Altruism and Selfishness in Believable Game Agents: Deep Reinforcement Learning in Modified Dictator Games

Damon Daylamani-Zad and Marios C. Angelides

Abstract—**This article focuses on using deep reinforcement learning, specifically proximity policy optimization, to train agents in a social dilemma game, modified dictator game, in order to investigate the effect of selfishness and altruism on the believability of the game agents. We present the design and implementation of the training environment, including the reward functions which are based on the findings of established empirical research, with three agent profiles mapped to the three standard constant elasticity of substitution (CES) utility functions, i.e., selfish, perfect substitutes, and Leontief, which measure different levels of selfishness/altruism. The trained models are validated and then used in a sample game, which is used to evaluate the believability of the three agent profiles using the agent believability metrics. The results indicate that players find altruistic behaviour more believable and consider selfishness less so. Analysis of the results indicates that human-like behavior resulting from the application of artificial intelligence evolves from perceived human behavior rather than the observed. The analysis also indicates that selfishness/altruism may be considered as an extra dimension to be included in the believability metrics.**

Index Terms—**Agents, believability, deep reinforcement learning, dictator game, proximal policy optimization (PPO).**

I. Introduction

Most video games would benefit from the application of artificial intelligence (AI) either from personalization and management, to supporting progression through a game, or through serving as opponents implemented as individual agents, groups of agents, or central intelligence [1].

Believability and human-likeness of these agents have always been a challenging area of research. Researchers have been working on various aspects of the agent behavior from decision-making and strategic planning to weapon selection and pathfinding. Many approaches including hierarchical task network [2], evolutionary [3] and genetic algorithms [4], fuzzy clustering [5], and neural networks and reinforcement learning (RL) [6] have been used to address various aspects of this challenge.

RL has gained more popularity in recent years, especially amongst game developers, due to its core principle of "behavior is reward-driven" [7], [8]. In RL, an agent learns a behavior through interacting with the environment which would reward the agent based on these interactions. Games lend themselves well to RL approaches as most games already have rewards systems in place, e.g. score, health, mana, and leveling [9].

The degree of human-like behavior has always been considered a measure for successful AI [10], [11] which in turn increases the believability of agents, hence increasing immersion [12]–[14]. Producing human-like decision-making behavior would allow for more enjoyable gameplay experience. This would be achievable by using models generated through RL, especially when the reward function is mapped to utility functions based on human observation.

Games such as prisoner's dilemma and dictator game have been used to guide research in social dilemmas. These games tend to be simple in terms of mechanics but yield interesting results for human behavior in social settings [15], [16]. In recent years, there has been great interest in experimenting with implementation of such games using RL [17]. The observations gained from these experiments have allowed researchers to have more control over the parameters of their experiments [18]–[21]. As state-of-the-art seeks for a performance closer to that of humans, it is becoming increasingly possible to consider integrating such trained models into games, to increase agent believability. One aspect that requires further research is altruistic behavior and decision-making in game agents.

This research aims to investigate the use of RL to create believable game agents by training them to exhibit human-like altruistic behavior. It further aims to establish an understanding of the effect of selfish or altruistic behavior on the agents' believability. The research is ultimately motivated to create agents that would naturally learn to exhibit different levels of altruism and be able to respond in a meaningful way to dynamic situations. For the purpose of showcasing the results of our research, we have chosen a modified dictator game. The agents' brain would be trained using proximal policy optimization (PPO), a cutting-edge RL approach [22]. The reward function is created based on utility functions that have been evolved from human experiments [23]. Once trained, the brains will be deployed in inference mode into a game. The behavior of the trained agent brains has been validated for believability through empirical research.

This article starts with presenting related works on deep RL and PPO, and continues with the social dictator game, and

Manuscript received August 28, 2019; revised February 13, 2020; accepted April 17, 2020. Date of publication April 27, 2020; date of current version September 15, 2021. *(Corresponding author: Damon Daylamani-Zad.)*

The authors are with the College of Engineering, Design and Physical Sciences, Brunel University London, London UB8 3PH, U.K. (e-mail: damon.daylamani-zad@brunel.ac.uk; marios.angelides@brunel.ac.uk).

Digital Object Identifier 10.1109/TG.2020.2989636

its modified version which form the basis of the hypothesis underlying our work. The article then presents the design and implementation of the RL approach that has been used to create altruistic agents, presenting the setting, reward functions, and hyperparameters used during training. The results of training are then presented, and the trained models are validated against the aim. The proceeding section discusses the empirical research with which believability of the agents produced using our approach has been evaluated. Section IV presents the design of the experiment, methods used, the participants, and materials. The results of the experiment are presented in detail and analyzed for drawing conclusions. Finally, the article concludes with discussion of the implications of the results and proposes future research directions.

II. Related Work

This section presents the related work and state-of-the art in using machine learning in games. From there, it will follow to discuss the two main themes underlining this research: RL and the dictator games.

A. Machine Learning in Video Games

Using AI techniques in games is an established field of research in both academia and industry. With the success and increased popularity of deep learning, these methods have also been widely applied in video games [9], [24]–[26]. While *supervised learning* is being used in games, they still rely on large data sets of player behavior and most times require further training using methods such as RL [27]. *Unsupervised learning* is also being used in games; however, the research in this area is in its early stages [9] and it shares the challenge of requiring large sets of data.

RL has been a popular and suitable approach in games due to its reward-based nature. Games can easily be mapped to environments which the agent can interact with and receive reward based on the agent's actions and decisions. However, the challenge of RL lies in the type of environment and sparsity or availability of reward signals. The algorithm needs to trace back the reward gained, such as winning the game, and propagate it back to the chain of actions that leads to a successful or unsuccessful reward signal. The research in [17]–[19] demonstrates that RL has been successfully used to map social dilemma scenarios into environments with suitable reward signals. This shows that RL is a promising method to tackle the aim of this research.

Evolutionary approaches, including *neuroevolution* [28] and *evolution strategy* [29], [30], have been widely used in training neural networks for games. These approaches are derivative-free optimizations as opposed to gradient-descent-based approaches previously mentioned. These approaches are population based and they maintain a distribution over network weight values and employ a large number of agents acting in parallel using samples from the distribution. The parallelization allows for faster computation compared to methods such as RL. However, this performance comes at a cost. These approaches treat the neural network optimization as a black-box. This black-box approach means the inner workings of the network are not considered during the training and only the overall outcome is used in deciding the fittest networks weights that are passed on to the next generation [30]. While this is acceptable in scenarios with sparse reward signal, in scenarios with richer reward signals, these approaches do not perform with the desired behavior.

This research aims to train game agents to exhibit believable altruistic behavior, the goal environment would be reward-rich and multiagent. The agents would be receiving a multitude of reward signals based on their actions. The aim is to train these agents to exhibit different levels of selfishness/altruism based on a partially observable environment. The amalgamation of these decisions in the long run would lead to a win/lose result; however, the reward signals would be plenty. The most important point in this research is that altruism is based on expectation of future results which may be achieved long in the future. Yet, during an episode, there are always unpredictable results from the environment which can be problematic for evolutionary approaches as they do not consider the inner workings of neural networks. Yet, RL shows clear advantage in allowing emergent behavior in multiagent environment with rich reward signals and seemingly random environmental rewards [31]–[33].

Hence, to allow for the further development of this research, RL has been adopted as the approach in this research. This choice allows for further development of the research toward its ultimate goal. The next subsection will describe RL in further detail.

B. Reinforcement Learning

RL is formed of agents that interact with an environment over time. Through these interactions, the agent receives a reward and the aim of the agents is to maximize their rewards through repeating various interactions available for each state. At each time step t, an agent is at state s_t where $s_t \in S$. In this state, the agent can take an action a_t where $a_t \in \mathbb{A}$. This action would result in agent receiving a reward r_t and moving to s_{t+1}. The probability of transition from one state to another is represented with a transition probability function $P(s_{t+1}|s_t, a_t)$. The state-actions reside in a policy matrix Π which holds state-action sets that define the actions available in each given state. Hence, each state s_t and action a_t set creates a policy $\pi(s_t, a_t)$. The reward of each policy is defined through a reward function $R(s, a)$, which is based on the dynamics of the environment. The agent will continue interacting until it reaches a final state, at which point it will calculate the total reward and then reset itself [34].

Most RL algorithms use *Q-value* for estimating the expected future rewards of state–action pairs. Most popular model-free RL approach includes Q-learning [35], which finds an optimal policy for any finite Markov decision process by creating a Q-table consisting of optimal Q-values. Deep Q-networks [36] combine Q-learning with convolutional neural networks (CNN), allowing the CNN to learn from high-dimensional sensory inputs. Trust region policy optimization (TRPO) [37] is an effective algorithm for optimizing large nonlinear policies, especially neural networks. PPO [22] is also a policy gradient optimization approach that has the benefits of TRPO but is simpler to implement, more general, and has better sample complexity.

RL aims to maximize the expected value of total reward for all consecutive steps starting from the current state. In other words, it identifies the most optimal state–action policy starting from the current state, where future steps would lead into even higher rewards. For each state s_t, we define the weight of Δt steps in the future as $\gamma^{\Delta t}$. γ, which is known as the discount factor, defines how much the agent cares about future reward. γ is defined as $0 < \gamma \leq 1$ and is typically assigned a value between 0.7 and 0.99.

For state–action, Q-function is defined as $Q(s, a)$, which corresponds to the expected future rewards of action a in state s. The true optimal function is defined using Bellman equation presented in

$$Q^*(s, a) = r + \gamma \max_a Q^*(s', a'). \tag{1}$$

Considering that a suboptimal value of Q-function in a state would be a step in the correct direction and these values would update as the learning progresses, the values within the Q-table for step t are updated based on

$$Q'(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a)) \tag{2}$$

where α is the learning rate.

Deep RL [36], [38] uses deep artificial neural networks as estimators. Most popularly, CNNs or recurrent neural network (RNN), mainly long short-term memory (LSTM) [39], are used in deep RL. In order to use Q-value in deep RL, (1) is updated to include the network parameters as presented in

$$Q(s_t, a_t) = r_t + \gamma \max_{a'} Q(\Phi_t, a'; \theta^-) \tag{3}$$

where Φ is the preprocessed equivalent to state s_t and θ stands for the parameters in the neural network (weights).

The *Q*-values for some of the actions in a state can have such small differences that algorithms may not be able to have real preferences between them. Therefore, an advantage function has been devised which defines how good an action a_t is compared to the average action of the specific state. The following equation presents how the advantage is calculated, where $V(s)$ is the average *Q*-value of state s:

$$A(s, a) = Q(s, a) - V(s) : V(s) = \frac{\sum_i^N Q(s, a_i)}{N}. \tag{4}$$

PPO is based on TRPO's approach which adopts the actor-critic architecture and belongs to the policy gradient category [22], [37]. These approaches use temporal difference (TD) error to determine the update step size in a continuous space. They define η as expected discounted long-term reward which should be always increasing. The expected discounter long-term reward for policy π is defined in

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots} \left[\sum_{t=0}^{\inf} \gamma^t r(s_t) \right]. \tag{5}$$

Hence, for a new policy $\tilde{\pi}$, the expected return can be viewed in terms of its advantage over previous policy π as $\eta(\tilde{\pi})$ as presented in

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \ldots \sim} \left[\sum_{t=0}^{\inf} \gamma^t A(s_t, a_t) \right]. \tag{6}$$

The value of $\eta(\tilde{\pi})$ can be approximated to $L_\pi(\tilde{\pi})$ presented in

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(s, a) A_\pi(s, a) \tag{7}$$

where ρ is the discounted visitation frequencies presented in

$$\rho_\pi(s) = \sum_{i=0}^{\infty} \gamma^i P(s_i = s). \tag{8}$$

Hence, the objective function of TRPO is defined as

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_{old}}(s_t, a_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t [r_t(\theta) \hat{A}_t] \tag{9}$$

where $r_t(\theta)$ is the ratio between the new and the old policies, $\hat{A}_t$ is the estimated advantage at time t, and $\hat{\mathbb{E}}_t$ is the empirical expectation over timesteps. The idea of TRPO has constraints that disallow too much policy change. This can be both constraining and resource-intensive. Therefore, PPO modifies TRPO's objective function with a penalty for having policy updates that are too large, instead of constraints. The clipped TRPO objective function is presented in

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t), \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t]. \tag{10}$$

Finally, PPO's objective function is presented in

$$L^{\text{CLIP}+VF+S}(\theta) = \hat{\mathbb{E}}_t[L^{\text{CLIP}}(\theta) + c_1 \, L^{VF}(\theta) + C_2 S[\pi_\theta](s_t)] \tag{11}$$

which is a lower bound of (10) and removes the KL divergence constraint. Therefore, the computation for PPO is much less resource-intensive.

PPO has gained much popularity due to its simple implementation and good performance. Open AI use PPO as their baseline RL algorithm [40] and Unity has also incorporated it into their machine learning toolkit [41]. In our implementation, we will be using PPO; the specific setup parameters are presented in Section III.

C. Dictator Game

The dictator game, also known as giving game, is a well-known game in the social-psychology and economics and it is used to examine altruism and selfishness in human beings. The original game [42] is defined as a game in which subjects (dictators) decide how much, if any, of an endowment to give to another player. While the game is typically performed as a one-shot [16], there have been many successful implementations of iterated dictator game [43]. The game is typically performed in an anonymous setting and it is common for over 50% of subjects to give some money away.

Andreoni and Miller [23] present a thought-provoking setup for the dictator game to examine the consistency of preference for altruism. Their results showed that over 98% of their subjects

TABLE I
ALLOCATION CHOICES [23]

Budget	Token Endowment	Hold Value	Give Value	Relative Price of Giving
1	40	3	1	3
2	40	1	3	0.33
3	60	2	1	2
4	60	1	2	0.5
5	75	2	1	2
6	75	1	2	0.5
7	60	1	1	1
8	100	1	1	1
9	80	1	1	1
10	40	4	1	4
11	40	1	4	0.25

exhibited behavior that is consistent with maximizing utility. They have mapped the results to the three standard constant elasticity of substitution (CES) utility functions: selfish, perfect substitutes, or Leontief.

1) *Selfish*: Those who prefer to keep everything.
2) *Perfect substitutes*: Those that give everything away when the price of giving is less than one, yet keep everything when the price of giving is greater than one.
3) *Leontief*: Those that always divide the surplus equally.

For their experiment, they used a modified dictator game where each subject is given a menu of choices with different endowments and prices for payoffs which the subjects had to make a decision for each one. Assuming that the total endowment in a choice is m and the payoff for person i is defined as $p_i \in P$, payoff to self is defined as p_s and payoff to other as p_o. In their experiment, $m = p_s + \lambda p_o$, where λ is the price of payoffs. The utility of each player is defined as $U_s = u_s(p_s, p_o)$.

The allocation choices provided to the subjects are presented in Table I. The allocation choices were designed so that each one presents a convex budget set. While budgets 7, 8, and 9 are choices like the standard dictator game, the other choices present scenarios where the endowment is an income variable. For example, in budget one, the price of payoff to self is 0.33, which means that giving one token raises the other subject's payoff by one point, and reduces subject's own payoff by three. According to their results, each of the three CES utility functions can be defined as follows.

1) *Selfish:* $U(p_s, p_o) = p_s$.
2) *Perfect substitute:* $U(p_s, p_o) = \min(p_s, p_o)$.
3) *Leontief:* $U(p_s, p_o) = p_s + p_o$.

The three utility functions and the setup of the experiment create the basis of the RL environment presented in this article.

III. Proposed Design and Implementation, Training, and Validation

This section discusses the implementation of the training environment and the setup used to train the agents. We discuss how we deploy the three profiles presented in the previous section, define reward functions for each based on their utility functions, design the training environment, and implement PPO. We aim at creating agents that learn to behave as their assigned profile based on their respective reward function.

A. Profiles

As presented in the previous section, this article presents research on training several agents with each having their own profile. In the case discussed in the previous section, each profile is based on one of the three CES utility functions. The training is set up in a way that the agents are unaware of each other's decisions. In order to achieve homogeny among the agents and ensure compatibility, the agents share the exact implementation, environment, and training variables. The only parameter in their profile that is unique to each one is their reward function. Therefore, each agent will receive rewards based on their profile. Section III-C discusses these rewards and their implementations based on CES utility functions.

B. Training Environment Setup

An agent training environment was created. The environment is partially observable and consists of agents in training. The agents are unaware of each other and will be independently trained. Each agent is presented with the 11 budgets presented by [23], illustrated in Table I, in each round. The agent would make a decision for each budget on the menu, receiving a reward for each decision. As mentioned before, the decision is to distribute the total endowment of m between itself and another agent who is nonobservable. We can define the payoffs in terms of the distribution decision. If we consider a hold decision of d, where $0 \leq d \leq 1$, the hold payoff (payoff to self) would be $p_s = d \times m$ and give payoff (payoff to other) would be $p_o = \lambda(1 - d) \times m$. This can be defined as a single continuous decision in RL. Once all 11 decisions required are made, a round terminates, and the agent resets and continues with a new round of decisions.

In order for the agent to form a policy, an observation vector $\vec{O}$ is defined. The observation vector includes the parameters of the current budget and the distribution proportion, dis, which is defined as p_s/p_o. The distribution proportion represents the proportion of hold over give of each decision. Hence, the observation vector for decision i can be defined in

$$\vec{O_i} = \{T_i, h_i, g_i, dis_i, p_{s_i}, p_{o_i}\} \tag{12}$$

where T_i is the token endowment, h_i is the hold value, and g_i is the give value. In this equation, for each decision d_i, the value of p_{s_i} is computed in

$$p_{s_i} = d_i \times T_i \times h_i \tag{13}$$

and the value of p_{0_i} can be calculated in

$$p_{o_i} = (1 - d_i) \times T_i \times g_i. \tag{14}$$

C. Rewards Based on Profiles

Each decision d_j in round i would receive reward r_{i_j} where the sum of all rewards in round r_i is designed to be normalized as $0 \leq (r_i = \sum_j r_{i_j}) \leq 1$. The reward functions are defined based on the three standard utility functions presented in the previous section. As mentioned before, we define three profiles for three different agents: Selfish (Sel), Perfect Substitute (Sub), and Leontief (Leo). Based on the utility function of each profile,

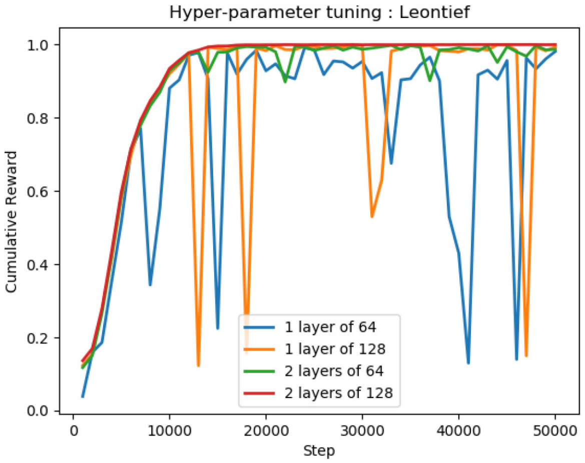

Fig. 1. Tuning the hyperparameters: comparing four different networks of 1 × 64, 1 × 128, 2 × 64, and 2 × 128. The 2 × 128 has the most stable results as well as the highest rewards. Plot presents the mean cumulative episodic reward (*Y*-axis) over timesteps of simulation (*X*-axis) during training and evaluation of Leontief agent.

we have defined their respective reward function in

$$cr_{\mathrm{Sel}_i} = \frac{p_{s_i} = d_i \times T_i \times h_i}{T_i \times h_i} = d_i$$

$$r_{\mathrm{Sub}_i} = \begin{cases} d_i, & \text{if } PG_i > 0 \\ 1 - d_i, & \text{if } PG_i < 0 \\ \frac{\min(d_i,(1-d_i))}{\max(d_i,(1-d_i))}, & \text{if } PG_i = 0 \end{cases}$$

$$r_{\mathrm{Leo}_i} = 1 - |(d_i \times h_i) - ((1 - d_i) \times g_i)|. \tag{15}$$

The Selfish will always hold and therefore its reward is calculated as the proportion of hold choice divided by the the payoff of choosing to hold all. The behavior of the Perfect substitute should be based on the price of giving, PG. As can be seen from Table I, for each budget i, PG_i is defined as h_i/g_i. Finally, Leontief behavior requires a fair distribution of the endowment between hold and give. Considering (13) and (14), to meet the Leontief behavior, we must ensure that $p_s = p_o$; therefore, $d_i \times T_i \times h_i = (1 - d_i) \times T_i \times g_i$, from which we can deduce $d_i \times h_i = (1 - d_i) \times g_i$. Considering the condition $d_i \times h_i + (1 - d_i) \times g_i = 1$, we can deduce that in this scenario to meet the Leontief behavior, we must ensure $d_i \times h_i = (1 - d_i) \times g_i = 1/2$. Hence, to meet the Leontief behavior, we need to promote a reward value as presented below. Equation (15) presents the reward functions for each profile.

D. PPO Hyperparameters and Training

As mentioned earlier, we have implemented PPO with clipped objective as presented in (11). In the algorithm, $\vec{D}_i$ represents the partial trajectory for policy π_i and $r_t(\theta)$ is the ratio between the new and the old policies. The implementation is based on communication with unity in [41], which has been used as the game simulation environment.

Algorithm 1: PPO with Clipped Objective.

```
procedure PPO$\vec{\Pi}, \vec{\Theta}, \epsilon, S$
        $\triangleright$ INPUT: policies, policy parameters, clipping threshold, maximum steps
    for $i = 0 \rightarrow S$ do
            $\triangleright$ Collect partial trajectories
        $\pi_i \leftarrow \Pi(\theta_i)$
        $\vec{D}_i \leftarrow D_i(\pi_i)$
            $\triangleright$ Estimate advantages
        $\hat{A}^{\pi_i} \leftarrow \sum_j^S \vec{D}_j(\pi_i)/S$
            $\triangleright$ take k steps of minibatch
        $L^{\mathrm{CLIP}}_{\theta_k}(\theta) \leftarrow \hat{\mathbb{E}}_{\pi_k}[\sum_{t=0}^{T}[\min(r_t(\theta)\hat{A}_t^{\pi_k}), \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_k}]]$
            $\triangleright$ Policy update
        $\theta_{i+1} \leftarrow \mathrm{argmax}_\theta L^{\mathrm{CLIP}}_{\theta_i(\vec{\Theta})}$
    end for
end procedure
```

TABLE II
HYPERPARAMETERS FOR TRAINING

optimizer	Adam
learning rate	0.0003
batch size	100
gamma	0.99
epsilon	0.2
maximum steps	50,000
number of layers	2
hidden units	128

The hyperparameters for training are summarized in Table II. Adam optimizer [44] has been used as stochastic gradient descent optimization algorithm. The hyperparameters were tuned through a random search approach and testing multiple combinations of hyperparameters based on estimations and empirical results. Fig. 1 demonstrates the distributions of the cumulative rewards per step of the Leontief agent training for four different networks. As illustrated, increasing the complexity, the model is increasingly more stable in its learning and has less instances of forgetting/resetting.

Training and evaluation were performed for each agent separately. All three profiles managed to arrive at the maximum mean cumulative episodic reward of $r = 1$. Fig. 2 presents the mean cumulative episodic rewards for each profile. Fig. 3 presents decision distributions for budget 1 from Table I during training for each profile.

E. Validating Trained Models

In order to test the validity of the trained model, they were then added to a test environment in inference mode. They would each iterate 50 times through the 11 budgets within the menu and their decisions were recorded. A "random" agent was added as a control baseline. The random agent's decision is a random value for hold, where $d_{\text{random}} = \mathrm{U}(0, 1)$. As can be observed in Fig. 4, the random (control) decision tends to stay around the 50% hold decision as on a normal distribution. In contrast, the Selfish is behaving as expected and consistently holds all endowments (keeps all). The Leontief is also acting as expected, reducing the

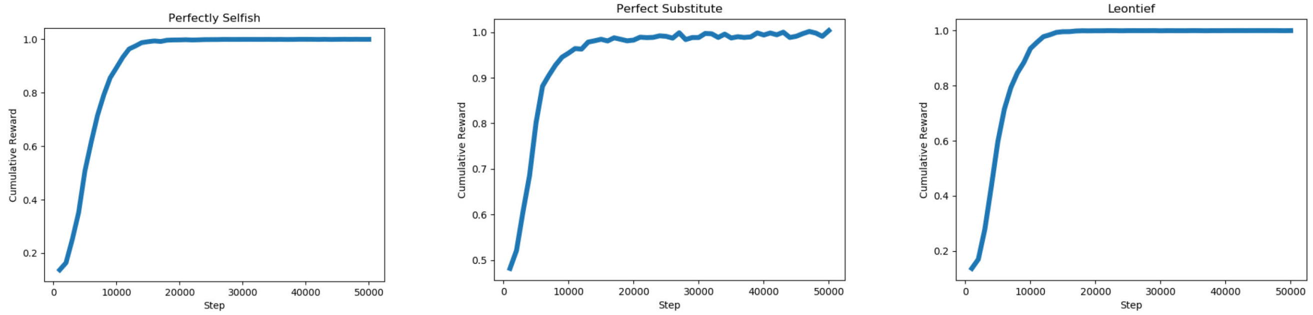

Fig. 2. Plots presenting the mean cumulative episodic reward (*Y*-axis) over timesteps of simulation (*X*-axis) during training and evaluation for selfish (left), perfect substitute (middle), and Leontief (right).

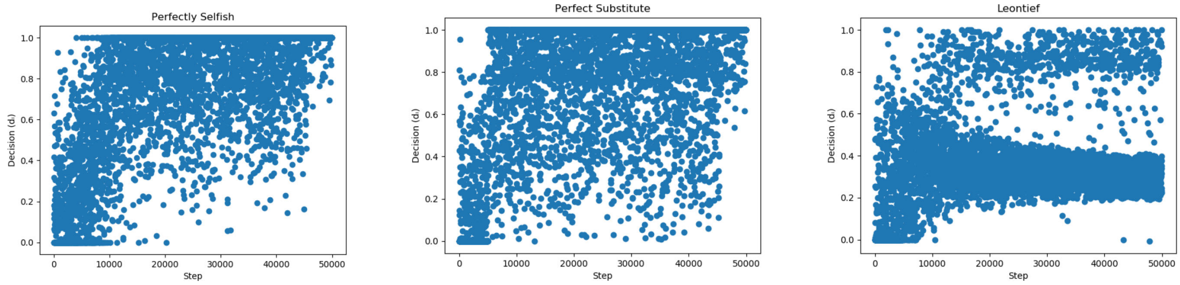

Fig. 3. Plots presenting the decision, d_i (*Y*-axis) over timesteps of simulation (*X*-axis) during training and evaluation for selfish (left), perfect substitute (middle), and Leontief (right) for budget 1.

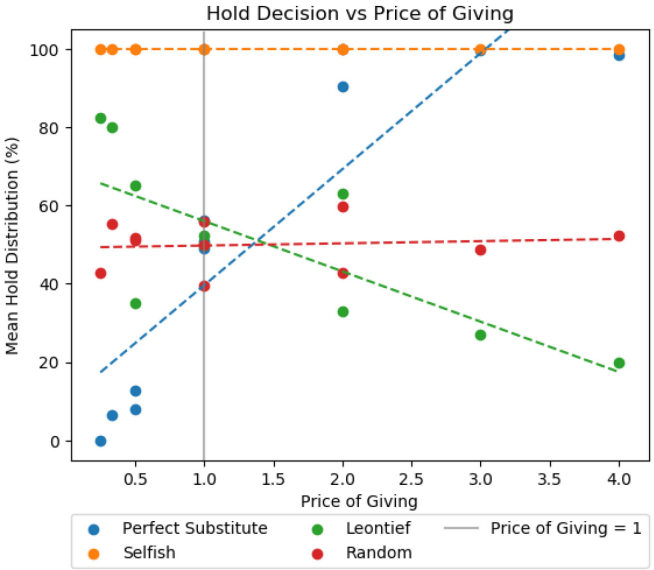

Fig. 4. Plot presenting the mean hold decisions for each agent profile (including random) per the price of giving for each budget (scatter points). The dotted lines depict the trendline of hold decision per price of giving.

hold decision, as the price of giving goes up, in order to maintain an equal balance of distribution. Finally, the Perfect Substitutes are giving everything away when price of giving is less than one and keeping everything when the price of giving is greater than one. Fig. 5 illustrates the mean hold and give payoffs for each agent profile per budget number, which allows for a closer confirmation of the intended behavior. A Wilcoxon matched pairs signed-rank test [45] was performed to determine whether there is a significant difference in the decision-making of the agents over 50 iterations. The test showed that the differences are significant ($p = 0.000$) as summarized in Table III. Therefore, it is possible to deduce that the agent's behavior are unique and this uniqueness is statistically significant.

IV. Believability Experiment

This section presents a test game that was developed in order to use the trained models in inference mode, where it can be evaluated against human behavior. The experiment involves 30 users who play with the agents and rate the agents using the agent believability metrics proposed by [12].

A. The Game: Shield Raid

In order to evaluate the performance of the agents, a test game, *Shield Raid*, is developed which is presented in Fig. 6. The game was implemented using *Unity Engine* and C#. In *Shield Raid*, players would need to charge toward turrets that are shooting at them, only using their shields. Each player can choose how much of its limited power charge to use for themselves or share between their comrades. Upon either player reaching the tower, both players would win. If the player shield reaches zero, the player loses. The game uses the trained agents from the previous stage in inference mode. They use the same observation vector and make a d on the observation.

The participants (blue character) receive the amount of shield energy given to them by the agent (green character) and would need to charge the turrets based on what they have received. It is important to note that the bullets from the turrets do not reach the player's starting point, and hence the player would need to charge in order for the game to progress.

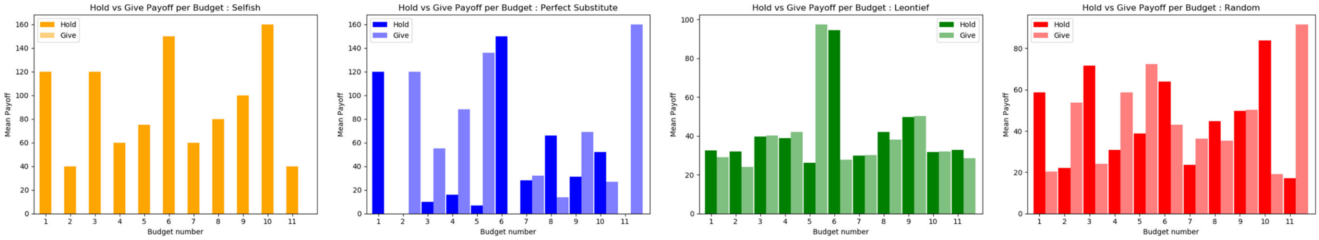

Fig. 5. Plot presenting the mean value of hold and give payoffs for each agent profile (including random) per each budget.

TABLE III
STATISTICAL SIGNIFICANCE TEST 95% CONFIDENCE INTERVAL

	p
Selfish - Leontief	0.000
Selfish - Perfect Substitute	0.000
Selfish - Random	0.000
Leontief - Perfect Substitute	0.000
Leontief - Random	0.000
Perfect Substitute - Random	0.000

B. Believability Questionnaire

The experiment uses the agent believability metrics proposed by [12]. This metric defines eight dimensions to the believability of an agent which the audience can identify. These dimensions include awareness, behavior understandability, personality, visual impact, predictability, behavior coherence, change with experience, and social. As the agents in this experiment visually look the same as the player character, the visual impact dimension has been excluded. The seven remaining dimensions were composed into a questionnaire where each dimension is presented as a question on a Likert scale of five answers that range from 1=strongly disagree to 5=strongly agree.

C. Participants

In total, 30 participants (11 female and 19 male) aged between 19 and 30 (mean = 22.9, SD = 2.81) are recruited. The participants are recruited from current undergraduate students of Computer Science, Games Development and Digital Media at a U.K. university. All participants regularly play games.

D. Procedure

At the start of the experiment, the participants are briefed on how to play the game. Each participant is then moved to an individual cubical with a laptop with the game preloaded. Each participant then charges ten times with each of the three agent profiles called agent A (selfish), agent B (perfect substitute), and agent C (Leontief). The order of playing the agents is selected at random. The 10 charges are chosen randomly from the 11 budgets presented in Table I.

At the end of each set of ten charges, players are asked to complete the believability metric questionnaire for the respective agent. The complete experiment lasts 20 min on average. The participants are encouraged to discuss their responses as they fill out the questionnaire. Their discussion comments are then used for qualitative analysis.

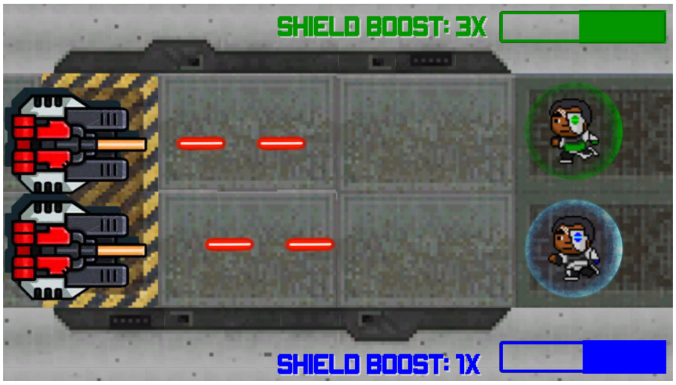

Fig. 6. A screenshot of *Shield Raid* game which uses the trained agents in inference mode for believability testing.

E. Results and Analysis

Overall, the participants find the three agents believable and were able to detect personalities in them. All three agent profiles score a mean believability of above 3.5/5, which is considerably high. Fig. 7 illustrates the mean believability for each agent profile. The selfish agent scores a mean believability of $M = 3.84$ (SD $= 0.28$) across all categories, perfect substitute scores $M = 4.19$ (SD $= 0.21$), and Leontief scores $M = 4.5$ (SD $= 0.23$). Hence, the Leontif was considered the most believable and the selfish the least believable. This is an interesting conclusion of the results that show the increased level of altruism leads to an increased believability of the agents. A Wilcoxon matched pairs signed-rank test is performed to determine whether there is a significant difference in the believability of agents. The results of the test are summarized in Table IV, which illustrates that the differences are statistically significant.

Table V presents the detailed summary of the results for each metric in the questionnaire. The selfish agent is regarded as the least social and the least to change with experience. This is a thought-provoking observation from the participants, as this is the most common behavior in humans as observed by [23]. The selfish agent will always hold the full endowment (shield energy) and would not give any to the other player. While this is regarded as a popular behavior in humans, the participants are expecting some share of the energy in the iterations of the charges in the game. They report that selfish is not learning from previous instances of the game and is not social. Despite this, the participants consider the selfish agent's behavior understandable, coherent, and highly predictable. They report that it is aware of

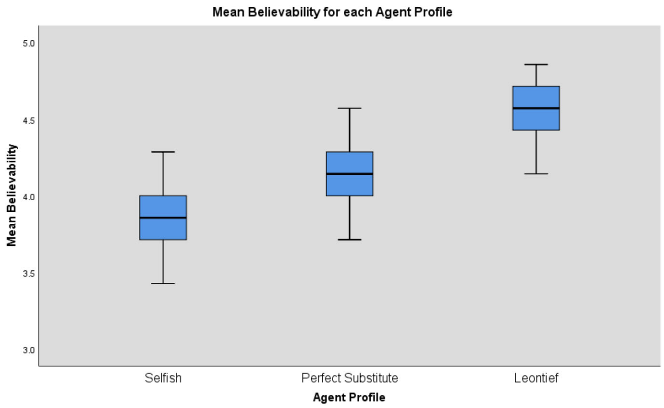

Fig. 7. Box-plot of the mean believability score for each agent profile, illustrating high believability for all agents. This also highlights Leontief as the most believable and selfish as the least believable.

TABLE IV
STATISTICAL SIGNIFICANCE TEST 95% CONFIDENCE INTERVAL

	p	Z
Selfish - Perfect Substitute	0.000	-4.090
Selfish - Leontief	0.000	-4.690
Leontief - Perfect Substitute	0.000	-4.355

TABLE V
BELIEVABILITY METRIC QUESTIONNAIRE RESULTS

Metric	Selfish		Perfect Substitute		Leontief	
	Mean	SD	Mean	SD	Mean	SD
Awareness	4.43	0.91	4.66	0.47	4.76	0.42
Understandability	4.46	0.56	4.5	0.5	4.6	0.55
Personality	4.3	0.52	4.33	0.47	4.6	0.48
Predictability	4.23	0.71	3.6	0.71	4.26	0.77
Coherence	4.4	0.55	3.8	0.7	4.4	0.55
Change with Exp.	3.06	1.36	4.2	0.74	4.36	0.61
Social	2	0.81	4.26	0.62	4.5	0.56

TABLE VI
HUMAN DISTRIBUTION [23] VERSUS AGENT BELIEVABILITY

	Selfish	Perfect Substitute	Leontief
Human distribution	47.2%	22.4%	30.4%
Agent believablity	3.84	4.19	4.5

TABLE VII
MULTINOMINAL LOGISTIC REGRESSION FOR BELIEVABILITY METRICS

Metric	χ^2	p
Awareness	9.003	1.000
Understandability	478.377	0.000
Personality	24.329	0.330
Predictability	68.743	0.000
Coherence	37.208	0.022
Change with Exp.	69.419	0.009
Social	122.155	0.000

its surroundings (budget changes and charges) and conclude that it exhibits a clear personality.

The perfect substitute agent is regarded as the most unpredictable and incoherent. This is a reasonable conclusion by the participants, as ten charges might not be enough for them to recognize the decision pattern of this agent. However, the participants find this agent profile much more aware of its surroundings (it acknowledges the other player by sharing the energy), understandable, social, and dynamic toward change.

Leontief agent is considered as the most believable agent profile. The participants find it the most aware and most coherent of the three. The Leontief behavior leads to equal gains for both players during a charge. This is a behavior that the participants identify clearly and value as being intelligent and the most believable behavior. They can predict its behavior as they find it coherent. Due to the equal sharing, they also identify Leontief as the most social of the three agents. It is significant that some participants mention that the Leontief agent behaves as they would. This might suggest a degree of resonance with the participants' own image of altruism and selfishness which in turn might affect this opinion. This suggests further research into the effect of players' self-image on the perceived believability of AI agents.

V. CONCLUDING DISCUSSIONS

This article presented an approach to create agents that exhibit different levels of selfishness and altruism in their behavior. The agents are trained using deep RL with PPO. Reward functions are defined based on the findings in [23]. The trained agents are validated against the aims and then incorporated into a test game in inference mode. The game was played by 30 participants who then rated the believability of the resulting agents using the agent believability metrics [12].

The experiment results provide thought-provoking observations in relation to the believability of the different agents based on their level of selfishness/altruism, as well as significant implications for believability metrics especially in relation to the players' self-perception. The two main implications of this research are as follows.

1) The definition of human-like behavior should not be solely based on human observation but, rather, it should also include perception of human behavior.
2) Altruism/selfishness could be considered as a dimension of believability metrics. It effects and is effected by behavior understandability, predictability, behaviour coherence, change with experience, and social metrics.

These implications are discussed below, in detail, as the concluding discussions of this research.

The participants rated the three agents in terms of believability. While all three were high on the believability scale, the Leontief ($M = 4.5$/5, SD $= 0.23$) was the most believable, followed by the perfect substitute ($M = 4.19/5$, SD $= 0.21$), and selfish ($M = 3.84/5$, SD $= 0.28$) was the least believable. The Leontief behavior, which aims to distribute the endowment equally, has been rated as the most believable, whereas the selfish behavior, which keeps all the endowment for the agent, has been rated the least believable. The perfect substitute behavior, which tends to give everything away when the price of giving is less than one but keeps everything otherwise, is rated more believable than selfish but less believable than Leontief.

This observation is not in characteristic of observed human behavior. As presented in Table VI and [23], the majority of their participants behave selfishly, and the least number of their participants behave as perfect substitutes with Leontief somewhere

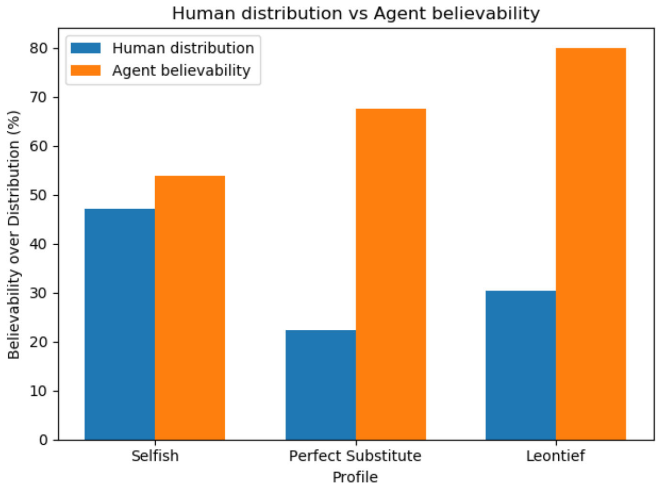

Fig. 8. Bar chart presenting the agent believability compared to the human distribution for each profile type. For clarity of comparison, the value for believability is calculated as the percentage in which each believability score was above mid-point believability (2.5).

in the middle. As such, the expected result should have matched this distribution; however, the selfish agent is rated the least believable, and perfect substitute as the second most believable. The contrast depicted in Fig. 8 shows that players consider altruism as a more believable trait compared to selfishness. It is noteworthy to consider that both *Shield Raid* and the dictator game have a cooperative nature which promotes altruism. The player perception might be different in an adversarial scenario, which warrants further research.

The above results allow us to consider more closely the definition of human-like behavior in the statement: "a more human-like behavior is considered a measure for successful AI" [10], [11]. Our results indicate that the definition of human-like behavior might not be always based on human observation, rather, in some cases, it should be based on the human perception of human behavior. The difference between human behavior and the perception of human behavior has been an ongoing discussion in various communities [46]–[48].

The results also indicate that the level of altruism/selfishness exhibited by the agents had a clear effect on their believability. The agents were similar in every other way except for their altruistic behavior. In order to validate this observation, a multinomial logistic regression (MLR) [49] was performed to investigate the effect of each dimension of the believability metrics on altruism. The likelihood ratio tests show that all coefficients of the model are zero and therefore statistically significant ($p = 0.000$). Both Pearson and deviance χ^2 statistics are statistically insignificant ($p = 1.00$ for both), illustrating that the model fits the data well. Table VII illustrates the detailed results of the MLR. Based on these results, we can deduce that altruism is effected by behavior understandability, predictability, behaviour coherence, change with experience, and social metrics as these have statistically significant effects.

The identified effect of altruism on believability suggests that the believability metrics could benefit from an extra dimension on selfishness/altruism which would allow further insight into the perception of believability of agents. There is also evidence for further research into possible quantification of selfishness/altruism in agents which could lead to the development of more believable and immersive agents within games and other AI industries.

References

[1] D. Daylamani-Zad, L. B. Graham, and I. T. Paraskevopoulos, "Chain of command in autonomous cooperative agents for battles in real-time strategy games," *J. Comput. Educ.*, vol. 6, no. 1, pp. 21–52, 2019.

[2] H. Hoang, S. Lee-Urban, and H. Muñoz-Avila, "Hierarchical plan representations for encoding strategic game AI," in *Proc. Artif. Intell. Interactive Digit. Entertainment*, 2005, pp. 63–68.

[3] C. A. Overholtzer and S. D. Levy, "Evolving AI opponents in a first-person-shooter video game," in *Proc. Nat. Conf. Artif. Intell.*, vol. 20, no. 4, 2005, p. 1620.

[4] N. Cole, S. J. Louis, and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," in *Proc. IEEE Congr. Evol. Comput.*, 2004, vol. 1, pp. 139–145.

[5] B. Gorman and M. Humphrys, "Imitative learning of combat behaviours in first-person computer games," in *Proc. Conf. Comput. Games*, Univ. Wolverhampton Sch. Comput. Inform. Techn., vol. 1, 2007, pp. 1–6.

[6] I. Borovikov *et al.*, "Winning isn't everything: Training agents to playtest modern games," in *Proc. Conf. Artif. Intell.*, AAAI Press, 2019, vol. 1, pp. 1–9.

[7] H. Wang, Y. Gao, and X. Chen, "RL-DOT: A reinforcement learning npc team for playing domination games," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 1, pp. 17–26, Mar. 2010.

[8] F. G. Glavin and M. G. Madden, "Adaptive shooting for bots in first person shooter games using reinforcement learning," *IEEE Trans. Comput. Intell. AI Games*, vol. 7, no. 2, pp. 180–192, Jun. 2015.

[9] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Trans. Games*, vol. 12, no. 1, pp. 1–20, Mar. 2020.

[10] R. Kurzweil, R. Richter, R. Kurzweil, and M. L. Schneider, *The Age of Intelligent Machines*. Cambridge, MA, USA: MIT Press , 1990, vol. 579.

[11] E. Rich and K. Knight, Learning in Neural Network. New York, NY, USA: McGraw-Hill, 1991.

[12] P. Gomes, A. Paiva, C. Martinho, and A. Jhala, "Metrics for character believability in interactive narrative," in *Proc. Int. Conf. Interactive Digit. Storytelling*, Springer, 2013, pp. 223–228.

[13] C. Pacheco, L. Tokarchuk, and D. Pérez-Liébana, "Studying believability assessment in racing games," in *Proc. 13th Int. Conf. Foundations Digit. Games*, ACM, 2018, p. 20.

[14] M. Mateas, "An oz-centric review of interactive drama and believable agents," in *Artificial Intelligence Today*. Berlin, Heidelberg, Germany: Springer, 1999, pp. 297–328.

[15] T. N. Cason and V.-L. Mui, "Social influence in the sequential dictator game," *J. Math. Psychol.*, vol. 42, no. 2–3, pp. 248–265, 1998.

[16] N. Bardsley, "Dictator game giving: Altruism or artefact?" *Exp. Econ.*, vol. 11, no. 2, pp. 122–133, 2008.

[17] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proc. 16th Conf. Auton. Agents MultiAgent Syst*, International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 464–473.

[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[19] T. W. Sandholm and R. H. Crites, "Multiagent reinforcement learning in the iterated prisoner's dilemma," *Biosystems*, vol. 37, no. 1–2, pp. 147–166, 1996.

[20] W. Wang, J. Hao, Y. Wang, and M. Taylor, "Towards cooperation in sequential prisoner's dilemmas: A deep multiagent reinforcement learning approach," 2018, *arXiv:1803.00162*.

[21] N. Anastassacos and M. Musolesi, "Learning through probing: A decentralized reinforcement learning architecture for social dilemmas," 2018, *arXiv:1809.10007*.

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

[23] J. Andreoni and J. Miller, "Giving according to GARP: An experimental test of the consistency of preferences for altruism," *Econometrica*, vol. 70, no. 2, pp. 737–753, 2002.

[24] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley, and C. H. Yong, "Computational intelligence in games," in *Computational Intelligence: Principles and Practice* G. Y. Yen and D. B. Fogel, Eds., Piscataway, NJ, USA: IEEE Comput. Intell. Soc., 2006.

[25] L. Galway, D. Charles, and M. Black, "Machine learning in digital games: A survey," *Artif. Intell. Rev.*, vol. 29, no. 2, pp. 123–161, 2008.

[26] G. N. Yannakakis and J. Togelius, "A panorama of artificial and computational intelligence in games," *IEEE Trans. Comput. Intell. AI Games*, vol. 7, no. 4, pp. 317–335, Dec. 2015.

[27] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[28] S. Risi and J. Togelius, "Neuroevolution in games: State of the art and open challenges," *IEEE Trans. Comput. Intell. AI Games*, vol. 9, no. 1, pp. 25–41, Mar. 2017.

[29] I. Rechenberg and M. Eigen, "Optimierung technischer systeme nach prinzipien der biologischen evolution," *Frommann-Holzboog Verlag, Stuttgart*, 1973.

[30] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.

[31] B. Baker *et al.*, "Emergent tool use from multi-agent autocurricula," 2019, *arXiv:1909.07528*.

[32] M. Jaderberg *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[33] J. Z. Leibo *et al.*, "Malthusian reinforcement learning," in *Proc. 18th Int. Conf. Auton. Agents MultiAgent Syst.*, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1099–1107.

[34] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.

[35] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.

[36] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[37] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[38] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[39] F. Liu *et al.*, "3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 5678–5687.

[40] OpenAI. "Proximal policy optimization," 2017. [Online]. Available: https://openai.com/blog/openai-baselines-ppo/

[41] A. Juliani *et al.*, "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.

[42] R. Forsythe, J. L. Horowitz, N. E. Savin, and M. Sefton, "Fairness in simple bargaining experiments," *Games Econ. Behav.*, vol. 6, no. 3, pp. 347–369, 1994.

[43] V. Capraro and J. Kuilder, "To know or not to know? Looking at payoffs signals selfish behavior, but it does not actually mean so," *J. Behav. Exp. Econ.*, vol. 65, pp. 79–84, 2016.

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

[45] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 196–202.

[46] D. Haski-Leventhal, "Altruism and volunteerism: The perceptions of altruism in four disciplines and their impact on the study of volunteerism," *J. Theory Social Behav.*, vol. 39, no. 3, pp. 271–299, 2009.

[47] M. Blow, K. Dautenhahn, A. Appleby, C. L. Nehaniv, and D. C. Lee, "Perception of robot smiles and dimensions for human–robot interaction design," in *Proc. 15th IEEE Int. Symp. Robot Human Interactive Commun.*, 2006, pp. 469–474.

[48] R. Binns, M. Van Kleek, M. Veale, U. Lyngs, J. Zhao, and N. Shadbolt, "'It's reducing a human being to a percentage': Perceptions of justice in algorithmic decisions," in *Proc. CHI Conf. Human Factors Comput. Syst.*, ACM, 2018, p. 377.

[49] W. H. Greene, *Econometric Analysis*, 7th ed. Boston, MA, USA: Pearson Education, 2012.

Damon Daylamani-Zad received the B.Sc. degree in software engineering from the University of Tehran, Iran, in 2005, and the M.Sc. degree in multimedia computing and the Ph.D. degree in electronic and computer engineering from Brunel University, London, U.K., in 2006 and 2013, respectively.

He is a Senior Lecturer in AI and games with the Digital Media Division, Department of Electronic and Computer Engineering, Brunel University, where he is also an EPSRC Research Fellow. He has published his research findings widely in journals and edited books, and presented his work at several conferences including those hosted by the IEEE. His current research interests include applications of artificial intelligence and machine learning in games, collaborative games, serious gaming, and player modeling and personalisation, especially in massively multiplayer online games (MMOGs), as well as application of evolutionary algorithms in creative computing.

Dr. Daylamani-Zad is a Fellow of the British Computing Society.

Marios C. Angelides received the B.Sc. and Ph.D. degrees from the London School of Economics (LSE), London, U.K.

He is a Professor of Creative Computing and Head of the Digital Media Division, Department of Electronic and Computer Engineering, Brunel University, London, U.K. He was a Lecturer with the LSE. In 1995, he authored his first book titled *Multimedia Information Systems*. He has published over 200 articles in journals, conference proceedings, and edited books.

Prof. Angelides is an Associate Editor and an Editorial Board Member of the *Computer Journal* (Oxford University Press) and an Editorial Board Member of *Multimedia Tools and Applications* (Springer). He is a Chartered Engineer (C.Eng.) and a Chartered Fellow of the British Computer Society (FBCS CITP).