Reward-Reinforced Generative Adversarial Networks for Multi-Agent Systems

Changgang Zheng, Shufan Yang, Juan Marcelo Parra-Ullauri, Antonio Garcia-Dominguez, and Nelly Bencomo

***Abstract*—Multi-agent systems deliver highly resilient and adaptable solutions for common problems in telecommunications, aerospace, and industrial robotics. However, achieving an optimal global goal remains a persistent obstacle for collaborative multi-agent systems, where learning affects the behaviour of more than one agent. A number of nonlinear function approximation methods have been proposed for solving the Bellman equation, which describe a recursive format of an optimal policy. However, how to leverage the value distribution based on reinforcement learning, and how to improve the efficiency and efficacy of such systems remain a challenge. In this work, we developed a reward-reinforced generative adversarial network to represent the distribution of the value function, replacing the approximation of Bellman updates. We demonstrated our method is resilient and outperforms other conventional reinforcement learning methods. This method is also applied to a practical case study: maximising the number of user connections to autonomous airborne base stations in a mobile communication network. Our method maximises the data likelihood using a cost function under which agents have optimal learned behaviours. This reward-reinforced generative adversarial network can be used as a generic framework for multi-agent learning at the system level.**

***Index Terms*—multi-agent, reinforcement learning, GAN, reward-reinforced GAN, airborne base station (ABS).**

I. Introduction

PRACTICAL applications of multi-agent systems, such as autonomous airborne base stations, need accurate real-time state estimation and learning capabilities to achieve optimised trajectory planning with a maximised global goal (the maximum number of users connected). The airborne base stations have a high degree of autonomy, serving their own users within the signal coverage range. At the same time, however, multiple airborne stations need to be coordinated with each other to serve the users in a constantly changing environment: users move around, and neighbouring base stations produce signal interference. It is challenging to create optimised trajectories for all agents (base stations) at the same time, while considering the current state of neighbouring agents and their performed actions.

Traditional centralised algorithms for multi-agent systems allowing agents to share their information with a central node are computationally expensive. Reinforcement learning-based distributed algorithms can only explicitly share information with their neighbours, which is computationally efficient compared with centralised algorithms. Reinforcement learning has had great successes in solving multi-agent collaborative tasks when three constraints are met: i) having an environment, ii) having a reward generation process either through an approximated function or a simple greedy method, and iii) having agents interacting with the environment. However, those methods often have high variance in their results. In addition, agents may start to compete instead of collaborating with each other due to scarce resources, such as communication channel capacities. To date, most solutions concentrate on the learning of individual agents, but neglect approaches at the system level [1].

A conventional reward mechanism in a reinforcement learning framework is designed to produce a cost-benefit assessment of a given action, and subsequently apply a high or low reward in a heuristic search approach [2]. Hjeldm *et al*. proposed an approach to intelligent drone tracking using reinforcement learning [3]. This type of mechanism required continuous feedback from the environment, which means this technique scaled poorly to a large team with multi-agents; furthermore, traditional Bayesian approaches to reinforcement learning problems cannot model the inherent variability of the action of the state. Deep Q Network (DQN) reinforcement learning method used multiple hidden layers of a neural network to fit a state value and a state-action value distribution into a single agent [3].

The goal of classical reinforcement learning was to find an optimal policy that could maximize rewards. Instead of searching for a policy, researchers proposed a heuristic search which aimed to improve an estimated state value function, in order to minimize the expected distance between the value function's output and agents's tates [4]. Since heuristic search methods were impossible to emulate all states in an environment. Other researchers focused on finding a reward function that 'experts'

Manuscript received September 7, 2020; revised January 4, 2021, February 23, 2021, and April 14, 2021; accepted April 26, 2021. This work was supported by U.K. EPSRC Project Twenty20Insight (EP/T017627/1). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Amit Konar. *(Corresponding authors: Shufan Yang.)*

Changgang Zheng is with the Department of Engineering Science, University of Oxford, Oxford OX1 2JD, U.K. (e-mail: 2289258Z@student.gla.ac.uk).

Shufan Yang is with the School of Computing, Edinburgh Napier University, Edinburgh EH11 4BN, U.K., and also with the Center of Medical and Industrial Ultrasonics, University of Glasgow, Glasgow G12 8QQ, U.K. (e-mail: s.yang@napier.ac.uk).

Juan Marcelo Parra-Ullauri, Antonio Garcia-Dominguez, and Nelly Bencomo are with the School of Engineering and Applied Science, Aston University, Birmingham B4 7ET, U.K. (e-mail: j.parra-ullauri@aston.ac.uk; a.garcia-dominguez@aston.ac.uk; nelly@acm.org).

This article has supplementary downloadable material available at https://doi.org/10.1109/TETCI.2021.3082204, provided by the authors.

Digital Object Identifier 10.1109/TETCI.2021.3082204

are implicitly optimising either from states to actions, or from states to reward values, often called inverse reinforcement learning [5]. Two distinct approaches of inverse reinforcement learning were explored: the first method was proposed to seek a way for directly approximating the reward function by tuning inputs [5]; the second one was focused on learning a policy that matched its action with demonstrated behaviour [6]. The first approach depended on selecting a complete reward structure or set of feature functions which can be hard to generalise since various environments may use different reward features [7]. The second approach was sensitive for deterministic actions since the optimization became theoretically impossible if the underlying reward function was a non-convex function [8]. The inverse reinforcement learning method had generalisation issues since it used "experts" to demonstrate how the optimal behaviour should be, which was unrealistic in real-life. Furthermore, agents based on inverse reinforcement learning methods were only assumed to follow an optimal policy and were prone to only take suboptimal actions by the agents.

By contrast, generative adversarial networks (GANs) have shown remarkable results at generating data that imitates a data distribution in a multi-agent system [9]. For example, an extended generative adversarial imitation learning framework was proposed in [10] to train an action policy network for autonomous vehicles, with only one action of the vehicle modelled. Although a large number of studies on image data augmentation tasks provided promising results, to our knowledge, far fewer research efforts have been devoted to the application of adversarial training to collaborative multi-agent systems.

Inspired by the study [11], we propose a deep generative model and an opponent discriminator model followed by a two-player min-max game formula [3] as the single learning process for modelling the behaviour of the entire team. We built on a reward mapping method that combines adversarial generative networks with the use of reinforcement learning to produce domain-specific rewards. The generative model is used to examine the distribution of the value function for all agents, in order to reduce the training steps while optimising the overall result. As aforementioned, we applied our model to a practical case study: the optimised deployment of airborne base stations in mobile communications. In our approach, the generator is trained to generate a predicted reward map and trained adversely with discriminator, so that the networks optimise the properties of distributed multi-agent behaviours in an adversarial fashion. Our results show that Reward-Reinforced Generative Adversarial Networks (RR-GANs) are able to achieve the best global goal values, compared to other state-of-the-art reinforcement learning methods. This is thanks to how our model can represent the distribution of the value function, replacing the approximation of Bellman updates via an adversarial learning method.

II. Methodology

The modern field of reinforcement learning is based on optimal control [12], which came into use during the 1960's to describe the problem of designing a controller to minimize a measurement of a dynamic system's behaviour. The Bellman equation, as defined in (1), states that the value of the initial state must equal the value of the expected next state, plus the expected reward along the way. Instead of modelling the expectation of each value, we design a new method which applies Bellman's equation to provide an approximated optimal value function. The distribution of Z is characterized by the interaction of two distributions: the reward R distribution, the probability that the tuple (s, a) at time t will lead to the next distribution Z at time $t + 1$.

$$Z(s,a) \equiv^{D} R(s,a) + Z(\acute{S}, \acute{A}) \tag{1}$$

To solve the Bellman equation, we aim to recover a estimation probability distribution over the reward function from "expert" demonstrations using generative adversarial modelling. At the initial stage, agents exploit the environment and use maximum likelihood rewards to choose best actions. At the execution stage, the generative modelling is used for generated reward mapping. Over time the agent is supposed to customize its actions to the environment so as to maximize the sum of this reward.

A. Generic GAN

The generator G uses the original environment input o and a randomly selected reward experience n to generate the predicted reward map p. Or, in mathematical notation, $G : \{o, n\} \rightarrow p$. The adversarial discriminator D tries to classify the o concatenating with p and the o concatenating with real reward map n.

$$\begin{aligned}&\mathcal{L}_{GAN}(G, D)\\ &\quad = \mathbb{E}_{o,p}[\log D(o, p)] + \mathbb{E}_{o,n}[\log(1 - D(o, G(o, n))]\end{aligned} \tag{2}$$

As shown in (2), the G aims at minimizing the objective $\mathcal{L}_{GAN}$ and the D tries to maximize the objective $\mathcal{L}_{GAN}$.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{o,p,n}\left[\|p - G(o, n)\|_2\right] \tag{3}$$

We substitute the definition given by (3) into (4). The generator objective for optimising the divergence between generated data and the input real data is:

$$G_{global} = \arg\min_{G}\max_{D} \mathcal{L}_{GAN}(G, D) + \lambda\mathcal{L}_{L1}(G) \tag{4}$$

B. Rr-Gan

The data used for learning is obtained by a greedy method. An agent only takes the action when the number of connected users increases, where a reward is given. At the learning stage, the user distribution (a part of the environment) is received as the input to the generator network: the target location is the one with the maximum reward.

The GAN is a multi-agent system with N agents and the loss function $\mathcal{L}_{GAN}(G, D)$ is denoted as $f([\theta_1, \theta_2, \ldots, \theta_N], \phi)$, where θ_N is the parameter vector of the N^{th} agent and ϕ is the parameter vector of the environment (i.e. the user distribution), and where the objective functions of generator are $f(\theta_N)$ and the objective function of discriminator $-f(\theta_N)$. The optimum is a Nash equilibrium defined as in (5).

$$\Theta \in argmax f(\Theta, \phi^*), \phi^* \in argmax - f(\Theta^*, \phi) \tag{5}$$

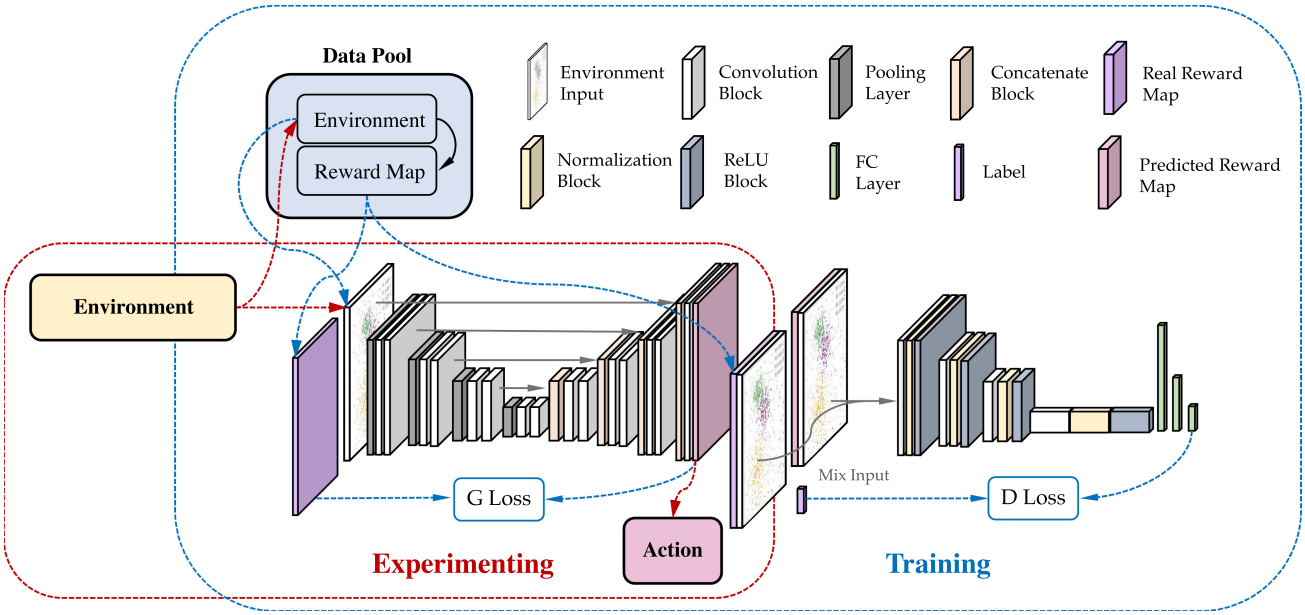


Fig. 1. General diagram of the proposed RR-GAN.

The maximum likelihood estimator solves $\frac{d\mathcal{L}}{dx} = 0$, where $x = (\Theta, \phi)^T$. The Nash-equilibrium points pose the following properties:

$$\frac{d\mathcal{L}_{GAN}}{d\Theta} = \begin{bmatrix} df(x*) \\ -df(x*) \end{bmatrix} = 0 \tag{6}$$

$$T(x^*) = \begin{bmatrix} \frac{d^2 f(x^*)}{d^2\theta} & \frac{df(x^*)}{d\theta d\Phi} \\ -\frac{d^2 f(x^*)}{d\theta d\Phi} & -\frac{d^2 f(x*)}{d^2\Phi} \end{bmatrix} \leq 0 \tag{7}$$

We substitute the definition of $T(x)$ and gradient of $\frac{d\mathcal{L}_{GAN}}{d\theta}$, as given by (6) and (7) into (8). Using consensus optimization [13], the objective function of the generator can add a 2-norm as a regularization term. The updated rule for the generator network becomes:

$$x_{k+1} = x_k + \alpha \frac{d\mathcal{L}_{GAN}}{d\theta} + \gamma T(x)^T \tag{8}$$

When γ and α are proper values using hypothesis testing, the optimisation will be locally stable at the Nash equilibrium if $T(x)$ is inevitable. Hence, a stationary distribution exists, and the GAN architecture can converge to this stationary distribution. Furthermore, to avoid low convergence speed or even divergence, we used the regulated RMSProp optimisation [14]. The empirical results reported in the results section demonstrate the ability of our method to solve the reinforcement learning tasks. Further mathematical proof requires an argument similar to [14].

RR-GAN is used to generate the predicted reward maps from the current environment input and randomly selected past rewards from each agent. This method reduces the fitting difficulty of the network of the discriminator D. The network mainly needs to learn from the data source (i.e. user distribution), and the time varying environment (i.e. how many users are within range of each airborne base station). In the initial stage an agent is placed in a situation without knowledge of any goals or other information about other agents. As an agent acts in the environment, a reinforcement reward governs its actions: increasing the number of connected users is rewarded. By only giving the agent a reward when connected users are reached, the agent learns to achieve its goals. However, since each agent competes (due to signal interference), generative modelling is used to generate reward mapping at each execution stage. Over time, the agent customizes its actions to the environment to maximize the sum of the rewards.

As shown in Fig. 1, real-time information from the environment (in this case, the user distribution) allows the network to self-adjust. Agents will act according to the predicted reward map for moving in the right direction. At the training stage, all agents will use a greedy method to exploit the maximum users connections in their own signal coverage areas to generate a reward map (blue arrows in Fig. 1). After several iterations, all agents adjust their behaviours according to the generated reward maps (red arrows in Fig. 1).

[15].

The pseudo code of RR-GAN is shown below:

Two aspects can be modified to enhance the stability of GAN training: model setup, and optimization methods. In this work we used the adversarial learned kernels in a batch training.

In our experiment, the generative neural network is composed of both the generator model and discriminator model. Our network structure follows Goodfellow's published work [16]. The generator model consists of a U-net network, which is composed of two fully connected layers and eight deconvolutional layers. The discriminator is composed of five convolution layers followed by two fully connected layers. The convolution and deconvolution layer come with a batch normalization layer.

Algorithm 1: Reward-Reinforced GAN.

Data: Environment E, Predicted Reward Map $R_{predicted}$, Reward Map R
Result: Trained Reward-Reinforced GAN parameters θ

```
Randomly generate environment E;
Exploit rewards using greedy methods and store reward map R
for every epoch do
    Train Generator by using stored E and R, (Update θ);
    Train Discriminator by using stored R and R_predicted;
    for every iteration do
        if environment changed then
            Generator generate R_predicted;
            Find global optimal position;
        end
        Do action toward global optimal
    end
end
```

TABLE I
DETAILED MODEL ARCHITECTURE—REWARD MAP PREDICTION NETWORK GENERATOR

Layer name	Block type	Output resolution	Output depth
Down 1	Conv Block	100×100	64
Down 2	Bottleneck	50×50	128
Down 3	Bottleneck	25×25	256
Down 4	Bottleneck	12×12	512
Down 5	Bottleneck	6×6	1024
Up 1	Bottleneck	12×12	512
Up 2	Bottleneck	25×25	256
Up 3	Bottleneck	50×50	128
Up 4	Bottleneck	100×100	64
Out Conv	Conv Block	100×100	n

TABLE II
BOTTLENECK ARCHITECTURE

Layer name	Out Direction	Kernel size	Stride size
Conv 1+BatchNorm+ReLU		3×3	1
Conv 2+BatchNorm+ReLU	Up	3×3	1
Max Pool	Down	2×2	2

The output layer of the generator uses the sigmoid function as the activation function while all other and discriminator network are based on ReLu as activation functions. Both generator and discriminator networks are trained under the Adam solver [17] with a learning rate of 0.0001 [18], [19]. The parameters are selected with reference to some other GANs [18], [19].

The specific structures of the generator and the discriminator are shown in Tables I, II, and III.

III. EXPERIMENTAL DESIGN AND RESULTS

In this section, we applied the proposed approach to control the trajectory of airborne base stations in a mobile communication system. It is worth noting that this approach can solve many other tasks; for instance, trajectory prediction for industrial robotic collaboration in warehouses [20]. More importantly, it can also be applied to common multi-agent problems: multi-agent learning can exhibit unexpected interactions between agents as they gravitate toward an equilibrium [21].

TABLE III
DETAILED MODEL ARCHITECTURE—REWARD MAP PREDICTION NETWORK DISCRIMINATOR

Layer name	Kernel size	Output depth	Stride size
Conv 1+BatchNorm+ReLU	4×4	64	1
Conv 2+BatchNorm+ReLU	4×4	128	1
Conv 3+BatchNorm+ReLU	4×4	256	1
Conv 4+BatchNorm+ReLU	4×4	512	1
Conv 5+BatchNorm+ReLU	4×4	512	1
Fully Connected 1+Dropout	—	128	—
Fully Connected 2+ReLU	—	1	—

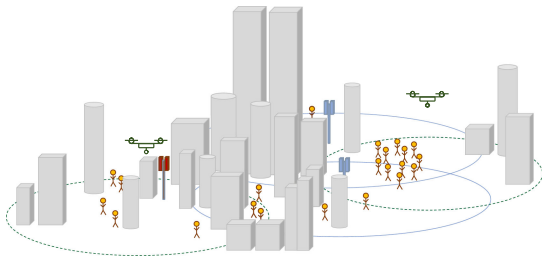


Fig. 2. Possible Airborne Base Stations using conditions: a static base station (red) has malfunctioned; people gather at an area with poor coverage from existing base stations.

A. Problem Statement

In a mobile communication network, there are various scenarios where sudden spikes in connection demands can be generated, such as large social events (e.g. business campaigns, political rallies, university opening ceremonies or other unexpected events), or emergencies like the malfunction of the existing mobile base stations. Airborne base stations can provide a fast response to these situations, as shown in Fig. 2. It is important to precisely control the movement of the airborne base stations, in order to connect the maximum number of users to the mobile network.

While providing mobile communications with airborne base stations, interference and noise are two potential impairments to signal quality [22]. Two parameters model communication channel (signal-to-interference-plus-noise ratio (SNIR), and reference signal received power (RSRP)) can be computed as the Power Density multiplied by the Antenna Effective Area [22], [23]. SNIR and RSRP are used for evaluating the potential signal quality between the mobile station and a user device. A threshold is used to distinguish the connectivity between users and base stations. The overall attenuation effects can be represented by path loss (*free-space loss*). The behaviour of the airborne base stations depends on their proximity to the neighbouring airborne base stations, and how many users are being served. Our objective is to learn a joined distribution of the reward map of each airborne base station for making long-term predictions about the optimised number of users staying connected to a mobile network.

TABLE IV
COMMUNICATION AND ENVIRONMENT PARAMETERS

Parameters	Value
Communication Parameters	
Lowest SINR requirment, θ_μ	0dB
UAV-base station antenna directivity angle, ϕ_{ap}	60°
Carrier frequency, f_c	2.4GHz
Drone transmition power	40dBm
Bandwidth	200kHz
Noise power spectral density	$10^{-20.4}$W/Hz
Environment Parameters	
Random seed	19
Length of the area	100m
Width of the area	100m
Drone step size	10m
Cluster number	4
Ratio of users of each cluster	4:5:6:6
Total number of users, N_u	1050
Drone numbers, N_d	1,2,4,8
User Height, h_u	1.5m
Drone Height, h_d	30m

The communication model follows the formula in (9). All parameters in Table IV are remained the same in all experiments.

$$L_s = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{9}$$

In (10), c is the speed of light in metres per second; EIRP is the Equivalent Isotropic Radiated Power (the drone transmit power) in watts; f_c is the carrier frequency in hertz; d is the distance between the user and the airborne base station in metres and the wavelength is $\lambda = \frac{c}{f_c}$.

$$RSRP_{n,u} = \frac{EIRP}{L_s} = \frac{EIRP\ c^2}{(4\pi f_c d)^2} \tag{10}$$

In (10), the RSRP for the link between the user u and the airborne base station n is calculated according to the EIRP, and the free space path loss is given in (9).

$$SINR_{n,u} = \frac{RSRP_{n,u}}{N + \sum_{\forall i \neq n} RSRP_{i,u}} \tag{11}$$

The Signal to Interference plus Noise Ratio (SINR) is defined in (11), where N is the noise power in Watts.

As an example, the distribution of users (and their mobile phones) can be modeled by a bivariate distribution consisting of a mixture of distinct Gaussian clusters [24]. The probability of users appearing can be treated as a mixture of two time-invariant 2-dimensional Gaussian distributions varying with time [25], [26]. Each airborne base station have many users (mobile phones) [24]. For each Gaussian user cluster, the "user appearing" probability also follows a 2-dimensional Gaussian distribution. The users in each mobile communication cell follows the distribution in (12).

$$(X_{pl}, Y_{pl}) \sim N\left(X_{gc}, \sigma^2_{gc1}, Y_{Gc}, \sigma^2_{gc2}, \rho_{gc}\right) \tag{12}$$

The distribution $f_{pl}(x, y)$ of user locations follows a 2-dimensional Gaussian distribution, where X_{gc}, Y_{Gc} is the cluster center. The standard deviation, including $gc1$, $gc2$, $pl1$ and $pl2$, are random variables.

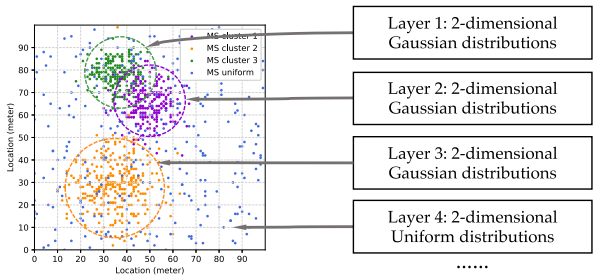


Fig. 3. User distribution.

$$(X_{cn}, Y_{cn}) \sim N\left(X_{pl}, \sigma^2_{pl1}, Y_{pl}, \sigma^2_{pl2}, \rho_{pl}\right) \tag{13}$$

The user distribution $f_{\text{cluster}}(x_{cn}, y_{cn})$ for cluster n, in (13), follows a 2-dimensional Gaussian distribution [27].

$$f_{MS}(x, y) = \frac{1}{k+1}\left(\text{uniform} + \sum_{n=1}^{k} \text{cluster}_n\right) \tag{14}$$

The distribution of users is defined in four layers in our simulation, with three 2-dimensional Gaussian layers (layer 1, layer 2 and layer 3) and one uniform distribution layer (layer 4), as shown in Fig. 3. The orange cluster in Fig. 3 shows the biggest size of the cluster compared with the green and purple clusters, where these three clusters simulate an emergency scenario with many users. The fourth layer is a uniform distribution layer, where the normal distribution is used to simulate a general scenario. The number of users at each Gaussian-distributed layer are 300, 250, and 200 respectively. The standard deviations of each cluster are 10, 7, and 6 (14).

B. Results

Each airborne base station chooses one action among five options at each iteration: move "east," "west," "south," or "north," or stay at the same spot. All base stations start at the right bottom corner. If an airborne base station moves in a direction which increases the number of users, the reward for this airborne base station will increase. However, if this airborne base station is too close to the neighbouring base station, the neighbouring base station will lose users. The global goal is to provide connectivity for the maximum number of users.

We compare the performance of the proposed method with the following baseline models: Q-learning, SARSA, DQN and k-means. All the baseline models use the same input features, and are trained with the same iterations and the same learning rate.

- Q-learning is a model-free off-policy algorithm [28], which provides agents with the capability of learning to act optimally in Markovian domains [29]. The reward map is exploited and updated through the Q-table [30]. Fig. 4 shows the dispersion of the global rewards over each training episode in the airborne base station system using the Q-learning algorithm. As the graph describes, the global rewards stabilize around values between 250 and 300, with

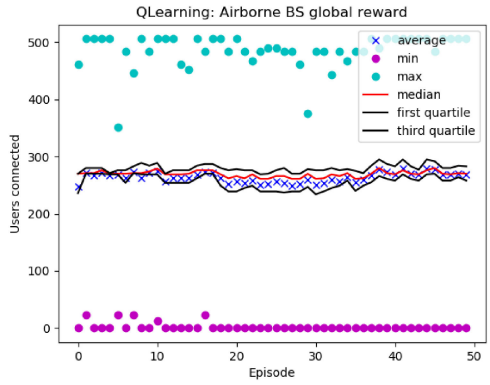


Fig. 4. Reward analysis of Q-learning.

a median of 269 connected users. The maximum value of the global reward is not within the first and third quartiles.

- SARSA applies the same policy in both data generation and evaluation. SARSA follows a Markovian (history-independent) structure [30]–[32]. Under this structure, the value function can be expressed in an algorithmic form known as the Bellman equation, and used to update Q-values in the lookup tables.
- A Deep-Q Network (DQN) can directly use deep learning methods to bridge the divide between high-dimensional inputs and agent actions [33]. The Deep Neural Network predicts the Q value of the current or potential states and actions. Two convolutional neural networks constantly update its parameters to learn the optimal option.
- A clustering network divides data objects with high similarity into clusters [34]. Currently popular clustering methods are k-means and mean-shift. In this work, the k-means algorithm is used as one of the baseline methods. It is interesting to test whether a clustering simulation of the user distribution favours the clustering method. The position of the ABS is randomly chosen, and their moving direction is controlled using L2 distance followed by (15).

$$a_j = \frac{1}{|c_i|} \sum_{x \in c_i} x \tag{15}$$

The positions of the ABSes a_j are updated with the number of users x in each center. As shown in Fig. 5, the k-means method actually performed very poorly, especially in the scenarios with 4 and 8 airborne base stations.

Fig. 5 compares the performance of RR-GAN against the four other baseline methods. As illustrated in Fig. 5, the RR-GAN method attains the highest average percentage of connected users over all airborne base stations, even with an increased number of airborne base stations. Since other methods do not consider the positions of the other base stations and the uncertainty during the initial learning period, RR-GAN can achieve the highest number of user connections.The theoretical best performance is calculated using a heuristic method. The performance of random position experiments is simulated when Airborne base stations moves following a random walk distribution. It is worth noting that the DQN method cannot minimize the divergence of the generated sample distribution and over-fits the training set by maximizing the likelihood, which reduces model generalizability.

C. Multi-Agent Reinforcement Learning

The Markov game framework, proposed in Littman (1994)'s seminal work has long been used to develop multi-agent reinforcement learning algorithms [35]. When autonomous agents operate with a shared reward scheme, each agent can have an identical value function or Q-function [36]. On the other hand, when agents cooperate, a team-average reward method is implemented [37]. In the case of competitive setting, it is typically modelled as Zero-sum Markov games [38]. In the present case study, ABSes are in a mixed setting, where cooperative and competitive actions among agents co-exist. Compared with single-agent reinforcement learning system, multi-agent reinforcement learning presents a more difficult challenge, providing the non-convergence of policy-based methods when action dimension increases exponentially due to the number of agents. One possible solution for the scalability issue is to assume additionally the factorized structures of either the value or reward functions with regard to the action dependence as proposed by Guestrin *et al.* (2002) [39]; while Kok and Vlassis (2004) improved the original heuristic ideas, Sunehag *et al.*(2018) provided empirical progress in their study [40] [41].

We compared the performance of the proposed RR-GAN with the one of the recent developed factorization methods known as Value-Decomposition Networks (VDN) [41] in order to evaluate the multi-agent reinforcement learning in a cooperative and competitive setting. Fig. 6 shows the performance comparison with VDN and our RR-GAN method. In VDN, the vector factorisation DQN network transformed the original joint action-value function into an easily factorizable one, with the same optimal actions, which provides better performance compared with DQN [41]. However, from the three experiments (2, 4 and 8 ABSes), RR-GAN has superior performance with especially larger margins in our case study. It is suggested that those non-cooperative behaviour may more aggressively impact on the overall reward. Furthermore, the performance for the 4 and 8 ABSes cases in VDN performed even worse than baseline performance. It is discussed that factorizing the value function is a very challenging task, which is a concern evidenced in [42].

D. Robustness Comparison

The potential application environments for airborne base stations are complex, real-world domains, which need to overcome the problems of high sample complexity and brittle convergence properties, requiring less meticulous hyperparameter tuning. Therefore, here different learning rates, greedy factors and user distribution are all tested; to allow the ABS to be fully aware of the neighbouring ABS, we also tested on 100 rounds, 1,000 rounds, 10,000 rounds and 100,000 rounds. The reward map is generated based on various amounts of exploration, stopping if there is no higher total reward for the last n rounds of stochastic

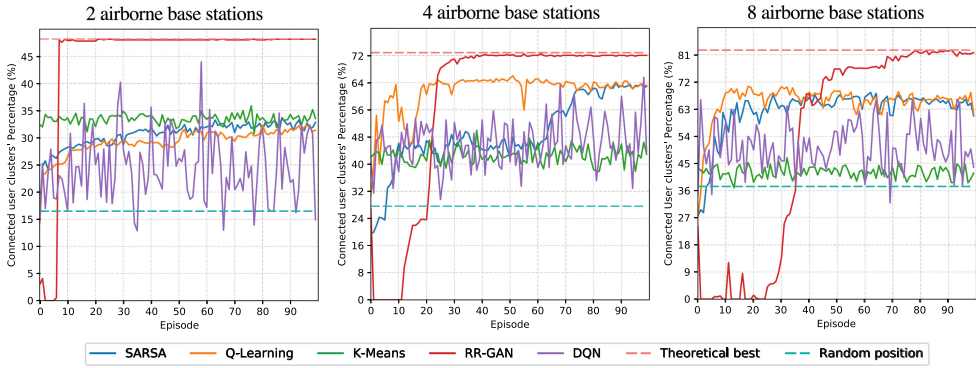


Fig. 5. Comparison among Q-learning, SARSA, k-means, DQN and RR-GAN.

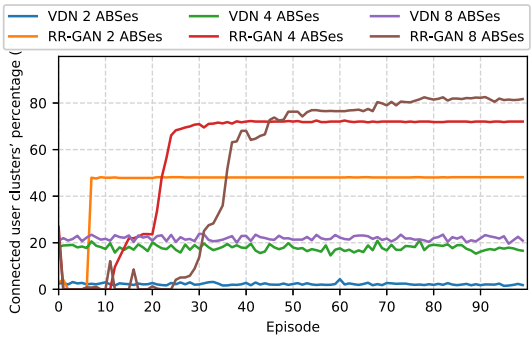


Fig. 6. Comparison among RR-GAN and VDN.

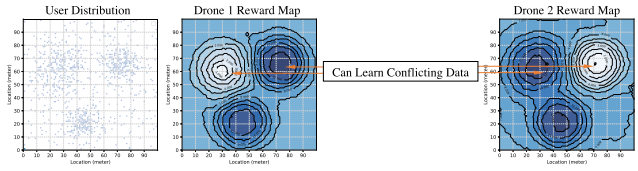


Fig. 7. Learning from neighbour.

exploring. The different number of n rounds is used to test network performance. Fig 8 shows a testing environment with 2 ABSes (airborne base stations), 4 ABSes, and 8 ABSes.

For all rounds, the time taken to reach maximum performance follows the same trend, particularly so for 8 airborne base stations. For 2 or 4 ABSes, a shorter exploring time may introduce oscillations in performance, but all achieve a similar plateau within a longer time frame.

A comparison of robustness with other baseline algorithms can be found in Fig. 9. The greedy parameter is a key parameter in evaluating the algorithm's performance to value their new actions. During the changing of the value of greedy parameters, we can investigate how each method responds with learning information and timing needed for training. With all else held equal, the greedy policy is tested under the values 0.7, 0.8, and 0.9, with the learning rate and discount factor fixed to 0.1 and 0.5, respectively. The user environment is generated with the same random seed.

As shown in Fig. 9, final performance improves as the learning rate approaches 1. Additionally, the larger the greedy factor is, the faster the algorithm will converge. Although initial efficacy is low, RR-GAN has a steeper convergence rate when compared to other baseline methods and, most importantly, is the only method to reach the global optimal. It is noted the DQN method is not chosen in the greedy method comparison. The DQN method hasn't performed well since state transition probability is approximated with one hidden layer neural network.

E. Scalability Test

To investigate the impact of the number of user clusters on the performance of RR-GAN, we ran a scalability test to investigate how much the number of user clusters inevitably affected the RR-GAN method. As shown in Fig. 10, even with an increased number of user clusters, RR-GAN performed consistently well with 2 airborne base stations, 4 airborne base stations and 8 airborne base stations.

F. Learning From the Neighbouring Base Station

Airborne base stations will interfere with each other if they are too close. When airborne base stations move close to a user cluster, the reward increases; however, if a neighbouring airborne base station has already moved towards that centre, the reward decreases. This experiment uses an environment of two airborne base stations to demonstrate that our RR-GAN can be made aware of the neighbouring base station via checking the history rewards of neighbouring ABSes. As shown in Fig. 7, the first channel of the reward map is generated for the first ABS. This airborne base station gains awareness of the fact that the left user cluster has the second optimal reward. For that reason, the network predicts that the second airborne base station will move to that place in next epoch, which results in the first ABS not moving into that direction. Similarly, the second ABS, with the help of the second channel of the reward map, will predict and find the global optimal for the first ABS and automatically reduce the rewards. In this manner, RR-GAN can successfully predict the neighbouring ABS trajectory to achieve the best global goal.

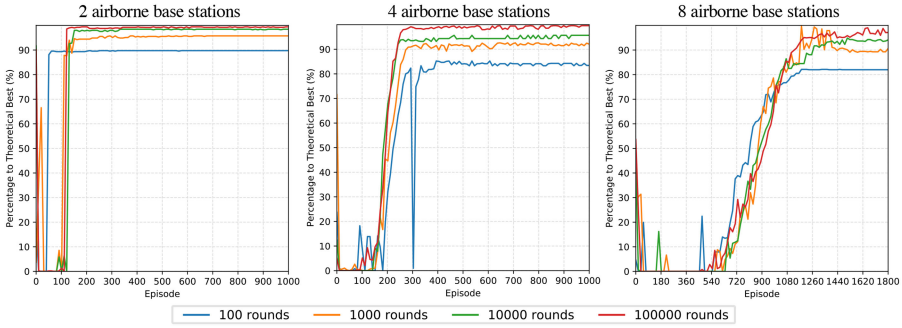


Fig. 8. Robustness comparison with various greedy methods.

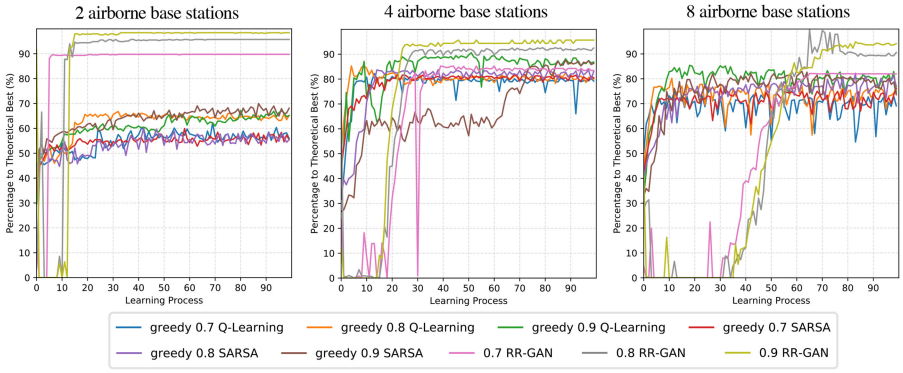


Fig. 9. Robustness comparison among Q-learning, SARSA, K-means, DQN, and RR-GAN with various greedy methods.

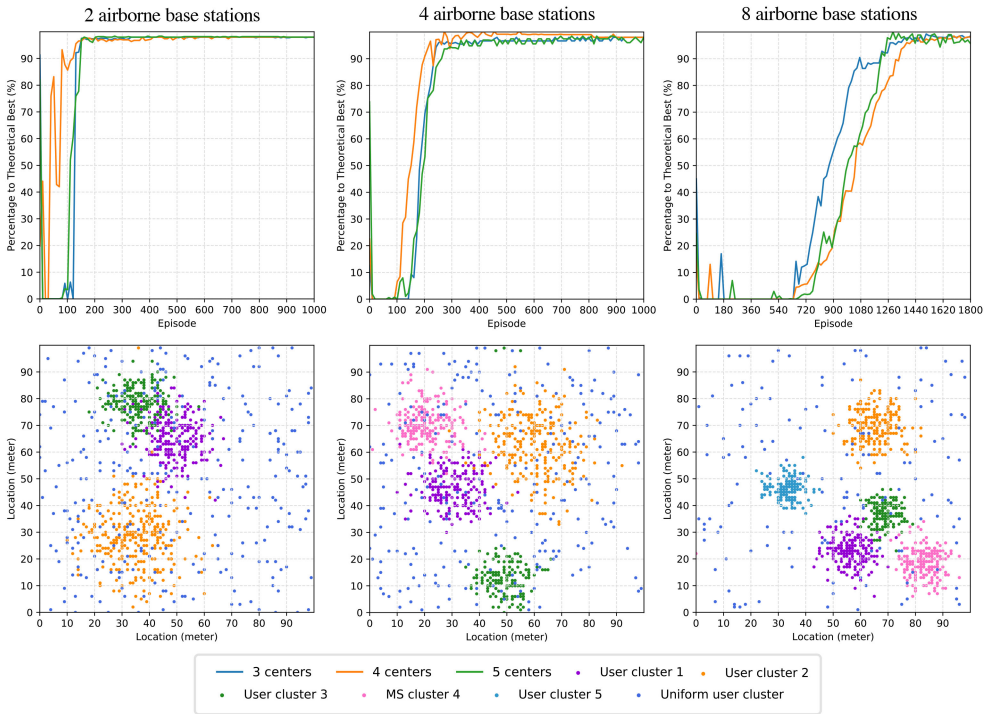


Fig. 10. Robustness comparison among environment with different amount center.

IV. Conclusion and Discussion

In this paper, a Reward-Reinforced GAN (RR-GAN)-based generic framework for multi-agent systems is proposed, which are potentially generalised to any multi-agent system. We propose to use a generator network as an implicit indication of the distribution of users, in order to achieve the maximum global goal. Our case study demonstrates how RR-GAN has the best system performance compared with other baseline methods (the source code can be accessed via GitHub link: https:\\github.com\Changgang-Zheng\Reward-reinforced-Generative-Adversarial-Network\). In terms of future developments, while this work demonstrated the advantages of using RR-GAN for multi-agent learning, several challenges still remain:

- **Convergence Failures:** The convergence properties for generative neural networks constitute an open research question. The theoretical condition for the adversarial model is when the loss function is a class of convex optimisation algorithms [15]. In that case, the adversarial model can be guaranteed to find a unique solution. However, when neural networks are used for the generator and the discriminator (as in this paper), it is not always guaranteed to converge to a unique solution. We used Nash equilibriums from game theory to demonstrate the possibility of converging for an adversarial model when a small change in probabilities for discriminator leads to a situation where two conditions hold: the generator did not change and has no better strategy in the new environment. At the training stage, it is a challenge to modify GAN design so that the discriminator can be trained optimally, avoiding the issue of convergence failures. The current research is investigate how to move away from unimodal distributions as a natural relaxation to solve potential convergence failures [43].
- **Uncertainty Quantification:** Accurately estimating user movements is commonly based on Bayesian methods, which introduces epistemic uncertainty into the model parameters (cluster centres and random seeds) [44]. Gaussian processes present issues of model inadequacy and parameter uncertainty when scaling to high-dimensional problems. Our RR-GAN network creates out-of-distribution samples so that classifiers can be explicitly taught about uncertain inputs. We evidence the complications in the training process from computer simulations, the implications of which for real-life experimental data are still unknown.
- **Explanation of uncertainty:** Creating explanations for the causes of model uncertainty [45] is a relatively underexplored area. In a GAN, uncertainty may arise because an input is unlike the training data and has been constrained by a set of known features in a previous unseen combination. This type of explanation has only been very recently explored by Merric and Taly to calculate the variance of Shapley values [46], despite the fact that it remains a challenging research area which may produce important consequences for assessing multi-agent learning systems.

Acknowledgment

We would like offer our sincerest gratitude to Paulo Valente Klaine and João P.B. Nadas, who provided an initial discussion for this project.

References

[1] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5792–5799.

[2] N. K. Long, K. Sammut, D. Sgarioto, M. Garratt, and H. A. Abbass, "A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 4, pp. 523–537, Aug. 2020.

[3] Y. Liu, X. Wang, G. Boudreau, A. B. Sediq, and H. Abou-zeid, "Deep learning based hotspot prediction and beam management for adaptive virtual small cell in 5G networks," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 1, pp. 83–94, Feb. 2020.

[4] S. Russell, "Learning agents for uncertain environments," in *Proc. 11th Ann. Conf. Comput. Learn. Theory*, 1998, pp. 101–103.

[5] A. Y. Ng *et al.*, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, vol. 1, 2000, pp. 2–10.

[6] C. L. Baker, R. Saxe, and J. B. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 113, no. 3, pp. 329–349, 2009.

[7] A. Coates, P. Abbeel, and A. Y. Ng, "Apprenticeship learning for helicopter control," *Commun. ACM*, vol. 52, no. 7, pp. 97–105, 2009.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[9] A. R. Shirazi and Y. Jin, "A strategy for self-organized coordinated motion of a swarm of minimalist robots," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 5, pp. 326–338, Oct. 2017.

[10] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4565–4573.

[11] G. L. Guimaraes *et al.*, "Objective-reinforced generative adversarial networks (organ) for sequence generation models," 2017, *arXiv:1705.10843*.

[12] R. E. Bellman, *Dynamic Programming*, Princeton, NJ, USA: Princeton University Press, 1957.

[13] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of GANs," 2017, *arXiv:1705.10461*.

[14] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," 2017, *arXiv:1701.04862*.

[15] Z. Pan *et al.*, "Loss functions of generative adversarial networks (GANs): Opportunities and challenges," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 4, no. 4, pp. 500–522, Aug. 2020.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–11.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1125–1134.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2223–2232.

[20] J. Li, H. Ma, and M. Tomizuka, "Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6658–6664.

[21] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (GANs)," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 224–232.

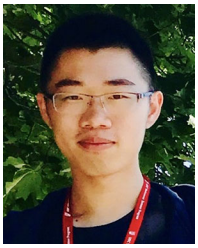
[22] F. Afroz, R. Subramanian, R. Heidary, K. Sandrasegaran, and S. Ahmed, "SINR, RSRP, RSSI and RSRQ measurements in long term evolution networks," *Int. J. Wireless Mobile Netw.*, 2015, pp. 113–123.

[23] M. L. Rocca, "RSRP and RSRQ measurement in lte," *Laroccasolutions Technol. Serv.*, Feb, vol. 2, pp. 9–19, 2015.

[24] F. Ricciato, P. Widhalm, M. Craglia, and F. Pantisano, *Estimating Population Density Distribution From Network-Based Mobile Phone Data*. Publications Office of the European Union, 2015.

[25] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. ACM, 2011, pp. 1082–1090.

[26] H. Gao, J. Tang, and H. Liu, "Exploring social-historical ties on location-based social networks," in *Proc. 6th Int. Proc. AAAI Conf. Artif. Intell. Conf. Weblogs Social Media*, 2012, pp. 14240–14252.
[27] K. Murphy, "Conjugate Bayesian analysis of the Gaussian distribution," def, 1(2σ2), 16, pp. 1–28, 2007.
[28] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 1038–1044.
[29] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.
[30] E. A. Petter, S. J. Gershman, and W. H. Meck, "Integrating models of interval timing and reinforcement learning," *Trends Cogn. Sci.*, vol. 22, no. 10, pp. 911–922, 2018.
[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[32] V. G. Lopez and F. L. Lewis, "Dynamic multiobjective control for continuous-time systems using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2869–2874, Jul. 2019.
[33] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[34] A. Akarsu and T. Girici, "Fairness aware multiple drone base station deployment," *IET Commun.*, vol. 12, no. 4, pp. 425–431, 2017.
[35] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learn-Ing," in *Machine Learning Proceedings 1994*. Amsterdam, The Netherlands: Elsevier, 1994, pp. 157–163.
[36] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Part C. (Applications Reviews)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.
[37] T. Doan, S. Maguluri, and J. Romberg, "Finite-time analysis of distributed td (0) with linear function approximation on multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2019, pp. 1626–1635.
[38] G. Brockman *et al.*, "Openai gym," 2016, *arXiv:1606.01540*.
[39] C. Guestrin, S. Venkataraman, and D. Koller, "Context-specific multiagent coordination and planning with factored mdps," in *Proc. AAAI Conf. Artif. Intell./IAAI*, 2002, pp. 253–259.
[40] J. R. Kok and N. Vlassis, "Sparse cooperative q-learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 61–72.
[41] P. Sunehag *et al.*, "Value-decomposition networks for cooperative multi-agent learning," 2017, *arXiv:1706.05296*.
[42] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learning. PMLR*, 2019, pp. 5887–5896.
[43] J. Li, A. Madry, J. Peebles, and L. Schmidt, "On the limitations of first-order approximation in gan dynamics," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2018, pp. 3005–3013.
[44] J. R. van Dorp, "A dependent project evaluation and review technique: A bayesian network approach," *Eur. J. Oper. Res.*, vol. 280, no. 2, pp. 689–706, 2020.
[45] S. Yang, K. Wong-Lin, I. Rano, and A. Lindsay, "A single chip system for sensor data fusion based on a drift-diffusion model," in *Proc. IEEE Intell. Syst. Conf.*, 2017, pp. 198–201.
[46] L. Merrick and A. Taly, "The explanation game: Explaining machine learning models with cooperative game theory," 2019, *arXiv:1909.08128*.

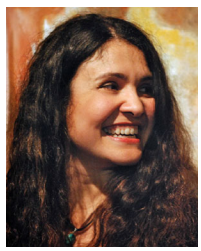


Changgang Zheng was graduated from the University of Electronic Science and Technology of China and the University of Glasgow joint school in 2020. He is the D.Phil. student with the University of Oxford and a Member of Jesus College. He is currently a Member of the Computing Infrastructure Group and working under the In-network machine learning project to explore the potential use of commodity programmable switches.

Shufan Yang recived the Ph.D. degree in computer science from the University of Manchester in 2010. She is the Associate Professor with Software Engineering Technology Subject Group, School of Computing, Edinburgh Napier University. She is a Charted Engineer and a Fellow for British Computer Society and IEEE Senior Member. Dr. Shufan Yang's current research interests include cognitively inspired AI, reinforcement learning, deep learning, computer vision and AI+healthcare (ambient intelligent systems for healthcare). In the past she has also worked on system-on-chip and computational neuroscience.

Juan Marcelo Parra-Ullauri received the degree of Engineer on Electronics and Telecommunications from the University of Cuenca, Ecuador, in 2017. He is currently the Ph.D. candidate with Aston University in the U.K. His research interests include model-driven engineering, autonomous self-adaptive systems and explainability in AI-driven systems. Juan is interested in applying these approaches to cyber-physical systems, Internet of Things and ambient assisted living contexts.

Antonio Garcia-Dominguez is a Lecturer with Aston University in Birmingham, U.K. His research interests include software testing, model-driven engineering and novel methods for engineering education. Antonio is the Lead Developer of the open source Eclipse Hawk model indexing framework, and a core committer of the Eclipse Epsilon family of model management languages.



Nelly Bencomo is a Senior Lecturer with Aston University in the U.K. Her research interests include software engineering include decision-making under uncertainty, model-driven engineering, autonomous systems, artificial intelligence, and requirements engineering. She was awarded EU MC and U.K. Leverhulme Individual Fellowships. She is the Leader of the EPSRC project Twenty20Insight.