ICIC Express Letters ICIC International ©2010 ISSN 1881-803X
Volume **4**, Number **5**, **October 2010** pp. **1–6**

Adaptive Intelligence for Automated Game Learning

Hasan Mujtaba[1], Graham Kendall[2], and Rauf Baig[1]
[1]Department of Computer Science
FAST- National University
{hasan.mujtaba; rauf.baig;}@ nu.edu.pk
[2] School of Computer Science
University of Nottingham
gxk@cs.nott.ac.uk

ABSTRACT. *Is it possible to develop a computer program that can learn to play different video games by itself depending on the interactions with other players? Can video game characters learn new skills through interacting with human players? Can we make video games more interesting by allowing in-game characters to behave according to human player's strategy? These are just some of the questions that video game developers and artificial intelligence researchers are working on. In this paper we present an evolutionary approach that uses a modified particle swarm optimization algorithm and artificial neural networks, to answer these questions by allowing the agents to respond to changes in their surroundings. Video games usually require intelligent agents to adapt to new challenges and optimize their own utility with limited resources and our approach utilizes adaptive intelligence to improve an agent's game playing strategies. This research is directly applicable to video games research and evolutionary gaming. The approach presented here can be further extended to develop intelligent systems for the exploitation of weaknesses in an evolutionary system.*

Keywords: Evolutionary Computation, Swarm Intelligence, Evolutionary Game Playing, Particle Swarm Optimization, Artificial Neural Network.

1. Introduction

Traditional game playing programs have largely ignored adaptability in the learning process. In this research we investigate adaptive intelligence from the perspective of a game playing agent. We argue that an intelligent game playing agent is not designed to play a specific game, but rather should be able to learn from its environment. A number of Artificial Intelligence (AI) game players have claimed to be successful because they can play games at the level of a human expert. However, this does not prove their intelligence. Given the number of humans, how many world chess champions, or checkers experts are there? Is the rest of the human population *unintelligent*? Another issue game playing agents face is their predictable behavior. Predictability reduces the fun in playing the game multiple times, decreasing their replay value. We believe that intelligent and adaptive AI will lead to more interesting game playing. In this paper we focus on the development of such adaptive AI based game players. The proposed methodology enables agents to accept new information from the environment and update their existing knowledge space. This automated

knowledge update mechanism allows the agents to explore and exploit new opportunities provided by their environment. This means that agents can be trained without any human intervention, relying solely on the information provided by the environment. Such adaptive intelligence will lead to superior mimicking of natural intelligence. This research is not only useful in creating more entertaining games but is also beneficial for other intelligent systems that require learning in dynamic and uncertain environments.

2. Problem Statement and Preliminaries

Chellapilla and Fogel [1] proposed an alternative approach to developing computer game programs. In their study, a new type of checkers program, called Blondie24, is evolved by utilizing artificial neural networks (ANN), without pre-injecting human knowledge. ANN based controllers were also used in the NERO framework developed by Stanley et al. [2] and also by Yannakakis et al. [3] in a simulated world called Flatland. In [3] they experimented with agents controlled by ANNs with fixed architectures and trainable weights. Yannakakis and Hallam also evolved ANN controllers for the games of Pac-Man and Dead End [4]. Yet another effort in this field has been made by Lucas for the game of Cellz [5]. Kendall and Su [6] evolved ANN based agents to deal with an environment in which all the input variables are not available at the start but are gradually added over time. ANN based agents are also evolved by Messerschmidt and Engelbrecht for playing Tic-tac-toe [7] and by Franken and Engelbrecht for playing checkers [8]. Frayn used an evolutionary approach for evolving game playing strategies for Monopoly [9]. An interesting experiment, using an evolutionary approach in the context of games, is presented in [10]. Freed et al. [11] argue that generating human-like players will make games more appealing and enjoyable. This idea is further supported by the investigation of the correlation between the believability of non-playing characters and the satisfaction of the player [12]. Iida et al. work on the measures of entertainment in board games was the first attempt in this area. They introduced a general metric of entertainment for variants of chess games depending on average game length and possible moves [13]. Crispini [14] discusses criteria to make simple online games appealing. The outcome of his work hypothesizes challenge, diversity and unpredictability as the primary criteria for enjoyable opponent behaviors. Pedersen et al. [15] presents criteria that collectively define interest on any predator/ prey games, as follows:

1. When the game is neither too hard nor too easy.
2. When there is diversity in the opponents' behavior.
3. When the opponents' behavior is aggressive rather than static.

Other works that deal with optimizing player satisfaction and modeling player experiences include [16 - 18]. Some other applications using game theory, are presented in [19 - 20]. Our work is different from all previous work. We believe, to the best of our knowledge, this area has not been investigated before. Our approach to game playing has the following main features.

1. Automated game learning from scratch without pre-injected knowledge.
2. Learning to play games based on exposure to new players.
3. The agent itself decides when it has to invoke new learning and discard previously learned (but now obsolete) information.
4. The ability to learn the game to the level of the human player, making the game more interesting to play (rather than too challenging or too easy).

3. Automated Game Learning

For evolutionary purposes, game playing strategies can be represented in different ways. In our approach, we choose to represent each game playing strategy as an ANN. Connection weights of these ANNs are evolved using a variant of a Particle Swarm Optimization (PSO) algorithm. Our PSO algorithm is based on the *lbest* PSO with a change in the sub-swarm connection architecture. Evolution starts (Algorithm 1) by creating a random population of P particles. These particles comprise S overlapping sub-swarms. Each of the sub-swarms contains 5 particles. Each sub-swarm is connected to the next by the corner particles. In this manner the sub-swarms create a ring like formation, where data is shared via the connecting particles. Each of the particles (that is a, game playing strategy) is allowed to play a series of games against a scripted player. The number of wins and losses is monitored and fitness of the strategy is calculated on the basis of these statistics. Once all the agents have played their games, we can find the *gbest* (global best particle i.e. the particle with the highest fitness value in the entire population) and *lbest* (local best particle i.e. the particle with the highest fitness within a sub-swarm). These values are then used in the PSO equations [21]. Particles with poor fitness values should be encouraged to explore other areas than their current neighborhood. While particles with better fitness should be allowed to extensively search their current locality for better solutions. To place a limit on how far the particle can jump from its current position we use velocity clamping using an *exploration parameter*. This parameter is adjusted dynamically based on the fitness value of the sub-swarm. After a number of iterations, the population is forced to explore new opportunities that may be available in the environment. For this reason we determine the worst performing sub-swarm and force it to reinitialize its strategy. However care must be taken in order to avoid reinitializing a sub-swarm that is currently improving its fitness value.

ALGORITHM 1. Automated Game Learning Technique

1. Initialize parameters
2. Create S randomly initialized sub-swarms of P particles
3. While iteration < T_t
    a. Allow agents to play the game
    b. Calculate fitness of the current strategy
    c. Locate *gbest* and *lbest* particles
    d. Adjust exploration parameter
    e. Update positions of the particles
    f. After every T iterations explore new opportunities
        i. Detect worst performing sub-swarm
        ii. Force worst performer to create a new strategy
    g. Increment iterations
4. Go to Step 3

4. Experimentation

For our experimentation we choose two games; Connect4 and Pacman (we modified the standard version of the games for simplicity). We choose an ANN architecture that was applicable to both of these games. The entire board was input to the ANN (we use same sized board for both games). In both games empty squares were given an input of 0,

opponent (or ghost square) were set to -1 and "own" squares were set to 1 (for Pacman, squares with pills were set to 1). The learning agents are never told which players or game they are playing. They are just informed about the fitness evaluation of their current strategy. Using this fitness evaluation they must take steps to improve their performance in order to survive. For Connect4 each agent plays against a scripted player. Scripted players were intentionally set to a mediocre level. Playing against a mediocre level player presents agents with a suitable challenge by forcing them to learn to play the game rather than applying a random strategy. A few points are worthy of note:

1. Our focus is on enabling the agents to learn to play new games without relying on human training.
2. Game playing strategies are represented using ANNs.
3. We are not trying to create agents that are an expert at the games they play, hence our aim is not to find the optimal ANN architecture, although this would be interesting future work.
4. ANN weights are trained using a variant of PSO algorithm.
5. We do not expect agents to find the optimal game playing strategy. Instead they must learn to play the game as well as they can within a limited amount of time.
6. In order to test if the agents have learned anything about a game, they play against random strategies. If their performance is better than these random strategies then we can deduce that (some) learning has occurred.
7. Fitness values are normalized to highlight the underlying learning pattern and make the learning for different games comparable with one another.

For our first experiment we allow the agents to learn to play the game of Connect4 for the first 1000 iterations and then Pacman for the next 1000 iterations. During this time they may not evolve optimal game playing strategies but they do learn to play the game at a beginner level. It is important to note that this learning occurs from scratch. The learning approach for our first experiment is the standard PSO algorithm. This algorithm is not adaptive in nature and it ignores the changes occurring in its environment. The PSO algorithm forces all agents to move towards the best solution in the population. A change in the surrounding, e.g. changing the rules of the game, adding new challenges to the game, etc., can render this *old* best solution obsolete. Due to its ignorance of the surrounding environment, the PSO algorithm will fail to realize this change and all particles will be forced to move towards the outdated solution. This will lead to a decrease in the population fitness. This fact is evident from figure 1.

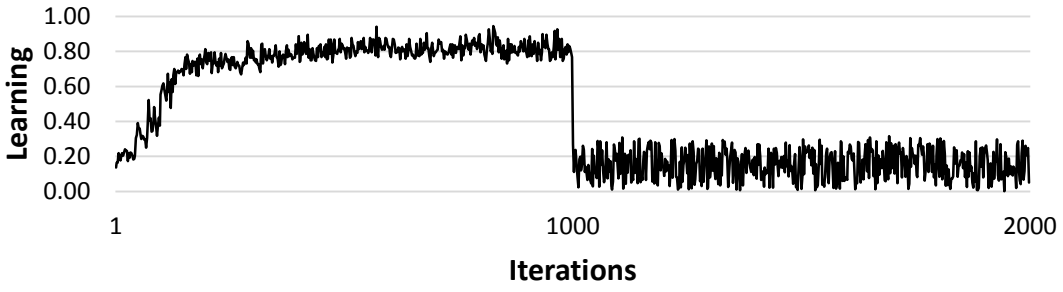

FIGURE 1. Ignoring the changes occurring in the surroundings

After a change in the game rules (from Connect4 to Pacman, which requires entirely different game playing strategies), the fitness of the population decreases drastically and

never recovers. The only way for the algorithm to recover is to get *lucky* and find a strategy by chance. For our second experiment, we use the same setup but this time instead of using the standard PSO for learning we use our own learning algorithm. As mentioned earlier our algorithm uses a modified version of the PSO algorithm. Our approach is more focused towards monitoring the environment for change. Once a change occurs, agents modify their current strategies to better adapt to it. As shown in figure 2, after 1000 iterations agents are introduced to new game players who play a different game from the once they have already learned. These new players require the agents to update their strategies in order for the agents to win. Relying on old strategies leads to poor initial fitness values. These fitness values indicate to the algorithm that something about the game has changed. New opportunities must be explored to formulate strategies better suited for this new game.

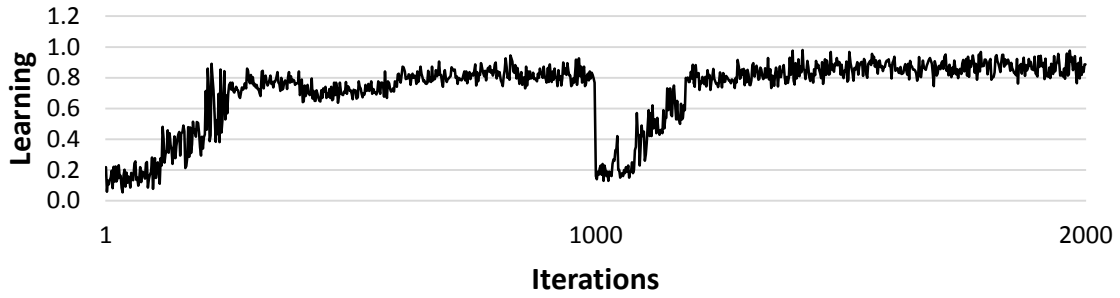

FIGURE 2. Reacting to changes occurring in the surroundings

5. Conclusions

In this paper we have presented an approach to adaptive intelligence which allows intelligent game playing agents to adapt to new challenges and optimize their own utility with limited resources. Our proposed approach overcomes the limitation of traditional approaches which fail to adapt to changes in the environment, our approach with its adaptive nature, allows the agents to adjust their behavior according to new challenges. This gives the agents the ability to learn in the presence of unseen scenarios (like learning to play new games) without relying on any pre-injected knowledge. The whole learning process is automated and stimulated by exposure to new players and situations. We believe this research will lead to better game AI and improving the replay value of video games. The learning algorithm assumes nothing about its surrounding environment, its objectives and this independent nature allows it to be applicable to other domains.

REFERENCES

[1]. K. Chellapilla, and D. B. Fogel, "Evolving an Expert Checkers Playing Program without Using Human Expertise", IEEE Trans. on Evolutionary Computation, 2001, Volume 5, Issue 4.

[2]. K. Stanley, and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies", IEEE Trans. on Evolutionary Computation, 2002, Volume 10, Issue 2.

[3]. G. N. Yannakakis, J. Levine, and J. Hallam, "Emerging Cooperation with Minimal Effort: Rewarding Over Mimicking", IEEE Trans. on Evolutionary Computation, 2007, Volume 11, Issue 3.

[4]. G. N. Yannakakis and J. Hallam, "A Generic Approach for Obtaining Higher Entertainment in Predator/Prey Computer Games," Journal of Game Development, 2005, Volume 1, Issue 3.

[5]. S. M. Lucas, “Cellz: A Simple Dynamical Game for Testing Evolutionary Algorithms”, IEEE Congress on Evolutionary Computation, 2004.
[6]. G. Kendall and Y. Su, “Imperfect Evolutionary Systems”, IEEE Trans. on Evolutionary Computation, 2007, Volume 11, Issue 3.
[7]. L. Messerschmidt and A. P. Engelbrecht, “Learning to Play Games Using a PSO-Based Competitive Learning Approach”, IEEE Trans. on Evolutionary Computation, 2004, Volume 8, Issue 3.
[8]. N. Franken and A. Engelbrecht, “Comparing PSO Structures to Learn the Game of Checkers from Zero Knowledge”, IEEE Congress on Evolutionary Computation, 2003.
[9]. C. M. Frayn, “An Evolutionary Approach to Strategies for the Game of Monopoly®”, Proceedings of IEEE Symposium on Computational Intelligence and Games, 2005.
[10]. R. G. Reynolds, Z. Kobti, T. A. Kohler, and L. Yap, “Unraveling Ancient Mysteries: Reimagining the Past Using Evolutionary Computation in A Complex Gaming Environment”, IEEE Trans. on Evolutionary Computation, 2005, Volume 9, Issue 6.
[11]. M. Freed, T. Bear, H. Goldman, G. Hyatt, P. Reber, A. Sylvan, and J. Tauber. Towards more human-like computer opponents. In Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, 2000.
[12]. N. A. Taatgen, M. van Oploo, J. Braaksma, and J. Niemantsverdriet. How to construct a believable opponent using cognitive modeling in the game of set. In Proceedings of the fifth international conference on cognitive modeling, 2003.
[13]. Hiroyuki Iida, N. Takeshita, and J. Yoshimura. A metric for entertainment of board games: its implication for evolution of chess variants. In R. Nakatsu and J. Hoshino, editors, IWEC2002 Proceedings, 2003.
[14]. N. Crispini. Considering the growing popularity of online games: What contributes to making an online game attractive, addictive and compelling. Dissertation, SAE Institute, London, October 2003.
[15]. C. Pedersen, J. Togelius and G. N. Yannakakis, "Modeling Player Experience for Content Creation", IEEE Trans. on Computational Intelligence and AI in Games, 2009, Volume 1, Issue 2.
[16]. G. N. Yannakakis, and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction", IEEE Trans. on Computational Intelligence and AI in Games, 2009, Volume 1, Issue 2.
[17]. G. N. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences," IEEE Systems, Man and Cybernetics; Part A: Systems and Humans, 2009, Volume 39, Issue 6.
[18]. G. N. Yannakakis, and J. Hallam, "Entertainment Modeling through Physiology in Physical Play," International Journal of Human-Computer Studies, 2008, Volume 66, Issue 10.
[19]. V. N. Marivate, and T. Marwala, “Introduction of Social Learning in Board Games”, ICIC Express Letters, September 2009, Volume 3, Issue 3(A).
[20]. Y. Konishi, N. Araki, Y. Iwai and H. Ishigaki, "Optimal Integrated Design for Mechanical Structure and Controller Using Bargaining Game Theory", ICIC Express Letters, March 2009, Volume 3, Issue 1.
[21]. A. P. Engelbrecht, “Fundamentals of Computational Swarm Intelligence”, John Willey & Sons, 2005.