Computer and Information Science; Vol. 9, No. 3; 2016
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

A Synthetic Player for Ayò Board Game Using Alpha-Beta Search and Learning Vector Quantization

Oluwatobi, A. Ayilara[1], Anuoluwapo, O. Ajayi[1] & Kudirat, O. Jimoh[1]

[1] Computer Science and Engineering, Obafemi Awolowo University, Nigeria

Correspondence: Anuoluwapo Ajayi, Computer Science & Engineering, Obafemi Awolowo University, Nigeria. E-mail: anuajayi@yahoo.com

Received: April 21, 2016 Accepted: May 6, 2016 Online Published: June 19, 2016

doi:10.5539/cis.v9n3p1 URL: http://dx.doi.org/10.5539/cis.v9n3p1

Abstract

Game playing especially, Ayò game has been an important topic of research in artificial intelligence and several machine learning approaches have been used, but the need to optimize computing resources is important to encourage significant interest of users. This study presents a synthetic player (Ayò) implemented using Alpha-beta search and Learning Vector Quantization network. The program for the board game was written in Java and MATLAB. Evaluation of the synthetic player was carried out in terms of the win percentage and game length. The synthetic player had a better efficiency compared to the traditional Alpha-beta search algorithm.

Keywords: intelligence,board game,win ratio, computing resources

1. Introduction

Developers are utilizing various computing techniques to create efficient games that will run in reasonable time to encourage interest from users. Ayò is an African game board made of a wooden substance. It is a rectangular board with twelve circular carved out pits (pockets) arranged in two rows of six pits, and each pit containing four seeds. The game belongs to the class of two-person, zero-sum game (Van den Herik, Uiterwijk, &Van Rijswijck, 2002). In playing the game, a player selects one of the non-empty pits and starts sowing in a counter clockwise direction and whenever the sowing reaches the originated pit, it is ignored. Seeds are captured when the last seed falls in an enemy's pit containing 1 or 2 seeds. Additional seeds are captured in pits immediately preceding the captured pit if they contain 2 or 3 seeds. Capturing occurs only in the enemy's pits. The game ends when any player has obtained 25 or more seeds. Mancala is the general name for the many variations (Awale, Oware,Awari, et al.) of the game played throughout Africa, as well as in other parts of the world. Refer to (Allis, Van Der Meulen, & Vanden Herik, 1991; Allis, 1994; Romein & Bal, 2002) for further information. However, one limiting feature of strategically played board games like Ayò is that they consume computational resources (Romein & Bal, 2002), due to their huge game states. Awale for instance, has a game tree complexity of 10^{31}(Allis, 1994; Van den Herik et al., 2002). This feature makes playing the game on most low-cost devices computationally challenging.

There is no doubt that numerous literature on computer games (Gomboc, Buro,& Marsland, 2005; Guid et al., 2006; Al-Mahmuda, Mubin, Shahid, & Martens, 2010) exist, and the need to design more storage efficient algorithms for games has encouraged the proposition of several techniques. However, only few exist in terms of design and architecture for resource optimization.Allis (1994) proposed a Minimax Search Algorithm (MSA), which employed the backward induction to predict game in Awale. However, the main difficulty posed by minimax search is how to develop and apply an evaluation function to a game tree. A database containing all board positions with 35 or less seeds to improve the playing strength of Awale was constructed in (Lincke & Marzetta, 2000). The retrograde analysis was proposed to find the optimal play for all the possible board positions of Awari (Romein & Bal, 2002; Romein & Bal, 2003). However, both endgame databases and retrograde analysis techniques are expensive to implement as the game requires more storage space. Meta-heuristic based hybrid technique was proposed to mine endgame databases for relevant features that are useful in the construction of a static evaluation function (Davis & Kendall, 2002; Daoud, 2004). Akinyemi, Adebiyi, and Longe (2009) proposed a refinement based heuristic technique for human-like decision making for playing *Ayò* game. The performance of Minimax search and aggregate Mahalanobis distance function in Evolving an Ayo Player was investigated by (Randle, Olugbara & Lall, 2012).

The Learning Vector Quantization (LVQ) network is an algorithm widely used in the classification of high-dimensional data (Biehl, Ghosh,& Hammer, 2007). Successful applications of LVQ include medical image analysis, fault detection, or classification of satellite spectral data (Bojer, Hammer, & Koers, 2003; Kuncheva, 2004; Schleif, 2006). Its popularity can be attributed to the modest implementation of its procedures and the fact that its complexity can be controlled during training according to specific needs. Both, the training algorithm and the resulting classification scheme are fairly easy to implement. In practice, the computational effort of LVQ training usually scales linearly with the training set size, and its classification depends on the (fixed) number of learning techniques (prototypes) andinput dimensionality (Biehl et al., 2007). This study therefore, proposes a synthetic player (called Ayò) that combined the Alpha-beta search algorithm with LVQ to enhancethe efficiency of Alpha-beta search.

2. The Synthetic Player Framework

The section describes the techniques and tools used in the design and implementation of the synthetic player for the game.

2.1 Alpha-Beta Pruning

Alpha-beta pruning is an algorithm that minimizes the game-tree search for some wrong moves. It simply identifies moves that are not beneficial and eliminate them from the game tree (Jones, 2008). The algorithm simply calculates and maintains two variables (alpha and beta) during the depth-first search of the game tree. The alpha variable defines the best move that maximizes the Alpha-beta search's best move while the beta variable defines the best move that minimizes the opposing player's best move.Although, Alpha-beta pruning method provides its best performance when the game tree is arranged such that the best choice at each level is the first one to be examined by the algorithm. That is, it only removes only one leaf node from the tree, but in larger game trees, this can result in fairly valuable reductions in tree size. In most cases, it will not necessarily remove large portions of the game tree. While simulating the playing of *Ayò* boardgame between Alpha-beta search and Random player (a computer program that uses stochastic algorithm to select any non empty pit to play), it was discovered after a careful analysis of the game data (playing patterns) generated by both players that Alpha-beta search made few wrong moves (though infrequent) by not distributing seeds from pits with heavily loaded seeds (called Odù in Yorubálanguage*)* as appropriate.Figure1illustrates an example of this problem, assuming the transition from state n to state n+1. For a given state n in the game, where it is the Alpha-beta's turn to make a move. Unfortunately, Alpha-beta search distributed the four seeds in pit E (Figure 1a) instead of distributing the 17 seeds in pit C (see Figure1b)which could have fetched additional 5 seeds (two seeds in pit c and 3 seeds in pit d as gains(see Figure1c). Unfortunately, Alpha-beta search had missed the opportunity to gain additional five seeds in that given state (state n).Not playing heavily loaded pits at the appropriate time may unnecessarily elongates the game length, which implicitly may affect computing resources.We intend to avoid this problem by combining LVQ with Alpha-beta to minimize this effect. The introduction of LVQ is to help in determining the appropriateness at which heavily loaded pits (Odù)are played.

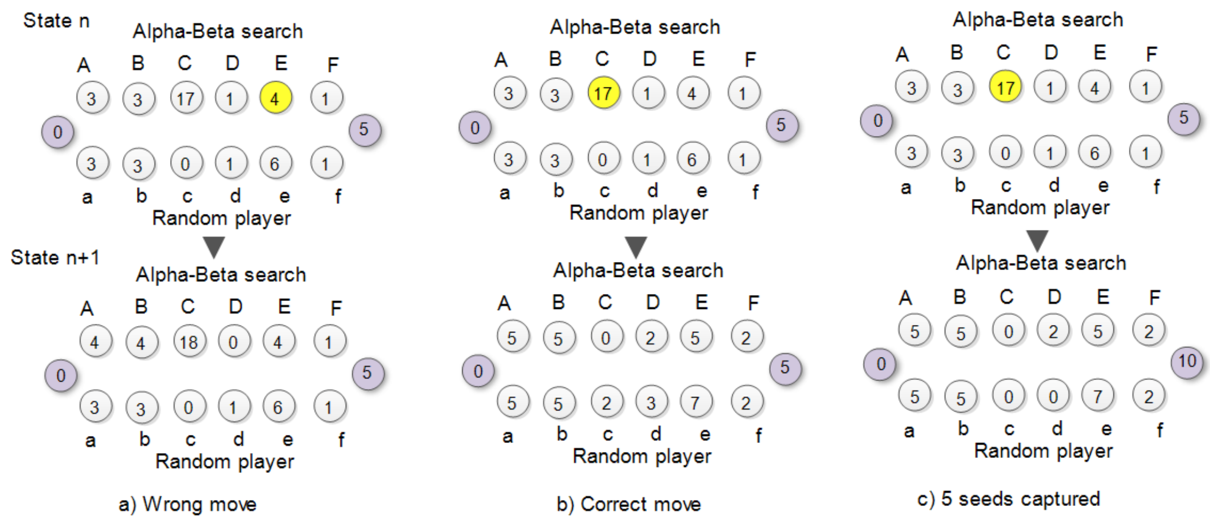


Figure 1. Wrong move by Alpha-beta search (a) versus the right move (b)

2.2 Learning Vector Quantization Network

To minimize the Alpha-beta limitation, the key states (locations of heaped seeds that when played will shorten playing length) were determined experimentally using simulated games between Alpha-beta search and Random player. The key states were then used to train the LVQ in order to optimally select and play seeds in the heavily loaded pits appropriately, to reduce game length. The LVQ network architecture adapted from (Demuth & Beale, 2001) and depicted in Figure2, has *R* number of elements in the input vector, S^1 number of competitive neurons, and S^2number of linear neurons.The network has two layers, the competitive and the linear layer. The neurons in a competitive layer distribute themselves to recognize frequently presented input vectors while the linear layer transforms the competitive layer's classes into target classifications defined by the user.The LVQ for the Ayò player has eight (8) neurons in the hidden layer and one neuron in the output layer. The training data (derived experimentally)for the network is the vector [33.2, 27.2], which represents the class percentages or frequencies at which seeds are heaped in the pits (indices D, C, B, and A) of the *Ayò* player. In practice, these pits have large number of seeds as observed from the several simulated games played by the Alpha-beta search against the Random player. The network learning rate was 0.01 and its learning function was Learning Vector Quantization 1 weight learning function (learn 1v1). The LVQ network was trained for a maximum of 500 epochs.The following equations were used as the LVQ input.

$$c = argmax_{i:i\in\{7,8\ldots,12\}} pit_i \tag{1}$$

$$d = \frac{\sum_{i=7}^{12} pit_i}{6} \tag{2}$$

$$v = \begin{bmatrix} c \\ \lfloor d \rfloor \end{bmatrix} \tag{3}$$

where c is the pit (belonging to Ayò player) which contains the largest number of seeds, variable d is the averagenumber of seeds in the pits belonging to the syntheticplayer, and $\lfloor d \rfloor$is the largest integer less than or equal to d. The input v is passed to the competitive layer which used the winner takes all Hebbian learning approach to determine the network's output, that is, the particular heavily loaded pit (Odù) to play.

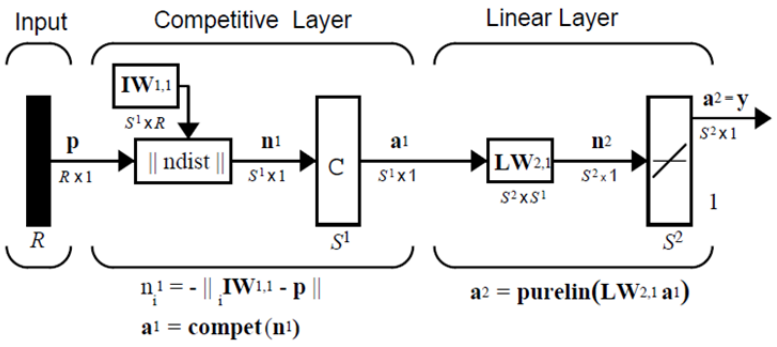


Figure 2. LVQ network architecture

2.3 Performance Evaluation

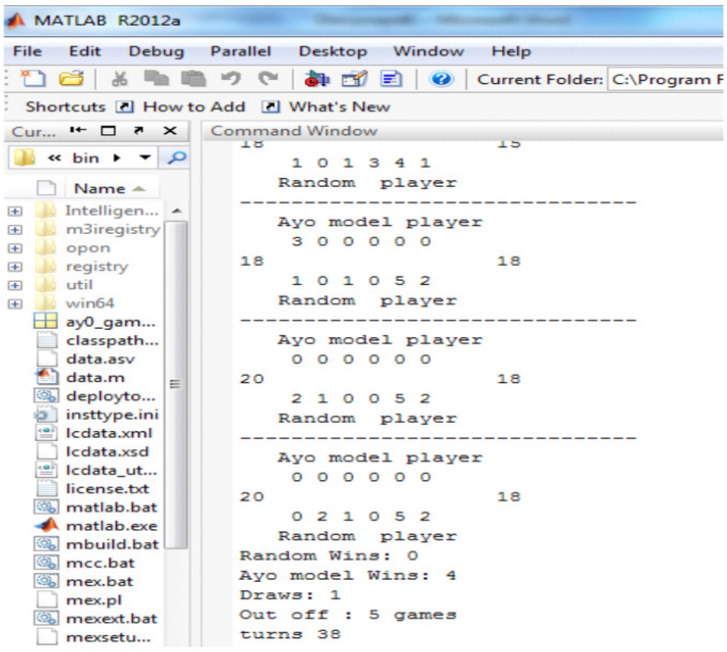
The Ayò board game was simulated in both Java and MATLAB R2012a environment. Java language was used to implement the Alpha-beta search algorithm due to its recursive nature. Moreover, creating recursive algorithms in Java can be easily accomplished. The MATLAB software includes Java Virtual Machine (JVM) software, that enables Java interpreter to be used via MATLAB commands, purposely to assist in the creation and running of programs that access Java objects. This feature enables MATLAB users to take advantage of the special capabilities of the Java programming language. The Java class implementing the Alpha-beta search algorithm was made available in the MATLAB workspace by placing it on the static Java class path and then invoked within the MATLAB environment. The Alpha-beta search algorithm implemented in this study used a 4-level depth minimax search together with the LVQ described above to evolve the Ayò player. The simulation was run on an Intel (R) Pentium 2.20 GHz machine. Figure 3 depicts sample simulation result. To compare the performance of Alpha-beta search with the synthetic player (LVQ_Alpha-beta), several games between Alpha-beta search versus Random player (a program having its playing patterns or moves unpredictable) andLVQ_Alphabeta versus Random player were simulated. Four simulation experiments were conducted as follows. The first experiment consisted of 1000 simulated games between Alpha-beta search versus Random player; and LVQ_Alphabeta against Random player. In the second experiment, 4000 simulated games between Alpha-beta versus Random and LVQ_Alphabeta against Random player were simulated. Similarly, the third and forth experiments consisted of 7000 and 10000 simulated games between the opposing players as configured for experiments 1 and 2 above. All experiments were repeated x times and the average game length for games played between LVQ_Alphabeta and Random player; and the average game length for games between Alpha-beta search against Random player were recorded and compared. Also recorded were the number of games won and loss by both LVQ_Alphabeta and Alpha-beta search against the Random player. The average game length (avl) is as defined as follows.

$$avl = \frac{1}{n}\left(\sum_{i=1}^{n} t_i\right) \qquad (4)$$

Where $t_i$isthe average turns made by eitherLVQ_Alphabeta or Alpha-beta search against the Random playerin game i, and n is the total number of games played. The notation for the Win percentage is given as:

$$winPercentage = (games\ won)/(total\ games\ played\ excluding\ draws) \times 100 \qquad (5)$$

The average number of turns used by all the three players- Random, Alpha-beta search and *Ayò* were noted. The Win percentage of games (excluding draws) played between the Random player and the Alpha-beta search, and the Random player versus the *Ayò* (LVQ-Alpha-beta) player were also recorded.



```
18                         15
     1  0  1  3  4  1
    Random   player
-------------------------------
    Ayo model player
     3  0  0  0  0  0
18                         18
     1  0  1  0  5  2
    Random   player
-------------------------------
    Ayo model player
     0  0  0  0  0  0
20                         18
     2  1  0  0  5  2
    Random   player
-------------------------------
    Ayo model player
     0  0  0  0  0  0
20                         18
     0  2  1  0  5  2
    Random   player
Random Wins: 0
Ayo model Wins: 4
Draws: 1
Out off : 5 games
turns 38
```

Figure3. Sample screenshot of the board game

3. Results

The average game lengths recorded for both LVQ_Alpha-beta and Alpha-beta search against the Random playerare shown in Table 1. The overall average game lengths for the two players were 36.48 and 51.93 respectively, resulting in approximately 30% reduction in game length usingLVQ-Alphabeta. Also shown in Table 1, is the percentage number of games (excluding draws) won by both Alpha-beta search and LVQ-Alphabeta players. The percentage numbers of games won by both players against the Random player were approximately70% and 83%, respectively.The results obtained in this study showed that the LVQ has enhanced the Alpha-beta search performance with shorter game length and with a higher win ratio. Thus, the addition of LVQ to Alpha-beta search has brought significant improvement in efficiency. The implication of this result is that Ayo will reduce computing processing time and also run conveniently on low cost handheld devices.

Table 1.Results of Random player versus LVQ_Aphabeta and Alpha-beta

	Win Percentage		Average game length (95%CI)	
Experiment	LVQ_Alphabeta	Alpha-beta search	LVQ_Alphabeta	Alpha-beta search
#1	82.7(0.51)	70.5(0.55)	35.6(3.26)	47.9(8.21)
#2	83.5(0.82)	70.6(0.59)	35.9(2.78)	47.6(7.01)
#3	83.3(0.68)	70.4(0.27)	37.6(3.32)	54.4(6.35)
#4	82.6(0.55)	70.3(0.18)	36.8(2.51)	57.8(9.85)

4. Conclusion and Future Direction

In this study, a synthetic player utilizing Alpha-beta search and LVQ network is presented. The synthetic player (Ayò) was simulated in MATLAB environment and evaluated in terms of the game length and win percentage.The result of performance evaluation has indicated that LVQ was able enhance Alpha-beta search by producing improved synthetic player for the Ayo board game that will optimize computing resources most importantly, the low-cost devices with limited capabilities.

References

Akinyemi, I. O., Adebiyi, E. F., & Longe, H. O. D. (2009). Critical analysis of decision making experience with a machine learning approach in playing Ayo game. *Engineering and Technology, 56*, 49-54.

Al-Mahmuda, A., Mubin, O., Shahid, S., & Martens, J. (2010). Designing social games for children and older adults: Two related case studies. *Entertainment Computing, 1,*147–156.http://dx.doi.org/10.1016/j.entcom.2010.09.001

Allis, L. V., Van der Meulen, M., & Van den Herik, H. J. (1991). Databases in Awale. In: Levy,D.N.L. &Beal,D.F. (Eds.).*Heuristic Programming in Artificial Intelligence, 2*, 73–86. Ellis Horwood, Chichester.

Allis, L. V. (1994). *Searching for solutions in games and artificial intelligence* (Unpublished Ph.D. thesis). University ofLimburg, Maastricht.

Biehl, M., Ghosh, A., & Hammer, B. (2007). Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, *8*, 323-360.

Bojer, T., Hammer, B., & Koers, C. (2003). Monitoring technical systems with prototype based clustering. In Verleysen,M. (Eds.). *European Symposium on Artificial Neural Networks*, 433–439. Evere, Belgium.http://dx.doi.org/10.1.1.13.5198/citeseerx.ist.psu.edu/viewdoc/summary?

Daoud, M., Kharma, N., Haidar, A., & Popoola, J. (2004). Ayo, the Awale player, or how better representation trumps deeper search. *Proceeding of the IEEE Congress on Evolutionary Computation*, 1001-1006.

Davis, J. E., & Kendall, G. (2002). An Investigation, using co-evolution, to evolve an Awale player. *Proceedings of Congress on Evolutionary Computation*, 1408-1413.

Demuth, H., & Beale, M. (2001). *Neural Network Toolbox for Use with MATLAB*. Massachusetts, MA: The MathWorks, Inc.

Gomboc, D., Buro, M., & Marsland, T. A. (2005). Tuning evaluation functions by maximizing concordance.*Theoretical Computer Science*, *349*, 202–229.http://dx.doi.org/*10.1016/j.tcs.2005.09.047*

Guid, M., & Bratko, I. (2006). Computer analysis of world chess champions.*International Computer Games Association Journal*, *29*, 65–73.

Jones, M. T. (2008). *Artificial intelligence a systems approach*. Massachusetts, MA: Infinity Science Press.

Kuncheva, L. I.(2004). In Roli, F., Kittler, J. &Windeatt,T. (Eds.), *Multiple Classifier Systems*, 1–15.Berlin: Springer.

Lincke, T. R., & Marzetta, A. (2000). Large endgame databases with limited memory space. *International Computer Games Association Journal*, *23*, 131-138.

Randle, O. A., Olugbara, O. O., & Lall, M. (2012). Investigating the performance of Minimax search and aggregate Mahalanobis distance function in evolving an Ayo/Awale player. *International Journal of*

Computer, Electrical, Automation, Control and Information Engineering, *6*, 953-956.

Romein, J. W., & Bal, H. C. (2002). Awale is solved. *International Computer Games Association Journal*, *25*, 162-165.

Romein, J. W., & Bal, H. C. (2003). Solving the game of Awale using parallel retrograde analysis.*IEEE Computer*, *36*, 23-33.http://dx.doi.org/10.1.1.57.7918/citeseerx.ist.psu.edu/viewdoc/summary?

Schleif, F. M., Villmann, T., & Hammer, B. (2006). Local metric adaptation for soft nearest prototype classification to classify proteomic data. In Bloch, I.,Petrosino, A.,&Tettamanzi, A.G.B. (Eds.).*International Workshop on Fuzzy Logic and Applications*, 290–296. Berlin.

Van den Herik, H. J., Uiterwijk, W. H. M., & Van Rijswijck, J. (2002). Games solved: Now and in the future. *Artificial Intelligence*, *134*, 277–311.http://dx.doi.org/10.1016/S0004-3702 (01)00152-7