Diverse Behavior Is What Game AI Needs: Generating Varied Human-Like Playing Styles Using Evolutionary Multi-Objective Deep Reinforcement Learning

Ruimin Shen[†,2], **Yan Zheng**[†,1], **Jianye Hao**[1,*], **Yinfeng Chen**[2], **Changjie Fan**[2]
[1]College of Intelligence and Computing, Tianjin University, China.
[2]Fuxi AI Lab, Netease, Inc., Hangzhou, China.
{yanzheng, jianye.hao}@tju.edu.cn, {shenruimin,chenyingfeng1,fanchangjie}@corp.netease.com

Abstract

Designing artificial intelligence for games (Game AI) has been long recognized as a notoriously challenging task in game industry, as it mainly relies on manual design, requiring plenty of domain knowledge. More frustratingly, even spending a lot of efforts, a satisfying Game AI is still hard to achieve by manual design due to the almost infinite search space. The recent success of deep reinforcement learning (DRL) sheds light on advancing automated game designing, significantly relaxing human competitive intelligent support. However, existing DRL algorithms mostly focus on training a Game AI to win the game rather that the way it wins (style). To bridge the gap, we introduce EMO-DRL, an end-to-end game design framework, leveraging evolutionary algorithm, DRL and multi-objective optimization (MOO) to perform intelligent and automatic game design. Firstly, EMO-DRL proposes the *style-oriented learning* to bypass manual reward shaping in DRL and directly learns a Game AI with an expected style in an end-to-end fashion. On this basis, the *prioritized multi-objective optimization* is introduced to achieve more diverse, nature and human-like Game AI. Large-scale evaluations on a *Atari* game and a commercial massively multiplayer online game are conducted. The results demonstrates that EMO-DRL, compared to existing algorithms, achieve better game designs in an intelligent and automatic way.

Introduction

Gaming is at the heart of the entertainment business, and game market is rapidly growing along with fierce competitions. According to the latest *Global Games Market Report* (Newzoo 2019), there are over 2.5 billion active gamers across the world, and over $152.1 billion will be spent on games in 2019. With such a huge and competitive market, the game quality like the entertainment and attraction becomes especially important, as they greatly determinate the success of a game. Unchanging game design sharply degrades the player's enthusiasm, resulting in losing users or even the failure of a game. To keep the entertainment, game companies put a lot of efforts to continuously improve game design (e.g., diverse designs), but often achieve limited progress (Alt 2004). Hence, a more effective and intelligent game designing approach is of great importance and meaning for the game company.

Due to the complexity and heavy reliance on domain knowledge, currently, game design is mainly dependent on human designers. Specifically, behavior trees (BTs) (Millington 2019), are extensively adopted by game companies for designing games, including *Halo* (Isla 2005; Isla 2008), *Spore* (Electronic Arts Inc. 2009), *DEFCON* (Lim, Baumgarten, and Colton 2010), *Bloshock* (Irrational Games 2012) and *Red Dead Redemption 2* (Rockstar Games 2018). However, BTs is built on rules, requiring abundant domain knowledge and labor cost. Besides, contrived rules may be contradictory, leading to potential bugs. Lastly, the more rules used in BTs, the harder it can be maintained, restricting its effectiveness in large scale games. Thus, a more intelligent game design technique towards better entertainment assurance is urgent in the game industry.

Deep reinforcement learning (DRL) has achieved tremendous success in intelligent game playing, from video games (Mnih et al. 2015) to complex chess game (Silver et al. 2016), from first person shooting game (Lample and Chaplot 2017) to real-time strategy game (Foerster et al. 2018). Despite promising, directly adopting DRL for automatic game design still suffers from two major issues. First, most DRL researches focus on finding a policy (game AI) to win the game without considering the way it wins, thus lacking the ability of controlling the policy's behavior. Another issue lies in the trait of DRL that optimizes the policy towards one single objective, resulting in the inability of learning policy with diverse behaviors, which however is critical for improving the game entertainment. Even more, optimizing towards single objective makes the learned policy behave too extremely and unnaturally, thereby leading to bad gaming experience.

To address above limitations in automatic game design, we propose a new algorithm, named EMO-DRL, combining the evolutionary algorithm (EA) with DRL techniques. To effectively control the learned policy's behavior, EMO-DRL proposes the *style-oriented learning* (SOL) to achieve automatic parameters tuning of the reward function for DRL, thereby guiding the policy learning towards the desired behavior. On the other hand, to achieve more nature and human-like behaviors, EMO-DRL proposes the *prioritized*

[*]Corresponding author. † Equal contribution.

multi-objective optimization (PMOO) to optimize policies towards multi-objectives, increasing the diversification of the learned policies. Another advantage is that, compared to optimizing towards single direction, PMOO leverages non-dominated sorting and crowding distance sorting to ensure the learned policies distributed among multiple directions, where more nature and human-like policies can be obtained. In addition, EMO-DRL proposes the asynchronized evolution (AE) - a highly parallel computational framework - to dramatically reduce the learning time in order to efficiently respond to the fast-changing requirements in the game design industry.

For evaluation, EMO-DRL is adopted to design game AIs for the Atari game pong (Mnih et al. 2013) and a commercial massively multiplayer online game (MMOG). Empirical results show that, compared to existing baselines, EMO-DRL can achieve not only specific behavior learning in a end-to-end fashion, but also more diverse, nature and human-like behavior effectively and efficiently.

Preliminaries

Markov Decision Process

Game playing is a process of incessant interactions where the player (i.e., agent) need to take a sequence of actions based the observations (e.g., images) to achieve a specific objective (e.g., winning). Game playing can be modeled as a Markov Decision Process (MDP), which consists of a tuple (S, A, R, T, γ), where S is the set of states and usually referred to as the observations, A is the action space that the player used for game playing, $R(s, a)$ is the reward function $S \times A \rightarrow \mathbb{R}$ indicating whether the action a taken at state s is good or bad, $T(s, a, s')$ is the transition function $S \times \mathbf{A} \times S \rightarrow [0, 1]$ giving the possibility of transiting into the new state s' after taking action a at state s, and $\gamma \in [0, 1]$ is a discount factor (Sutton and Barto 2018).

Asynchronous Advantage Actor-Critic

Asynchronous Advantage Actor-Critic (A3C) (Mnih et al. 2016) is one of the state-of-the-art DRL algorithms, aiming at training an agent to play games intelligently and automatically. In general, A3C maintains a policy $\pi(a|s; \theta)$ and an estimation of the value function $V(s; \theta_v)$ parameterized by θ and θ_v, respectively. During game playing, the agent (actor) will execute the action a following $\pi(a|s; \theta)$ and receive an immediate reward signal r_t, which will be used by the critic to learn the value function $V(s; \theta_v)$. As a result, the value function $V(s; \theta_v)$, in turn, will be leveraged to optimize the agent's policy $\pi(a|s; \theta)$ to maximize the cumulative rewards $\sum_{t=1}^{\infty} r_t$ received during entire game playing process using the following gradient:

$$\nabla_\theta \log \pi(a_t|s_t; \theta)[r_t + V(s_{t+1}) - V(s_t)], \tag{1}$$

where s_t, a_t, r_t, s_{t+1} are sampled by the actor[1]. Notable, A3C leverages multiple asynchronous agents (critics) to simultaneously explore the environment, dramatically speeding up the efficiency of exploring, sampling and policy learning.

Multi-Objective Optimization

Multi-Objective Optimization (MOO) (Deb et al. 2000) is an effective optimization method for solving optimization problems with multiple objectives. Different from standard evolutionary algorithm (e.g., genetics algorithm) evaluating the solution using a scalar fitness value and optimization from a single objective perspective, MOO extends the fitness value to vectors and achieve optimization from a multi-objective perspective. Consequently, the way of selecting "better" offspring changes from scalar-based comparison to vector-based comparison, requiring to evaluate solution from multiple angles. The advantage is that offspring evolved from MOO can simultaneously achieve high performance regarding multiple objectives and better diversity among multiple objectives (Van Moffaert and Nowé 2014).

Problem Formulation

To make a commercial game popular and successful, game companies spend plenty of time in designing different diverse behaviors for game artificial intelligence (Game AI) to keep its attraction and entertainment. For instance, in a online combat game, designers need to design various roles with different playing styles to attract the players (NetEase Games 2016). Another example is a action-adventure game (Rockstar Games 2018), where over 12,000 lines of code are written for only one single game character to ensure its behavioral diversity. This designing work heavily relies on the domain knowledge and are mostly done manually, thus requiring tremendous human and time cost. For most game companies, currently there is a lack of effective approach of designing diverse game character behaviors automatically.

Different Game AIs behave differently from a visual perspective (e.g., aggressive/defensive behaviors in a combat game), and the behavior characteristics of a Game AI playing the game is also referred to as the style in game industry. In general, designers not only need a Game AI with a specific style, but also nature and human-like Game AIs with diverse styles. For this purpose, given a Game AI, its style (denoted by $\mathfrak{S}_\pi$) is quantify as follows:

$$\mathfrak{S}_\pi = [v^I] \tag{2}$$

where I is a business indicator and v^I is a statistical scalar measuring the performance of π in term of I during game playing. For instance, to measure whether a Game AI has an aggressive style, we can use the harmness indicator, and the total damage $v^{I_{\text{harm}}}$ that Game AI produces during game playing can be used for measuring the aggressiveness. Normally, one indicator is enough for measuring a certain type, and, unless stated otherwise, $\mathfrak{S}_\pi$ is regarded as a scalar hereafter.

Practically, designing a Game AI for a combat game to beat the player is relative simple, however, what really hard

[1] For better understanding, the advantage function $A(s, a) = r_t + V(s_{t+1}) - V(s_t)$ is measure by 1-step return, which however can also be measured by n-step return.

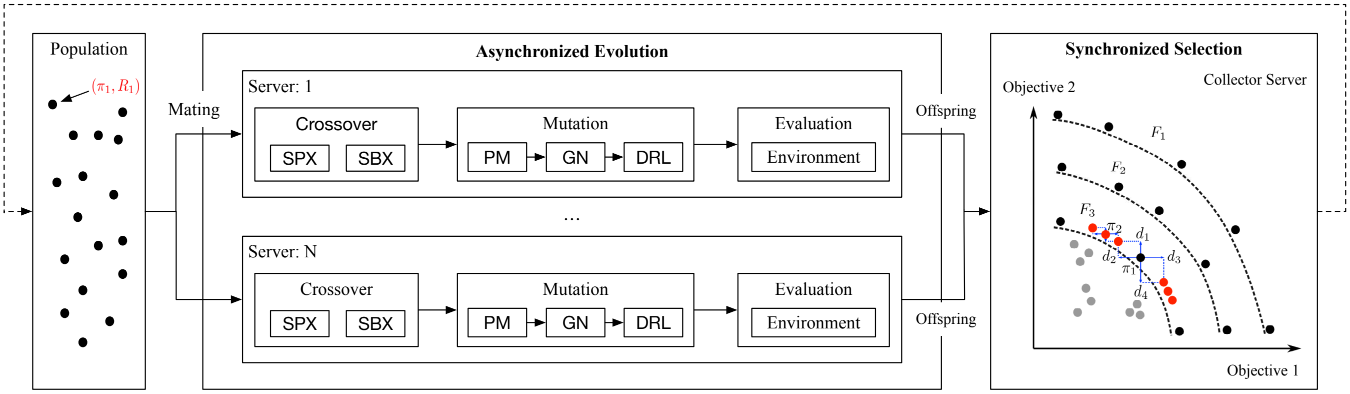

Figure 1: Overall framework of EMO-DRL

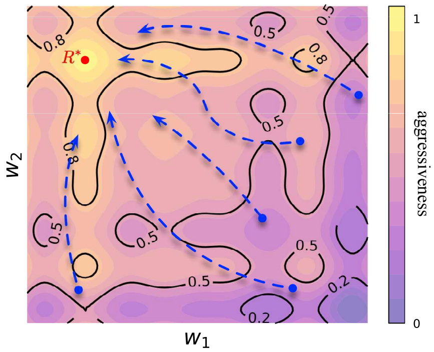

Figure 2: The aggressiveness of policy π learned by different weights. The x- and y-axis are weights while color depicting the extent of a policy being a aggressive style.

is to design one with a specific style that in need. This is because DRL learn a policy π^* according to the reward function R, which thus in turn determines the style of policy π^* as follows:

$$\mathfrak{S}_{\pi^*} \sim \pi^* \sim Game(S, A, R, T, \gamma) \tag{3}$$

Normally, R consists of indicators that only focus on winning, lacking the ability of controlling the style $\mathfrak{S}_{\pi^*}$. However, to overcome this, more style-related component r_i is adopted to shape reward R as follows:

$$\mathfrak{S}_{\pi} \sim R = \sum_{i \in n} (w_i * r_i), \tag{4}$$

where n style-related item r_i are linear combined with the corresponding weights w_i to adjust the learned styles. Despite theoretically possible, desired style is still hard to obtained for two major reason: 1) the choice of weights requires abundant domain-knowledge; 2) Even a slight change in weights, due to the nonlinear relationship between the weights and styles, would make the specific style unpredictable. Specifically, Fig. 2 is a schematic diagram briefly describing the relationship between weights and the target style (an aggressive style), where one can find that most of the weight combinations (in the x- and y-axis) are incapable of achieving significant aggressiveness (yellow areas). Only a few combinations (around the red dot R^*) can lead to very aggressive style, which however are barely possible to find by manual parameters tuning. On the other hand, even if such combination can be found, manual tuning requires tremendous labor- and time-cost, as multiple styles are normally required for diverse behavior designing. Therefore, the first challenge in designing diverse behavior is to how to learn a policy with any specific style in need automatically and efficiently.

Another challenge is that DRL aims at maximizing the reward function, making the learned style too extreme and unnatural. For instance, in a combat game, aggressive policy learned by DRL will always attack the player without even moving. Meanwhile a defensive one learned by DRL will always avoiding the player without any attacks. Both styles behavior too extreme to be accepted as feasible solutions. Practically, more natural styles (e.g., 70% aggressive and 30% defensive styles) are regarded as better solutions. However, for designer, such nature style is hard to accurately achieved, since the corresponding reward function (weights) is barely possible to be found manually.

The last challenge is how to ensure the important abilities of learned styles. Specifically, in a combat game, ability of winning the game should be always guaranteed regardless of aggressive or defensive styles. Achieving such an complex style, for designer, is even harder than a nature one, since multiple traits (e.g., winning and aggressiveness) should be achieved at the same time. Even more, winning and defensiveness, to some extent, are contradictory since they will respectively facilitate and suppress attack behaviors, increasing the difficult of learning such complex styles.

To the best of our knowledge, most game companies still do game AI design by manually tuning parameters in reward function, and suffer from all aforementioned issues, which however can be all solved by the proposed EMO-DRL effectively and efficiently.

Algorithm

To achieve diverse nature and human-like playing styles, EMO-DRL is proposed by leveraging DRL on the basis of EA. Specifically, EMO-DRL proposes the SOL to bypass the manual parameters tuning and directly obtain the specific

Algorithm 1: EMO-DRL

```
1  input   : $n$: the size of the population
2  output  : $P$: the population of policies.
3  // Initialize $P$ with randomized $(\pi_i, R_i)$
4  $P = \{\{\pi_1, R_1\}, ..., \{\pi_n, R_n\}\}$;
5  // Initialize $Q$ with a synchronized set
6  $Q = \{\}$;
7  repeat
8      // Asynchronized Evolution
9      $p_1, p_2, ..., p_u = mating(P)$;
10     $q_1, q_2, ..., q_v = crossover(p_1, p_2, ..., p_u)$;
11     for $q \in \{q_1, q_2, ..., q_v\}$ do
12         $q = evaluate($DRL-Train$(mutate(q)))$;
13     $Q = Q + \{q_1, q_2, ..., q_v\}$
14     // Synchronized Selection
15     if $|Q| \geq n$ then
16         $P$ = Diverse-Select$(P \cup Q, n)$;      ▷ (see Alg.2)
17         $Q = \{\}$;
18 until the stop criteria is satisfied;
19 return $P$;
```

style in an effective end-to-end manner. On the other hand, EMO-DRL leverages PMOO together with the SOL to learn divers nature and human-like style. It is worth mentioning that EMO-DRL adopts distributed parallel architecture to boost the learning efficiency to satisfy the fast-changing demands in designing diverse styles.

Overview

From a higher perspective, EMO-DRL is built on an evolutionary process consists of two parts: *Asynchronized Evolution* and *Synchronized Selection* (As shown in Fig. 1). In the beginning, EMO-DRL initializes and maintains a population of candidates (black dots). Then, in asynchronized evolution, all candidates are mutated and evolved to create offspring (new styles) through automatic reward shaping, and the newborns' style will be evaluated by environment interaction (game playing). After evaluation, in the synchronized selection, only better offspring will be kept for the subsequent evolution while the rest be eliminated. We referred to this entire evolutionary process as the SOL, by which varied and human-like Game AIs can be efficiently obtained in an end-to-end fashion.

Style-Oriented Learning

Based on the evolutionary framework (Jaderberg et al. 2017; Li et al. 2019), SOL eliminates the intermediate manual weights tuning and directly learns policies with target styles. The algorithm flow of SQL is outlined in Alg. 1. Specifically, SOL initializes a population of candidates, each of which is a pair of π (e.g., a neural network parameterized by θ) and R (parameterized by w). To create new styles, SOL leverages the SPX and SBX to crossover the neural network in π and weights in R, respectively. After that, GN and PM are adopted to mutate the neural network in π and the weights in R [2].

Once finished, the resulting parameters θ' and w' are automatically fine-tuned, and the newborn $(\pi_{\theta'}, R_{w'})$ are trained using A3C algorithm (Mnih et al. 2016) (line 12). After training, each individual is evaluated to obtain their style $\mathfrak{S}_\pi$ (a scalar) measuring by some business indicators (e.g., harmness). Normally, individuals with higher $\mathfrak{S}_\pi$ will be kept for further evolution, while the rests be eliminated. This evolutionary cycle repeats until reaching a certain number of iterations. In this way, policies in the last population will have higher $\mathfrak{S}_\pi$, meaning that all policies have a significant certain style (e.g., aggressive style) without any manual parameters tuning.

Different from standard EA, SOL uses DRL-Train after mutation (line 12) to achieve a more efficient learning due to the power of gradient-based optimization. Besides, diverse-select is employed to achieve diverse behavior learning (line 16), which will be detailed in the following section.

It is worth mentioning that, to respond to the fast-changing requirements in game designing, SOL is built on a distributed parallel architecture. Specifically, SOL divides the evolutionary process into two part and distributes them into multiple servers to boost efficiency. As shown in Fig. 1, SOL divides the population and distributes them to multiple servers to achieve asynchronized evolution including: crossover, mutation (including DRL training) and evaluation (asynchronized part in Alg. 1). Each time a new policy, it will be send back to a collector (right part in Fig. 1) and be added into a synchronized set Q, and the server will repeat this process from the beginning. All servers will running efficiently without communication or blocking each other. Once the size of Q reached a threshold n, SOL utilizes PMOO to select better offspring as the next new population (line 16 in Alg. 1). Comparing to computing resources EMO-DRL needs, it greatly reduces the time for obtaining policies with specific styles, which is of more importance for game companies.

Generating Diverse Styles

Despite SOL providing a efficient way to achieve desired style, learning only one style at a time introduces unnecessary time cost, especially when the designer needs multiple styles for game designing. More importantly, single-objective optimization often make the learned policy performs too extremely to be accepted as effective solutions, resulting in bad game experience. For instance, neither an aggressive style only attack without moving, nor a defensive one that only run away is an acceptable design. By contrast, more nature policies (e.g., 70% aggressive and 30% defensive styles) are practically required. To address this, we propose the Prioritized Multi-objective Optimization (PMOO) combined with the SOL to generate more nature Game AIs with multiple styles simultaneously.

[2] SPX, SBX, PM and GN are short for the Single-point Crossover (Deb and Agrawal 1994), Simulated Binary Crossover (Deb and Kumar 1995), Polynomial Mutation (Deb and Goyal 1996) and Gaussian Noise (Such et al. 2018), respectively

To achieve this, PMOO is responsible for guiding policies evolve towards multiple styles. Comparing to evolving towards single style (i.e., using scalar $\mathfrak{S}_\pi$), PMOO needs to evaluate a policy's style from multiple aspects to achieve a diverse offspring selection, leading to two major differences: 1) the way to measure a policy's style and 2) the way to select offspring.

To evaluate a policy from a multi-style perspective, we measure the style using multiple indicators as follows:

$$\mathfrak{S}_\pi = [v^{I_0}, \cdots, v^{I_i}, \cdots, v^{I_n}] \tag{5}$$

where each v^{I_i} measuring the performance of policy π regarding indicator I_i. For instance, in combat game, each policy is evaluated by $\mathfrak{S}_\pi = [v^{I_{\text{agg}}}, v^{I_{\text{def}}}]$ with $v^{I_{\text{agg}}}$ and $v^{I_{\text{def}}}$ measuring the extend of π being aggressive and defensive style, respectively. In this way, each policy is evaluated from multiple aspects, providing the foundation of subsequent diverse offspring selection.

The way of selecting better offspring differs because each policy's type is no longer measured by a single scalar, but a style vector. Thus, to better capture the capability relation of different policies and select better offspring, we define the policy domination relation as follows:

As a result, comparing to vanilla SOL selecting policies with higher scalar $\mathfrak{S}_\pi$ from a single-objective perspective, PMOO selects the ones, which can dominate others, as better offspring. However, policies may not dominate each other (i.e., $\pi_0 \not\succ \pi_1$ and $\pi_1 \not\succ \pi_0$) since π_0 may have larger in the i-th dimension, but smaller in the j-th dimension (i.e., $\mathfrak{S}^{(i)}_{\pi_0} > \mathfrak{S}^{(i)}_{\pi_1}$ and $\mathfrak{S}^{(j)}_{\pi_0} < \mathfrak{S}^{(j)}_{\pi_1}$), meaning that π_0 has a better aggressiveness than π_1, but a worse defensiveness. From the multi-style learning perspective, it hard to say which one is better, since we want to find both styles simultaneously. To identify these policies, we define the *non-dominated set* as follows:

NDS represents the "best" individuals in the population, since candidates in the NDS are not dominated by any others in the NDS. It provides an effective way to group candidates together, whereby the population P can be divided into different NDSs for selecting better individuals.

Alg. 2 gives the pseudo-code[3] and Fig. 1 (right part) visualizes the offspring selection using two optimization objectives (two styles). Given a population P, the Pareto frontier is identified using non-dominated sorting algorithm (ND_Sort) (Deb et al. 2000), and added into the new population, then removed from the original population. This process repeats until the size of the new population reaches a pre-defined threshold (line 3-7 in Alg. 2). Each Pareto frontier is a NDS (e.g., $F_1, F_2, F_3, ...$ in Fig. 1), containing a number of candidates. and that F1 generally exceeds F2 because $\forall \pi \in F1.(\nexists \pi' \in F2, \pi' \succ \pi)$. Once the size of the new population P' adding the current Pareto frontier F_i exceeds the threshold n, only $n - |P'|$ candidates in the F_i will be selected in to the P' to ensure the size of the new population exactly n (Line 8-12). An intuitive example is given in Fig. 1, where only part of the F_3 can be selected. To achieve this, the crowding distance sorting algorithm (CD_Select) is adopted (Line 10) to measure the density of each candidates in F_3, and select sparse candidates to construct a more diverse offspring. In Fig. 1, the density of π_1 is computed based on the distance between two nearest surrounding neighbors regarding in terms of two objectives (i.e., $d1 + d2 + d3 + d4$). Afterwards, candidates with sparse density will be selected as the new offspring. In this way, EMO-DRL will push some policies along with two objectives as far as possible (extreme aggressive/defensive styles) and evenly spread the rest polices among these two extreme styles. The resulting population contains all kinds of mixed styles (i.e., 70% aggressive and 30% defensive styles), where designers can choose any nature style they prefer as the Game AI.

Algorithm 2: Diverse-Select

```
   input : $P$: the population, $n \leq |P|$: the number of
           candidates to be selected
   output: $P'$: the selected population
 1 $P' \leftarrow \emptyset$;
 2 loop
 3     // Select the Pareto frontier $F$ from $P$
 4     $F = $ ND_Sort$(P)$;                      ▷ see Appendix
 5     if $|P'| + |F| \leq n$ then
 6         $P' \leftarrow P' \cup F$;
 7         $P = P \setminus F$
 8     else
 9         // Squash set by crowiding distance
10         $F' \leftarrow$ CD_Select$(F, n - |P'|)$ ; ▷ see Appendix
11         $P' \leftarrow P' \cup F'$ ;
12         break
13 return $P'$
```

Guaranteeing Important Abilities

Intuitively, defensive and aggressive styles can be measured and obtained by $\mathfrak{S}_\pi = [v^{I_{\text{agg}}}, v^{I_{\text{def}}}]$ with $v^{I_{\text{agg}}}$ and $v^{I_{\text{def}}}$ measuring the extent of π being a aggressive and defensive styles. However, using a scalar v^{I_i} has a potential limitation in creating complex human-like behaviors like "hit-and-run".

Consider a two-player combat game, for an agent, the ability to win should be firstly guaranteed regardless of its playing styles. Specifically, a defensive but winnable style (e.g., "hit-and-run") tries to keep away from the opponent, but still need to attack to win the combat. However, such complex (def-win) style is hard to achieve by using only a scalar $v^{I_{\text{def}}}$, since scalar is not enough to encode complex behaviors. For instance, $v^{I_{\text{def}}}$ can only measuring the extent of avoiding the opponent. Maximize $v^{I_{\text{def}}}$ results in a Game AI that always avoiding the opponent and cannot win. To overcome this, instead of single indicator I_{def}, PMOO proposed to use a vector of indicators (i.e., I_{win} and I_{def}) and measure the extent of π being a "def-win" style (denoted by $\vec{v}^{I_{\text{def-win}}}$) as follows:

$$\vec{v}^{I_{\text{def-win}}} = [v^{I_{\text{win}}}, v^{I_{\text{def}}}], \tag{6}$$

[3] More details can be found in appendix: http://bit.ly/33WAEsE

where two scalar $v^{I_{\text{win}}}$ and $v^{I_{\text{away}}}$ measures a complex behavior together. Consequently, with the style measurement changing from using a scalar to vector, $\mathfrak{S}_\pi$ adapters into a vector of vectors as follows:

$$\mathfrak{S}_\pi = [\vec{v}^{I_0}, \cdots, \vec{v}^{I_i}, \cdots, \vec{v}^{I_n}], \tag{7}$$

where each vector $\vec{v}^{I_i}$ measures the degree of being a complex style I_i. Thus, the domination relation (in Def. **??**) need to be adapted for handling the new $\mathfrak{S}_\pi$. PMOO proposes the prioritized comparison for vectors as follows:

$$\begin{aligned} \vec{v}^{I_i}_{\pi_0} > \vec{v}^{I_i}_{\pi_1} \text{ if and only if } \exists j \in [0, n], (\vec{v}^{I_i}_{\pi_0})_{(j)} > (\vec{v}^{I_i}_{\pi_1})_{(j)} \\ \text{and } \forall k \in [0, j), (\vec{v}^{I_i}_{\pi_0})_{(k)} = (\vec{v}^{I_i}_{\pi_1})_{(k)} \end{aligned} \tag{8}$$

where $(\cdot)_{(i)}$ represents the i-th value in the vector. In this way, PMOO will select offspring with a better $\vec{v}^{I_{\text{def-win}}}$ with higher value in both $[v^{I_{\text{win}}}, v^{I_{\text{away}}}]$ dimensions, whereby the def-win style is achievable with high $v^{I_{\text{win}}}$ and $v^{I_{\text{away}}}$ simultaneously. It is worth-mentioning that, by leveraging $\mathfrak{S}_\pi = [\vec{v}^{I_{\text{def-win}}}_\pi, \vec{v}^{I_{\text{agg-win}}}_\pi]$, EMO-DRL can not only learn complex styles (e.g., def-win, agg-win), but also guarantees their victory in game.

Experiments

This section presents empirical results on a classic Atari game *pong* and a commercial MMOG game *Treacherous Water Online*. Comparison between EMO-DRL and A3C are performed to verify their effectiveness in generating Game AI, especially generating more diverse, nature and human-like Game AIs. Besides, we visualize the behaviors of all learned Game AIs for further comparison [4]. It is worth mentioning that, due to parameters tuning heavily-dependents on the domain knowledge, to ensure the fairness as far as possible, experienced front-line designers are invited to tune the parameters for A3C in the following experiments.

Atari Game

Atari pong is a widely used benchmark in the DRL community for evaluating the performance of Game AIs, whose first priority is to control the green paddle shown in Fig.3 to beat the human player. To obtain more diverse styles, both methods uses the same reward shaping as follows:

$$R = w_1 * r_{\text{env}} + w_2 * r_{\text{busy}} + w_3 * r_{\text{lazy}} \tag{9}$$

where r_{busy} and r_{lazy} are shaping items, affecting the resulting styles. r_{env} is the environment reward and will be 1 if score otherwise 0. r_{busy} will be 1 if the paddle move and 0 otherwise. r_{lazy} will be 1 if no action is taken and 0 otherwise. Notable, A3C manually tunes the weight w_i to obtain desired Game AIs, however, EMO-DRL leverages automatic tuning.

Firstly, we investigate the effectiveness of EMO-DRL in generating Game AI with different styles. Then, detailed quantitative comparisons of the Game AIs learned by EMO-DRL and A3C is given to further demonstrate the advantages of EMO-DRL.

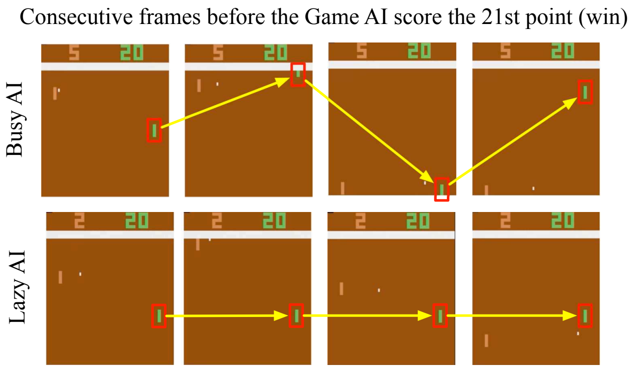
Consecutive frames before the Game AI score the 21st point (win)
Busy AI
Lazy AI
5
20
2
20

Figure 3: Visualization of the win-busy and win-lazy styles learned by EMO-DRL.

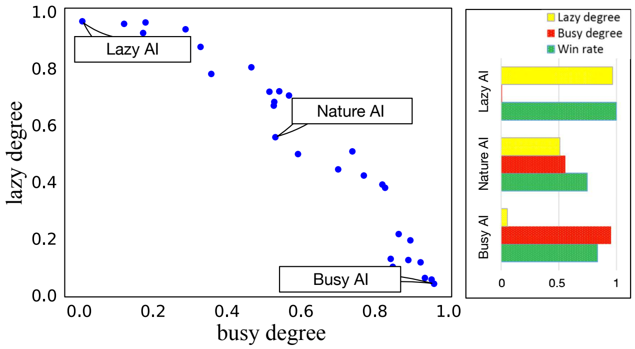

Figure 4: Distribution of styles learned by EMO-DRL among two objectives (left), and quantitative analysis of three styles in term of three related indicators (right).

Generating Diverse Styles Fig. 3 visualizes the busy AI and lazy AI styles learned by EMO-DRL, where one can find that two styles varies greatly. Specifically, during game playing, the busy AI moves a lot while the lazy AI does not. Moreover, both styles can win the game, practically proving that EMO-DRL can not only learn a Game AI to win, but also control the way it wins[5]. Another advantages is that, comparing to A3C, EMO-DRL achieve this in a end-to-end way without any manual parameter tuning.

Quantitative Comparisons To further investigate the advantages of EMO-DRL, statistical indicators are adopted to quantitative measure the Game AI's behavior during game playing, and detailed comparisons are conducted.

The behavioral data of a Game AI during game playing is collected and calculated using $I_{\text{win-rate}}$, I_{busy} and I_{lazy} indicators, measuring the winning rate, busy degree and lazy degree, respectively. $I_{\text{win-rate}}$ measures the win rate after 10 runs. I_{busy} measures the cumulative time of agent moving, and I_{lazy} the cumulative time of agent not moving. On this basis, Game AIs learned by EMO-DRL is quantify and visualized in Fig. 4,

[4] More videos of the learned Game AIs can be found in our website: https://sites.google.com/view/emo-drl/home

[5] Videos of diverse styles in Atari game: https://youtu.be/lE-5OiZbaPY

Table 1: Averaged statistical results (10 runs) regarding related indicators of learned styles in Atari game.

Indicators		Atari Game win rate(score)	busy degree	lazy degree
EMO-DRL	Busy AI	**15.2**	**0.954 (best)**	0.055
	Nature AI	**13.6**	0.557	0.531
	Lazy AI	**18.2**	0.008	**0.966 (best)**
A3C	Busy AI	-9.9	0.908	0.215
	Nature AI	-5.0	0.304	0.857
	Lazy AI	11.8	0.257	0.853

where one can find that EMO-DRL not only learn extreme styles (lazy- and busy-win styles) along with two objectives, but also more nature AIs evenly distributed among them. Quantitative analysis of three styles demonstrates that busy AI and lazy AI performs extreme behavior, however, more nature styles are also learned by EMO-DRL. Designers can pick whichever varied nature AI in need.

On the other hand, the comparison of the Game AIs learned by EMO-DRL and A3C is conducted, and Tab. 1 describes the results. One finding is that the busy AI learned by A3C has low score (-9.9), meaning that such AI fails to win the game. Another finding is that the nature AI learned by A3C not only lose the game (-5.0), but also fails to perform nature due to its high lazy degree (0.857). This further prove that nature styles is hard to learned by manual parameters tuning, let alone winning the game at the same time. By contrast, the busy AI and lazy AI learned by EMO-DRL not only achieve higher busy (0.954) and lazy degree (0.966) than the ones learned by A3C respectively, but also be able to win the game simultaneously. This demonstrates the effectiveness of EMO-DRL in learning complex style automatically. Another advantages is that the nature AI learned by EMO-DRL achieve 0.557 busy degree and 0.531 lazy degree, making itself a suitable nature Game AI, which however is unlearnable by A3C.

Treacherous Water Online

In this experiments, a real-world commercial MMOG game (TWO) is adopted for evaluation. To make our study feasible, we only selected one combat scenario, where a easy-to-kill wizard need to beat a hard-to-kill fighter (shown in Fig. 5). Both A3C and EMO-DRL need to learned diverse styles by tuning the parameters of the following shaped reward function:

$$R = w_1 * r_{\text{env}} + w_2 * r_{\text{aggressive}} + w_3 * r_{\text{defensive}} \qquad (10)$$

where $r_{\text{aggressive}}$ and $r_{\text{defensive}}$ encourage attacking and avoiding the opponent, respectively. r_{env} is the environment reward and will be positive if wins otherwise negative. r_{busy} will be positive if the agent behavior aggressive and negative otherwise. $r_{\text{defensive}}$ will be positive if being defensive and negative otherwise.

Generating Diverse Styles Fig. 5 (c,d) visualizes the footprint of the aggressive AI (Agg AI) and defensive AI (Def AI) learned by EMO-DRL, where one can find that two styles varies greatly. Specifically, during game playing, the Agg AI always attack, barely moving, producing shorter trajectories. By contrast, Def AI creates longer trajectories by running around the field to avoid the opponent (known as the "hit-and-run"). More importantly, both styles succeed in defeating the opponent, confirming the ability of EMO-DRL in generating varied human-like styles[6].

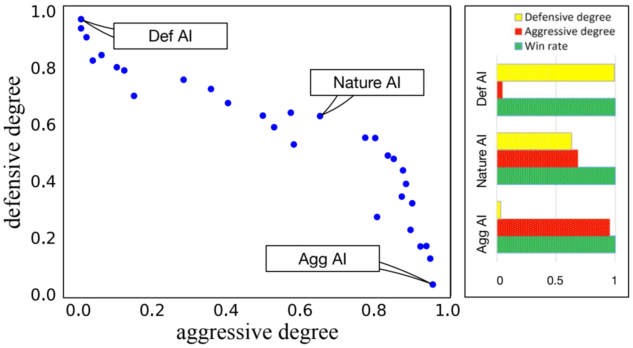

Figure 5: (a) is a round fighting ground in TWO and (b,c,d) visualize the footprint of different styles in game playing. The blue and yellow dots depict trajectories of Game AI and its opponent, respectively

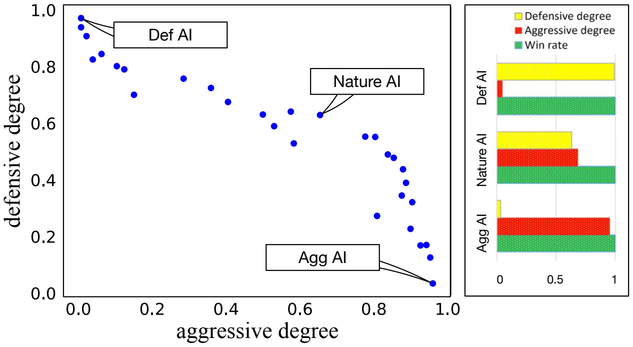

Figure 6: Distribution of styles learned by EMO-DRL among two objectives (left), and quantitative analysis of three styles in term of three related indicators (right).

Quantitative Comparisons Statistical indicators $I_{\text{win-rate}}$, I_{agg}, and I_{def} are adopted to quantitative measure the behavior of the learned styles by measuring the winning rate, aggressive degree and defensive degree, respectively. $I_{\text{win-rate}}$ measures the win rate after 10 runs. I_{agg} measures the cumulative time used for defeating the opponent. It will be larger if defeats soon and smaller otherwise. I_{def} calculates the running time of the Game AI avoiding the opponent, and will be larger is always avoiding the opponent and smaller otherwise. Fig. 6 visualizes the distribution of the statistical results. Styles learned by the EMO-DRL are evenly distributed between two dimensions, where more varied and nature policies exists.

Lastly, quantitative comparison results of learned styles

[6] Videos of diverse styles in TWO game: https://youtu.be/n6O-IzXTVvc

Table 2: Averaged statistical results (30 runs) regarding related indicators of learned styles in TWO game.

Treacherous Water Online Game

Indicators		win rate	aggressive degree	defensive degree
EMO-DRL	Agg AI	**1.000**	**0.955 (best)**	0.032
	Nature AI	**1.000**	0.685	0.632
	Def AI	**1.000**	0.048	**0.993 (best)**
A3C	Agg AI	0.233	0.891	0.136
	Nature AI	0.760	0.188	0.916
	Def AI	0.633	0.164	0.938

are given in the Tab. 2. First, EMO-DRL achieve better AIs in terms of two objectives than A3C (i.e., 0.955 in aggressive degree and 0.993 in defensive degree). Second, EMO-DRL succeed in achieve more nature AIs by automatic tuning, while A3C fails. Lastly, the aggressive AI learned by A3C fails to win, however, EMO-DRL can achieve an aggressive AI to win efficiently and automatically.

Conclusion

This paper proposes the EMO-DRL algorithm to generate Game AIs with varied human-like playing styles efficiently and automatically. SOL provide an end-to-end fashion to guide the policy learning towards the desired behavior accurately. Moreover, PMOO is introduced to achieve human-like behaviors, and increasing the diversification of the learned policies. Finally, a distributed parallel architecture is adopted to boost the computational efficiency. The empirical results demonstrate that the EMO-DRL algorithm not only generates varied game styles with better diverse and extreme behaviors, but also obtains higher winning scores.

References

[Alt 2004] Alt, G. 2004. The suffering: A game ai case study. In *Challenges in Game AI workshop, Nineteenth national conference on Artificial Intelligence*, 134–138.

[Deb and Agrawal 1994] Deb, K., and Agrawal, R. B. 1994. Simulated binary crossover for continuous search space. *Complex Systems* 9(2):115–148.

[Deb and Goyal 1996] Deb, K., and Goyal, M. 1996. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics* 26(4):30–45.

[Deb and Kumar 1995] Deb, K., and Kumar, A. 1995. Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems. *Complex Systems* 9(6):431–454.

[Deb et al. 2000] Deb, K.; Agrawal, S.; Pratap, A.; and Meyarivan, T. 2000. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation - NSGA-II. *PPSN.*

[Electronic Arts Inc. 2009] Electronic Arts Inc. 2009. Spore. https://www.spore.com/.

[Foerster et al. 2018] Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. *AAAI.*

[Irrational Games 2012] Irrational Games. 2012. Bioshock. http://www.bioshockgame.com/.

[Isla 2005] Isla, D. 2005. Handling complexity in the halo 2 AI. In *Game Developers Conference, GDC.*

[Isla 2008] Isla, D. 2008. Halo 3-building a better battle. In *Game Developers Conference, GDC.*

[Jaderberg et al. 2017] Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; Fernando, C.; and Kavukcuoglu, K. 2017. Population Based Training of Neural Networks. *CoRR.*

[Lample and Chaplot 2017] Lample, G., and Chaplot, D. S. 2017. Playing FPS Games with Deep Reinforcement Learning. *AAAI.*

[Li et al. 2019] Li, A.; Spyra, O.; Perel, S.; Dalibard, V.; Jaderberg, M.; Gu, C.; Budden, D.; Harley, T.; and Gupta, P. 2019. A Generalized Framework for Population Based Training. *CoRR.*

[Lim, Baumgarten, and Colton 2010] Lim, C.-U.; Baumgarten, R.; and Colton, S. 2010. Evolving behaviour trees for the commercial game defcon. In *European Conference on the Applications of Evolutionary Computation*, 100–110. Springer.

[Millington 2019] Millington, I. 2019. *AI for Games, Third Edition.* CRC Press.

[Mnih et al. 2013] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR.*

[Mnih et al. 2015] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

[Mnih et al. 2016] Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. *ICML.*

[NetEase Games 2016] NetEase Games. 2016. A chinese ghost story online game. https://qnm.163.com/jobs/new/.

[Newzoo 2019] Newzoo. 2019. Global games market report. https://newzoo.com/solutions/standard/market-forecasts/global-games-m

[Rockstar Games 2018] Rockstar Games. 2018. Red dead redemption 2. https://www.rockstargames.com/reddeadredemption2/.

[Silver et al. 2016] Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature.*

[Such et al. 2018] Such, F. P.; Madhavan, V.; Conti, E.;

Lehman, J.; Stanley, K. O.; and Clune, J. 2018. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv: 1712.06567*.

[Sutton and Barto 2018] Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

[Van Moffaert and Nowé 2014] Van Moffaert, K., and Nowé, A. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *J. Mach. Learn. Res.*