Hindawi
Advances in Human-Computer Interaction
Volume 2018, Article ID 5681652, 12 pages
https://doi.org/10.1155/2018/5681652

Review Article

Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review

Mohammad Zohaib

Department of Computer Science, BMS College of Engineering, Bangalore 560 019, Karnataka, India

Correspondence should be addressed to Mohammad Zohaib; zohaib27may@gmail.com

Received 31 May 2018; Accepted 14 October 2018; Published 1 November 2018

Academic Editor: Hideyuki Nakanishi

Dynamic difficulty adjustment (DDA) is a method of automatically modifying a game's features, behaviors, and scenarios in real-time, depending on the player's skill, so that the player, when the game is very simple, does not feel bored or frustrated, when it is very difficult. The intent of the DDA is to keep the player engrossed till the end and to provide him/her with a challenging experience. In traditional games, difficulty levels increase linearly or stepwise during the course of the game. The features such as frequency, starting levels, or rates can be set only at the beginning of the game by choosing a level of difficulty. This can, however, result in a negative experience for players as they try to map a predecided learning curve. DDA attempts to solve this problem by presenting a customized solution for the gamers. This paper provides a review of the current approaches to DDA.

1. Introduction

The concept of the video game is continuously changing. The early games like Computer Space and Pong of the early seventies were limited to commercial arcades, but now they are seen in multiple platforms such as cell phones, tablets, computers, and other devices. People are spending in excess of 3 billion hours weekly on gaming [1], which goes to show the extent of change it has brought to our lives.

Entertainment is but one aspect; games are now moving into reality, and the invisible boundaries separating games and reality are now becoming increasingly obscure [2]. Video games now extend to realms of healthcare [3] and education [4]. Experts have studied methods to assess how playing video games affect motor learning and its scope of improving patient involvement with therapy, especially commercial games which could be linked with specialized controls [5].

Although technology in gaming continues to evolve, a general discontent of players with the existing games has been observed due to their limitations in offering challenge levels to suit individual traits of the player like dexterity, learning and adapting ability, and emotional characteristics [6, 7]. Static levels of difficulty that are selected manually can no longer avoid boredom in players as they, in all probability, would be unable to decide on the challenge level that matches their abilities [8]. Also, constantly calling out the players to select the difficulty levels could distract them and interrupt the game [9]. The fun factor in games depends on three factors: challenge, fantasy, and curiosity [10]. Creating an adequate level of challenge is not easy when players with varying skills are pitted against each other. When an opponent is beaten effortlessly, the game appears boring. Again, in the face of a vastly superior opponent, the game turns frustrating. These two extremes lessen fun, since an optimal challenge is not offered. Csikszentmihalyi [11] first proposed that players, when kept away from the states of boredom or frustration, travel through a "flow channel" (Figure 1) and this was incorporated into a gaming scenario by Koster [8].

This model indicates how the difficulty of a task directly relates to the performer's perception. The flow channel shows that the difficulty level can be gradually enhanced, as sufficient time exists for the players for learning and improvement to meet this challenge [11]. Thus, the model prevents the frustration of difficult situations and the boredom of simple ones. In a different study, Malone [10] suggested that if the fantasy, challenge, curiosity, and control in games could be

Table 1: DDA research studies since 2009.

Year	Journal Papers	Conference Papers	Theses	Books	Total
2009	1	2	1		4
2010	1	7	1		9
2011	2	5		1	8
2012	3	8	2		13
2013	2	5	3	1	11
2014	1	4	1	1	7
2015	1	5	1		7
2016	3	5	3		11
2017	5	7	2		14
2018	0	0	0	0	0

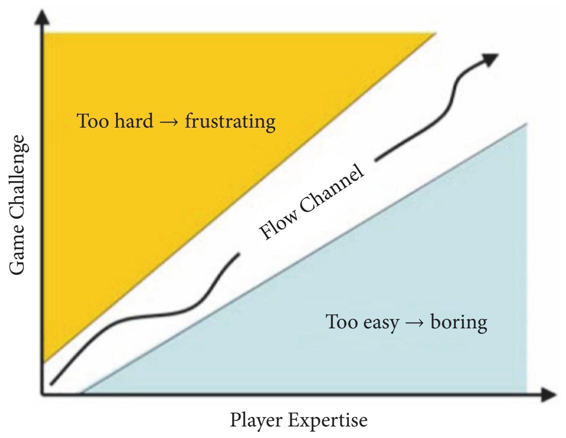

Figure 1: Flow channel concept proposed by Csikszentmihalyi.

balanced and associated with the gradual enhancement of difficulty level stated earlier, it could be possible that the ensuing game could keep the player entertained. Peeters [12] in her study suggested designing an automated platform for scenario-based training so that learners could engage in personalized autonomous training where agent-based notions such as beliefs, desires, and intentions can be used to deal with the gamer's competency and skills.

Numerous studies have been addressing the problems of static levels and have proposed the dynamic difficulty adjustment (DDA) technique that allows the automatic mapping of playing experience with the individual skills. DDA is a technique of automatic real-time adjustment of scenarios, parameters, and behaviors in video games, which follows the player's skill and keeps them from boredom (when the game is too easy) or frustration (when the game is too difficult). The essence of the DDA is to retain the interest of the user throughout the game and to offer a satisfactory challenge level for the player [13]. Andrade et al. suggested that DDA must cater the following three basic needs of games [14]:

(1) The game needs to automatically track the player ability and rapidly adapt to it

(2) The game must follow the player's improving or falling level and maintain a balance in accordance with the player's skill

(3) The process of adaptation must not be clearly perceived by the players, and successive game states need to have coherence with the earlier ones

Before applying the DDA, an understanding of the term "difficulty" is necessary. Though abstract, certain aspects need to be considered to assess and measure difficulty. Some of them are characteristics of design [15], number of resources [16], number of losses or victories [17], and so on. Nevertheless, DDA is not as easy as merely giving a player some healthier items in times of trouble. It needs an estimate of time and an entry at the right instant, as keeping the player absorbed is complicated in an interactive sense [16].

2. DDA Studies in the Last Ten Years

After 2009, there have been many research studies related to methods to develop or improve DDAs, including innovative applications in diverse fields. It is notable that the number of research papers in 2012 and 2017 is almost three times the number of research papers presented in 2009 (Table 1).

In this study, we have focused on DDA research studies undertaken after 2009 and have presented the important categories observed over the last decade (2009–2018). Going by the data presented in Table 1, we observe that, in the last decade, there has been a significant increase in the number of research papers on DDA over the years, and it was the highest in 2017.

Figure 2 depicts the DDA research studies carried out in the last ten years, including journal and conference papers, thesis work, and book chapters for every year.

3. Classification of DDA Approaches

Various methods for DDA are proposed in the literature (Table 2). The one common aspect in all methods is a requirement to measure (in a manner that may be implicit or explicit) the level of difficulty being faced by the player at any given instant. These measures are estimated by heuristic functions, also called challenge functions.

They assign a value for any game state that is indicative of the difficulty level of the game felt by the player at any given moment. Typical examples of heuristics in use are success rates of hits, numbers of pieces won and lost, life points,

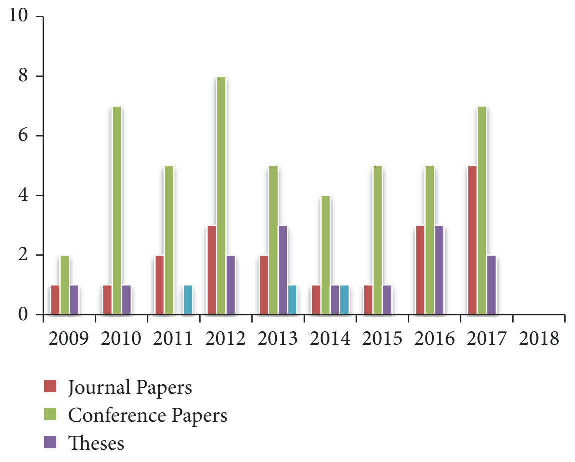

FIGURE 2: DDA studies over the past decade.

TABLE 2: List of DDA approaches.

Author(s)	Approach
Xue et al.	Probabilistic Methods
Pedersen, Togelius, and Yannakakis	Single and multi-layered perceptrons
Spronck et al.	Dynamic scripting
Hunicke and Chapman	Hamlet System
Hagelback and Johansson	Reinforcement Learning
Li et al.	Upper Confidence Bound for Trees and Artificial Neural Networks
Ebrahimi and Akbarzadeh-T	Self-organizing System and Artificial Neural Networks

completion time for assigned tasks, or any other metrics for calculating game scores.

There are several ways we can classify approaches to DDA.

3.1. Probabilistic Methods. A study on a framework that sees DDA as a problem of optimization was carried out [18]. This approach maximized player engagement all through the game. They modeled the progression of the player on a probabilistic graph (Figure 3) that maximized engagement as a well-defined objective function.

A dynamic programming technique having high efficiency was utilized to solve it. They assessed the DDA implementation using a mobile game by Electronic Arts, Inc. The group treated by DDA showed a clear increase of core engagement metrics, e.g., total number of plays and duration of game, while being revenue neutral when evaluated with the control group that did not have enabled DDA. This framework can be extended to a variety of game genres. DDA can be successfully applied to other genres if an appropriate progression model is constructed. The states for level-based games can be established by two important facets: trial and level. For games that are more complex having multiple or nonlinear progressions (e.g., role-play games), too, the states having varied dimensions can be defined. The graph would then be more complex since more states and links would be included.

Segundo et al. [19] proposed the creation of a parameter manipulating method for DDA, which aims to enhance the pleasure of gaming. The proposed method utilizes probabilistic calculations that could be deployed in a challenge function. A sample of students was provided with a questionnaire to assess whether a significant statistical difference existed in the understanding of game difficulty, game play, and the desire to play often with and without the method. The results indicated that the DDA version showed better results than the other versions with regard to game play and the desire to play often.

In a study [20], it was proposed that both online and offline learning techniques could be used for DDA. In the offline learning, a genetic algorithm was applied to create a fuzzy rulebase for game tactics during play to manipulate the computer-controlled adversaries. In the online learning, a probabilistic method was used for adapting the game strategies to the player. The level of difficulty of the game can be adjusted in accordance with the preference of the player seeking a challenge. The results demonstrated the superior capability of the evolved offline rulebases and the efficacy of the suggested online learning method for DDA.

Bunian et al. [21] developed a modeling technique by use of data gathered from players involved in a Role-Playing Game (RPG). The proposed technique has 2 features: (i) a player's Hidden Markov Model (HMM) tracking in-game traits for modelling individual differences and (ii) use of the HMM output to generate features of behaviors for classifying real-world characteristics of players that include expertise along with the big five personality traits. The results showed the prediction capability for some of personality traits, like conscientiousness and expertise. A logistic regression model was trained considering the composition of the freshly created behavioral features for 66 participants. A three-fold cross validation was used as the dataset was small. The prediction accuracy for conscientiousness and expertise category was 59.1% and 70.13%, respectively.

Bayesian optimization techniques were used in a study [22] to design games which maximize the engagement of users. Participants were paid to attempt a game for a short period, following which they could continue to play without payment or quit voluntarily. Engagement was measured by their persistence, estimates of duration of other players, and a survey after the game. Utilizing Gaussian surrogate-based process optimization, experiments were conducted to establish game design features, especially those affecting difficulty leading to maximum engagement. The converging outcomes indicated that overt difficulty manipulations were effectual in modifying engagement only with the covert manipulations, demonstrating the user's self-perception of skill as being critical.

Hintze, Olson, and Lehman [23] proposed the idea of orthogonal coevolution and verified its effectiveness in a game that was browser-based modified from a scientific simulation. The outcomes demonstrated that evolving adversaries together with evolved friends could lead to seamless DDA and permit gamers to experience more diverse

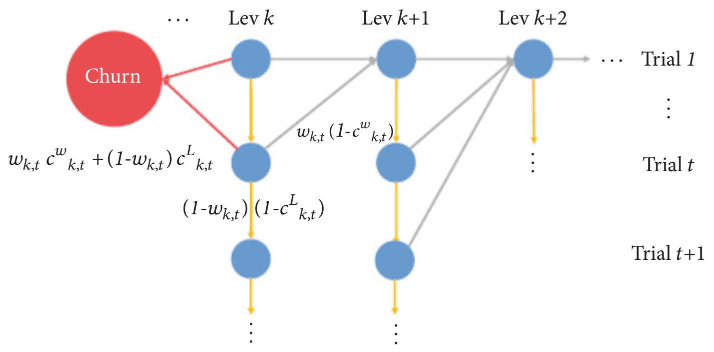
Lev k
Lev k+1
Lev k+2
Churn
Trial 1
Trial t
Trial t+1
$w_{k,t}\, c^{w}_{k,t} + (1-w_{k,t})\, c^{L}_{k,t}$
$w_{k,t}\,(1-c^{w}_{k,t})$
$(1-w_{k,t})\,(1-c^{L}_{k,t})$

FIGURE 3: Probabilistic graph showing the player's progression model in a typical level-based game.

situations. They concluded that such an orthogonal coevolution could be of promise for adjusting gaming difficulties.

3.2. Single and Multilayered Perceptrons. In a study by Pedersen, Togelius, and Yannakakis [24], the relationship between parameters of level design of platform games, player experience, and individual characteristics of play were studied. The studied design parameters had relation to the size and placement of level gaps and the presence of direction changes; and the constituents of a player's experience comprised frustration, fun, and challenge. A neural network model, which mapped between characteristics of playing behavior, design parameters of levels, and player emotions, was trained utilizing game session data and evolutionary preference learning.

Data was gathered from the Internet. Users were inducted through messages on mailing lists and blogs and sent to a web page which contained a Java applet initiating the game and a questionnaire. After playing the games and completing the questionnaire, all the characteristics (gameplay, controllable, and player experience) were recorded in a repository on a server. After analyzing this data, they attempted a function approximation based on gameplay and controllable characteristics to record emotional choices utilizing neuroevolutionary preference learning. This data representing the function was full of noise, since the choices of the players were highly subjective and the style of playing varied. All of this, coupled with the meager training data amount, suggests the usage of a function approximator that is robust. An artificial neural network (ANN), being a nonlinear function, is a suitable option for the approximation in mapping between data and reported emotions. Therefore, a simple single-neuron (perceptron) was used to learn the relationship between characteristics (ANN data input) and the analyzed emotional choice. The primary purpose for the use of a single neuron rather than a multilayered perceptron (MLP) here was that the trained function approximator needed to be analyzed. Though MLP can approximate the function more accurately, it is simpler for us to visualize the derived function when presented by a single-neuron ANN. Learning was obtained by artificial evolution by adopting the preference learning method [25]. A generational genetic algorithm was deployed, utilizing a goodness-of-fit function which measured the variation between the recorded emotional preferences and the corresponding model output. Results showed that there was high accuracy of prediction of challenge (77.77%), frustration (88.66%), and fun (69.18%) using a single-neuron model, which recommends using more elaborate nonlinear approximators. The study also discussed how the models generated could be used to generate game levels automatically, which would improve the player experience.

In another study, Shaker, Yannakakis, and Togelius [26] demonstrated the automatic generation of personalized levels for platform games. They built their model on the earlier work by Pedersen, Togelius, and Yannakakis [24]. At first, single layer perceptrons (SLP) were used to approximately evaluate the affective level of the players. The input subsets were selected by the sequential feature selection. To generate content customized to suit real-time player experience automatically in real-time, predicting emotions, to some extent, from controllable features is necessary. For this, the rest of the controllable features not already in the chosen feature subset were forcibly entered in the input of the multiple layer perceptron models, and the topology of the networks was made optimal for the highest accuracy of prediction.

In this study, dynamic adaptation to changes in playing styles was assessed. The model's capacity to generalize over players of various types was tested. To carry this out, two artificial intelligence (AI) agents were deployed for play in turns, while tracking the growth of the fun value. The experiment commenced from a level generated at random. The agents played 100 levels with an agent switch after every 20 levels. The result showing the variation in fun level across 100 levels is shown in Figure 4.

It is seen that the fun value is about 70% for the initial 20 levels when the first agent plays, increases to 80% when the next agent plays for 20 levels, and drops down to 70% when the first is brought back to the game. It clearly shows the model's capability to adjust to the player type. As a further test, the same trial was repeated on 4 human players in a reduced set of 12 levels. The result of this trial is illustrated in Figure 5, which shows the progress of fun over 48 levels. The results are similar to those obtained from the AI agents. It clearly indicates that the model robustly adapts to an individual player generalizing over various kinds of players.

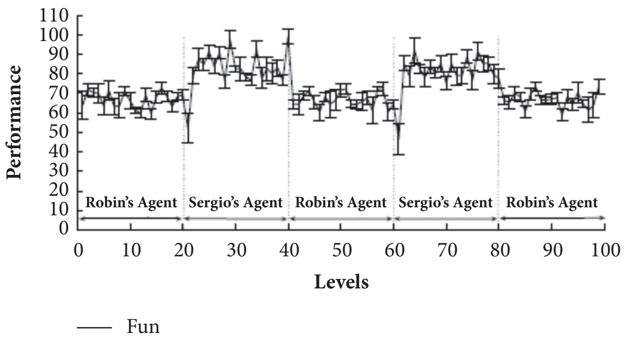

FIGURE 4: Two-agent optimized levels of fun.

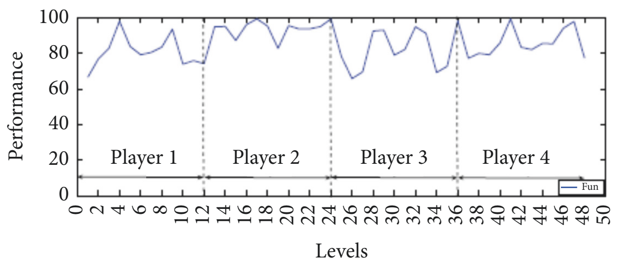

FIGURE 5: Four-player optimized levels of fun.

A study [27] constructed computational models of a player's experience derived from interaction in gameplay for use as fitness functions for content creation in games. A classic platform game's modified version was used for their experiments, and player data was collected from the Internet. They used the preference learning method for generating models of player experience. Feature selection was utilized to lower the features in the model. The data from training of nonlinear perceptrons was used to approximate the mapping functions between controllable features and selected gameplay. They presented the results of optimal construction of multilayer perceptrons (MLP) and the MLP model performances. They finally discussed the ways by which induced models could generate game content automatically.

Most DDA methods are based on the intuitions of designers, which do not reflect real-world playing patterns. Therefore, Jennings-Teats, Smith, and Wardrip-Fruin [28] created Polymorph that used methods from machine learning and level generation to analyze player skill and level difficulty, thereby dynamically creating levels in a 2D platformer game having continuously desired challenges. The DDA problem was addressed by generating a machine-learned difficulty model in a 2D platformer game using a model of the existing skill of the player. Multilayer Perceptrons accessed from play traces are used. These traces are gathered using a web-based tool which assigns users with various short-level components and rates them on a difficulty level. The Polymorph model utilizes the models of difficulty to choose the suitable level segment for the existing performance of the player.

Carvalho et al. [29] presented a method for generating gameplay sessions for endless games. This genre still remains largely unexplored in literature. The method uses a four-step process starting from the generation of required content to placing the content through the gameplay sessions. A robust evaluation technique was also designed. This technique utilizes both features that can be adjusted by a designer and gameplay items gathered from gameplay sessions. The usage of 2 neural networks is a new technique that agrees with the idea of game as a service and supports it throughout the life cycle of the game. The two neural networks have different purposes: the first receives merely features that are controllable as input, and the other receives both non-controllable and controllable features as input. Both the neural networks adjust chunk (fixed-size segments of the game on which gameplay elements are placed) difficulty as their output. Thus, the first network is used in the initial development stages, where one has access to only controllable features of these chunks, and the other is used to periodically adjust the game once it is made available. Both neural networks are Multilayer Perceptrons, each having a hidden layer.

3.3. Dynamic Scripting.

3.3. Dynamic Scripting. Dynamic scripting is an online unsupervised learning approach for games. It is computationally rapid, robust, efficient, and effective [30]. It operates many rulebases in the game, running one for every opponent type. These rules are designed manually utilizing domain-specific information. With the creation of a new opponent, the rules that constitute the script guiding the opponents are taken from the rulebase based on their type. The probability of a script rule selection depends on the weight value allotted to the rule. The rulebase adjusts by amending the values, reflecting the rates of failure or success of the related script rules.

In this approach, learning takes place progressively. On completing an encounter, the rule weights used in the encounter are treated depending on their effect on the result. The rules leading to success have their weights increased, while those leading to failure have their weights decreased. The remaining rules are adjusted accordingly so that the sum of all the rulebase weights remains constant. Dynamic scripting is used to generate fresh opponent tactics while increasing the level of difficulty of the game's AI to match the level of experience of the human player (Figure 6).

There are three different enhancements to this technique allowing the opponents to learn playing a balanced game:

(1) High-fitness penalizing: The weight balancing provides rewards in proportion to the fitness value. To obtain mediocre rather than optimal behavior, the weights can be amended to reward mediocre values of fitness and punish superior values.

(2) Weight clipping: The maximum value of weight decides the highest optimization level that a learned tactic can reach. A high value for the maximum permits the weights to increase to high values, so that soon the most effective rules will nearly always be chosen. This results in scripts having near to optimal values. Similarly, low values for the maximum hamper growth of weights. This creates a large variation in scripts generated, many of which would be nonoptimal. This method automatically varies the maximum value, thereby enforcing a balanced game.

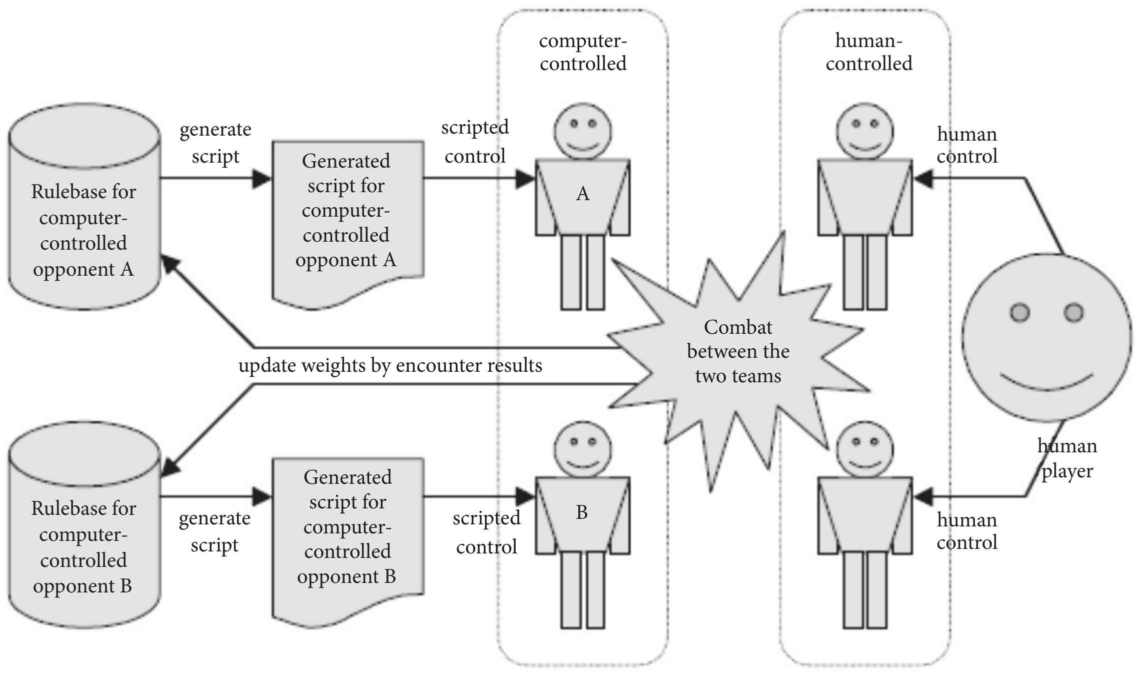
computer-controlled
human-controlled
Rulebase for computer-controlled opponent A
generate script
Generated script for computer-controlled opponent A
scripted control
A
human control
Combat between the two teams
update weights by encounter results
human player
Rulebase for computer-controlled opponent B
generate script
Generated script for computer-controlled opponent B
scripted control
B
human control

Figure 6: The dynamic scripting process.

(3) Top culling: Similar to weight clipping, it uses a similar mechanism for adaptation for the maximum value, the difference being that, here, weights are allowed to grow above the maximum value. However, rules having weights in excess of the maximum value do not get chosen for a generated script. As a result, frequent wins of computer-controlled opponents cause effective rules to be rejected, making opponents use weak tactics. Conversely, frequent losses would cause rules with high weights to become selectable, making opponents use weak tactics.

Experiments conducted by the authors showed that DDA by dynamic scripting is effective. It was also seen that the three approaches were tested; high-fitness penalizing was not successful, but the two other approaches did well.

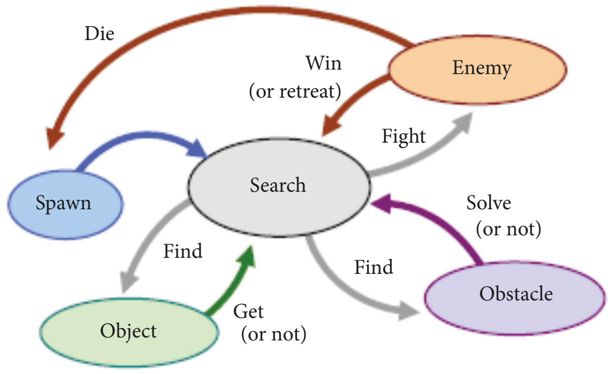

Figure 7: A simple FPS game state transition illustration.

3.4. Hamlet System. Most games use the concept of inventory, i.e., the store of items a player gathers and takes all through the game. The relative ampleness or lack of items in inventory directly impacts the experience of the players. Games are designed to control the exchange of items between the player and the world [31].

These maps of producer-consumer links can be seen as an economy or a dynamic system. Hamlet, a DDA system built by Hunicke and Chapman [13], uses methods taken from Operations Research and Inventory Management. It studies and adjusts supply and demand of the inventory in the game so as to manipulate the game difficulty. The system is essentially a group of libraries maintained in the engine of Half Life. Hamlet has functions in the following:

(1) Managing game statistics in accordance with statistical metrics defined in advance

(2) Deciding adjustment tasks and rules

(3) Carrying out those tasks and rules

(4) Presenting data and system settings

(5) Creating traces for playing rounds

Hamlet uses metrics for monitoring incoming game information as the players advance through the game world. It predicts the future state of the players from this information. Whenever an unwanted but preventable state is predicted, Hamlet steps in and tweaks game settings as required. Essentially, it tries to anticipate when the player is struggling repeatedly and nearing a state where his existing resources can no longer meet requirements. When this struggle is detected, Hamlet intervenes to assist the player to continue the game.

Keeping up a player's challenge and interest level is not an easy task. One popular approach is the flow model proposed by Csikszentmihalyi [11]. In an FPS environment (Figure 7), the gameplay can be illustrated with a fairly simple state transition picture.

Players engage in cycles of seeking, fetching, solving, and battle. Each new level creates new foes and hurdles. Difficulty levels and skill get enhanced with time. Hamlet is tailored to keep players within the Csikszentmihalyi's flow channel by promoting some states and demoting others. The basic aim is to keep the players in engaging loops of interaction for durations that are most appropriate based on their skill and experience gathered. To sum up, Hamlet looks to

(1) Evaluate when adjustments are required
(2) Decide on the changes
(3) Make changes seamlessly

When a player is struggling, in many FPS games, it is observed that constant inventory shortfalls occur in locations where the player's existing resources do not meet the required demands. By noting trends in the inventory expenses of players, probable shortfalls are looked for, thereby identifying probable opportunities for adjustment. The evaluation process begins with the establishing of metrics to assess data. The damage data, based on its probability distribution, are analyzed. Inventory theory equations provide a basis for modeling the player's overall inventory and flow. Shortfalls are predicted based on total damage probabilities. Hamlet accordingly takes reactive and proactive action by making adjustments. Adjustment protocols are defined in the Hamlet system. Adjustment actions, together with cost estimations, form adjustment policies.

3.5. Reinforcement Learning. Games are played by a variety of players who use varied gaming patterns and strategies. Hence, a static game AI cannot deal with the gaming styles of all kinds of gamers. A game AI that is adaptive, therefore, can create varied gaming experiences for different playing styles and thus add interest and repeatability of play to a game. Such mechanisms have been studied with interest in recent years. For example, evolutionary algorithms by Togelius et al. [32] were used to create racing tracks that were popular with players.

Hagelback and Johansson [33] in a study observed that players enjoy playing an evenly matched game against opponents who adapt to their styles. To this end, Tan, Tan, and Tay [34] developed an adaptive AI for games that promotes even play rather than beating opponents. Here, a dynamic computer controlled opponent adapts its behavior, according to its opponent's moves. This DDA technique uses adaptive AI in the game to adjust game behaviors and parameters in real time automatically in response to the skill of the player. It can keep the player engrossed for longer periods and enhance their experience.

As mentioned, here, DDA is carried out in real time. The adaptive AI of the game requires being adept enough to make unforeseeable but rational judgments like human players but must not display the overtly thoughtless behavior. The AI also needs to be able to correctly assess its opponent in the beginning of the game itself and adjust its playing style to opponent's skill. This study proposed two adaptive algorithms, adaptive unichromosome controller (AUC) and the adaptive duochromosome controller (ADC) that utilized concepts from evolutionary computation and reinforcement learning [34] to adaptively play in real-time. Two metrics, winning percentage difference ($|W-L|$ and D to be minimized, where W, L, and D are wins, losses, and draws) and mean score difference ($|s1-s2|$ to be minimized and max(s1,s2), where s denotes scores of players 1 and 2), were used as indicators of entertainment value. First, the game is so designed that the AI has the capability to beat the player. Second, the game AI can make deliberate mistakes, termed as artificial stupidity; therefore, the players remain interested in the game.

The training and adaptation of the AUC take place in real time when the game is in session. As its name suggests, AUC stores a single chromosome that maps to seven numbers, one corresponding to each behavior component. Each number indicates the probability of deploying a behavior component when a waypoint is crossed. The expected behavior mapped by this chromosome exemplifies a victory strategy. The chromosome tailors the proficiency of the opponent by mapping a behavior set which would be sufficient to beat him. The chromosome is initialized randomly at the beginning of every game. Whenever a waypoint is crossed, a set of rules updates the chromosome. The assumption here is that the complement of an expected victorious strategy is a losing one. The ADC and AUC are similar except that the former does not assume that the complement of an expected victorious strategy is a losing one. Instead, two sets of chromosomes are maintained, one winning and one losing chromosome, all through the game. The chromosomes are updated by different rules for wins and losses.

These controllers were tested against static controllers of varied driving traits to simulate various styles of play like heuristic controllers, neural network controllers, reverse enabled controllers, predictive fast controllers, etc. The effects of changing the mutation and learning rate were studied for both controllers (algorithms). The pattern of the difference of scores was assessed and both achieved score differences of 4 or less in at least 70.22% of the games. Wins and losses were also well scattered across the sequence of games played consecutively. It was also seen that the AUC had a low memory footprint, and the ADC was capable of maintaining a lesser number of drawn games, which could keep the player interested. The final values of the chromosomes showed that the algorithms choose various combinations of behavior components to deal with different opponents. Both controllers were able to learn proper sets of behavior components for the various opponents by way of winning percentage and mean score. Also, both were able to generalize satisfactorily to different opponents.

Sekhavat [35] suggested a personalized DDA method for a rehab game that manages difficulty settings automatically, based on a real-time patient's skills. Concepts of reinforcement learning were used as a DDA technique. It was shown that DDA has multiple objectives, in which objectives could be evaluated at different times. To solve this issue, it was proposed to use Multiple-Periodic Reinforcement Learning (MPRL) which enables the evaluation of various objectives of DDA in separate time periods. The experiments showed that MPRL performed better than available Multiple-Objective Reinforcement Learning methods in user satisfaction and enhancing the patient motor skills.

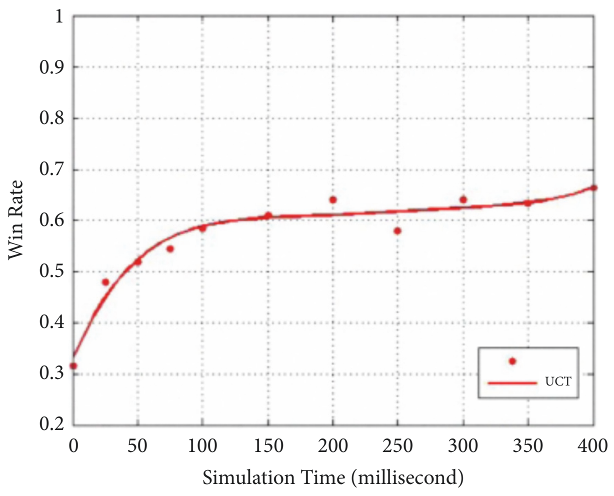

Figure 8: DDA from UST.

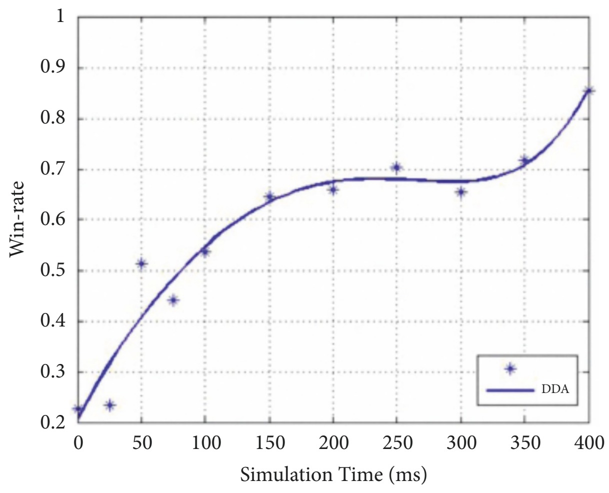

Figure 9: DDA from ANN.

3.6. Upper Confidence Bound for Trees and Artificial Neural Networks. Li et al. [36] developed a DDA technique using artificial neural networks (ANN) from data derived from the upper confidence bound for trees (UCT). The Pacman game was used as a test-bed for this study. Considering that UCT is a computing intelligence method, UCT performance significantly correlates with the duration of simulation [37]. Figure 8 illustrates DDA process from UCT-created data.

Here, the x-axis denotes simulation time, which is in the range 0–400 ms. The y-axis denotes win-rates of opponents (ghosts) which are in the range 30%–70%. The curve rises steeply in the period 0–100 ms; this period has a higher number of test data. After 100 ms, the curve flattens. The win-rate attains a maximum at 400ms. The reason for the stability of the win-rate is because of UCT being a stochastic simulation approach. In the interval 0–100 ms, the sample space is bigger and the stochastic outcomes become more accurate. Hence, the UCT performance is drastically improved. Also, after crossing 100 ms (threshold value), the accuracy of results is still fairly good so that the win-rate grows smoothly even at higher values of simulation time. UCT can be used as DDA in real-time games, too. By merely adjusting the UCT simulation time, we obtain game opponents of increasing difficulty levels.

ANN can be trained from UCT-created data. Even though the UCT approach can be deployed as DDA for games like Pacman, it is not practical to be used for complex online games because of UCT's computational intensiveness. But then since UCT's performance can be tweaked by varying the simulation time, ANN offline training becomes possible by running the UCT-created data with changed simulation times. Thus, DDA can be generated from UCT-created data, bypassing the computational intensiveness. In this study, the 3-Layered Feed-Forward Artificial Neural Network model in WEKA was used for the implementation.

DDA can also be created from ANN. The weights and bias of ANN are reserved in MDB files. Opponents are managed by ANN by loading MDB files.

Figure 9 illustrates the DDA from ANN. The x-axis ranging within 0–40ms is the same as Figure 7. The y-axis represents win-rate of opponents (ghosts) controlled by ANN from data created by UCT with changed simulation time ranging within 20%–86%. The curve rises steeply in the range 0–100ms. After 100ms, the curve rises steadily and the win-rate peaks at 400ms.

The performance of the opponent's neural network depends on the training data quality. With insufficient incidences for a certain route, the ANN training remains poor. With ample incidences for all routes, the trained ANN performs well. With the increasing growth of simulation time, the UCT data achieve greater precision, which in turn creates ANN that is better trained. Comparing the two curves, we note that the DDA curve tends to rise from a minimum to a maximum simulation time. Hence, a valid DDA curve can be derived by ANN training from UCT based data. Thus, UCT is a good computation intelligence algorithm which performs better when the simulation time increases. It can therefore be used as a DDA tool by tweaking the simulation time. UCT can also create data to train ANN.

A data-driven approach for DDA was proposed by Yin et al. [38]. The objective was to match the player's performance to the required conditions laid down by the designer. The data pertaining to dynamic game states and in-game player performance were used for taking decisions on adaptation. Trained ANNs were utilized to map the relationship between player performance, dynamic game state, adaptation decisions, and the game difficulty that resulted. The predicted difficulty enables effective adaptation of both magnitude and direction. An experiment on a training game application demonstrated the efficacy and stability of the suggested approach.

3.7. Self-Organizing System and Artificial Neural Networks. In another study [39], a self-organizing system (SOS) was developed, which is a group of entities that presents global system traits through local interactions while not having centralized control. This method proposes a new technique that tries to adjust the difficulty level by creating an SOS of

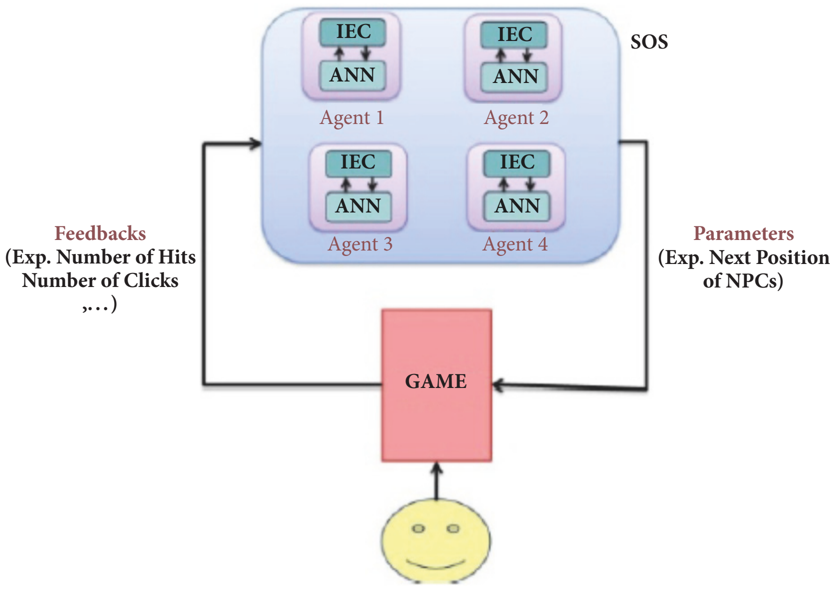

Figure 10: Illustration of self-organizing algorithm.

Non-Player Characters (NPCs) that are not in the player's control. To track human player traits, ANNs are used in the system. ANNs need to adapt to players having varied levels of skills and traits; therefore an evolutionary algorithm having adaptation skills was developed which modifies ANN weights (Figure 10).

Pacman game has been used as a test bed. There are two agents in the game: Pacman (the player) and ghost (opponent). The authors have considered four types of Pacmans having varying intelligence levels. The first is Cost-Based Pacman, a local agent who locates his subsequent location based on the position of his nearby ghosts. The second, named Distance-Based Pacman, is a global agent, who considers locations of all ghosts prior to deciding the next move. The third, neither fully global nor local, is named Nearest Distance-Based Pacman. The three Pacmans represent players having varying levels of skill. The fourth, Random Pacman, moves randomly and is not classified in any of these categories.

A neuroevolutionary controller for each Ghost was developed in order that they adapt to the different skill levels of the players. They decide the next position based on environmental precepts. Every ghost was given a feedforward neural network having a concealed layer. The topologies of the networks were decided during the game. Ghosts are first trained offline. This offline training helps to create chromosomes that perform better than random chromosomes. The Cooperative Coevolution Algorithm is used to train ghosts offline [40]. A subpopulation of chromosomes is considered for every ghost, which have neural network connection weights. The best performers in every subgroup are chosen as representatives. Next, to assess chromosomes of each subgroup, a game is arranged between the representatives and these chromosomes. On completion of the game, its fitness is assessed and assigned. This process is repeated till all the chromosomes of every ghost are mapped. After assessment of all chromosomes in the subgroups is carried out, the genetic algorithm selects, crosses over, and mutates for producing new chromosomes. When the stop requirements are met, this sequence of events halts.

Next, online learning of the controllers takes place. Before the game starts, the required chromosomes are loaded in the ghost controllers. As the skill levels of the players are still unknown, intermediate chromosomes are selected for all ghosts. The subgroups are sorted on the basis of individual fitness; the median chromosomes are selected. The game begins when the chromosomes are loaded in the neural networks. The game runs for a short duration after which the system fitness is assessed. Neural controllers are trained using the Interactive Evolutionary Computation (IEC) where the fitness function replaces human evaluation [41]. As human evaluation results in fatigue, IEC optimizes systems effectively. System fitness is indirectly assessed using player feedback, e.g., number of keys pressed, occasions of key switches, and Pacman's wall hits. After each duration, these numbers are analyzed to assess fitness.

The results show that this system is capable of adapting to many skill levels by selecting proper factors that hasten convergence to the optimal requirements.

A study [42] explored the use of NEAT and rtNEAT neuroevolution techniques to create intelligent adversaries for games having real-time strategies. The primary objective is to convert the challenge created by the adversaries to match the recompetence of the player in a real-time situation, thereby resulting in a greater entertainment value experienced by the player. The study introduced the application of the neuroevolution techniques to Globulation 2 (G2), real-time strategy game for DDA. Initially, NEAT was used to optimize the functioning of G2 nonplayer characters and two suggested challenge factors were investigated by offline trials in G2. The results indicated that the aggressiveness factors and warrior numbers are contributors to challenge because neuroevolved agents obtained succeeded in outperforming all typical AI nonplayer characters available for playing the game.

3.8. Affective Modeling Using EEG. Earlier researchers studied heuristic approaches based on a game state. Generally, in a game that tracks an ongoing score, decisions on DDA application can be taken when the difference between the scores of the players exceeds a threshold value, i.e., when one player becomes stronger than the other, and assuming that this would result in boredom in the stronger and frustration in the weaker player.

Stein et al. [43] have proposed a different method—measuring the excitement of players and setting in motion the game levels when the level of excitement dips below a threshold value. They have attempted to address the main issue of gaming experience directly, rather than depending on heuristic scores to decide when they are bored with the game.

An affective-state regulation technique was implemented by using headsets to decipher electroencephalography (EEG) signals and the mechanism to modify the signal to an affective state.

Next, assessing this affective state, DDA is deployed by the game. Two studies were conducted. In the first, the relationship between the EEG signals and game events (GE) was investigated. The results showed a significant correlation between the indicator for short term excitement (STE) and GE. Playing experiences were attempted to be enhanced by maximizing STE. In the second study, this EEG-initiated DDA was compared to (1) a typical heuristic technique which used elapsed duration and game status and (2) a control game without DDA. A case study was presented for the EEG-initiated DDA approached in a customized version of the Boot Camp game. The study confirmed that (1) players preferred the EEG-initiated DDA to the two other choices and (2) this method greatly increased the excitement level of the players. The study also indicated that the option of the initiating strategy is significant and greatly impacts experience of the players.

Afergan, Mikami, and Kondo [44] used functional near-infrared spectroscopy for collecting passive brain-sensing information and detecting long durations of overload or boredom. Using these physiological signals, a simulation was adapted for optimizing real-time workload that permits the system to adjust the task for the user at each moment in a better manner. To demonstrate this concept, they conducted laboratory experiments where participants, in a simulation, were assigned path planning for several unmanned aerial vehicles. The task difficulty was varied based on their state by the addition or deletion UAVs and they found that errors could be decreased by 35% over and above a baseline level. The results indicated that fNIRS brain sensing can be used to detect real-time task difficulty and an interface constructed that enhances user performance by DDA.

Fernandez et al. [45] adapted the levels of difficulty of a basic 2D platform game, working on and building levels automatically. The method proposed consisted of DDA and Rhythm-Group Theory, a procedural content development approach, along with attention levels gathered from EEG data. Trials were planned in a manner that players needed to perform 5 varied levels automatically created by their performances and EEG data collected through a biosensor during play. Results indicated that the method adapted successfully to the difficulty levels as per the status of the player. Additionally, the method calculated difficulty utilizing calculated real-time values to decide the level.

4. Future Work

There are many opportunities for higher research into DDA by creation of level structures. Researchers can go beyond the standard 2D platformer video game genre and apply the same DDA concepts to different genres. Also, more research is required in new search-based techniques for identifying optimal levels. Researching player models other than agent types could be another promising area for more research. As the fitness function is a crucial element in game design, increasing its complexity by the addition of more variables that could consider many other aspects of play is yet another promising area.

An interesting research could be to investigate the possibility of covering traits like playing style. The concept of mapping the human player and developing a player model accordingly is yet another possibility. A player model that includes more behavioral aspects could yield interesting observations.

Many player modeling techniques exist currently. Integrating a few of these with present DDA approaches could open up some possibilities of interest and yield more DDA techniques tailored to a gamer’s preference.

5. Conclusions

DDA techniques have been proven in the literature to be useful tools for incorporation in complex and dynamic systems. This investigation has presented a review on DDA applications and directions in many diverse kinds of games in the past decade highlighting some of the most representative types for every application. There are numerous application studies of DDAs in various domains, including generalizations and extensions of DDAs. The number of approaches presented here is neither complete nor exhaustive but merely a sample that demonstrates the usefulness and possible applications of AI techniques in modern video games.

Conflicts of Interest

The author declares that they have no conflicts of interest.

References

[1] J. McGonigal, *Reality Is Broken: Why Games Make Us Better and How They Can Change The World*, Vintage, London, UK, 2012.

[2] G. Calleja, "Digital games and escapism," *Games and Culture*, vol. 5, no. 4, pp. 335–353, 2010.

[3] A. DeSmet, D. Thompson, T. Baranowski, A. Palmeira, M. Verloigne, and I. De Bourdeaudhuij, "Is participatory design associated with the effectiveness of serious digital games for healthy lifestyle promotion? A meta-analysis," *Journal of Medical Internet Research*, vol. 18, no. 4, 2016.

[4] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers & Education*, vol. 59, no. 2, pp. 661–686, 2012.

[5] K. Lohse, N. Shirzad, A. Verster, N. Hodges, and H. F. Van der Loos, "Video Games and Rehabilitation," *Journal of Neurologic Physical Therapy*, vol. 37, no. 4, pp. 166–175, 2013.

[6] P. Sweetser and P. Wyeth, "GameFlow," *Computers in Entertainment*, vol. 3, no. 3, 2005.

[7] K. M. Gilleade, A. Dix, and J. Allanson, "Affective videogames and modes of affective gaming: Assist me, challenge me, emote me," in *Proceedings of the 2nd International Conference on Digital Games Research Association: Changing Views: Worlds in Play (DiGRA '05)*, 20, 16 pages, Vancouver, Canada, June 2005.

[8] R. Koster, *A theory of fun for game design. Sebastopol (Cali.)*, OReilly Media, 2014.

[9] J. Chen, "Flow in games (and everything else)," *Communications of the ACM*, vol. 50, no. 4, pp. 31–34, 2007.

[10] T. W. Malone, "Toward a theory of intrinsically motivating instruction," *Cognitive Science*, vol. 5, no. 4, pp. 333–369, 1981.

[11] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, Harper Row, New York, NY, USA, 2009.

[12] M. M. M. Peeters, *Personalized Educational Games-Developing agent-supported scenario-based training [Ph.D. thesis]*, The Dutch Graduate School for Information and Knowledge Systems, 2014.

[13] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Proceedings of the Challenges in Game Artificial Intelligence AAAI Workshop*, pp. 91–96, San Jose, Calif, USA, 2004.

[14] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 12, 7 pages, Edinburgh, United Kingdom, 2005.

[15] R. A. Bartle, *Designing Virtual Worlds*, New Riders, Berkeley, Calif, USA, 2006.

[16] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE '05)*, pp. 429–433, Valencia, Spain, June 2005.

[17] S. Poole, *Trigger Happy Videogames and The Entertainment Revolution*, Arcade Publishing, New York, NY, USA, 2007.

[18] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games," in *Proceedings of the the 26th International Conference*, pp. 465–471, Perth, Australia, April 2017.

[19] C. V. Segundo, K. Emerson, A. Calixto, and R. P. Gusmao, "Dynamic difficulty adjustment through parameter manipulation for Space Shooter game," in *Proceedings of SB Games*, Brazil, 2016.

[20] H. Hsieh, *Generation of Adaptive Opponents for a Predator-Prey Game*, Asia University, 2008.

[21] S. Bunian, A. Canossa, R. Colvin, and M. S. El-Nasr, "Modeling individual differences in game behavior using HMM," in *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*, 2017.

[22] M. M. Khajah, B. D. Roads, R. V. Lindsey, Y.-E. Liu, and M. C. Mozer, "Designing engaging games using Bayesian optimization," in *Proceedings of the 34th Annual Conference on Human Factors in Computing Systems, CHI 2016*, pp. 5571–5582, San Jose, Calif, USA, May 2016.

[23] A. Hintze, R. S. Olson, and J. Lehman, "Orthogonally evolved AI to improve difficulty adjustment in video games," in *European Conference on the Applications of Evolutionary Computation*, vol. 9597 of *Lecture Notes in Computer Science*, pp. 525–540, Springer International Publishing, Cham, Switzerland, 2016.

[24] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in Super Mario Bros," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 132–139, Milano, Italy, September 2009.

[25] G. N. Yannakakis and J. Hallam, "Game and player feature selection for entertainment capture," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games*, pp. 244–251, Honolulu, Hawaii, USA, April 2007.

[26] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, pp. 63–68, Stanford, Calif, USA, October 2010.

[27] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 54–67, 2010.

[28] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: A model for dynamic level generation," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*, pp. 138–143, Stanford, Calif, USA, October 2010.

[29] L. V. Carvalho, A. V. M. Moreira, V. V. Filho, M. Túlio, C. F. Albuquerque, and G. L. Ramalho, "A Generic Framework for Procedural Generation of Gameplay Sessions," in *Proceedings of the SB Games 2013, XII SB Games*, São Paulo, Brazil, 2013.

[30] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent AI with dynamic scripting," *International Journal of Intelligent Games & Simulation*, vol. 3, no. 1, pp. 45–53, 2004.

[31] Z. Simpson, "The In-game Economics of Ultima Online," in *Proceedings of the Game Developers Conference*, San Jose, Calif, USA, 2000.

[32] J. Togelius, R. DeNardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in *Proceedings of the Workshop Adaptive Approaches Optim. Player Satisfaction Comput. Phys. Games*, p. 70, 2006.

[33] J. Hagelback and S. J. Johansson, "Measuring player experience on runtime dynamic difficulty scaling in an RTS game," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 46–52, Milano, Italy, September 2009.

[34] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 4, pp. 289–301, 2011.

[35] Y. A. Sekhavat, "MPRL: Multiple-Periodic Reinforcement Learning for difficulty adjustment in rehabilitation games,"

in *Proceedings of the 5th IEEE International Conference on Serious Games and Applications for Health, SeGAH 2017*, Perth, Australia, April 2017.

[36] X. Li, S. He, Y. Dong et al., "To create DDA by the approach of ANN from UCT-created data," in *Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, pp. V8-475–V8-478, Taiyuan, China, October 2010.

[37] J. Yang, Y. Gao, S. He et al., "To Create Intelligent Adaptive Game Opponent by Using Monte-Carlo for Tree Search," in *Proceedings of the 2009 Fifth International Conference on Natural Computation*, pp. 603–607, Tianjian, China, August 2009.

[38] H. Yin, L. Luo, W. Cai, Y.-S. Ong, and J. Zhong, "A data-driven approach for online adaptation of game difficulty," in *Proceedings of the 2015 IEEE Conference on Computational Intelligence and Games, CIG 2015*, pp. 146–153, Tainan, Taiwan, September 2015.

[39] A. Ebrahimi and M.-R. Akbarzadeh-T, "Dynamic difficulty adjustment in games by using an interactive self-organizing architecture," in *Proceedings of the 2014 Iranian Conference on Intelligent Systems, ICIS 2014*, Iran, February 2014.

[40] M. A. Potter and K. A. de Jong, "Cooperative coevolution: an architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[41] H. Takagi, "Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.

[42] J. K. Olesen, G. N. Yannakakis, and J. Hallam, "Real-time challenge balance in an RTS game using rtNEAT," in *Proceedings of the 2008 IEEE Symposium on Computational Intelligence and Games, CIG 2008*, pp. 87–94, Perth, Australia, December 2008.

[43] A. Stein, Y. Yotam, R. Puzis, G. Shani, and M. Taieb-Maimon, "EEG-triggered dynamic difficulty adjustment for multiplayer games," *Entertainment Computing*, vol. 25, pp. 14–25, 2018.

[44] D. Afergan, E. M. Peck, E. T. Solovey et al., "Dynamic difficulty using brain metrics of workload," in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, CHI 2014*, pp. 3797–3806, Toronto, Ontario, Canada, May 2014.

[45] H. D. B., K. Mikami, and K. Kondo, "Adaptable game experience based on player's performance and EEG," in *Proceedings of the 2017 Nicograph International (NicoInt)*, pp. 1–8, Kyoto, Japan, June 2017.