Generating Behavior-Diverse Game AIs with Evolutionary Multi-Objective Deep Reinforcement Learning

Ruimin Shen[1,⁎], **Yan Zheng**[2,3,⁎], **Jianye Hao**[2,4,5,†], **Zhaopeng Meng**[2]
Yingfeng Chen[1,†], **Changjie Fan**[1], **Yang Liu**[3]
[1]Fuxi AI Lab, NetEase, [2]College of Intelligence and Computing, Tianjin University
[3]Nanyang Technological University, Singapore
[4]Noah's Ark Lab, Huawei, [5]Tianjin Key Lab of Machine Learning
{ruimingshe}@corp.netease.com, {yanzheng, jianye.hao, mengzp}@tju.edu.cn
{yingfengchen01, changjiefan}@corp.netease.com, yangliu@ntu.edu.sg

Abstract

Generating diverse behaviors for game artificial intelligence (Game AI) has been long recognized as a challenging task in the game industry. Designing a Game AI with a satisfying behavioral characteristic (style) heavily depends on the domain knowledge and is hard to achieve manually. Deep reinforcement learning sheds light on advancing the automatic Game AI design. However, most of them focus on creating a superhuman Game AI, ignoring the importance of behavioral diversity in games. To bridge the gap, we introduce a new framework, named EMOGI, which can automatically generate desirable styles with almost no domain knowledge. More importantly, EMOGI succeeds in creating a range of diverse styles, providing behavior-diverse Game AIs. Evaluations on the Atari game, a commercial game indicate that, compared to existing algorithms, EMOGI performs better in generating diverse behaviors and significantly improves the efficiency of Game AI design.

1 Introduction

Gaming is at the heart of the entertainment business, and the game market is rapidly growing along with fierce competition. According to the latest *Global Games Market Report* [17], there are over 2.5 billion active gamers across the world, and over $164 billion will be spent on games in 2020. With such a vast and competitive market, the game quality like the entertainment and attraction becomes especially important, as they greatly determinate the success of a game. For instance, a Game AI with monotonous behaviors in a battle game is tedious and will sharply degrade the player's enthusiasm, resulting in losing users or even the failure of a game. To keep the entertainment, game companies put many efforts (e.g., continuously generating new behaviors), but often achieve limited progress [1]. Consequently, a more effective approach to create behavior-diverse Game AIs is of great importance and meaning for game companies.

One natural way to create Game AIs is to model the human players' behavior [7; 19; 5; 24]. However, these data-driven methods require sufficient data in advance and are prone to led the AI's behavior biased to these training data. Besides, imitating from the pre-collected human behaviors may implicitly restrict the behavioral diversity and sacrifices the possibility of discovering more styles.

Another piratical way to create Game AIs with controllable behavior is the behavior tree (BT) [13], which is extensively adopted in designing Game AIs, including *Red Dead Redemption* [20], *Bloshock* [8] and *Halo* [9]. However, BT is a rule-based method, requiring abundant designer expert knowledge and labor cost in designing rules. Besides, contrived rules may be contradictory, leading to potential bugs. Lastly, the more rules in BT, the harder it can be maintained, restricting its effectiveness in large scale games.

Deep reinforcement learning (DRL) has achieved great success in generating competitive Game AIs (referred to as the policies in DRL) for various kinds of games [16; 21; 11; 6; 26]. However, DRL mostly focuses on winning the game, restricting its ability to generate diverse behaviors. The aforementioned Game AIs design methods suffer from the following limitations: 1) the necessity of human behavioral data; 2) heavy dependence of designer expert knowledge and substantial labor costs in searching a desirable behavior; 3) lacking the ability to generate diverse behaviors.

To address these, we propose a new framework, named Evolutionary Multi-Objective Game Intelligence (EMOGI), which combines the power of evolutionary multi-objective optimization (EMO) and DRL to generate behavior-diverse Game AIs with barely prior human knowledge. It is worth mentioning that EMOGI requires no pre-collected human behavioral data and directly generates Game AIs from scratch. To minimize the dependence of prior knowledge and labor costs in behavior searching, EMOGI leverages the power of evolutionary algorithm to achieve automatic parameter tuning of the reward function in DRL, guiding the policy learning towards the desirable behavior automatically. On the other hand, to generate diverse behaviors, the prioritized multi-objective optimization is introduced, which leverages the non-dominated sorting and crowding distance sorting to

⁎ The first two authors contributed equally.
† Corresponding author.

ensure the learned policies distributed among multiple objectives, where various kinds of behavior-diverse Game AIs can be selected for games.

For evaluation, EMOGI is adopted to design behavior-diverse Game AIs for a Atari game [15] and a commercial massively multiplayer online game. Empirical results show that, compared to existing baselines, EMOGI can achieve not only generate Game AIs with barely prior human knowledge but also behavior-diverse Games AIs effectively and efficiently.

2 Preliminaries

2.1 Markov Decision Process

Game playing is a process of successive 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 $S \times A \rightarrow \mathbb{R}$ implying 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 [23].

2.2 Asynchronous Advantage Actor-Critic

Asynchronous Advantage Actor-Critic (A3C) [14] is one of the state-of-the-art DRL algorithms, aiming at training an agent to play games. 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.

2.3 Multi-Objective Optimization

Evolutionary multi-Objective optimization (EMO) [3] is an effective method for solving optimization problems with multiple objectives. Different from the standard evolutionary algorithm (e.g., genetic algorithm) which evaluates the solution using a scalar fitness value and optimizes from a single objective perspective, EMO uses vectors and achieve optimization from a multi-objective perspective. In such a way, offspring evolved from the EMO can simultaneously achieve high performance regarding multiple objectives and better diversity among multiple objectives [25].

3 Problem Formulation

Generating behavior-diverse Game AIs is critical for guaranteeing a game's entertainment and popularity. This section formulates this problem by three parts: 1) necessary notions; 2) generating a policy with desirable behaviors and 3) the challenges in generating diverse behaviors.

Notions Game AI refers to an intelligent agent who can play games using different policies π, resulting in different behaviors. From the design perspective, behaviors with strong characteristics are more attractive (e.g., an aggressive Game AI in combat games). To measures aggressiveness, a natural way is to count the opposite duration spent in games, as an aggressive Game AI tends to finish the game quickly. Similarly, the average distance between agents is a feasible measurement for defensiveness since a defensive Game AI will keep a large distance from the opponent to avoid damage.

Formally, the opposite duration and average distance are referred to as the game business indicators (denoted by I_{duration}[2] and I_{distance}). Given a policy π, the aggressiveness and defensiveness can be measured by:

$$\mathfrak{S}_{\text{agg}}(\pi) = I_{\text{duration}}, \mathfrak{S}_{\text{def}}(\pi) = I_{\text{distance}}, \tag{2}$$

where $\mathfrak{S}_{\text{agg}}(\pi)$ and $\mathfrak{S}_{\text{def}}(\pi)$ denote the extent of a policy exhibiting an aggressive/defensive behavior, respectively. It is worth mentioning that $\mathfrak{S}(\pi)$ is used for measuring the policy after the policy is learned, not guiding policy learning.

Behavior Generation Leveraging DRL to generate a policy with desirable behavior characteristics is non-trivial, requiring tremendous domain knowledge and labor-costs. One reason is that the behavior of a DRL-learned policy π is determined by the reward function $R(s, a)$, which mostly focuses on winning the game. To generate desirable behaviors, one effective method is the reward shaping, adding behavior-related reward items in reward function $R(s, a)$ to affect a policy's behavior [18; 22]. Specifically, to encourage aggressive/defensive behaviors, the reward function can be shaped as follows:

$$R_w(s, a) = (w_0 * r_{\text{win}} + w_1 * r_{\text{damage}} + w_2 * r_{\text{injury}}), \tag{3}$$

where r_{damage} and r_{injury} encourage damaging the opponent and avoiding getting hurts, respectively. Different weights $w = \{w_0, w_1, w_2\}$ lead to different behaviors. Increasing w_1 and w_2 tends to create aggressive and defensive behaviors, respectively. Intuitively, any desirable behavior can be learned by DRL with appropriate weights.

It should be emphasized that the reward shaping item r_i and indicator value I are different. The former constitutes the reward function, guiding DRL towards a desirable behavior, and the latter measures the degree of a policy having a certain behavior characteristic. Besides, indicators like I_{duration} and I_{distance} can only be measured after playing, making them unusable in the reward shaping.

[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.

[2] $I_{\text{duration}} = 1 - t$, where t is the normalized time spent in games.

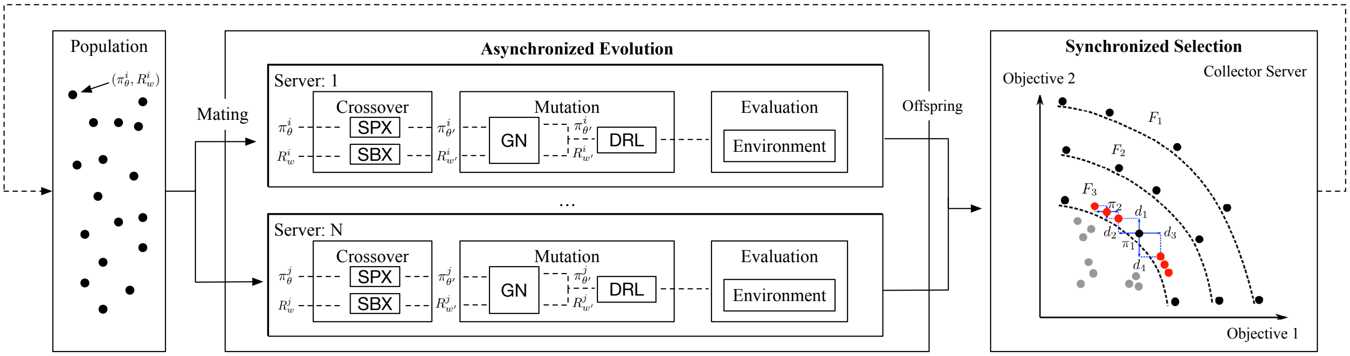


Figure 1: Overall framework of EMOGI

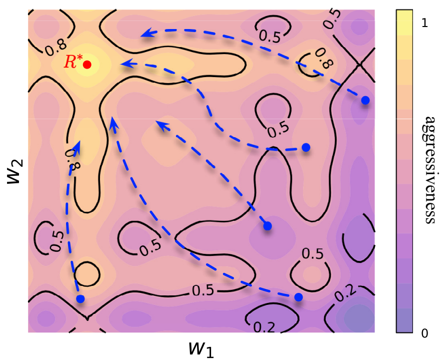


Figure 2: The aggressiveness $\mathfrak{S}_{\text{agg}}(\pi)$ of policy π learned by different weights . The x- and y-axis are weights in reward function while color depicting the extent of a policy being a aggressive style.

Challenges Manually tuning weights requires abundant domain-knowledge. Even a slight change in the weight combination may result in unpredictable behaviors. Fig. 2 is a schematic diagram briefly describing the relationship between weights and an aggressive behavior, 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 create target behavior, which, however, are hard to find by manual parameter tuning.

4 EMOGI

We propose a new framework, named EMOGI, to generate behavior-diverse Game AIs with barely prior human knowledge. Overall, EMOGI is built on an evolutionary framework like population-based training (PBT) [10; 12]. However, in addition to the policy, EMOGI treats the reward function as a part of the candidate (denoted by (π_θ^i, R_w^i) in Fig. 1). By evolving the w, EMOGI achieve automatic weights tuning, guiding DRL learning towards the desirable behavior. Another innovative difference is that EMOGI leverages multi-objective optimization to select policies with different behaviors, guaranteeing the diversity of the populations.

Algorithm 1: EMOGI

1 **Input :** n: the size of the population
2 **Output:** P: the population of candidates.
3 $P = \{\{\pi_\theta^1, R_w^1\}, ..., \{\pi_\theta^n, R_w^n\}\}$ ▷ Initialize randomly
4 $Q = \{\}$ ▷ Initialize Q with a synchronized set
5 **repeat**
6 // Asynchronized Evolution
7 $p_1, p_2, ..., p_u = mating(P)$
8 $q_1, q_2, ..., q_v = crossover(p_1, p_2, ..., p_u)$
9 **for** $q \in \{q_1, q_2, ..., q_v\}$ **do**
10 $q = evaluate(\textit{\textbf{DRL-Train}}(mutate(q)))$
11 $Q = Q + \{q_1, q_2, ..., q_v\}$
12 // Synchronized Selection
13 **if** $|Q| \geq n$ **then**
14 P = ***Diverse-Select***$(P \cup Q, n)$ ▷ (see Alg.2)
15 $Q = \{\}$
16 **until** *the stop criteria is satisfied*;
17 **return** P;

4.1 Generating Single Behavior

EMOGI aims at generating desirable behaviors by automatic weights tuning without human intervention. Specifically, as shown in Alg. 1, EMOGI initializes a population of candidates, each of which is a pair of π_θ (a neural network parameterized by θ) and $R_w(s,a)$ (parameterized by w). To create new offspring, EMOGI performs single-point crossover (SPX) [2] on network parameters with a minimal performance loss, and simulated binary crossover (SBX) [4] on reward weights (Line 8-10). Then Gaussian noise (GN) is adopted to mutate both parameters. As shown in Fig.1, SPX and GN are adopted to crossover and mutate the network parameters θ, leading to new θ'. As for reward weights w, EMOGI uses the SBX and GN to achieve crossover and mutation, resulting in new w'. Once finished, the newborn policy $\pi_{\theta'}$ is trained using a DRL algorithm, guided by the newly tuned reward function $R_{w'}$ towards new behaviors (line 10). After training, each candidate is evaluated to obtain their performance regarding a given game business indicator. Assume an aggressive behavior is required and duration indicator I_{duration} is adopted to measure a policy's aggressiveness (i.e., $\mathfrak{S}_{\text{agg}}(\pi) = I_{\text{duration}}$). Individuals with higher I_{duration} will be kept for further evolution, while the rests are eliminated.

This evolutionary cycle repeats until reaching a certain number of iterations. In this way, policies in the last population will have high $\mathfrak{S}_{\text{agg}}(\pi) = I_{\text{duration}}$, achieving a significant aggressive behavior without any manual parameters tuning.

To boost efficiency, EMOGI divides the population and distributes them to multiple servers to achieve asynchronized evolution including: crossover, mutation, DRL training and evaluation (asynchronized part in Alg. 1). Each evolved candidates will be sent to the collector server and added into a synchronized set Q. Once the size of Q reached a threshold n, PMOO is adopted to select better offspring as the next new population (line 14 in Alg. 1).

4.2 Generating Diverse Behaviors

Evolving the whole population towards one behavior style will restrict the population's behavioral diversity. To address this, EMOGI proposes the prioritized multi-objective optimization (PMOO), where different behavior styles are regarded as different optimization objectives. PMOO optimizes policies towards multiple style, requiring policies to be evaluated from multiple aspects and resulting in two differences: 1) how to measure a policy and 2) how to select better offspring.

PMOO uses multiple indicators to measure a policy's performance regarding different behavioral characteristics. Take a combat game for instance, given a policy π, the duration and distance indicators $I_{\text{duration}}, I_{\text{distance}}$ are adopted to respectively measure the aggressiveness and defensiveness of π (denoted by $\mathfrak{S}_{\text{agg}}(\pi), \mathfrak{S}_{\text{agg}}(\pi)$). Thus, policy π can be measured regarding two kinds of behavioral styles as follow:

$$[\mathfrak{S}_{\text{agg}}(\pi), \mathfrak{S}_{\text{def}}(\pi)] \equiv [I_{\text{duration}}, I_{\text{distance}}] \tag{4}$$

where $\mathfrak{S}_{\text{agg}}(\pi)$ and $\mathfrak{S}_{\text{def}}(\pi)$ are regarded as two optimization objectives, spanning a two-dimensional optimization space, where each policy is located (right part in Fig. 1).

For offspring selection, PMOO proposes the domination relation to achieve vector-based comparison between policies. Specifically, we say policy π_0 dominates π_1 (denoted as $\pi_0 \succ \pi_1$) if and only if $(\mathfrak{S}_{\text{agg}}(\pi_0) > \mathfrak{S}_{\text{agg}}(\pi_1) \wedge \mathfrak{S}_{\text{def}}(\pi_0) \geq \mathfrak{S}_{\text{def}}(\pi_1))$ or $(\mathfrak{S}_{\text{agg}}(\pi_0) \geq \mathfrak{S}_{\text{agg}}(\pi_1) \wedge \mathfrak{S}_{\text{def}}(\pi_0) > \mathfrak{S}_{\text{def}}(\pi_1))$. Intuitively, $\pi_0 \succ \pi_1$ means π_0 is better than π_1 regarding at least one optimization objective, while the rest no worse.

The set of all "best" candidates constitutes a Pareto-optimal frontier (denoted by F_i in Fig. 1). Each policy in F_i cannot dominate each other (e.g., π_0 is more aggressive while π_1 is more defensive). The visualization of offspring selection is depicted in Fig. 1 (right part), and the pseudo-code is outlined in Alg. 2. Given a population P, the Pareto-optimal frontier is identified using the non-dominated sorting [3], and added into the new population, then removed from the original population (e.g., $F_1, F_2, F_3, ...$ in Fig. 1). This process repeats until the size of the new population reaches a predefined threshold (line 3-7 in Alg. 2). Note that, F_1 generally exceeds F_2 because $\forall \pi \in F_1.(\nexists \pi' \in F_2, \pi' \succ \pi)$. 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 can be selected in to the P' (Line 8-12). An intuitive example is given in Fig. 1, where only part of the F_3

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    $F = \textbf{ND\_Sort}(P)$ ▷ select the Pareto frontier from P
4   **if** $|P'| + |F| \leq n$ **then**
5      $P' \leftarrow P' \cup F$
6      $P = P \setminus F$
7   **else**
8      $F' \leftarrow \text{CD\_Select}(F, n - |P'|)$ ▷ squash set.
9      $P' \leftarrow P' \cup F'$
10   **break**
11 **return** P'

can be selected. To achieve this, PMOO measures the density of each candidate in F_3 and selects sparse candidates to construct a more diverse offspring (line 8). Intuitively, the density of π_1 is computed based on the distance between two nearest surrounding neighbors regarding in terms of two objectives (i.e., $d1 + d2 + d3 + d4$). Candidates with sparser density will be selected as the new offspring.

In this way, EMOGI can generate not only desirable behaviors by evolving policy towards two objectives (aggressive/defensive styles), but also behaviors (e.g., a neutral style) evenly distributed among two objectives. This guarantees the behavioral diversity in the population, where designers can choose whichever in need for games.

4.3 Generating Complex Behaviors

A single indicator is insufficient to create complex behaviors like "hit-and-run", which exhibits a defensive behavior but need to win eventually. Maximizing I_{distance} will result in a behavior that is always avoiding without attacking.

To create such complex behaviors, EMOGI proposes to evaluate the extent of a policy exhibiting complex behaviors as follows:

$$\mathfrak{S}_{\text{agg}}(\pi) = [I_{\text{win-rate}}, I_{\text{duration}}], \mathfrak{S}_{\text{def}}(\pi) = [I_{\text{win-rate}}, I_{\text{distance}}] \tag{5}$$

where the win rate indicator $I_{\text{win-rate}}$ is introduced and $I_{\text{win-rate}}$ measures the win rate of a given policy. To compare policies regarding a complex behavior (e.g., "hit-and-run"), we propose a prioritized element-wise comparison as a complement to the domination relation. For instance, π_0 is better than π_1 in achieving a "hit-and-run" behavior only when: 1) it has a larger $I_{\text{win-rate}}$ regardless of I_{distance}; or 2) the same $I_{\text{win-rate}}$ but larger I_{distance}. This comparison guarantees that indicators in the front will be firstly considered than the latter. In this way, PMOO will select offspring with higher value in both $[I_{\text{win-rate}}, I_{\text{distance}}]$, whereby the "hit-and-run" behavior is achievable.

..

5 Experiments

This section presents empirical results on an Atari game *pong* and a commercial game *Justice Online* (JO). Comparisons be-

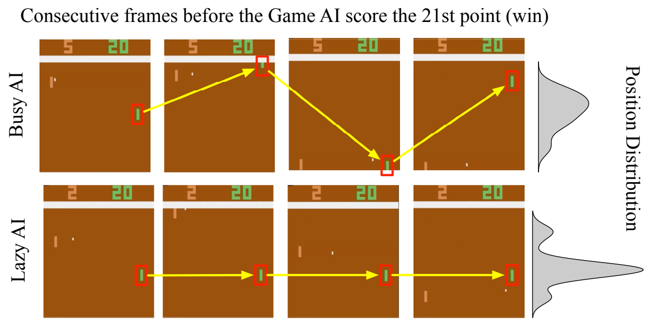


Figure 3: Visualization of different behaviors generated by EMOGI. The paddle position (in the vertical direction) are counted and plotted, where busy AI has a more dense distribution than lazy AI.

tween EMOGI and A3C are conducted to verify their effectiveness in generating behavior-diverse Game AIs. Besides, we visualize all generated behaviors for further comparison[3]. It is worth mentioning that, due to parameters tuning heavily dependents on domain knowledge, to ensure the fairness, experienced front-line designers are invited to tune the weights in reward function for A3C, and all baselines use the same hyper-parameters defined in [14].

5.1 Atari Game

Atari pong is a widely used benchmark where a agent controls the green paddle to win the game. We investigate the effectiveness of EMOGI in generating competitive Game AIs with diverse behaviors (e.g., busy/lazy styles). To be fair, all methods uses the same reward function as follows:

$$R = w_1 * r_{\text{win}} + w_2 * r_{\text{paddle-move}} + w_3 * r_{\text{act}}, \tag{6}$$

where $r_{\text{win}}, r_{\text{paddle-move}}$ and r_{act} encourage the agent to win, move positions and take fewer actions, respectively. Specifically, the $r_{\text{win}} = 1$ if the agent scores otherwise 0. The $r_{\text{paddle-move}} = 1$ if the the paddle's position changes otherwise 0 (the red rectangle). The $r_{\text{act}} = -1$ if any actions is taken otherwise 0. In A3C, weights are manually tuned by designers to create Game AIs, however, EMOGI leverages automatic tuning. The busy/lazy styles are treated as two optimizing objectives by EMOGI, which can be measured as follows:

$$\mathfrak{S}_{\text{buzy}}(\pi) = [I_{\text{win-rate}}, I_{\text{move-rate}}], \mathfrak{S}_{\text{lazy}}(\pi) = [I_{\text{win-rate}}, I_{\text{no-act-rate}}], \tag{7}$$

where $I_{\text{win-rate}}$ is the win rate. $I_{\text{move-rate}}$ calculates the percentage of frames when the paddle moves, and $I_{\text{no-act-rate}}$ counts the percentage of frames when the agent takes no actions.

Generating Diverse Behaviors Fig. 3 visualizes the different behaviors generated by EMOGI, where one can find that two styles vary greatly. Both AIs can win the game, but the busy one moves more frequently than the laze one[4]. Another evidence is that busy AI has a dense position distribution than laze AI. These complex behaviors are achieved by EMOGI with automatic parameter tuning without human intervention.

[3]More details in https://sites.google.com/view/ijcai20-emogi

[4]Videos of busy/lazy styles: https://youtu.be/1uEWhIxmGVc

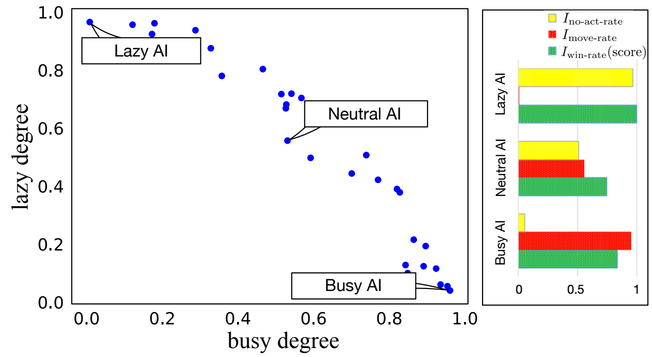


Figure 4: Distribution of styles generated by EMOGI among two objectives (left), and quantitative analysis of three styles in term of three related indicators (right). Values in all axis are normalized.

Table 1: Averaged evaluation results (10 runs) regarding related indicators of generated Game AIs in Atari game.

Indicator values		$I_{\text{win-rate}}$(score)	$I_{\text{move-rate}}$	$I_{\text{no-act-rate}}$
EMOGI	Busy AI	**15.2**	**0.954 (best)**	0.055
	Neutral 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
	Neutral AI	-5.0	0.304	0.857
	Lazy AI	11.8	0.257	0.853

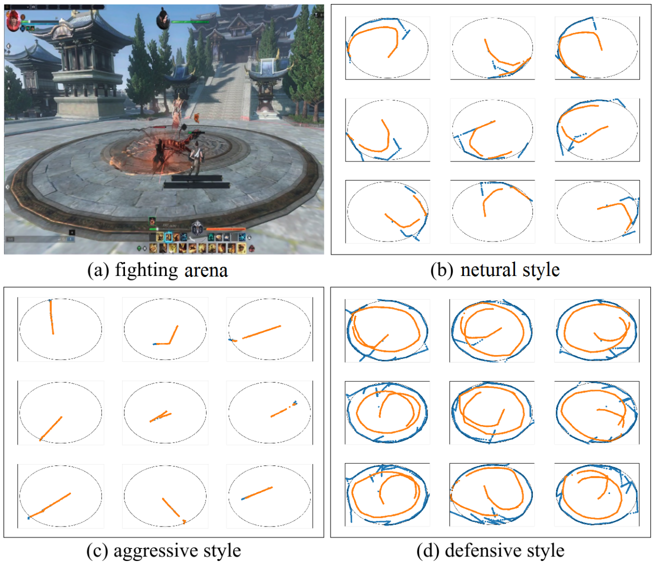
Quantitative Comparisons To investigate the effectiveness of EMOGI in generating diverse behaviors, Game AIs' behaviors are evaluated quantitatively regarding related game business indicators. Fig. 4 (left) visualizes the distribution of all Game AIs created by EMOGI among two optimization objectives (i.e., busy/lazy styles). EMOGI can generate not only extreme styles (lazy- and busy styles) along with two objectives but also more neutral AIs evenly distributed among them. Fig. 4 (right) demonstrates that all Game AIs can win the game (green bar) but exhibits different behaviors.

Evaluation results of Game AIs generated by EMOGI and A3C are summarized in Tab. 1. To be fair, both A3C and EMOGI are trained using the same number of runs (around 1 million episodes). One can find that A3C has limitations in generating desirable behaviors via manual parameter tuning. For instance, the busy AI high value in $I_{\text{move-rate}}$ but fails to win. Besides, the neutral AI, generated by A3C, not only lose the game but also fails to perform neutrally.

By contrast, the busy and lazy AIs created by EMOGI 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 EMOGI in learning complex style automatically. Another advantage is that the neutral AI, generated by EMOGI, achieves 0.557 busy degree and 0.531 lazy degree, making itself a suitable neutral Game AI, which, however, is unlearnable by A3C.

5.2 Justice Online

In these experiments, a real-world commercial game (JO) is adopted for evaluation. To make our study feasible, we only selected one combat scenario, where an easy-to-kill wizard needs to uses different skills to beat a hard-to-kill fighter



(a) fighting arena (b) netural style (c) aggressive style (d) defensive style

Figure 5: (a) is a round fighting arena in JO 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 (shown in Fig. 5). EMOGI and A3C are adopted to create different behaviors for the wizard (e.g., aggressive/defensive styles) by tuning the reward functions as follow:

$$R = w_0 * r_{\text{win}} + w_1 * r_{\text{damage}} + w_2 * r_{\text{injury}} \tag{8}$$

where $r_{\text{win}}, r_{\text{damage}}$ and r_{injury} encourage to win, attack and avoid the opponent, respectively. The r_{win} will be a large positive scalar if the agent wins otherwise 0. The $r_{\text{damage}}, r_{\text{injury}}$ are the damages to the enemy and the agent received, respectively. The agg/def styles are treated as two optimizing objectives by EMOGI, which can be measured as follows:

$$\mathfrak{S}_{\text{agg}}(\pi) = [I_{\text{win-rate}}, I_{\text{duration}}], \mathfrak{S}_{\text{def}}(\pi) = [I_{\text{win-rate}}, I_{\text{distance}}], \tag{9}$$

where $I_{\text{win-rate}}, I_{\text{duration}}$ and I_{distance} measure the win rate, opposite duration spent in game and averaged distance between agents, respectively. Note that, high I_{duration} means finish game quickly.

Generating Diverse Behaviors Fig. 5 (c,d) visualizes the behavior of the aggressive/defensive AIs (denoted by Agg/Def AI) generated by EMOGI, where one can find that two styles vary greatly. The Agg AI always attack, barely moving, producing shorter trajectories. By contrast, Def AI creates longer paths 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 EMOGI in generating complex and diverse behaviors[5].

Quantitative Comparisons Fig. 6 (left) visualizes the distribution of all Game AIs created by EMOGI among two optimization objectives (i.e., agg/def styles). A similar finding can be found that EMOGI can generate diverse and desirable behaviors efficiently and automatically. As shown in Fig. 6 (right), all generated Game AIs can defeat the opponent but in a different way. Def AI defeats the opponent by keeping a safe distance (in Fig. 5(d)). However, Agg AI exhibits aggressive behavior, like constant attack without moving (in Fig. 5(c)). Tab. 2 shows that EMOGI can generate better behaviors than A3C from a numerical perspective (e.g., Agg/Def AI with bold values). Besides, A3C fails to create an Agg AI to win, which, however, can be generated by EMOGI. It is worth mentioning that both A3C and EMOGI are trained using the same number of runs (around 27 million episodes). But A3C still fails to find a desirable behavior (e.g., Agg AI and Netural AI).

[5]Videos of diverse behaviors: https://youtu.be/5eps0YLY-lM

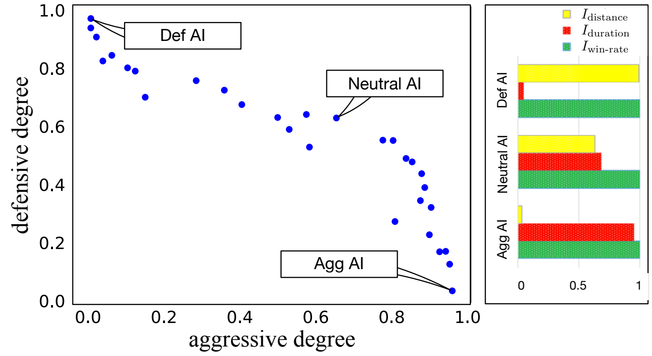


Figure 6: Distribution of styles learned by EMOGI among two objectives (left), and quantitative analysis of three styles in term of three related indicators (right).

Table 2: Averaged evaluation results (30 runs) regarding related indicators of generated Game AIs in JO game.

Indicator values		$I_{\text{win-rate}}$	I_{duration}	I_{distance}
EMOGI	Agg AI	**1.000**	**0.955 (best)**	0.032
	Neutral 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
	Neutral AI	0.760	0.188	0.916
	Def AI	0.633	0.164	0.938

6 Conclusion

This paper proposes the EMOGI to generate behavior-diverse Game AIs with barely domain knowledge. EMOGI leverages the evolutionary algorithm to create desirable behaviors without human intervention. Besides, PMOO is adopted to create behavior-diverse Game AIs efficiently. Last, EMOGI can be deployed to distributed servers to boost the efficiency of behavior generating. The empirical results demonstrate the effectiveness and advantage of EMOGI algorithm in creating diverse and complex behaviors.

Acknowledgment

The work is supported by the National key R&D program of China (Grant Nos.: 2018YFB1701700), National Natural Science Foundation of China (Grant Nos.: 61702362, U1836214), new Generation of Artificial Intelligence Science and Technology Major Project of Tianjin (Grant Nos.: 19ZXZNGX00010), Singapore National Research Foundation, under its National Cybersecurity R&D Program (Grant

Nos.: NRF2018NCR-NCR005-0001), National Satellite of Excellence in Trustworthy Software System (Grant Nos.: NRF2018NCR-NSOE003-0001) and NRF Investigatorship (Grant No.: NRF-NRFI06-2020-0001). We thank our industrial research partner Netease, Inc., especially the Fuxi AI Lab for their discussion and support with the experiments.

References

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

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

[3] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation - NSGA-II. *PPSN*, 2000.

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

[5] Anders Drachen, Alessandro Canossa, and Georgios N Yannakakis. Player modeling using self-organization in tomb raider: Underworld. In *2009 IEEE symposium on computational intelligence and games*, pages 1–8. IEEE, 2009.

[6] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual Multi-Agent Policy Gradients. *AAAI*, 2018.

[7] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Personas versus clones for player decision modeling. In *International Conference on Entertainment Computing*, pages 159–166. Springer, 2014.

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

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

[10] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population Based Training of Neural Networks. *CoRR*, 2017.

[11] Guillaume Lample and Devendra Singh Chaplot. Playing FPS Games with Deep Reinforcement Learning. *AAAI*, 2017.

[12] Ang Li, Ola Spyra, Sagi Perel, Valentin Dalibard, Max Jaderberg, Chenjie Gu, David Budden, Tim Harley, and Pramod Gupta. A Generalized Framework for Population Based Training. *CoRR*, 2019.

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

[14] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *ICML*, 2016.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A Riedmiller. Playing Atari with Deep Reinforcement Learning. *CoRR*, 2013.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

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

[18] Inseok Oh, Seungeun Rho, Sangbin Moon, Seongho Son, Hyoil Lee, and Jinyun Chung. Creating Pro-Level AI for a Real-Time Fighting Game Using Deep Reinforcement Learning. *arXiv.org*, April 2019.

[19] Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N Yannakakis. Imitating human playing styles in super mario bros. *Entertainment Computing*, 4(2):93–104, 2013.

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

[21] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[22] Halit Bener Suay, Tim Brys, Matthew E Taylor, and Sonia Chernova. Learning from Demonstration for Shaping through Inverse Reinforcement Learning. *AAMAS*, 2016.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[24] Julian Togelius, Renzo De Nardi, and Simon M Lucas. Towards automatic personalised content creation for racing games. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 252–259. IEEE, 2007.

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

[26] Zheng, Yan, Fan, Changjie, Xie, Xiaofei, Su, Ting, Lei Ma, Hao, Jianye, Meng, Zhaopeng, Yang Liu, Shen,

Ruimin, and Chen, Yingfeng. Wuji - Automatic Online Combat Game Testing Using Evolutionary Deep Reinforcement Learning. *ASE*, pages 772–784, 2019.