ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 2, July 2024

AN IMPROVED UTILITY-BASED ARTIFICIAL INTELLIGENCE TO CAPTURE NPC BEHAVIOUR IN FIGHTING GAMES USING GENETIC ALGORITHM

Supeno Mardi Susiki Nugroho[1], Lazuardi Ya'qub Affan[1,2], Mauridhi Hery Purnomo[1]

[1]Computer Engineering, Institut Teknologi Sepuluh Nopember
[2] Calcatz Studio

email: mardi@its.ac.id, lazuardiyaffan@gmail.com, hery@ee.its.ac.id

Abstract

In computer fighting games, the ability of players to play with Non-Player Characters (NPC) is essential. A poorly designed NPC leads to poor player engagements and unsatisfactory playing experiences due to predictable behaviors. A utility-based AI is game designer depended, thus leads to less-varied selections, therefore we propose an improved utility-based AI selected by genetic algorithm (GA) to determine the utility functions of each NPC action. The fitness functions of GA are determined by a rating method named ELO ratings which is usually used in chess games. The variety of decisions of human-like NPCs are generated by utility-based AI, which employ many forms of functions for calculating the value of the AI utility.

Tests on chromosomes in each generation were also carried out to obtain different responses. The Pearson Correlation coefficient is used to obtain an analysis of the influence of each assessment variable. The satisfaction level continues to increase along with the generation iteration with an average of 0.1443 which demonstrates the influences the satisfaction level of game users.

Keywords : Non Player Character, Behavior NPC, Utility based function, Genetic Algorithm, Satisfaction level

Received: 04-07-2024 | **Revised:** 17-07-2024 | **Accepted:** 22-07-2024
DOI: https://doi.org/10.23887/janapati.v13i2.82040

INTRODUCTION

In a game, especially those with action genres such as fighting, the feature of playing with a Non-Player Character (NPC) must be that a player can still play the game even without any other players. An NPC will feel more natural and engaging to play with if the action is not easy to predict, such as when he will chase, attack, defend, or launch an ultimate attack. The method that is used to achieve those abilities, that is artificial intelligence (AI), is implemented in the NPCs as their brain [1]. AI in modern games generally leads to three needs: the ability to move the character, to make decisions on movements or actions, and to think tactically or strategically [2]. In virtual characters, the characteristics most needed to be developed on AI are as follows.

1. Autonomous; so that NPCs can function effectively even without input from a human at run time.
2. Reactive, so that NPCs are conscious and responsive to changing situations.
3. Nondeterministic, so NPCs may give different outcome/behavior upon the same environments.
4. Cultural Authentic is behaving as an individual of the culture depicted.
5. Believable that it should keep the immersion by behaving as human beings [3]

The demand from players for challenging gameplay with intelligent interaction with the NPC over time led to the further development of methods used by game developers for implementing AI in the game's NPC. In popular AI methods such as Behaviors Tree (BT) ([4], defining problematic NPC Behaviors will cause the design of AI Behavior structures to be more complicated as it adds more branches and little changes (a slight change in a node may lead to changes on other nodes). In addition, decision-making is also less varied because it is limited only to the branching condition of each node. Therefore, AI methods are needed to make games easier to design, but they can generate complex behavior with more varied decision making, i.e., low predictability in the perspective of the players.

RELATED WORK

One of AI design methods for NPC in games is utility-based AI [5]. Utility-based AI works by identifying the action options found on the AI and taking the best option (priority) by scoring the usefulness of each action option

based on a particular state, called utility. This makes utility-based AI much more comfortable to design since scoring calculations for each action have no strict relation to other actions. In addition, decision-making is also more varied since the choice of evaluation process is calculated using continuous functions. Such algorithms have been used previously in similar task such as the control of a video game character such as the research by Ng Chee Hou et al [6], where they use a genetic algorithm to create a finite state machine to play Mario Bros competitively, which shows that this algorithm is a viable option for controlling video game characters. Anang et al [7] research optimal GA-based boxing movements. Another study by Edirlei et al [8], where it describes a dynamically generated quest system which produce game quests that is indistinguishable from one made by a professional game designer, this study allows more dynamic storylines where allow a more personalized experience. Another study by Charoenkwan et al [9], also shows that such algorithms can also be used to make games, or mini games of existing games, but it also shows the limitation of such methods where complicated behaviors will cause unpredictable results, another result that they found is the primacy of fitness function which is one of the most important factors in using a genetic algorithm. Lastly a study by Lin et al [10], which shows emergent properties of such algorithms, where it creates tactical formations which would allow such games AI to adapt to unforeseen environments, this not only shows that such algorithms could adapts based on existing conditions, but also could arrive at optimal solutions for such tasks.

This paper proposes an improved utility-based AI which the curves of actions are generated by chromosomes chosen by the GA function.

METHOD

A fighting game prototype is used in this research to implement the proposed method as illustrated in Fig. 1. The game consists of fighting game elements in general, such as face-to-face battle, health points, battle timer, attack combos, attack parrying, and ultimate attack.

In this game prototype system, there are two main phases. Those are procedures in the initialization phase and those in execution phase, as described in Fig. 2. In the initialization phase, the process begins with initializing population data using genetic algorithm. Next is the process of initializing utility-based AI as an evaluator in decision-making of action to be performed by the NPC. The process is then continued with the initialization of the initial condition for each NPC. In addition, various system components used, such as a timer (with a time limit of 120 seconds), game environment or stage, and user interfaces, are also prepared in this phase. Inside the user interfaces, users can view the attributes of each NPC, including the chromosome index of the population, ELO Ratings, the utility values of each action, the values of each parameter of consideration, and the history of actions that the NPC has done.

The next phase is the execution phase, the core phase of how the game works. At this phase, there is a cycle of the system that will keep checking events of the core gameplay, including player input command, time calculation, artificial intelligence process with utility-based AI, population evaluation process with a genetic algorithm, and other events including the appearance of healing items (the item that is used to increase the health point of the NPC) which is also an element that influences the flow of the game. In the implementation phase, an interruption process can pause the game for a moment or stop the game completely. NPC conditions checking is used to determine the events occurring in the game. For example, the determination of whether the fight is in progress, and checking whether the game is over or not is determined by the health point of each player. The game ends if a unit health point reaches zero or the time runs out.

IMPLEMENTATION OF THE CHARACTER'S FIGHTING GAME ELEMENT DESIGN

The design of character attributes as playable and non-playable (NPC) is based on previous works on characters' gameplay attributes [11]. Since the player encounters the enemy in the fighting game, the character attribute design is applied for both player and NPC as a specific enemy gameplay attribute in [11] is designed for multiple enemies. In our fighting game design, we limit the attack type to a physical attack that substitutes the element list. This type of attack is as follows:

- Chain Strong Attack
- Chain Normal Attack
- Ultimate Attack

We design attribute assignment input as in table 1. The distribution of Player HP follows equation 1. As HP increases based on the player's level, the HP' which is the targeted HP, increases according to the current HPi value and the next level(i+1).

$$\mathrm{zP}' = \sum_{i=0}^{\mathrm{N_p}} \mathrm{HP}_{(i+1)} + \left(\mathrm{HP}_{(i+1)} - \mathrm{HP}_i\right) \qquad (1)$$

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 2, July 2024

$$P(C_{St=i_{st}}) = \frac{C_{St=i_{st}}}{(C_{St=0}+C_{St=1}+\ldots+C_{St=i_{st}})} \quad (2)$$

$$D(max_{st}, St_i) = \sqrt{(max_{st} - St_i)^2} \quad (3)$$

$$St' = \begin{cases} \sum_{i=0}^{Np} St_i, if\ D(max_{st}, St_i) \geq i_{st}, St_i = i_{st} \\ \sum_{i=0}^{Np} St_i, if\ D(max_{st}, St_i) \geq 2, St_i = 2 \\ \sum_{i=0}^{Np} St_i, if\ D(max_{st}, St_i) \geq 1, St_i = 1 \\ 0\ ,otherwise \end{cases} \quad (4)$$

The character statistics distribution follows equations 2-4. Equation 2 is the attribute increase probability with ist as the contents of statistics to assign, which value is shown in Table 1 at no. 6.

For example, according to Table 1, the $P(C_{St=1})$ is the probability of statistics to assign as 'Strength', $P(C_{St=2})$ is the probability of stats to assign as 'Endurance'. $C_{St=1}$ is the number of 'Strength' values, and $C_{St=2}$ is the number of 'Endurance' values derived from Table 1.

Each attribute is randomised with results limited based on equation (3). D is the distance within the highest attribute value of x and the current p value. The max_{st} value in equations (3) and (4) depicts the desired maximum attribute value. Each level progression of the player is depicted by St_i, which will continue rising to its highest value Np. St_i is the attribute value that increases every level i.

Figure 1. Fighting Game Used in The Analysis

Table 1. Input Data for Player Gameplay Attributes Calculation

No.	Variable	Input
1.	Start Level	1
2.	Max Level	100
3.	Start HP	159
4.	Next HP	163
5.	List Element	['Chain Strong Attack', 'Chain Normal Attack', 'Ultimate Attack']
6.	List Stats Name	['Strength', 'Endurance', 'Speed', 'Luck']
7.	Max Stats Value	[74, 63, 65, 60]
8.	Stats to Assign	[2, 1]

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 2, July 2024

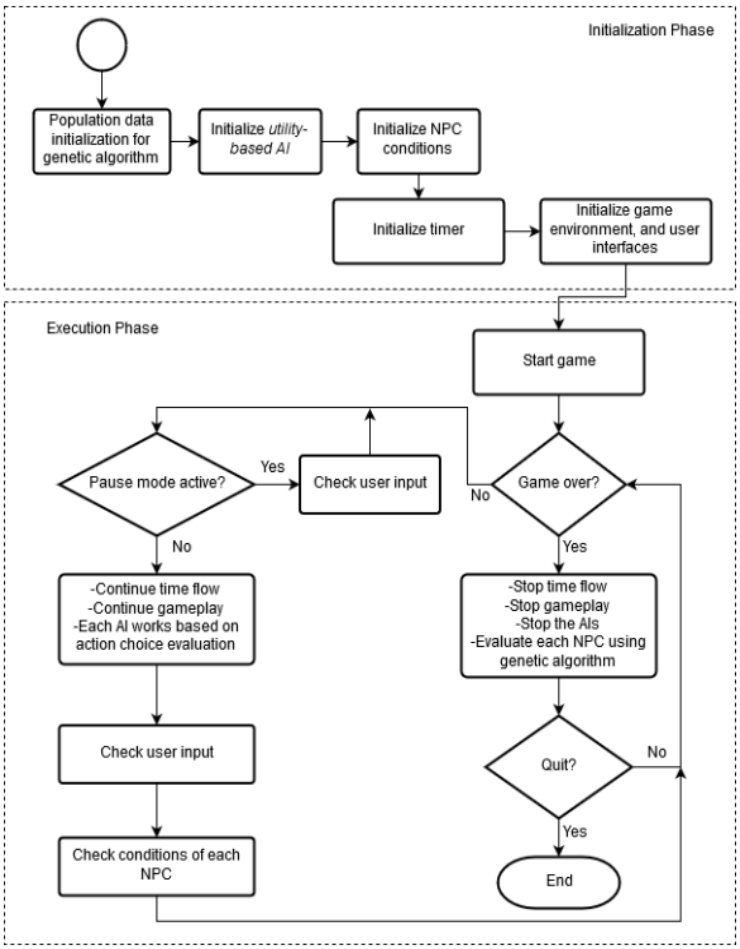


Figure 2. Research Procedures in System Design

PROPOSED UTILITY-BASE AI FOR NPC BEHAVIOUR

The NPC acts as an opponent in the game used in this study. To be able to work in accordance with its role as an opponent, it takes artificial intelligence (AI) that will act as the brain of the NPC. In this research, an AI process for decision-making of action using utility-based AI is applied.

In its application, the first thing to do is to determine the actions that can be done by the character, along with their consideration parameters. Each action has a utility value against one parameter of consideration. However, it is not impossible that action also has more than one consideration parameter.

Therefore, the expected utility calculation is used with the following formula.

$$EU = \sum_{i=1}^{n} D_i \, P_i \qquad (5)$$

In Eq. (5), D is the utility value of an action based on each parameter i, and P is the probability of the utility. This calculation is done for each action, and the action with the most significant expected utility will be chosen. This is called the Principle of Maximum Expected Utility [5]. This value can also be divided by the number of consideration parameters to get the average value.

The following is a list of actions that characters can carry out.

1. Chain Strong Attack
2. Chain Normal Attack
3. Chase Enemy
4. Chase Healing Item
5. Parry
6. Ultimate Attack

It is necessary to determine the forms of the utility functions for each consideration parameter in all actions. These functions will be modified using genetic algorithm later. Utility functions are created using a feature of Unity 3D called Animation Curve [12]-[14]. It is an object that holds a collection of keyframe objects located in a two-dimensional cartesian coordinate system. The keyframe's position can be moved using a handle, allowing the user to

create a curve that can be used as a function to evaluate value over time.

Using the Animation Curve, utility functions for all actions with each parameter of consideration are normalized based on the min-max values of their respective parameters. Those values are normalized into a range from zero as the minimum possible and one as the maximum possibility.

The following list explains the utility value calculation from each action based on their respective parameter.

1. Chain Strong Attack

Utility functions for this action are determined by distance and elapsed attack time of the enemy as can be seen in Fig. 3. The enemy distance parameter derived from the normal Gaussian distribution with its apex lies at 0.3 as in Figure 3a. This kind of attack has a longer execution time thus it is better utilized in the maximum distance of this attack can be delivered. Also, regarding the enemy executed attack time or after idling for a relatively long time as seen in figure 3b, which shows reverse parabolic function.

2.Chain Normal Attack

The utility function for the action has a short execution time which is better for utilizing this action at a short distance. As shown in figure 4, we design a utility function with an exponential decrease as further the distance of the enemy. The time parameter is derived from the sinusoid function that provides maximum utility value when the enemy finishes their attack or as a follow-up to another action.

3.Chase Enemy

This action determines whether chasing the enemy to engage then is more necessary than other chase actions. Enemy distance and character health point condition determine its utility value. A higher distance means the enemy is likely out of reach of the character attack range, thus motivating chasing the enemy. The high health point condition means the character is more confident to be engaged.

4.Chase healing item

The action of chasing a healing item affects the character health point condition, and the utility value is based on a sigmoid function which denotes that a lower health condition will motivate the character to chase healing item action. The higher enemy health point linearly will motivate higher utility value for this action. Lower distance from the character to the healing item position motivates the character to chase the action of the healing item.

5. Parry

Parry action enables the character to parry an enemy attack with its utility value derived from the distance s from the enemy elapsed attack time. As the distance from the character is lower, this will make the parry action utility value higher, as the enemy elapsed attack time is based on gaussian distribution that makes the utility value on its apex at 0.15 on normalized time.

6. Ultimate Attack

This ultimate attack action can be performed if a certain condition is fulfilled. This condition is determined by the power point gained from successful chain normal and strong attacks. Just like the previous attack, the distance of the enemy is paramount to successfully executing this attack. Thus, the utility value based on its distance is derived based on the optimal distance of attacking with this action. Furthermore, the utility function is also determined by power value based on the binary step function since this action can only be performed if a valid power condition is fulfilled.

Each calculation and evaluation of action options should be performed at run time, in the sense that utility values are not specified when the scenario is being designed but evaluated based on the exact situation occurring at the simulation time [15].

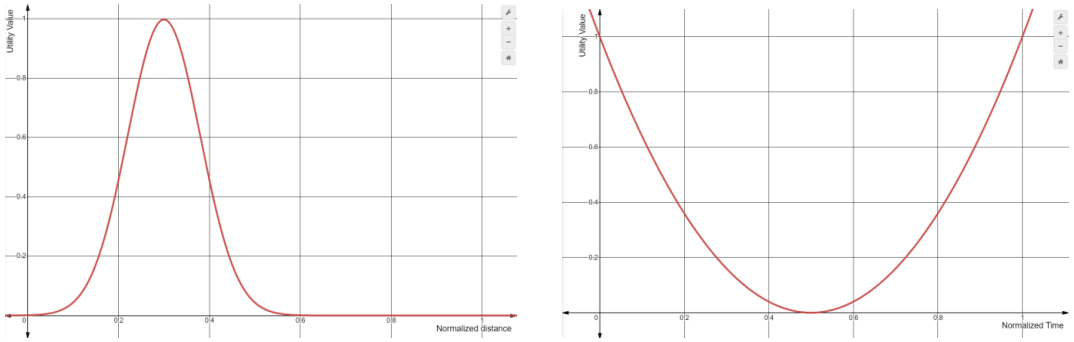
Proposed Genetic Algorithm to Optimize Utility Functions

GA is a problem-solving method that uses genetics as a problem-solving model. This method is a technique for finding approximate solutions to optimization problems by using the fitness function to get a better solution over each generation [16]. The architecture of GA implementation in Fighting Game by mapping each NPC character unit (Chromosome) to Utility Function (Gene) can be shown in Fig. 4. NPC characters in this game can influence AI.

The chromosome is modelled as an object that stores a collection of utility functions for all actions that stores a collection of utility functions for all actions based on each consideration parameter. Thus, any utility function can be called a gene.

A population of thirty chromosomes was used as the initial population. The initial chromosome has utility functions that have been created manually. The twenty-nine initial chromosome copies are created to make the other twenty-nine chromosomes. A light mutation is then applied to those twenty-nine new chromosomes by changing one gene (one form of function) of each new chromosome

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 2, July 2024



(a) (b)

Figure 3. Utility Functions for The Action of Strong Chain Attack (A) Based on The Parameter of Enemy Distance (B) Based on The Parameter of The Enemy Has Elapsed Attack Tim

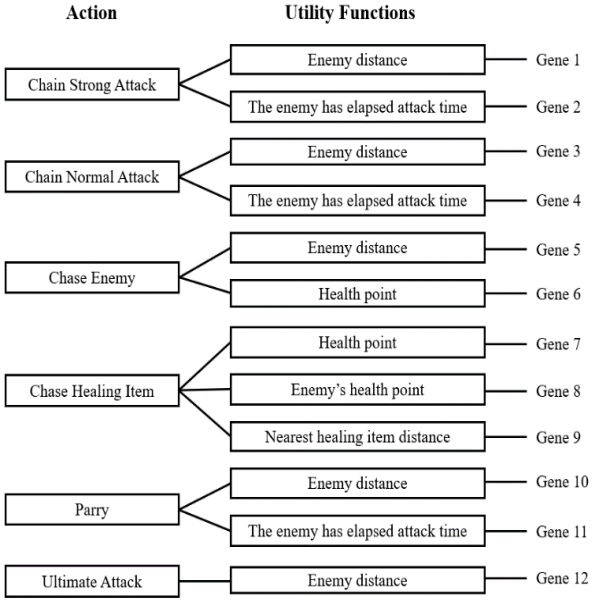


Figure 4. Design on GA Implementation in Fighting Game by Mapping Each NPC Character Unit (Chromosomes) to Utility Functions (Gene)

Defining the Fitness Function

The way to evaluate whether AI works correctly in a fitness function system is defined. The system must be robust, designed to provide a comparative, and capable of evaluating where new individuals are often added or removed. The system should also allow for variations in individual performance, which concludes the outcome of many fights, rather than relying on one performance condition (win or lose) as a basis for ranking since a match itself is not entirely dependent on one NPC's actions. The second NPC's (opponent) reaction contributes significantly to success or failure. In addition, the actions of each NPC will vary so that each battle between each pair will be different. Therefore, the ELO Rating system works because the performance of NPCs in any match is a random variable in a normal distribution. The NPC's overall rating must also be considered with the enemy's ability, winning, and losing outcomes. An equation is then applied to the rating to predict the NPC's winning probability. The result of the equation is then used to calculate how many points will be added to the winner's rating and how many points will be subtracted from the loser's rating. The way to calculate the expected score of NPCs, the following formulas are used.

$$E_{P1} = \frac{1}{1+10^{(R_{P2}-R_{P1})/400}} \quad (6)$$

$$E_{P2} = \frac{1}{1+10^{(R_{P2}-R_{P1})/400}} \quad (7)$$

In Eq. (6) and Eq. (7), P1 and P2 are the first NPC and second NPC. R1 and R2 are the ratings of both NPC. After the match between two NPCs, the ratings will be adjusted to the amount proportional to the difference between the expected score and the winning result. The formula used is as follows.

$$R'_{P1} = R_{P1} + K(S_{P1} - E_{P1}) \quad (8)$$

In Eq. (8), K is a factor that controls the strength of an adjustment, and S is a Boolean (zero or one), which indicates victory or defeat. Every 30 chromosomes in the population will be given 1000 ELO Ratings. More vital NPCs are expected to reach ELO Ratings of up to 1000. Conversely, weaker NPC has ELO Ratings below 1000. A round-robin tournament is held between all the existing chromosomes to find the ELO Ratings for each chromosome. Each of the 30 chromosomes will fight 30 others except itself, which counts as 29.

Each victory of a chromosome will increase the ELO Ratings of the chromosome. In contrast, ELO Ratings will be reduced if the chromosome loses. The difference in the addition or subtraction of ELO Ratings is applied from the calculation of the ELO Rating formula described, with a K factor value of 50.

Determine which chromosomes will survive to the next generation and which chromosomes will be used for crossovers are needed. After calculating through a round-robin tournament, sorting populations makes the selection based on ELO Ratings. From the sorted population, one-third of chromosomes with the highest ELO Ratings are taken to form a new population. In this case, out of a third of the population of 30 chromosomes, the new population consists of 10 chromosomes.

After obtaining one-third of the best chromosomes as the new population through the selection process, ten pairs of chromosomes are randomly selected. Against those ten pairs of chromosomes are then crossovers to get two new offspring.

That way, the population, which is one-third of the initial population (10 chromosomes), will be added with two new offspring from each chromosome pair with a total of 20 new chromosomes. So, the population becomes as much as the original number, which is 30 chromosomes.

After getting 20 new chromosomes through a crossover process, the chromosomes are not directly added to the population, but the mutation is done first. The mutation is done by altering the gene by randomly changing the position and gradient of the keyframe from the selected function's animation curve. The value of the mutation rate used in implementing this GA is 0.25.

This means that a quarter of all the genes in the new chromosomes are altered. A quarter of these genes are randomly selected from any chromosome so that each chromosome may have a different number of mutated genes. Once the mutation is done, the new chromosomes are inserted into the population, resulting in a new population for the next generation. ELO Ratings of the new population are returned to 1000 for the last tournament to produce new ELO Ratings. Every GA process that has been described is continuously repeated until it reaches the 40th generation.

Testing NPC Against Human

AI system testing is conducted to provide data on whether the NPC that implements utility-based AI by utilizing GA has given the most optimized utility functions at each action to produce the best behavior. Therefore, in the AI system testing, the ability of AI is tested by comparing the fitness value of each of the best chromosomes in each generation by re-holding a round-robin tournament. The best chromosome in a generation is the chromosome that has the highest ELO Ratings.

After several best chromosome samples are produced for each generation, to find a challenging AI when opposed by human players, human testing with 25 respondents of age above thirteen years is carried out. The age range was chosen because of the relatively hard difficulty level of the game, and those older players can better analyze and assess the behavior of the AIs. Out of the 40 chromosome samples from each generation, four samples representing the first generation to the latest generation were taken on the chromosomes in the 1st, 13th, 27th, and 40th generations. Every tester (human player) must then play these four chromosomes. Each tester must record their win (Boolean with a value of 1) or lose (Boolean with a value of 0) status, score the difficulty level (on a scale of five), and score the satisfaction level (on a scale of five). The average result of the winning status, difficulty level, and satisfaction level from all the respondents can be seen in Table 2.

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 2, July 2024

Table 2. The Average Result of The Winning Status, Difficulty Level, and Satisfaction Level

	1st Gen	13th Gen	27th Gen	40th Gen
Winning Status	0.64	0.36	0.44	0.4
Difficulty Level	2.77	3.34	3.81	3.87
Satisfaction Level	3.41	3.49	3.79	3.84

Based on the data in Table 2, chromosomes from the first generation to the newest generation tend to have an increasing value of difficulty level. It follows the expectation that AI training using GA can produce chromosomes that become more vital as the generation iterates. The decrease also follows the increasing value of the difficulty level in the winning status of human players. Although the value of winning status from the 13th generation to the 27th generation increased, the values afterward continued to decrease. While the value of satisfaction level continuously increases as generation iterates.

Correlations between winning status, difficulty level, and satisfaction level need to be measured. Calculations of the correlation coefficient between winning status and difficulty level, winning status and satisfaction level, and difficulty level are then performed. The correlation coefficient calculation result is then analyzed to deduce the criteria for challenging AI to be played by human players. The formula used to calculate the correlation coefficient is Pearson's correlation coefficient [17]–[19].

Based on Table 3, the winning status and the difficulty level have a strong negative linear correlation (-0.504), so the more complicated an AI is, the lower the possibility of a human player winning the match. Winning status and satisfaction level have a weak negative linear correlation (-0.037). Hardly any correlation indicates that winning status has almost no effect on satisfaction level because there are some testers who are satisfied when winning, while others are not satisfied when winning. While difficulty level and satisfaction level have a medium positive linear correlation (0.455), which means the testers are more satisfied when faced with a more challenging opponent.

Table 3. Correlation Between Winning Status, Difficulty Level, and Satisfaction Level

Measured Correlation	Correlation
Winning Status and Difficulty Level	-0.504
Winning Status and Satisfaction Level	-0.037
Difficulty Level and Satisfaction Level	0.426

RESULT AND DISCUSSION

AI system testing is conducted to provide data on whether the NPC that implements utility-based AI by utilizing GA [17], [18] has given the most optimized utility functions at each action to produce the best behavior. Therefore, in the AI system testing, the ability of AI is tested by comparing the fitness value of each of the best chromosomes in each generation by re-holding a round-robin tournament. The best chromosome in a generation is the chromosome that has the highest ELO Ratings.

Round robin tournament needs to be done again because the ELO Ratings used for fitness apply only to chromosome comparisons in one generation that has undergone a round-robin tournament in the same generation. The best chromosome samples are required in each generation to be held round-robin to produce ELO Ratings on each of these chromosomes to compare fitness (ELO Ratings) in different generations. After the round-robin tournament, the ELO Rating graph of the representational chromosomes of each generation is shown in Figure 5. The winning percentage graph is also shown in Figure 6.

The expected result is the increasing value of ELO Ratings from the first generation to the last generation (40th generation) caused by chromosomes which continuously improve their utility functions of each action in every iteration of its generation. In Fig. 5, the value of ELO Ratings as generation increases develops lucratively but results in a new peak or maximum value at specific points. This shows that the training process using GA can produce chromosomes with a more vital ability as generation increases, even though several generations of them must decrease ELO Ratings.

The graph in Fig. 6 also has a pattern like the ELO Rating graph in Fig. 5. The percentage of winnings on each chromosome as the fluctuating generation grows, resulting in a new top or maximum value at specific point.

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 2, July 2024

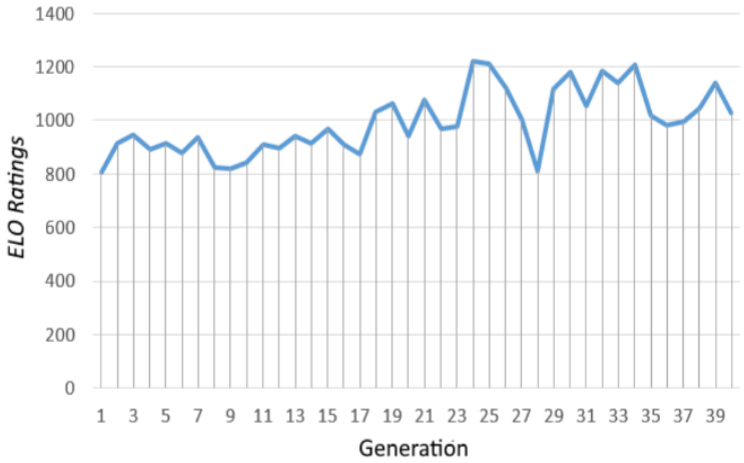


Figure 5. Relation of ELO Ratings (Fitness) Obtained Through Round-Robin Tournament of Each Representative of The Best Chromosomes in Each Generation

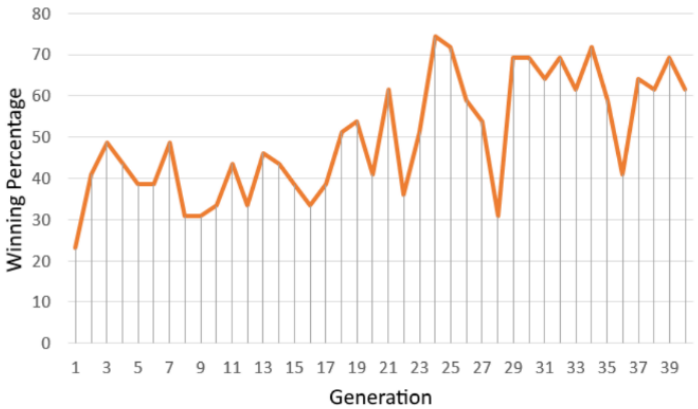


Figure 6. Winning Rate Obtained Through Round-Robin Tournament of Each Representative of The Best Chromosomes in Each Generation

Despite having the same pattern, the two graphs do not always have the same increasing or decreasing behavior. For example, an increase in value between the 4th generation to the 5th generation of the ELO Rating graph precisely followed by the decreasing value of the winning percentage of the 4th generation to the 5th generation. This occurs in accordance with the principle of ELO Ratings that the performance of a player (in this context is AI) is a random variable of a normal distribution, where in addition to considering the results of winning and losing, ratings or the overall ability of both an NPC and it is opponent also a consideration. It is good for NPC's action in game, which must mimicking human behavior.

CONCLUSION

Based on the results of implementation and experimentation, some conclusions can be drawn as follows:

1) AI system testing results show that the training process using GA, with NPC as a chromosome model and the utility functions of each action are modelled as genes, and ELO Ratings as the fitness function can produce AI chromosomes with more vital ability as generation iterates. Although several generations of them experienced decreasing ELO Ratings, as generation iterates, it creates new chromosomes with new highest ELO Ratings ensure generations.

2) Based on experiments on the chromosomes in the 1st, 13th, 27th, and 40th generations, each generation has divergent responses to each assessment on Winning Status, Difficulty Level, and Satisfaction Level. As in the first generation to the latest generation, the value of the difficulty level on the chromosomes increases by an average of 0.3637. This is relevant to the aim of this research that the implementation of AI using GA can produce obtrusive chromosomes as generations increase.

However, the winning status value tends to decrease, although the winning status value from the 13th generation to the 27th generation literally increased by 0.08, then decreased by 0.09% (from the 27th generation to the 40th generation). While the value of the satisfaction level continues to increase along with the generation iteration with an average of 0.1443.

Future works will be focused on the creation of utility-based AI with more detailed actions and more consideration parameters so that the chosen decisions are more precise and to add more characters to the game with a balanced ability since character variation and character selection is an important feature [20] in a fighting game. In addition, things to consider in the development of the Fighting Game regarding the intelligent behavior of independent NPC agents is comparing several approaches, including hybrid methods, by observing learning speed instead of the greatest fitness function values only.

ACKNOWLEDGMENT

This work was supported by Institut Teknologi Sepuluh Nopember as a part of the Upgrading Program on Undergraduate Thesis.

REFERENCES

[1] Christyowidiasmoro, R. C. A. Putra, and S. M. Susiki, “Measuring level of difficulty in game using challenging rate (CR) on 2D Real time Strategy Line Defense game”, Proc. - 2015 Int. Electron. Symp. Emerg. Technol. Electron. Information, IES 2015, pp. 218–222, 2016, doi: 10.1109/ELECSYM.2015.7380844.

[2] R. Dreżewski and J. Solawa, “The application of selected modern artificial intelligence techniques in an exemplary strategy game,” Comput. Sci., vol. 192, pp. 1914–1923, 2021

[3] K. Dill, “A Game AI Approach to Autonomous Control of Virtual Characters”, in Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011, 2011, no. 11136, pp. 1–11.

[4] Y. Hossain and L. Zaman, “NCCollab : collaborative behaviour tree authoring in game development”, in Multimedia Tools and Applications, 2022.

[5] J. Norstad, “An Introduction to Utility Theory”, Game AI, pp. 67–80, 2005, doi: 10.1201/9780429055058-6.

[6] N. C. Hou, N. S. Hong, C. K. On and J. Teo, "Infinite Mario Bross AI using Genetic Algorithm," 2011 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT), Semenyih, Malaysia, 2011, pp. 85-89, doi: 10.1109/STUDENT.2011.6089330.

[7] A. K. Adisusilo, M. Hariadi, A. Zaini, S. M. Susiki, “Optimizing of Boxing Agent Behaviour Using Genetic Algorithm,” Jurnal Ilmiah Kursor Vol 7,no. 2, pp. 55–6,2013

[8] E. Soares de Lima, B. Feijó and A. L. Furtado, "Procedural Generation of Quests for Games Using Genetic Algorithms and Automated Planning," 2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Rio de Janeiro, Brazil, 2019, pp. 144-153, doi: 10.1109/SBGames.2019.00028.

[9] P. Charoenkwan, S. W. Fang, and S. K. Wong, “A study on genetic algorithm and neural network for implementing mini-games”, Proceedings - International Conference on Technologies and Applications of Artificial Intelligence, TAAI 2010, 2010, pp. 158–165. doi: 10.1109/TAAI.2010.35.

[10] C. S. Lin and C. K. Ting, “Emergent tactical formation using genetic algorithm in real-time strategy games”, Proceedings - 2011 Conference on Technologies and Applications of Artificial Intelligence, TAAI 2011, 2011. doi: 10.1109/TAAI.2011.63. | IEEE Conference Publication | IEEE Xplore

[11] N. R. Widiyanto, S. M. S. Nugroho, and M. H. Purnomo, “The Calculation of Player ’ s and Non-Player Character ’ s Gameplay Attribute Growth in Role-Playing Game with K-NN and Naive Bayes”, 2020 International Conference on Computer Engineering, Network and Intelligent Multimedia, 2021, Cenim 2020, pp. 103–110

[12] Y. Jiang, B. Xiao, B. Yang, and X. Guo,”Study of plant animation synthesis by unity3D”, IFIP Adv. Inf. Commun. Technol., vol. 452, pp. 344–350, 2015, doi: 10.1007/978-3-319-19620-6_39.

[13] J. Wang and W. Zhu, “Design and Implementation of Virtual Animation Based on Unity3D”, 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), 2021, pp. 667–669,doi: 10.1109/ainit54228.2021.00134.

[14] H. Jeon, E. Chae, and H. Pak, “Study of Camera Path and Motion Data Creation for UNITY 3D Game Engine”, vol. 65, pp. 13–16, 2014, doi: 10.14257/astl.2014.65.04.

[15] K. Dill, E. R. Pursel, P. Garrity, and G. Fragomeni, “Design Patterns for the Configuration of Utility-Based AI”, Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012, 2012, no. 12146, pp. 1–12

[16] S. N. Sivanandam and S. N. Deepa, “Introduction to Genetic Algorithms”, Springer Science & Business Media, 2007.
[17] P. Bonanno and P. A. M. Kommers, ”Gender differences and styles in the use of digital games” , vol. 25, no. 1, 2005.
[18] R. Festl, M. Scharkow, and T. Quandt, "Problematic computer game use among adolescents, younger and older adults," Addiction, vol. 108, no. 3, pp. 592–599, 2013, doi: 10.1111/add.12016.
[19] W. Frencken, K. Lemmink, N. Delleman, and C. Visscher, "Oscillations of centroid position and surface area of soccer teams in small-sided games”, Eur. J. Sport Sci., vol. 11, no. 4, pp. 215–223, 2011, doi: 10.1080/17461391.2010.499967.
[20] B. K. Khotimah, M. Miswanto, and H. Suprajitno, “Optimisation of feature selection using genetic algorithm in naïve Bayes classification for incomplete data”, Int. J. Intell. Eng. Syst., vol. 13, no. 1, pp. 334–343, 2020, doi: 10.22266/ijies2020.0229.31.