Learning Human-like Behaviors using NeuroEvolution with Statistical Penalties

Luong Huu Phuc
School of Information Science
JAIST
Ishikawa, Japan
Email: luongphuc@jaist.ac.jp

Kanazawa Naoto
School of Information Science
JAIST
Ishikawa, Japan
Email: na-kanazawa@jaist.ac.jp

Ikeda Kokolo
School of Information Science
JAIST
Ishikawa, Japan
Email: kokolo@jaist.ac.jp

***Abstract*—In game artificial intelligence (AI), two common directions for developing non-human computer players are strong AI and human-like AI. Human-like AI aims at making computer agents behave like humans. In this direction, NeuroEvolution (NE), which is a combination of an artificial neural network (ANN) and an evolutionary algorithm (EA), had been frequently used to a make computer agent to behave like a human. Our research introduces a novel approach to create human-like computer agents in a platform game *Super Mario Bros.* (SMB) - we called it a 2D action game in this research. The approach utilizes statistical penalties to evaluate candidates created by NE algorithm. The penalties help in reducing mechanical actions of computer agents based on human data statistics, and the effects of statistical penalties are analyzed by asking human subjects to rate the human-likeness of agents. Experiments show that our method improves the human-likeness in the behavior of a computer agent.**

I. Introduction

Recently, many things in modern life are inspired by nature, and researchers tend to create every machine with artificial intelligence based on the intelligence of living things. In the viewpoint of Game AI, computer players are usually made for entertaining, but the important thing is games represent real world problems. Since games are simple, well-defined and easy to evaluate, many AI methods such as supervised/reinforcement learning, optimization, tree search, and reasoning are proposed to solve problems in this domain. Creating game AI means developing an AI agent for a certain problem, such as solving games, interacting with human players, automatically generating suitable game levels, or even teaching human players to play games.

A current attractive trend in this field is making intelligent machines which are able to beat human champions. Developing a strong AI, let say an AI champion - which is able to beat human experts, started to make impacts on society since 1997. The first machine, called Deep Blue [1], won against a human champion in a chess game. Furthermore, contemporary research on the deep neural network, AlphaGo [2], is the first computer player to defeat a human champion in a very complex game of Go.

Another interesting direction is developing human-like computer agents. The purpose of making a human-like agent differs from that of a strong AI agent. Strong AI agents' behaviors, in case of action games, tend to be somehow impossible to human players such as very precise movements or immediately responses. In contrast, human-like AI agents should react more naturally and reasonably. A reason for the necessity of human-like AI agents is sometimes human players may think of whom they are playing with. They prefer to play with humans rather than computer players. Another reason which is human-like computer players might be used to simulate how human players play through a game, so more suitable levels and stages can be designed. In case of teaching human players to play a game, a human-like AI is also required because human beginner players may not be able to follow the optimized movements of a strong AI player. Comparing to human players, the behavior of AI agents is very unnatural. Thus, the questions are *"Is it possible to make a computer learn to play like a human?"* and *"How to make computer players be more like human players?"*

This research mainly focuses on developing a human-like game AI agent. We attempt to use evolutionary algorithm with artificial neural network structure to create a computer player. Obviously, rule-based agent is a possible solution to implement human-like agents. However, defining rules for a computer agent to behave like a human in a specific game is very costly and arduous, to say nothing of its generality. Therefore, a flexible and creative approach is needed to address this problem. Because of the variety in types of computer games, only 1-player action game type is taken into account. The target of this research is to enable an AI computer agent to behave like a human player by using penalties to punish mechanical, non-human behaviors as well as unnatural behaviors.

To evaluate the human-likeness of a computer agent, a special procedure and human subjects are required. In the past, judging the intelligence of a machine was done by Turing Test, proposed by Alan Turing [3] in 1950. After being introduced, many researchers used extended versions of the Turing Test to evaluate the human-likeness of their AI agents. We employ a test, which allows human subjects to ranking abilities of a player from 1 (worst) to 5 (best), rather than the Turing Test to evaluate the proposed computer agent.

A testbed, which is used in this research, is Mario AI Benchmark 0.1.9 [4]. This testbed had been already used in competitions in 2009 and 2010, and it is a modified version of SMB by Nintendo, one of the most famous two-dimension

action games in the world. Strong AI agent in this game is already achieved by using A* search algorithm [5], and the most human-like AI is developed using NE algorithm [6]. However, the ideal human-likeness level of AI agents is not yet reached.

II. Related Work

Developing human-like computer agents in games is adopted as a field in game AI research since competitions in game AI become popular in the past ten years. Many competitions were held, and methods similar to Turing Test were promising for the human-likeness qualification of computer players in games. An extended version of the Turing Test is used in the competition 2K BotPrize Contest, and its results are shown afterward [7]. In 2012, another Turing Test competition was held in Mario AI Championship [8]. While 2K BotPrize Contest uses first-person shooter game as the domain, Mario AI Championship [9] focuses on a two-dimension action game. In the test, human subjects are asked to watch many videos of the game plays by human players and computer players. Then, they have to vote whether the player was human or machine.

A. UTˆ2 computer agent

The BotPrize competition employed the computer game Unreal Tournament 2004 (UT2004 - a first person shooter game) as a testbed. In this game, players are able to move around a three-dimension virtual world in a first person's viewpoint. The competition conducted the Turing Test to test the ability to imitate human players of computer players. The result of the competition shown that the UTˆ2 bot ranked the 2^{nd} place among computer players with humanness rating of 27.3%. The humanness equals the number of human judgments on total judgments in percentage. Comparing to human players, even the least human-like player achieved 35.5% of human-likeness, so the human-likeness of the UTˆ2 bot is still not at the level of human players.

The UTˆ2 computer player learns combat behavior by NE, and it uses human trace database. Additionally, an imitation technique is also used to observe the way human do rather than only combat. Since UTˆ2 bot is trained by NE with multiobjective to learn combat behavior, a set of three objectives is defined. The first objective is to maximize the dealt damage. The others are to minimize both the received damage and the number of collision events with level geometry. Detailed architecture and learning process, as well as multiobjective optimization, are shown in [10]. Directly training a computer player to play like a human player usually requires data collected from human players, and this computer agent is trained by this way. However, using human trace data may not be the best way due to the accuracy and validity of the collected human data. More data means longer training time, and lack of data results in the inefficient training or overfitting. If there is no human data for training the computer player, it might be difficult to make the computer player be more human-like, to say nothing of a huge number of human data requires effort to collect.

B. Human-like Mario Computer Agent

The winner of the Turing Test track which was held in the 2012 Mario Championship competition [8] was named VLS bot. After that, there are several approaches for producing human-like behaviors introduced and compared [11]. In this section, some notable approaches are briefly introduced.

1) VLS Bot: The VLS bot has its name of its contributors Vinay, Likith, and Stefan. It is inspired by a technique in the robotics field, namely Artificial Potential Field. This technique utilizes influence map in defining a number of things in potential fields. The author described four types of the field which are: (1) The field of progression, (2) the field of rewards, (3) the field of opponents, and (4) the field of terrain. The first field, progression field, makes Mario move right rather than left. The field of rewards make coins, mushrooms, blocks, and flowers attractive so that Mario will change his attention to these rewards. Next, the field of opponents provides dangerous positions for Mario to keep a distance from enemies or to kill them. Finally, terrain field ensures Mario to avoid gaps, dead ends and search for correct path ways. By using these potential fields, the computer will choose the action that will take it to the most attractive position. Moreover, tuning parameters for each field are also used, and the appropriated values are obtained by asking testers to evaluate the players.

The evaluation of human-likeness in the competition is calculated based on the number of human votes from human subjects. The result of the competition showed that VLS bot acquired 25.79% of score in the Turing Test track, while expert human players score 23.21%, and 30.95% is the score of novice human players. An explanation for the human-likeness of the VLS bot is that VLS bot spends time moving left, collecting most items, and interactive with enemies other than just running in the right direction. Unfortunately, defining such potential fields might lead to the same problem in rule-based approach that is the complexity of mixed and overlapping potential fields.

2) NeuroEvolution Bot: Artificial neural network (ANN) is commonly used in AI with supervised, unsupervised, or reinforcement learning method. However, the ANN can also be trained by using Evolution Algorithm (EA), so the definition of Neuro-Evolution is came from this combination. Similar to Direct Policy Search, which is used in weightlifting robot [12] and later on as a framework for robot model [13], NE defines an ANN as a solution in the population.

In SMB game, an existed computer agent using NE is basically trained by adjusting weights of the ANN [11]. The objective for training this agent is to make the AI agent to produce similar trajectories of Mario to human player's trajectories.

The results, in term of human-likeness, of this NE [11] showed that NE had 89 times of human judgments in total. Supervised learning method, which uses ANN Back Propagation, achieved 21 times. Comparing to human players, human players received 110 times of human judgments. Thus, NE is better than ANN with supervised learning, but it is still not able to achieve the human-likeness as human players.

Moreover, the NE agent in the result uses the pre-trained ANN obtained from the supervised learning agent, so the performance of NE alone without using the trained ANN structure in imitating human playing style is questionable.

3) A Agent with Biological Constraints:* According to the result of the competition, A* is the strongest computer agent to solve the game SMB in particular. Unfortunately, the behavior of computer agent which uses A* search algorithm looks very mechanical, and the performance of A* is too efficient that it might surpass even a human expert. To limit the unnatural actions and increase the human-likeness of A* algorithm, Fujii et al. [14] introduced the intentional error and artificial delay in actions of an AI agent. Biological constraints are applied by including noise into the input information of the agent, using information of the past for the current actions, and physical fatigue (by limiting the number of pressing keys). This approach made A* algorithm to be more human-like in Mario game than human expert players, but it may be less human-like in other game domains.

In the case of 2D action games, such as SMB, there were currently three human-like approaches, which are VLS, NE, and A* with biological constraints. VLS can be implemented by using hard coding, but its inflexibility is a drawback. NE is able to mimic one trajectory of a playing, but comparing the positions of the character in the game between computer agent and human players is costly and inflexible. Also, the cost of collecting more human data raises, and human traces data is required for the performance of this agent. Meanwhile, A* provides strong AI players as expert as human players, and biological constraints contribute to its human-likeness. The disadvantage of this method is the calculation time since it requires searching for a solution every single frame.

If there is a more general method to solve the problem of human-like computer agents, it might be possible to make computer players behave like a human in every types of games. Therefore, we introduce a new approach to tackle this problem.

III. Statistical Penalties

Generally, a penalty is a punishment which is given to a computer player with a specific behavior to show that bad behaviors are not allowed. The idea of this approach is to limit or remove mechanical behaviors. By this way, a computer player is able to behave like a human due to its selected behavior have less chance to be a mechanical behavior.

Considering behaviors of players in games, we can divide the behaviors into two subsets: human-like behaviors and mechanical behaviors. Considering two subsets A and B which are the sets of behaviors in a game, we call set A a set of human-like behaviors, and set B is a set which contains only mechanical behaviors. Thus, a computer player may be more human like if its behaviors belong to set A. In order to make this computer agent, we have to keep the size of set A larger than set B's. If set A is larger, the chance for human-like behaviors to occur will increase, so a computer player, by chance, will choose to human-like behaviors rather than mechanical behaviors. Hence, the issue now is how to trigger this situation.

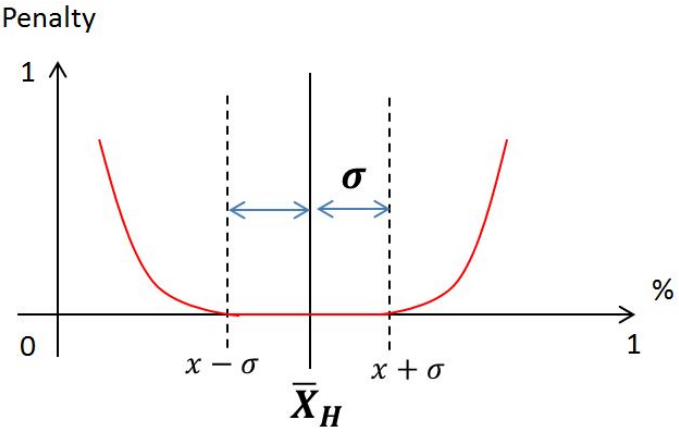

Fig. 1. A feature penalty example

We address this problem by using the concept of penalty. A penalty will be given to the computer players if it is likely to produce bad things, say less human-like behaviors and more mechanical behaviors. In other words, a computer player receives penalties in case of the number of human-like behaviors is not enough, or the number of mechanical behaviors exceeds the limitation. Consequently, thresholds must be defined for giving penalties, and a large penalty is given if the thresholds are excessively exceeded.

One essential thing in using penalties to punish bad behaviors of a computer agent is designing good thresholds. Randomly choosing values for thresholds is not the way this approach should work. We decide to use human data statistics to determine the threshold values for penalties. It means the threshold values are based on the average values obtained from human player data. The term *statistical penalties* is from this idea. The next point needs to be clarified is what kind of things should be given penalties to make a computer agent be able to produce human-like behaviors rather than mechanical behaviors.

A. Penalty feature

For computer agent in games, things such as the number of pressed buttons, the number of normal or illegal actions can be considered as features for punishment. Features can be defined differently according to the game. Hence, the penalty can be used to punish mechanical and non-human actions by introducing which features should be punished. The penalty features are defined based on human expert knowledge.

In particular, we choose the game SMB to implement the proposed idea. Instead of the original game, we employ a modified version, which had already been used in the last competition, as the environment.

1) Mario AI Benchmark 0.1.9: This is a modified version of Markus Perssons Infinite Mario Bros [9], namely Mario AI Benchmark, which is a public domain clone of the NintendoFLs classical game SMB. It is a very popular and famous

TABLE I
HUMAN DATA CHARACTERISTICS. VALUES MOST LIKELY DESCRIBE PLAYING STYLE ARE IN BOLD.

	Human			AI		
Instruction	Free	Coin	Speed	A*	NE	NE + Penalty
Stand	0.081 ± 0.057	0.192 ± 0.114	$\mathbf{0.033} \pm 0.029$	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
All-Left	0.039 ± 0.028	$\mathbf{0.136} \pm 0.036$	0.017 ± 0.014	0.008 ± 0.003	0.012 ± 0.002	0.052 ± 0.025
OnGround	0.216 ± 0.105	0.425 ± 0.129	$\mathbf{0.109} \pm 0.058$	0.041 ± 0.002	0.039 ± 0.004	0.061 ± 0.009
ChangeAction	0.333 ± 0.141	$\mathbf{0.611} \pm 0.137$	$\mathbf{0.200} \pm 0.079$	$\mathbf{0.120} \pm 0.003$	0.275 ± 0.011	0.280 ± 0.048
CollectedCoin	0.313 ± 0.123	$\mathbf{0.466} \pm 0.096$	0.272 ± 0.098	0.283 ± 0.049	0.306 ± 0.062	0.312 ± 0.054
KilledEnemy	0.338 ± 0.191	$\mathbf{0.418} \pm 0.206$	0.242 ± 0.120	0.326 ± 0.027	0.278 ± 0.021	0.278 ± 0.087
IllegalAction	$\mathbf{0.035} \pm 0.035$	0.071 ± 0.067	0.044 ± 0.041	0.000 ± 0.000	0.012 ± 0.002	0.099 ± 0.092
TimeSpent	0.429 ± 0.178	$\mathbf{0.814} \pm 0.183$	$\mathbf{0.261} \pm 0.107$	$\mathbf{0.142} \pm 0.005$	0.332 ± 0.017	0.356 ± 0.052

game, so it was made into a benchmark and used in many competitions and international academic conferences.

The gameplay implemented in this benchmark is similar to the gameplay of original Mario games, but it is comparatively simpler than the original version. The main objective in the gameplay is to navigate the Mario to reach the end of a level. Possible actions for navigating Mario are LEFT, RIGHT, UP, DOWN, JUMP, SPEED/SHOOT.

Collecting items, and killing enemies give additional scores to the player. Mario can be in one of three states: Fire, Big, and Small. Besides, there are certain types of enemies depending on the "difficulty setting".

2) Selected features: According to the game environment described in the previous section. We pick features that are necessary for accumulating the human-like performance of the Mario computer player.

To increase the human-likeness of a computer agent, features, which describe as the behaviors, such as the number of actions, the number of times that actions are changed, etc. might be included. For instance, "Left button" is almost not used in A* agent, but human players may use for moving backward to collect more coins. Selected features in this research are described as following:

- **Stand**: Since human players sometimes stop and think for a solution, the number of action, which no button is pressed, is used to describe this behavior.
- **All-Left**: LEFT action is rarely used in case computer player tries to reach the goal. For human players, they tend to move left to collect some items. Therefore, all actions where LEFT button is pressed are worth considered.
- **OnGround**: This feature is related to jumping behavior. Some computer controller use JUMP action in a particular pattern to navigate the Mario character to pass over obstacles. It may be better to have a similar amount of time to human players being on the ground or in the air.
- **ChangeAction**: We realized that some AI players change their actions extremely frequently, and some of them keep consistent action patterns. This kind of behavior seems to be impossible for human players, even with an expert.
- **CollectedCoin**: If a computer player is developed to collect many coins, it possibly collects all coins or too many coins in a level. However, human players sometimes leave some coins behind and move forward. For this reason, the number of collected coins should be compared.
- **KilledEnemy**: Similar to the case of collected coins. If human players know that they can win without interfering with enemies, they will just pass it.
- **IllegalAction**: An illegal action, for example, can be many or all buttons pressed at the same time. Useless and conflicted actions can also be considered as illegal actions.
- **TimeSpent**: Computer players tend to finish a game level quicker than human players, so they should have the same amount of time spending in the level for investigation.

To use penalties for these features, the idea is that if these features of a computer player and human players are significantly different, the punishment can be given to the computer player. As a consequence, a computer player with more human-like behavior, says it has less penalty or punishment, might be obtained by training with additional penalties beside the original score.

B. Human Data Analysis

The feature comparison for calculating penalties requires human statistical data, so we attempted to collect some human data by asking people to play the Mario game using the benchmark.

We have done an analysis on human behavior in this game to obtain statistical data for the training of our agent. Collecting human data takes time, and it is very costly. If we collect data from many human players, we need to categorize them by experience and level. Thus, we ask only five human players to participate in because it is enough to see the differences between behaviors of human players and computer players.

Five human players are given five levels in the Mario AI Benchmark with difficulty 0. Even with difficulty 1, normal human players feel it is hard to complete a level, and they

fail many levels, especially when Mario is not in Fire mode. Moreover, if more levels with the same difficulty are given to human players, they will get bored, so their behavior may be different. Additionally, three instructions are given to human players to examine if their behavior changes accordingly for each objective. These instructions are: Free to play $(Free)$, try to collect coins as many as possible $(Coin)$, and run as fast as possible $(Speed)$. Human players play each level one time, and they have to repeat for each instruction. $(Free)$ comes first, then $(Coin)$ and $(Speed)$ follow. In addition, we let a computer player implemented with A* algorithm play the same levels and compare with human players. The characteristics of human players and A* algorithm are shown in Table I.

In table I, the average percentage and standard deviation of actions and some features are calculated. The *All-Left* in the table means all action in which LEFT button is pressed such as LEFT and JUMP. Related features have their percentage value, which is the number of actions of the player in one level divided by the maximum number of actions, items, enemies, or time allowed for each level, rounded to three number after the decimal point for more precise analysis because the maximum number of actions in a game play is 3000.

According to the statistics shown in Table I, explanations for the behaviors of human players in this game are following:

- Typically, in 2D action games, a character should move right to clear a level, so the average percentage of *All-Left* action is very small comparing to other features.
- It is easily to see that the average percentage of *Stand* action, in case of *Coin* instruction is given, is higher than those in *Free* and *Speed* instructions. It is possible to think that human players tend to suspend and thinking about the solution to navigate Mario character to get some items in the level.
- Human players with *Coin* instruction pressed LEFT button the highest times comparing to the other two instructions. The reason might be collecting more coins requires Mario to go backward because some coins may be left while Mario traversing through a level.
- If human players are asked to play as fast as possible, they try to navigate the Mario to run right and jump just to pass over obstacles and enemies. It results in the average percentage of *All-left* and *TimeSpent* is small.
- Comparing *CollectedCoin* and KilledEnemy features, the statistics clearly shows that the average percentage of human players with *Free* instruction is between those in *Coin* and *Speed* instructions. This makes sure the behavior of human players in case of *Coin* instruction is to keep surround places safe for gathering items, and human players ignore items and enemies in case of *Speed* instruction.
- The A* computer players behavior is very different from human behaviors. Zero percentage for *Stand*, *IllegalAction*. There are a very small number of LEFT actions, times changing actions and very short duration while Mario character is on the ground. This AI finish a level faster than even human players with *Speed* instruction.

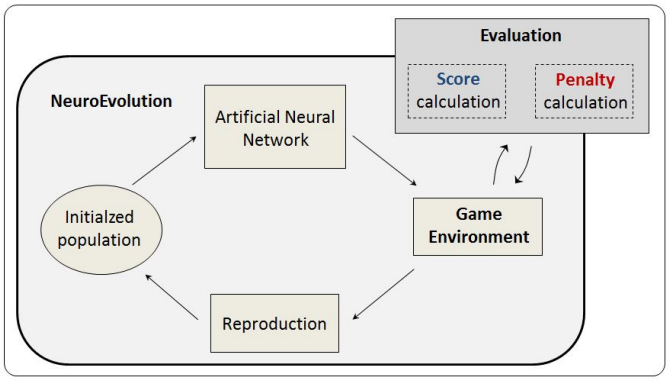

Fig. 2. Proposed method architecture

The analysis on human behavior has shown that the playing style of human players is different for each instruction. Based on these features, a computer player is expected to behave more like a human player if their average percentage of actions or features is similar or even almost the same as human players'. The statistical penalty existed for that reason, and how penalties are given is important to create human-like computer players.

C. Penalty calculation

To compute a penalty value, we take the average percentage of a feature obtained from human data statistic plus and minus its standard deviation to get the upper and lower limits. Then, if the percentage of the considered feature of a computer player falls into this range, no penalty is given, and vice versa.

Figure 1 shows an example of a feature penalty calculation. Suppose that the considered feature is the number of an action A, the average percentage of action A in human players' behavior is $\bar{X}_H$ percent, and the standard deviation is σ. Next, let a computer agent play the game, and its behavior is obtained. The percentage of action A of the computer player, x, is compared to the average percentage of human players. The penalty, which is represented by the red curve in figure 2, is calculated by using a quadratic function. If more penalty features are used, each individual penalty for each feature should be calculated.

The standard deviation is used in this case because not all human players play the same style. The variance in the behavior of human players exists, so some human players seem to have a play style similar to the average of human players than others. For that reason, it is better to use standard deviation to declare the range of value which penalty is not given.

The total penalty is calculated using equation (2). The sum of penalties is calculated as a summation of the squares differences from the mean. Since different features may be used, the weight of each penalty for each feature is needed to be balanced. If weights are not balanced, some penalties may not be effectively used to limit unnatural behavior. To

$$Penalty_{actionA} = \begin{cases} ((\bar{X}_H - \sigma) - x)^2 & if \quad x < \bar{X}_H - \sigma \\ (x - (\bar{X}_H + \sigma))^2 & if \quad x > \bar{X}_H + \sigma \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

$$TotalPenalty = \sum w_i * Penalty_i \tag{2}$$

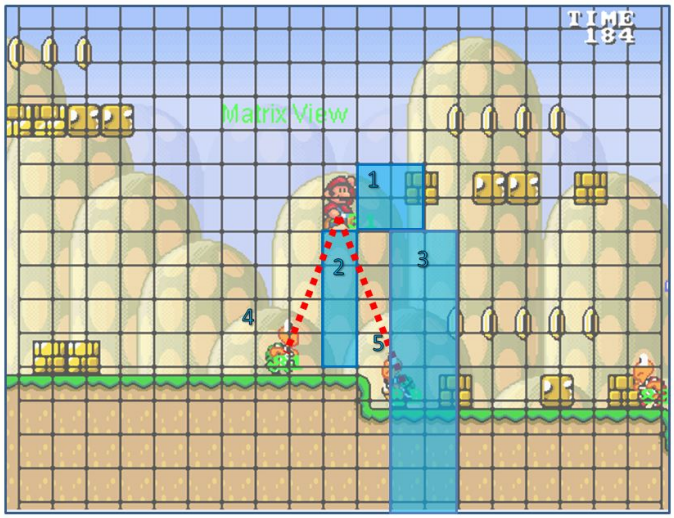

Fig. 3. Input feature setting

decide weights for penalty features, the values such as average numbers of action of human players should be approximated. Besides, if a feature is more important than others, its weight can be set to a higher value.

IV. Implementation

Previous works had shown that NE has the potential to imitate human players. Despite more a human-like agent in games such as A* with biological constraints in Mario was shown, it is only for 2D action games. The fact that A* is efficient for path finding if and only if there is enough time for calculation. More complex games might increase the calculation time of A*, so more general approach is needed. We propose an architecture in which NE and statistical penalties are integrated.

A. NeuroEvolution

NeuroEvolution is a combination of an artificial neural network (ANN) and an Evolutionary Algorithm (EA). In games, NE is used for decision making. The ANN in NE has the role of an input-output function. Its input is a game state, and output of the ANN is usually an action. Typically, the ANN in NE is usually a fully connected network, and it is trained by using EA to adjust the weights. The objective of NE is to maximize/minimize a fitness value corresponding to how much a problem is solved.

In our proposed method, we do not use a fully connected neural network because of its performance and inflexible structure. Instead, we employ an approach, which is similar to the NE of Augmenting Topologies (NEAT) proposed by Stanley and Miikkulainen in 2002 [15], to apply in our problem. NEAT uses a direct encoding to store information of an ANN, for example, nodes, connections, etc. The difference between NEAT and normal NE approach is that the ANN in the NEAT contains no connection and hidden nodes at the beginning.

B. Architecture

The architecture of our method is depicted in figure 2. It consists of three components: (1) *NeuroEvolution*, (2) *Environment*, (3) *Evaluation*. An approach similar to NEAT [15] is used to develop the first component *NeuroEvolution*. The *Environment* is the targeted game. In this research, SMB is the targeted game, and the benchmark of this game is used as a testbed. The *Evaluation* component is designed based on the *Environment*.

The process of this architecture consists steps similar to NE. At first, a set of artificial neural networks is initialized as the population. The structure of ANNs contains no hidden node and connection. Then, they will be placed into the environment one by one for evaluation. Unlike the normal evaluation step in NE, an additional calculation of penalties is taken into account because finding best solutions, which can solve the game, is not only the main target. The best solution, in term of solving games, must satisfy some requirements to additionally be the most human-like solutions. That is why penalty calculation is introduced. The fitness value for evaluating solution is computed by using equation (3). In the fitness function, *score* is the evaluation on the main target, and *penalties* is calculated based on penalties features. The calculation of total penalty is shown in the next section.

After being evaluated, the reproduction step handles the job of creating new solutions by mutating current solutions and replacing worse solutions in the population. Each individual ANN's structure is assumed as a solution, so mutation creates new solutions by altering the ANN's structure. An evolutionary strategy, called $(1 + \lambda)$-ES, is used in this step for the mutation process. More generally, λ mutants can be generated and compete with their parent, and the parent is replaced only if there is a better mutant.

C. Fitness function

To evaluate the human-like candidates, fitness function must be defined. The fitness function is calculated using the following equation:

$$Fitness = Score - TotalPenalty \quad (3)$$

By using this fitness function, candidates with a high total penalty value will be removed since they are not considered as human-like candidates. The *Score* is calculated by the game environment, and the *TotalPenalty* is calculated by equation (2). While *Score* guarantees the candidates is able to play the game, *TotalPenalty* is used to measure how much human-like a candidate is.

The objectives are maximizing score and minimizing total penalty. The fitness value is calculated by taking the score minus the total penalty. Note that the score and total penalty may lead to underestimating solutions. If the value of score is extremely big comparing to the total penalty, the total penalty might have no effect. To address this problem, a pair of weight values might be used for the *Score* and *TotalPenalty* respectively, but we consider only weights of penalty features in this research.

The weight of each penalty feature is adjusted according to the means calculated from human data. For instance, the mean of *OnGround* is 0.216, and the mean of *ChangeAction* is 0.333. If the weights of *ChangeAction* is 1.0, then 1.5 is the *OnGround*'s weight.

V. Experiments

To train our agent, we use the evolving neural network, which is able to alter both weights and structure because it is faster than the fully-connected neural network in term of training time. We did a small preliminary experiment on the performance of both methods using only score in the fitness function. While fully-connected NN requires 1.5 days for 2500 iterations, evolved ANN needs only 8 hours to achieve a similar score.

A. Settings

The input features include three areas and information from 2 nearest enemies $(4, 5)$ such as distance and angle to the enemies as shown in figure 3. Three areas are: (1) four cells in front of Mario, (2) spaces between Mario and the ground, (3) a couple of column in front of Mario to the bottom of the observation, The first area is to check whether or not there is any obstacle that blocks Mario to process forward. The second area is to measure the distance of the current position of Mario to the ground. The final area is to check if there is any gap in front of Mario. These three areas (1,2,3) return the value corresponded $\{true, false\}$ value. We also use Mario status (*isAbleToJump, isAbleToShoot, isOnGround, MarioMode*), Mario positions (x, y), and the position of Mario at one second ago as input features.

The population size of NE is set to 20, and λ is 10. Mutation chance is 0.5, and no crossover. The number of iterations for training is 2500 each level. The number of output is 6 corresponding to six actions in the Mario AI Benchmark. For fitness calculation, we use penalty features as described in the previous section III-A2 with the threshold values of human players in the case of *Free* instruction.

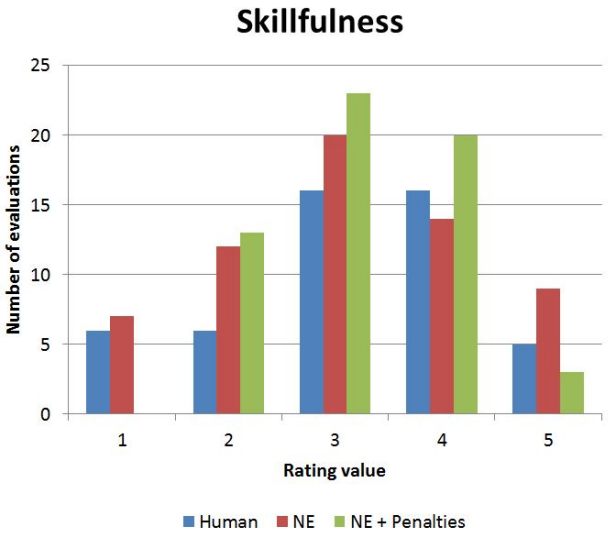

Fig. 4. Skillfulness comparison

TABLE II
THE STATISTICAL VALUE OBTAINED FROM THE T-TEST. $\alpha = 0.05$

Agent	Human	NE	NE + Penalties
Human	-	2.3×10^{-11}	2.0×10^{-4}
NE	-	-	1.7×10^{-3}

We asked 16 human subjects to evaluate 5 pairs of videos. We match videos of the three players (human, NE, proposed method) into pairs. Two videos must be from different players, but the same level is used. It is either human vs. NE, human vs. proposed method, or NE vs. proposed method. We also adjust pairs of videos to make all videos have the same chance to be appeared and evaluated.

B. Results

According to the evaluations from human subjects, we have summarized all the answers and comments. The comparison results are shown in figure 4, and 5.

Figure 4 shows the number of rating for each value from 1 to 5 corresponding to the skillful of each player. The blue column is the human players' performance, red column is for the computer agent, which using only NE approach, and the green column indicates the performance of our proposed method.

The important thing to be compared is the human-likeness. Our main target is to improve the human-likeness of computer agents. Figure 5 shows the human-likeness of human players, NE computer player, and our proposed computer player. In addition, statistics of the original NE and the proposed method, in which five levels similar to the case of human players are given for them to play, are also computed and shown in Table I for comparison.

The t-test is applied to the results shown in figure 5 in order to get the significance difference between means of each pair of agents. The result of the t-test is shown in Table II.

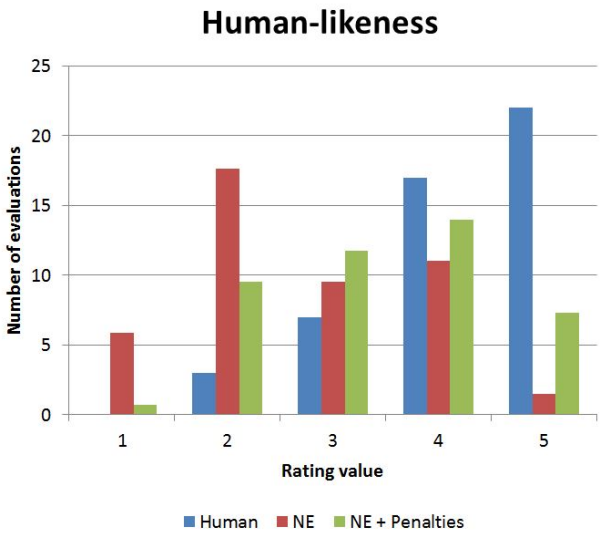

Fig. 5. Human-likeness comparison

According to the results, the average human-likeness of the proposed method is significantly better than the naive NE method, but it is still significantly worse than Human.

C. Discussion

According to the results of the assessments, which is described in the previous section, that we employed to evaluate our agent, we computed the overall score in both cases skillfulness and human-likeness. In average, the skillfulness of human players is 3.16. The computer agent, which was trained by the naive NE, scores 3.10, and our method is 3.22. The best solution in our method after training was more skillful than the best solution in NE without penalties.

The average human-likeness value of NE computer agent is 2.66. Our proposed method scores 3.41. To think that the proposed method has improved the human-likeness of NE by 0.75 points in average. Unfortunately, the gap between humans' score and our agent's is about 0.77 points.

The proposed method showed that the behavior of computer agents trained using NE is affected by the feature penalties. As shown in Table I, the statistics of the agent trained by the proposed method seem to converge to human statistic values. Because of the penalties calculation, the behavior of the computer agent is less mechanical so that human subjects think that it is clever. Hence, the skillfulness of the proposed method is high.

It is easy to see that human players are the most human-like in this case. Human players are ranked 5 in most of the evaluations. The least human-like player is NE. Our proposed method has improved the human-like of NE.

However, the human-likeness is not enough. The reason may be the input and penalty features are not enough, so more detailed/richer information is needed.

VI. Conclusion and Future works

In order to create human-like computer players, the statistical penalties are introduced. Since making computer players to generate similar actions to human players requires human data. Unlike previous methods that human data are directly used for computation, our proposed method requires only statistical data from human players. Therefore, we might not need to collect many data to train a computer agent. An analysis of human players' characteristics in the domain game should be done beforehand. The experimental results show that the proposed method increases the human-likeness of NE. If better features and training strategy are defined, the expected behavior of a computer player may be as human-like as human players.

Future works will focus on improving the efficiency of the training. Richer information and information of the past will be considered. Also, other game domain will be tackled to examine the generality of our method.

References

[1] J. Schaeffer and A. Plaat, "Kasparov versus deep blue: The re-match," *ICCA Journal*, vol. 20, pp. 95–101, 1997.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[3] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[4] J. Togelius, S. Karakovskiy, and R. Baumgarten, "The 2009 mario ai competition," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1–8.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[6] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, pp. 47–62, 2008.

[7] P. Hingston, "A turing test for computer game bots," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, pp. 169–186, 2009.

[8] N. Shaker, J. Togelius, G. N. Yannakakis, L. Poovanna, V. S. Ethiraj, S. J. Johansson, R. G. Reynolds, L. K. Heether, T. Schumann, and M. Gallagher, "The turing test track of the 2012 mario ai championship: entries and evaluation," in *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, 2013, pp. 1–8.

[9] S. Karakovskiy and J. Togelius, "The mario ai benchmark and competitions," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, pp. 55–67, 2012.

[10] J. Schrum, I. V. Karpov, and R. Miikkulainen, "Human-like combat behaviour via multiobjective neuroevolution," in *Believable bots*. Springer, 2013, pp. 119–150.

[11] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating human playing styles in super mario bros," *Entertainment Computing*, vol. 4, pp. 93–104, 2013.

[12] M. T. Rosenstein and A. G. Barto, "Robot weightlifting by direct policy search," in *International Joint Conference on Artificial Intelligence*, 2001, pp. 839–846.

[13] K. Ikeda, "Exemplar-based direct policy search with evolutionary optimization," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2005, pp. 2357–2364.

[14] N. Fujii, Y. Sato, H. Wakama, K. Kazai, and H. Katayose, "Evaluating human-like behaviors of video-game agents autonomously acquired with biological constraints," in *Advances in Computer Entertainment*. Springer, 2013, pp. 61–76.

[15] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, pp. 99–127, 2002.