Adaptive Game AI-Based Dynamic Difficulty Scaling via the Symbiotic Game Agent

Siphesihle Philezwini Sithungu, Elizabeth Marie Ehlers

Adaptive Game AI-Based Dynamic Difficulty Scaling via the Symbiotic Game Agent

Siphesihle Philezwini Sithungu and Elizabeth Marie Ehlers

Academy of Computer Science and Software Engineering,
University of Johannesburg, Johannesburg, South Africa
{siphesihles,emehlers}@uj.ac.za

Abstract. This work presents AdaptiveSGA, a model for implementing Dynamic Difficulty Scaling through Adaptive Game AI via the Symbiotic Game Agent framework. The use of Dynamic Difficulty Balancing in modern computer games is useful when looking to improve the entertainment value of a game. Moreover, the Symbiotic Game Agent, as a framework, provides flexibility and robustness as a design principle for game agents. The work presented here leverages both the advantages of Adaptive Game AI and Symbiotic Game Agents to implement a robust, efficient and testable model for game difficulty scaling. The model is discussed in detail and is compared to the original Symbiotic Game Agent architecture. Finally, the paper describes how it was applied in simulated soccer. Finally, experimental results, which show that Dynamic Difficulty Balancing was achieved, are briefly analyzed.

Keywords: Dynamic Difficulty Balancing, Adaptive Game AI, Intelligent Agent Design, Symbiotic Game Agents.

1 Introduction

Players interact with computer games uniquely and are, therefore, most likely to pursue unique strategies in order to win. This means that non-player characters (NPC) experience each player differently [1]. This phenomenon is foundational to the realisation of Adaptive Game Artificial Intelligence (AGAI): game AI that can change its behaviour based on the player's behaviour [2]. AGAI has a chance of providing entertainment to players. This adds more value to a game than static game AI since the value of a computer game is directly related to its level of entertainment [3] [4].

Dynamic Difficulty Balancing (DDB), on the other hand, refers to the act of automatically adapting the challenge the game presents to the player depending on the player's proficiency [5] [6]. This technique may help accommodate players with different playing abilities [7]. In addition, DDB aims to achieve an *even game* between the game AI and the human player. An even game is one where the difference between the number of wins and losses for each agent is relatively small [2] [4]. Furthermore, research done by Hagelback and Johansson [8] has shown that players First Person Shooter games enjoyed an even game more than one static difficulty.

In order to achieve successful DDB, the following three requirements must be met [9]: (1) the game should be quick in classifying and adapting itself to the player's proficiency, (2) efficient in identifying improvements and lapses in the player's performance and (3) adapting itself in a believable way.

Game AI may be implemented in various ways as an NPC controller. A natural approach to implementing an AI-based NPC controller is the use of an intelligent agent [10]. An agent typically has three main components: (1) perception, (2) decision-making and (3) action. A game agent architecture essentially follows the same convention. Therefore, the choice of the intelligent agent architecture to be used is critical to the success of a game. This work proposes the use of the Symbiotic Game Agent (SGA) architecture to achieve DDB.

SGA is a special kind of multi-agent system based on biological symbiosis [11], which is the formation of persistent relationships among candidates of different species. Symbiosis can be observed across a broad spectrum of animal and plant life on earth and is more potent than *lateral gene transfer* – a transferal of traits that may occur between different species. This is because symbiotic relationships between two entities may result in a more genetically, biochemically and behaviorally complex organism [12].

The rest of the paper is organized as follows: Section 2 is the problem background. Section 3 provides a literature review of similar works. Section 4 presents the model - AdaptiveSGA. Section 5 presents the experiment setup. Finally, Section 6 discusses experimental results, and Section 7 concludes the paper.

2 Problem Background

The problem this works aims to solve is designing an SGA-based model for efficiently performing AGAI-based DDB. Game AI that is adaptive alleviates the shortcomings of traditional game AI by allowing for the creation of NPCs that can effectively react to changing situations in unpredictable ways [3]. An essential part of DDB is measuring how challenging the player is finding a game to be at a given moment. This can be done by defining a *challenge function* [9] that maps the player's performance to a suitable difficulty level.

It is important to note that AGAI is not a requirement for DDB. DDB can be achieved in various ways and has been implemented in games without the use of AGAI. One common approach is to continuously modify game parameters (e.g. opponent health, the time provided to complete a task, game speed, etc.) to change the difficulty of a game. Although this approach may work, it does not solve the problem of making game AI adaptive in terms of behaviour [2]. The work presented here aims to achieve DDB through AGAI.

DDB is a worthwhile goal to achieve because it supports the concept of designing *weak AI* (i.e. AI that behaves reasonably intelligent to players) for games, as opposed to *strong AI* (i.e. AI that always makes the correct, desirable or optimal decisions and potentially surpassing human cognition).

3 Literature Review

3.1 Dynamic Difficulty Scaling through Adaptive Game AI

Tan, Tan and Tay [1] proposed two algorithms to achieve DDB through adaptive, behaviour-based AI: Adaptive Uni-Chromosome Controller (AUC) and Adaptive Duo-Chromosome Controller (ADC). Both AUC and ADC were behaviour-based controllers and were defined in terms of 7 possible behaviour states which were encoded as a chromosome vector. Each element of the vector stored a real number indicating the probability of that behaviour being activated.

Experimental results showed that the AUC and ADC controllers both achieved DDB by maintaining an even game against static controllers through 5000 game instances for each experiment. Moreover, both algorithms were able to achieve a score difference of 4 or lower for 70.22% of the time [1].

Spronck, Sprinkhuizen-Kuyper and Postma [4] also proposed the use of AGAI to achieve DDB. Their method utilized dynamic scripting [3] to generate new opponent strategies (AGAI) while also scaling the difficulty level of game AI (DDB). Three approaches were used to achieve this: (1) high fitness penalizing, (2) weight clipping and (3) top culling, which was the highest performing approach.

3.2 Dynamic Difficulty Balancing through Symbiotic Game Agents

Obodoekwe, Coulter and Ehlers [13] proposed the use of SGA to perform DDB in a serious game in order to maintain player immersion. Facial expression analysis was used to classify the expression on the player's face, and this information determined how the difficulty of the game would be adjusted.

The task of the facial expression recogniser was to continuously analyse the player's facial expression and compare it to the following basic expressions: anger, sadness, surprise, disgust, happiness and fear. The information on the player's emotion and duration was then passed to the play classifier, which modified the game's difficulty accordingly.

After a given time interval, the player's emotion and difficulty level were matched up with the player's score. A fitness value was then assigned to the emotion, difficulty and score combination, and a genetic algorithm was used to optimise the choice of difficulty for a given facial expression. This method performed well as it resulted in the overall increase in player scores over long periods of time [13].

4 AdaptiveSGA

The work presented here aims to achieve AGAI-based DDB through the SGA model. Although DDB has been implemented by Obodoekwe et al. through SGA [13], the paper did not mention the use of AGAI to achieve it. DDB was achieved by modifying two parameters: game speed and question difficulty. This does not necessarily change the game AI's behaviour. Moreover, the use of computer vision involves recording the

player's face and surroundings. Some players might not be comfortable with this approach from a privacy perspective.

The discussed approaches proposed by Spronck et al. and Tan et al. used AGAI to achieve DDB through dynamic scripting and adaptive controllers (AUC and ADC), respectively. However, the work presented here seeks to achieve DDB using SGA (Please refer to Fig. 1) due to the flexibility offered by this architecture.

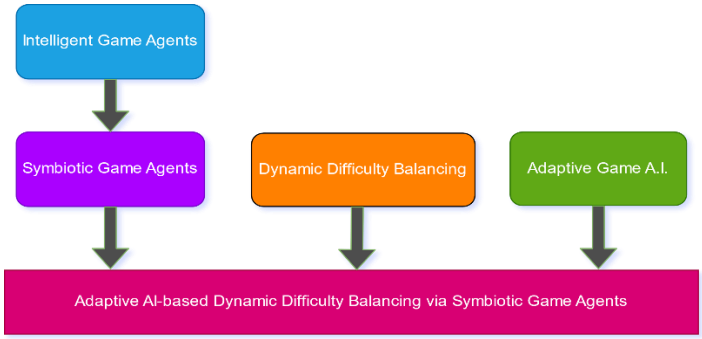

Fig. 1. A flow diagram indicating how the different concepts discussed thus far are used to realise the proposed model.

By using SGA, for example, one can swap symbiont agents without having to take their architecture into consideration. In addition, the swapping of agents may happen in real-time, without having to suspend gameplay. Cotterrell et al. performed this using a control symbiont agent. Please refer to Fig. 2 for the original SGA model by Cotterrell et al. [11].

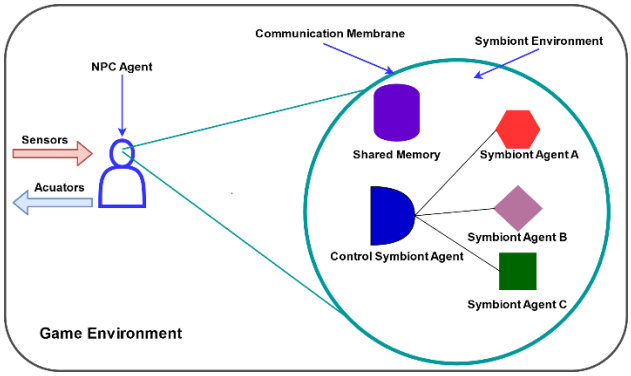

Fig. 2. The initial SGA model. Proposed by Cotterrell et al. Image adapted from [11].

The control symbiont agent is essentially an added agent with the purpose of facilitating communication among the various symbiont agents. In order to alleviate this issue, this work proposes an SGA model that does not make use of a control symbiont

agent. This is achieved by placing more responsibility on the communication membrane. Fig. 3 illustrates the proposed SGA model without a control symbiont agent.

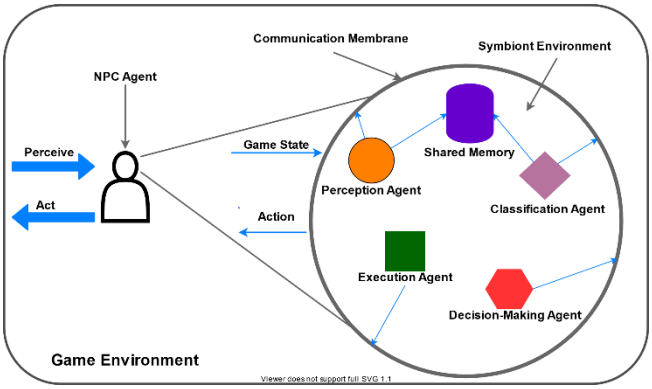

Fig. 3. The AdaptiveSGA model for achieving DDB through AGAI.

Game Environment. This is where the NPC agent interacts with the game's elements and other NPCs. Every action that the NPC performs affects the game environment accordingly. For games with dynamic environments, the environment, as well as the entities inside it, have the potential of changing the state of the NPC even if the NPC is idle.

Symbiont Environment. This environment is separated from the game environment by the communication membrane. Events that occur in the game environment cannot directly affect the symbiont environment. Symbiont agents interact with the communication membrane the shared memory within the symbiont environment.

Communication Membrane. Information about the game state is sent to the symbiont environment via the communication membrane. Furthermore, the communication membrane acts as a filter which allows only certain information to flow from the game environment into the symbiont environment and vice-versa.

4.1 NPC Agent and Symbiont Agents

The NPC Agent is the symbiotic agent and serves as the NPC controller. The NPC agent's internal workings are enabled by a multi-agent system of symbiont agents. The symbiont agents only interact with the communication membrane and shared memory.

Perception Agent. The role of the perception agent is to maintain a percept history in the shared memory for later use by the classification agent. This agent has the autonomy to decide in what format the data should be stored.

Decision-Making Agent. The decision-making agent is only responsible for decisions that the NPC makes in the game environment. Decisions relating to the symbiont

environment are made by the communication membrane in conjunction with the classification agent. The decision-making agent typically makes use of algorithms such as finite-state machines, decision trees, behaviour trees, etc. to help the NPC make appropriate decisions in the game environment.

Execution Agent. The execution agent's task is to process decisions made by the decision-making agent and translate them into literal actions. The reason for having two separate symbiont agents for decision-making and execution is to separate decision-making models from code that translates a decision to low-level actions. This provides flexibility because, in some cases, it may be unnecessary to replace a decision-making symbiont agent simply to modify the efficiency with which actions are executed.

Classification Agent. The task of the classification agent is to continuously analyse the historical data stored in shared memory in order to classify the player's current proficiency. Classifying proficiency helps in choosing the right difficulty for future game instances.

5 Experimental Setup

5.1 Prototype

In order to test the applicability and feasibility of the model, we applied it to the problem of achieving DDB in simulated soccer. The soccer game application was designed using the Java programming language, and a screenshot of it is shown in Fig. 4. The team in yellow is controlled by a static game AI while the team in blue is controlled by AGAI-based SGA. Some of the rules of real-world soccer were excluded in this prototype because they do not significantly contribute to the aim of this research.

Fig. 4. Simulated soccer. The team in yellow is controlled by a static game AI while the team in blue is controlled by AGAI-based SGA.

As such, goalkeepers were excluded from the game. The offside rule does not apply. When the ball leaves the pitch, it is placed at the centre of the pitch, and the game continues. Each match (game instance) was 1 minute 20 seconds with each update cycle occurring every 80 milliseconds. This resulted in each match lasting 1000 time steps. It should be noted that all real-world soccer rules can be incorporated into the prototype, although the problem that is being addressed by this work does not depend on their presence.

There were 8 available conditions an NPC could evaluate at any point during the game: (1) *Opponent's goals close?* (2) *Own goals close?* (3) *Teammate has the ball?* (4) *Opponent has the ball?* (5) *Do I have the ball?* (6) *Am I close to a teammate?* (7) *Teammate ahead?* (8) *Am I close to the ball?*

There were 7 available states for an NPC to transition to at any point during the game: (1) *Shoot*, (2) *Pass the ball to a teammate ahead*, (3) *Pass the ball to the closest teammate*, (4) *Make an attacking run*, (5) *Make a defensive run*, (6) *Chase the ball*, or (7) *Dribble.*

5.2 Achieving DDB through AGAI

Static Game AI Implementation. As mention in Subsection 5.1, the yellow team was controlled by a static game AI which did not use SGA to control the NPCs. The static game AI object made use of a finite-state machine (FSM) to make decisions and it executed the decisions itself. The FSM used by the static game AI followed the design depicted in Fig. 5.

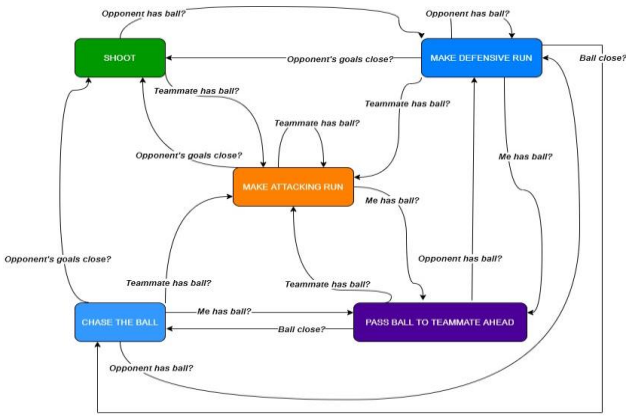

Fig. 5. FSM used by the static game AI to control the yellow team.

Adaptive Game AI Implementation. The blue team was controlled by the AGAI-based SGA controller. In order to achieve AGAI, we leveraged the flexibility that SGA brings as an agent design principle: the ability to swap in and out symbiont agents during the execution of the game without interruption. Three difficulty settings were

implemented: *EASY*, *MEDIUM* and *HARD*. The classification symbiont agent was responsible for predicting an appropriate difficulty setting.

The rules for choosing a difficulty setting were inspired by the concept of achieving an even game with the static game AI. The even-game approach is the same approach that was followed by Spronck et al. [4] and Tan et al. [1]. Therefore, the primary goal of the classification symbiont agent was to ensure that the adaptive team achieved an even game with the static team.

At the end of every match, the communication membrane provided the classification symbiont agent with the match result and the current difficulty setting. Upon predicting the next appropriate difficulty setting, the classification agent provided the new difficulty setting to the communication membrane. The communication membrane would then swap decision-making agents accordingly in order to scale the difficulty of the game (Please refer to Fig. 6).

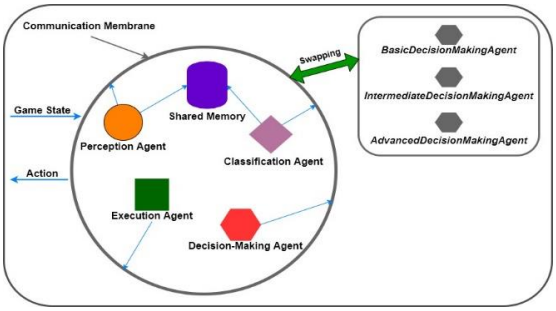

Fig. 6. An illustration of how the swapping of decision-making symbiont agents is performed by the communication membrane.

Three decision-making symbiont agents designed for this game: (1) *BasicDecisionMakingAgent*, (2) *IntermediateDecisionMakingAgent* and (3) *AdvancedDecisionMakingAgent*. Each decision-making symbiont agent used a different algorithm to reach decisions.

The *BasicDecisionMakingAgent* and *IntermediateDecisionMakingAgent* used FSMs of different logical structure and complexity. The *AdvancedDecisionMakingAgent* made use of a decision tree. Each decision made by the decision-making symbiont agent during the game was passed to the communication membrane, which further passed it as a percept to the execution symbiont agent, which determined the final action to be taken by the respective NPC. At the end of every match, the perception agent was responsible for storing the following match-related data in persistent storage for later analysis: (1) difficulty setting, (2) yellow team's score and (3) blue team's score.

6 Experimental Results

In order to test the ability of the SGA model to achieve AGAI-based DDB, we simulated 100 matches between the static game AI (yellow team) and the adaptive game AI (blue team). The aim of collecting this data was to determine if the blue team could

achieve an even game with the yellow team over 100 matches. We used Equation (1) [1] to measure how evenly matched the two teams were based on the results of each of the 100 matches.

$$W = L = \frac{(n-D)}{2} \quad (1)$$

where W is the number of wins by the blue team, L is the number of losses by the blue team and n is the number of matches, and D is the number of draws. According to a survey performed by Tan et al., players deem a large number of draws as more frustrating than. Therefore, having achieved an even game, it is also desirable to achieve a small number of draws. Moreover, a relatively high score difference between the two teams signifies an entertaining game [1].

6.1 Achieving an Even Game

Please refer to Fig. 7 for the trend in difficulty adjustment throughout the 100 matches. For most of the matches, the difficulty transitioned between *EASY* and *MEDIUM*. It was only in match 12 and 24 that the *AdvancedDecisionMakingAgent* (i.e. *HARD* difficulty) had to be swapped in order to maintain an even game.

With regards to achieving an even game, we defined the threshold for an uneven game as $|W - L| = 0.1 * n$. Therefore, if the difference between wins and losses exceeds 10% of the total games, we declared the encounter as uneven. The number of wins for the blue team was 35 ($W = 35$) while the number of losses was also 35 ($L = 35$), which means that $|W - L| = 0$. This means that the adaptive game AI was able to achieve an even game against the static game AI.

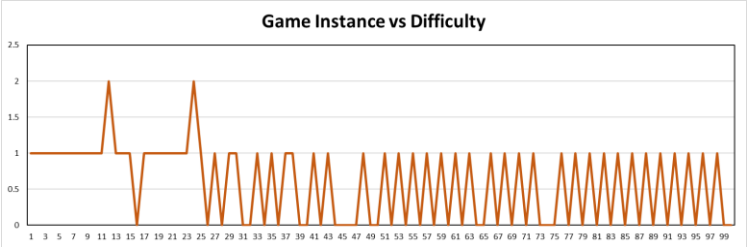

Fig. 7. A line graph depicting how the difficulty was adjusted at each game instance (match).

6.2 Achieving Entertainment Value

With regards to the number of draws, we defined the requirement for a non-frustrating game as $D < 0.1 * n$, which is 10% of the total games. The total number of draws was 30 ($D = 30$), which amounts to 30% of the total number of games. The adaptive game AI can be improved, in this aspect, so that the encounter causes the least amount of frustration for the opponent.

When looking at the score-difference history (Please see Fig. 8), we can see that 87% of all drawn matches took place within the first 50 matches. This means that the game

was less frustrating in the final 50 matches. Therefore, there is definitely room for improvement. Finally, in terms of score difference - which we translated into goal difference for a soccer game - the adaptive game AI achieved a fairly entertaining game with the goal difference between the two teams being 47. This is a promising result.

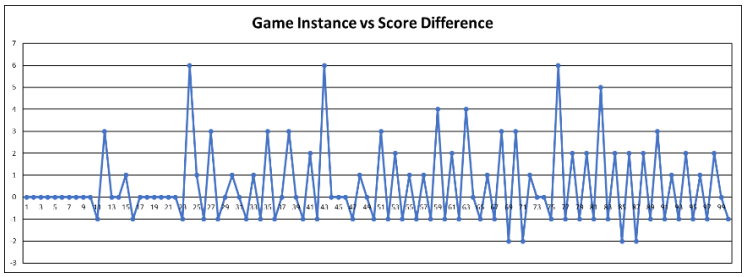

Fig. 8. A line graph depicting the score difference for each match (game instance) for all 100 matches.

7 Conclusion

The aim of this work was to achieve DDB through AGAI by making use of SGA architecture. According to experimental results, DDB was indeed achieved when using the even-game approach as a performance measure. However, there is still room for improvement with regards to reducing the level of frustration by further minimizing the number of draws. One approach may be to use a goal-based classification agent that incorporates historical data in its predictions.

The advantage of the presented model is that it requires no training phase. However, there is still the limitation that, once the adaptive game AI faces an opponent that is stronger than the *AdvancedDecisionMakingAgent*, there is no way to dynamically improve the *AdvancedDecisionMakingAgent* in real-time. This is similar to the problem faced by Tan et al.'s method in the sense that the adaptive controllers cannot learn. Once a player is as good as the controller's full potential, DDB will no longer be achievable. Future work will focus on the use of machine learning approaches to enable decision-making symbiont agents that can learn useful behavioural traits from opponents.

8 References

1. Tan, C.H., Tan, K.C., Tay, A.: Dynamic game difficulty scaling using adaptive behavior-based AI. 3, 289–301 (2011).
2. Cunha, E.M.C.A.M.: An Adaptive Game AI Architecture.
3. Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., Postma, E.: Adaptive game AI with dynamic scripting. 63, 217–248 (2006).
4. Spronck, P., Sprinkhuizen-Kuyper, I., Postma, E.: Difficulty scaling of game AI. Presented at the Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004) (2004).

5. Gomez-Hicks, G., Kauchak, D.: Dynamic game difficulty balancing for backgammon. Presented at the Proceedings of the 49th Annual Southeast Regional Conference (2011).
6. Andrade, G., Ramalho, G., Gomes, A.S., Corruble, V.: Dynamic Game Balancing: An Evaluation of User Satisfaction. 6, 3–8 (2006).
7. Hawkins, G., Nesbitt, K., Brown, S.: Dynamic difficulty balancing for cautious players and risk takers. 2012, 3 (2012).
8. Hagelback, J., Johansson, S.J.: Measuring player experience on runtime dynamic difficulty scaling in an RTS game. Presented at the (2009). https://doi.org/10.1109/CIG.2009.5286494.
9. Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Extending reinforcement learning to provide dynamic game balancing. Presented at the Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI) (2005).
10. Nareyek, A.: Intelligent agents for computer games. Presented at the International Conference on Computers and Games (2000).
11. Cotterrell, D., Ehlers, E.: Symbiosis, Game Agents and the Cloud. (2015). https://doi.org/10.5176/2251-1679_CGAT15.35.
12. Douglas, A.E.: The symbiotic habit. Princeton University Press (2010).
13. Obodoekwe, C., Coulter, D.A., Ehlers, E.M.: A Model for Improving Serious Game Player Immersion via Symbiotic Agents, Natural Language Processing, and Facial Expression Analysis. Presented at the International Conference on Computer Games, Multimedia & Allied Technology (CGAT). Proceedings (2017).