Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 156 (2019) 395–403

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Young Scientist Conference on Computational Science

Dynamic Difficulty Adjustment with a simplification ability using neuroevolution

Mariia Shakhova*, Aleksandr Zagarskikh

ITMO University, Saint-Petersburg, Russia

Abstract

This paper describes a concept of a dynamic difficulty adjustment system that learns and adapts online to the style of the player's in-game behavior based on the neuroevolution using additionally the base of catalogues. A neural network trained through evolutionary algorithms is used to achieve better adaptation. In this case, genetic algorithms alter the weights of the neural network. In addition to adaptation, the proposed method provides an opportunity to downgrade difficulty when it is necessary or respond to rapid changes in the player skill level. The test game in the genre of real-time first-person fighting is designed to validate the efficiency of the proposed model. Multilayer perceptron (MLP) architecture was selected as a neural network's topology to achieve the maximum correlation between an approximation of the loss function and speed of the neural network. The article describes the advantages and limitations of the proposed concept in comparison with other approaches.

Keywords: Dynamic Difficulty Adjustment; Neural Networks; Genetic Algorithm; Neuroevolution;

1. Introduction

Nowadays, the trend in game development is to adapt the game's difficulty according to the player's skills. This adaptation provides a new experience and creates an appropriate level of difficulty for a player. When a player feels challenge, he gains pleasure out of gameplay and therefore continues to play. Without this feeling, the game may become boring because a player can predict the sequence of game events or behavior of game characters. Therefore,

* Corresponding author. Tel.: +7-981-815-90-37.
E-mail address: shakhova.mary@gmail.com

1877-0509
10.1016/j.procs.2019.08.219

a player stops getting a new experience from the game and may stop playing. Conversely, if the difficulty level of the game is too high for a player, he may become angry and will not like it at all.

These problems can be solved with a Dynamic Difficulty Adjustment (DDA) concept, which maintains the game's difficulty level and tunes it according to player skills and style of his playthrough. DDA can influence the adaptation and behavior of agents driven by artificial intelligence. In particular, it can affect their characteristics (amount of health, strength, etc.), tactical reasoning, strategic planning, and the number of them. The process of adaptation can proceed during the game (online) or before the start of the game, between levels of the game or during the game pause (offline). Offline learning provides better performance than online; however, it has a lack of responsiveness. At the same time, online learning has a faster response to the player actions and the problem of low performance becomes less meaningful with the growth of computing powers.

Standard systems with predefined levels of game difficulty, where the player can choose it only when the game starts, have several disadvantages. First, the player cannot know exactly an appropriate difficult for him and as a result, his choice may be unsuitable. Second, such systems do not track and do not consider the progress of the player. Next, a player can learn behavior patterns of the enemy and consequently, gameplay becomes more predictable. The DDA concept aims to solve the problems mentioned above. Contrary to standard discrete systems, DDA provides a continuous approach to configure game difficulty. The development of a difficulty balancing system is a complex problem and requires advanced and modern methods. The research objective of this work is to develop a DDA system that will be learning online and adapting to the player's game style by using genetic algorithms to achieve better difficulty scaling. Furthermore, the additional requirement for this system is that it should be able to adjust the difficulty level, if the player's progress has changed a lot, unexpectedly. For example, the player had not played for a long time and as a result, his level of skill in the game has heavily decreased. The system should consider this and lower the level of the game difficulty.

This paper presents a system for adaptation of game difficulty level by methods of machine learning and algorithms of training neural networks through evolutionary approaches. It was developed by connecting game development technologies and neuroevolution methods [1]. Our system is designed according to modern requirements proposed by Andrade et al. [2] for difficulty scaling systems:

- The game must identify and adapt itself to the human player's initial level as quickly as possible
- The game must track the evolutions and regressions in the player's performance as close and as fast as possible
- In adapting itself, the behavior of the game must remain believable, since the user is not meant to perceive that the computer is somehow facilitating things

The remainder of this work is structured as follows. Section 2 presents a review of related works concerning DDA systems and neuroevolution algorithms. Section 3 explains the methods of adaptation that were used to fit all the requirements. Section 4 presents the results of a comparison with other approaches. The conclusion is discussed in section 5.

2. Related works

There are various implementations of DDA systems in games. The main requirement for them is to define the challenge function [3]. It is a heuristic function, which is used to evaluate a current level of difficulty at any given time by matching value to each state of the game. Indicators of the success of adaptation (such as health points, number of hits, amount of damage, etc.) are usually used as the parameters of the challenge function. An overview of some popular and modern approaches for building the DDA system is presented below.

2.1. Dynamic Scripting Method

Riwinoto et al.[4] use a dynamic scripting method in Game Arcade. The dynamic scripting monitors player behavior and generates various scripts for game agents controlled by artificial intelligence. For each agent, there is own database with a predefined set of rules. The behavior of characters is defined by the weights of these rules. The system contains events, that affects these weights when fired. For each agent, this effect has a unique behavior because

it is influenced by agent's inner properties. In the beginning, each rule has an equal chance to be in the generated script.

The main advantage of this method is that it fits well in systems with small amount of rules. Furthermore, each agent behaves diverse due to own unique rule base. In contrast, a game developer has to describe all possible rules and it may be burdensome and time-consuming. Besides, these rules cannot be changed during the gameplay process.

2.2. Skilled Experience Catalogue

Skilled Experience Catalogue [5] (SEC) is a system that balances the skill level of NPCs in real-time according to the player behavior. In the beginning, a series of training games is suggested to a player, and as a result, the system creates a catalogue of experience with learning policies. During the game this catalogue is used to adjust the policy of NPC according to the player skill level. At the same time, NPC continue evolving with help of reinforcement learning. SEC system was tested in the First Person Shooter game.

The mechanism of using milestones from the catalogue allows running only one SEC instance to tune the level of the whole group of opponents with artificial intelligence. SEC can be used in many genres of games. The disadvantage of this approach is the high requirements of system computational resources. Therefore, it has to have an accurate implementation of its components to affect game performance as less as possible.

2.3. Multiple-Periodic Reinforcement Learning

Sekhavat [6] proposes a rehabilitation game, in which for improving rehabilitation of patients a system takes into account individual progress and a behavior of each patient in real-time. This system uses Multiple-Periodic Reinforcement Learning (MPRL) to evaluate several objectives in DDA in a separate periods of time. Results show that MPRL improves patients motor skills and satisfy player's requirements better than Multiple-Objective Reinforcement learning. The limitation of this system is that it considers only player skill level, but there is also a requirement to learn his play style and individual characteristics. Besides, MPRL has low performance because it needs to handle different objectives. Moreover, MPRL requires special equipment like Kinect.

2.4. EEG-triggered dynamic difficulty adjustment

In this model [7], the authors suggests to measure a player excitement to control DDA. They are using special equipment, such as the Emotiv EPOC EEG headset, for collecting a passive brain-sensing information. In according to this data, they define when a player enjoys the game. Several experiments with multiplayer third-person shooter game were performed to demonstrate this concept. As a result, these experiments show that DDA based on EEG has better performance in terms of increasing the player excitement and improving the game experience compared to other models.

The main advantage of this method is that it gets data about player emotional state directly from his head. However, it requires special equipment, which can be expensive and difficult to find on the market. The other disadvantage is that it is hard to develop software for this type of headset. Moreover, game developer has to be familiar with an equipment specification.

2.5. A neural network with genetic algorithm

Weeks et al [8] propose the concept of controlling agents by a neural network, which weights are altered by genetic algorithm. This approach is called neuroevolution [9]. The authors developed the model to describe this concept that includes two characters: an ogre, controlled by the neural network, and a wizard, controlled by a primitive set of rules. The neural network receives a list of game properties that become considerations of the ogre's decision-making process. In the result, the neural network calculates a set of possible actions with weights. Than ogre determines the next move by choosing the action with maximum weight.

The first advantage of this method is that it provides a better performance compared to other learnings methods. Moreover, neuroevolution algorithms are well scalable, because it can be applied to any game artificial intelligent.

The disadvantage of the system that it is limited by a set of input factors and possible output actions. If the developer wants to change these parameters, he needs to train the neural network again. The other problem is that with online training of neural network the same result cannot be received on each game instance.

The result of approaches comparison is presented in Table 1. Most of the methods support online learning and not require any predefined rules. The main problem of all approaches except the SEC is that they cannot provide the possibility to downgrade difficulty when it is necessary. At the same time, there is only one limitation for the neural network with the genetic algorithm in this comparison. Moreover, neural networks are a flexible powerful tool that provides a high degree of accuracy.

Table 1. Approaches Comparison.

	Dynamic Scripting Model	SEC	MPRL	EEG-triggered	Neural network with genetic algorithm
Offline learning	-	+	-	+	-
Online learning	+	+	+	+	+
Predefined rules	+	-	+	-	-
Scalability	-	-	-	-	+
Requires special equipment	-	-	+	+	-
Downgrade difficulty	-	+	-	-	-

For these reasons, it was decided to use the approach of neuroevolution to design a model of difficulty level adaptation. With this model, the system monitors the player skill, the style of his playthrough and his level progress. Then it takes into account received information and alters the behavior of agents controlled by artificial intelligence, their number and inner properties (amount of health, strength, etc.). This process takes place in real-time to achieve continuous adaptation to player actions without any pre-training of neural networks. The base of catalogues is additionally used to provide the ability to downgrade the level of game difficulty when it is necessary or to respond to rapid changes in the player performance.

3. Method

The model proposed in this paper can be applied to any game with combat agents, where the loss function is defined and the neural network is pre-trained. The test game in the genre of real-time first-person fighting is designed to validate the efficiency of the proposed model. The description of the model and the game is described below.

3.1. Game description

The game contains two characters that combat with each other and use different weapons. One of the characters is controlled by artificial intelligence and another one by human-player. Characters have a set of own parameters, such as the life points, the quantity of stamina, strength and agility. Initially, the number of lives and the amount of stamina of all fighters equals 100. When the life points of one of the fighters fall to zero, the game ends and the other character becomes the winner. Each of the characters can attack the other in three directions (left, right, top) and with two types of strike (fast or strong). Moreover, they can block an enemy attacks if they raise up a shield in the direction of the hit. At the beginning of the battle, each fighter can choose a suitable weapon from his inventory. The list of available weapons is next: sword, axe, mace and spear. Each of the weapons has several benefits and weaknesses inherent in type. For this paper, the system takes into account only battle behavior and therefore fighters can only attack and block the hits, any other movements are temporarily inaccessible. The information about the current state of adaptation is saved after a certain amount of fight in order to be able to change the game difficulty level.

3.2. Behavior model

Behavior modelling is an important component in the implementation of the game difficulty adaptation. The character controlled by a computer analyzes the behavior of the player and responds to his actions in accordance with the specified challenge function. Every next step of this character is defined by a neural network, the weights of which evolves with genetic algorithm.

The behavior of agent consists of next actions: change weapon, attack and block. The agent's choice of weapon is defined by player's previously most used weapons. The final decision of agent about direction and type of attack is influenced by a maximum damage it can cause. The agent blocks a hit when he has an insufficient amount of stamina or decides that the next player's attack will be strong. Besides, the agent considers the player's history of attacks, for instance which types or directions of attacks the player chooses most frequently.

The fitness function is used for evaluation of difficulty adaptation success at any given moment. In this paper, the difference between the current difficulty level of agent and the player skill level is used as fitness function parameters.

The current difficulty level of the agent is calculated as an amount of damage inflicted on the player character by the agent during the fight. This damage defines as the difference between damage dealt by the agent and damage taken by the player's character. It is related to the fact that the part or all of the damage can be blocked by the shield. As a result, in fight number i the agent grade is calculated using equation 1:

$$grade_i = \frac{(d_d - d_t)}{h_p \cdot t}, \tag{1}$$

where d_d - damage dealt by the agent, d_t - damage taken by the player's character, h_p - the amount of player health, t - the duration of the fight.

For calculating the player's skill level, the system takes into account the ability of the player to block the hits and to deal maximum damage. As a result, the equation 2 was used for evaluating the player's skill level in fight number i:

$$skill_i = \frac{n_d}{n_m} \cdot \frac{(d_d^p - d_t^p)}{t}, \tag{2}$$

where n_d - the number of blocked hits, n_m - the number of missed hits, d_d^p - damage dealt by the player, d_t^p - damage dealt by the agent, t - the duration of the fight.

The fitness function (equation 3) is an evaluation of the difference between $skill$ and $grade$:

$$rank_i = skill_i - grade_i \tag{3}$$

After every fight, the weights of the neural network and some additional information are recorded. All data from previous fights are processed and ranked by value of $rank$ when the game loads. The weights associated with the value of $rank$ close to zero determine the best behavior of agent at a given time.

3.3. Neural network architecture

In this work, multilayer perceptron (MLP) architecture with an input layer of 18 neurons, three hidden layers with 72 neurons and one output layer with 13 neurons, was selected as the neural network's topology. The structure of neural network is illustrated in Figure 1. The neurons of the hidden layers have a sigmoid activation function. The output softmax layer is used to modelling probability distribution. It means that outputs show the probability of choosing decision option. The function of the agent grade was selected as the loss function of the neural network. The MLP architecture has been selected to achieve the maximum correlation between an approximation of the loss function and speed of the neural network.

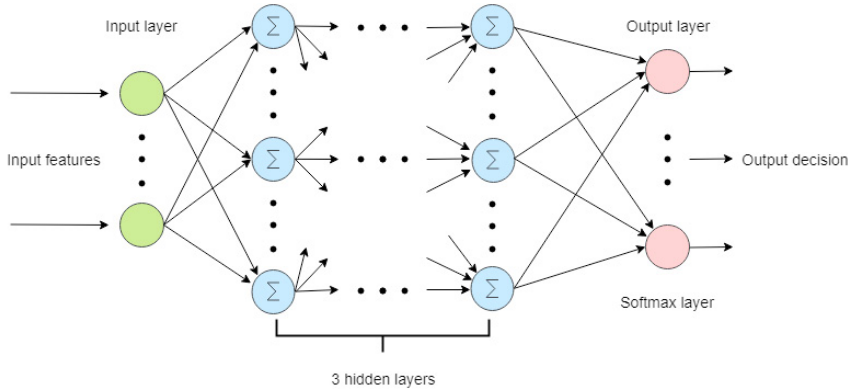

Fig. 1. Structure of neural network.

Next features are used as inputs: how much agent's or player's character's health and stamina points remains, which weapons agent or player's character choose (whether the weapon was used), whether the agent or the player's character was recently hurt, whether the agent or the player character recently blocked an attack, the fight duration, the player rank. All values of inputs are normalized to the interval from 0 to 1. The outputs are 13 values representing possible agent movements: strong or fast attack in one direction, block the hit in one direction or choose one of the available weapons.

3.4. Genetic Algorithm component

For altering the weights of the neural network, the system uses procedures of selection, crossover and mutation from the genetic algorithm [10]. New sets of weights are created by combining two of top previous ranking data sets. A two-point crossover is used in this method instead of the usual single-point crossover. It solves the problem of always swapping over the end genes. Furthermore, the two-point crossover can create combinations of genes that single-point crossover simply cannot provide. The weights of the neural network in each dataset are converted to long, binary strings. After that, the first parent is chosen by random from the two datasets and two crossover points are selected randomly. The binary string of the first parent is copied from the beginning to the first crossover point and from the second crossover point to the end of the string. The area of binary string between the two crossover points is taken from the second parent. For making mutations, the bits in the string are selected and are flipped randomly. Figure 2 shows overall algorithm of creating new sets of weights of the neural network. The result of this algorithm is a binary string, which is converted back into the neural network weights. In case where there is no previous data, the weights are populated randomly.

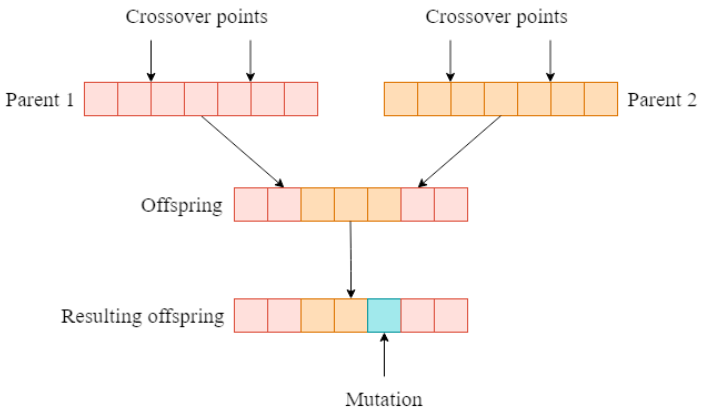

Fig. 2. The algorithm of creating new sets.

After every fight, the current weights with statistic information are saved. Before the new game fight starts, the statistic information is reset and a new set of weights is created. It is necessary to make several fights to get the weights with the grade close to zero. Over time, the weights with the highest grades are displaced by these sets. This process is connected with genetic algorithms, where new sets of weights are received by selection, crossover and mutation. For each fight, only one set is generated, that can be evaluated at one fight.

3.5. The component of catalogues

To provide better adaptation to rapid change of player skill level, the proposed approach uses a system of catalogues. For every game, the system creates the catalogue with information about the training of the neural network. The information about the current weights of the neural network is saved at predefined intervals. If the player skill level changes rapidly, the system can tune the level of difficulty by loading the appropriate catalogue. Furthermore, the player can change the level of difficulty himself, when it is necessary. This idea works while the player continues playing the same game instance. If he wants to start a new game, then a new catalogue is created.

Over time, a base of catalogues with different sets of the neural network weights is formed. In this base each set is matched to unique tactics (from weak to strong). It allows to collect the statistics of popular actions and weapons to use this information in the future. Besides, it creates possibility to analyze common types of players. The base of catalogues can be useful for researching which initial weights of the neural network evolve faster to achieve better adaptation.

4. Result

In this section all benefits and disadvantages of new approach are described based on the proposed model from the previous section. Moreover, this section shows how the model meets the requirements.

The first requirement is that the game must identify and adapt itself to the human player's initial level as quickly as possible. The proposed approach provides identification and adaptation to the player's skill level over time. This method is fast due to it uses simple topology of the neural network and modern algorithms of evolving its weights.

The second requirement stands for tracking the evolutions and regressions in the player's performance as close and as fast as possible. The proposed approach achieves it by using the neural system with input parameters directly connected with information of the current state of the player's character. This information is gathered after every fight and describes the actual player's skill level. Moreover, the behavior and parameters of the agent on the next fight is changed in accordance with this information. Besides, the system allows to track the evolution of the player's skill level by using the system of catalogues. They contain the information about the progression of agents, and it can be matched to the player performance.

The last requirement is that the agent's behavior must remain believable. In the proposed approach, the neural network with genetic algorithm provides constant agent's evolution and it is contributing to believable behavior because the agent learns with the player. In this way the agent's behavior cannot become unbelievably smart or dull. The system of catalogues helps to maintain this behavior by switching difficulty level when it is necessary.

Next step is a comparison of the proposed approach with other methods. The main advantage of this approach that it can react to drastic changes in the player skill progress by adjusting the difficulty level. It is achieved by the system ability to learn online in compared to the other methods. The proposed approach, unlike the dynamic scripting model, does not have any predefined rules. This allows to decrease the development time and to save additional resources. Moreover, the neural network with genetic algorithm generates diverse tactics for the agent than the dynamic scripts.

Skilled Experience Catalogue requires an additional series of training to fill the experience catalogue, therefore it needs some additional time to prepares the model. The described system with catalogues provides the fast adaptation without using any training phases. Besides, SEC needs more computational resources in contrast to the proposed model.

The proposed model does not require any special equipment in comparison with MPRL and EEG-triggered methods. For instance, MPRL requires Kinect device and EEG-triggered - Emotiv EPOC EEG headset.

As mentioned above, the base of catalogues is used to fix the neuroevolution problem of downgrading the difficulty level when it is necessary. But there are still some problems with the neuroevolution approach. First of all, the neural network in this model should be trained each time input and output parameters are changed. The other problem is that with online training of neural network the same result cannot be achieved on each game process.

5. Conclusion and future work

This paper provides the approach that aims to improve the game's difficulty adaptation in accordance with the player skill level. To achieve this, an architecture based on neural network, genetic algorithm and a base of catalogues was proposed. The neural network together with genetic algorithm provides an incremental evolution of agents controlled by artificial intelligence. The goal of this approach is to create the system of adaptation that can take into account any rapid changes in player skill level. As described in the previous section the base of catalogues solves this problem because it provides the possibility to tune the game difficulty level by saving the information about the training of the neural network in catalogues.

The main advantage of the proposed approach is that the system not only takes into account any changes of player skill level, but also it can simple downgrade the difficulty level if the player fails with it. Furthermore, the learning process of the neural network is almost online, and partly offline if the base of catalogues is filled with information. Besides, the proposed approach does not have any predefined rules and can be scalable, but neural network should be trained for each game individually.

In the future, this approach can be improved by applying other evolving algorithms [11] for neural networks instead genetic algorithms and by increasing the number of input parameters to add extra possible actions, for instance an attempt to dodge an attack. Moreover, the information from the base of catalogues can be used to collect the statistical data about players, for example, the most popular tactics and weapons, to apply this information in future behavior modeling.

Acknowledgements

This research is financially supported by The Russian Science Foundation, Agreement №17-71-30029 with co-financing of Bank Saint Petersburg.

References

[1] Stanley KO, Clune J, Lehman J, Miikkulainen R. Designing neural networks through neuroevolution. Nat Mach Intell 2018;1:24–35. doi:10.1038/s42256-018-0006-z.

[2] Andrade G, Ramalho G, Santana H, Corruble V. Extending Reinforcement Learning to Provide Dynamic Game Balancing 2005.

[3] Zohaib M. Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review. Adv Human-Computer Interact 2018;2018:1–12.

doi:10.1155/2018/5681652.

[4] Riwinoto, Miranto C, Muhaimin M. Implementation of Dynamic Scripting Method Using Top Culling in Game Arcade Into Space. Proc 2018 Int Conf Appl Eng ICAE 2018 2018:1–5. doi:10.1109/INCAE.2018.8579380.

[5] Glavin FG, Madden MG. Skilled Experience Catalogue : A Skill-Balancing Mechanism for Non-Player Characters using Reinforcement Learning. 2018 IEEE Conf Comput Intell Games 2018:1–8.

[6] Sekhavat YA. MPRL : Multiple-Periodic Reinforcement Learning for Difficulty Adjustment in Rehabilitation Games. 2017 IEEE 5th Int Conf Serious Games Appl Heal 2017:1–7. doi:10.1109/SeGAH.2017.7939260.

[7] Stein A, Yotam Y, Puzis R, Shani G, Taieb-Maimon M. EEG-triggered dynamic difficulty adjustment for multiplayer games. Entertain Comput 2018;25:14–25. doi:10.1016/J.ENTCOM.2017.11.003.

[8] Weeks M, Binnion D, Randall AC, Patel V. Adventure Game With A Neural Network Controlled Non-playing Character 2017. doi:10.1109/ICMLA.2017.0-129.

[9] Risi S, Togelius J. Neuroevolution in games: State of the art and open challenges. IEEE Trans Comput Intell AI Games 2015;PP:25–41. doi:10.1109/TCIAIG.2015.2494596.

[10] Yannakakis GN, Togelius J. Artificial Intelligence and Games 2018. doi:10.1007/978-3-319-63519-4.

[11] Yang X, Deng S, Ji M, Zhao J, Zheng W. Neural Network Evolving Algorithm Based on the Triplet Codon Encoding Method. Genes (Basel) 2018;9:626. doi:10.3390/genes9120626.