ICML Workshop RL4RealLife*.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS*.

Handa, A.; Pătrăucean, V.; Stent, S.; and Cipolla, R. 2016. SceneNet: An annotated model generator for indoor scene understanding. In *ICRA*.

Heess, N.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S.; Riedmiller, M.; et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv:1707.02286*.

Heinrich, J., and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv:1603.01121*.

Hendtlass, T. 2004. An introduction to collective intelligence. In *SAP*.

Hernandez-Leal, P.; Kartal, B.; and Taylor, M. E. 2018. Is multiagent deep reinforcement learning the answer or the question? A brief survey. *arXiv preprint arXiv:1810.05587*.

Jaderberg, M.; Czarnecki, W. M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A. G.; Beattie, C.; Rabinowitz, N. C.; Morcos, A. S.; Ruderman, A.; et al. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. In *CoRR*.

Johnson, M.; Hofmann, K.; Hutton, T.; and Bignell, D. 2016. The Malmo platform for artificial intelligence experimentation. In *IJCAI*.

Juliani, A.; Berges, V.-P.; Vckay, E.; Gao, Y.; Henry, H.; Mattar, M.; and Lange, D. 2018. Unity: A general platform for intelligent agents. *arXiv:1809.02627*.

Kleiman-Weiner, M.; Ho, M. K.; Austerweil, J. L.; Littman, M. L.; and Tenenbaum, J. B. 2016. Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *CogSci*.

Kolve, E.; Mottaghi, R.; Gordon, D.; Zhu, Y.; Gupta, A.; and Farhadi, A. 2017. AI2-THOR: An interactive 3D environment for visual AI. *arXiv:1712.05474*.

Kraemer, L., and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *NIPS*.

Lauer, M., and Riedmiller, M. 2004. Reinforcement learning for stochastic cooperative multi-agent systems. In *AAMAS*.

Leibo, J. Z.; d'Autume, C. d. M.; Zoran, D.; Amos, D.; Beattie, C.; Anderson, K.; Castañeda, A. G.; Sanchez, M.; Green, S.; Gruslys, A.; et al. 2018. Psychlab: A psychology laboratory for deep reinforcement learning agents. *arXiv:1801.08116*.

Leibo, J. Z.; Hughes, E.; Lanctot, M.; and Graepel, T. 2019. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv:1903.00742*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *ICML*.

Liu, S.; Lever, G.; Merel, J.; Tunyasuvunakool, S.; Heess, N.; and Graepel, T. 2019. Emergent coordination through competition. In *ICLR*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NIPS*.

Marivate, V. N. 2015. *Improved Empirical Methods in Reinforcement-Learning Evaluation*. Ph.D. Dissertation, Rutgers University-Graduate School-New Brunswick.

Matignon, L.; Laurent, G. J.; and Le Fort-Piat, N. 2012. Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems. *The Knowledge Engineering Review*.

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*.

Mordatch, I., and Abbeel, P. 2018. Emergence of grounded compositional language in multi-agent populations. In *AAAI*.

Myerson, R. B. 2013. *Game theory*. Harvard University Press.

Nichol, A.; Pfau, V.; Hesse, C.; Klimov, O.; and Schulman, J. 2018. Gotta learn fast: A new benchmark for generalization in RL. *arXiv:1804.03720*.

OpenAI. 2016. Openai universe. In *https://github.com/openai/universe*.

OpenAI. 2018. Openai five. *https://blog.openai.com/openai-five/*.

Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent bidirectionally-coordinated nets for learning to play StarCraft combat games. *arXiv:1703.10069*.

Perez-Liebana, D.; Samothrakis, S.; Togelius, J.; Schaul, T.; and Lucas, S. M. 2016. General video game AI: Competition, challenges and opportunities. In *AAAI*.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. VirtualHome: Simulating household activities via programs. In *CVPR*.

Qiu, W.; Zhong, F.; Zhang, Y.; Qiao, S.; Xiao, Z.; Kim, T. S.; and Wang, Y. 2017. UnrealCV: Virtual worlds for computer vision. In *International Conference on Multimedia*.

Savva, M.; Chang, A. X.; Dosovitskiy, A.; Funkhouser, T.; and Koltun, V. 2017. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*.

Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; et al. 2019. Habitat: A platform for embodied AI research. *arXiv:1904.01201*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv:1707.06347*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of Go without human knowledge. *Nature*.

Song, Y.; Wang, J.; Lukasiewicz, T.; Xu, Z.; Xu, M.; Ding, Z.; and Wu, L. 2019. Arena: A general evaluation platform and building toolkit for multi-agent intelligence. *arXiv preprint arXiv:1905.08085*.

Suarez, J.; Du, Y.; Isola, P.; and Mordatch, I. 2019. Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*.

Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; Casas, D. d. L.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; et al. 2018. Deepmind control suite. *arXiv:1801.00690*.

Tesauro, G. 1995. Temporal difference learning and TD-Gammon. *Communications of ACM*.

Tian, Y.; Gong, Q.; Shang, W.; Wu, Y.; and Zitnick, C. L. 2017. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. In *NIPS*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *IROS*.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. StarCraft II: A new challenge for reinforcement learning. *arXiv:1708.04782*.

Wu, Y.; Wu, Y.; Gkioxari, G.; and Tian, Y. 2018. Building generalizable agents with a realistic and rich 3D environment. *arXiv:1801.02209*.

Wydmuch, M.; Kempka, M.; and Jaśkowski, W. 2018. ViZDoom competitions: Playing doom from pixels. *IEEE Transactions on Games*.