Ayo, the Awari Player, or How Better Representation Trumps Deeper Search

Mohammed Daoud, Nawwaf Kharma[1], Ali Haidar and Julius Popoola
Dept. of Electrical and Computer Engineering,
Concordia University
1455 de Maisonneuve Blvd. W.,
Montreal, QC, Canada. H3G 1M8
[1]kharma@ece.concordia.ca

***Abstract*-Awari is a two-player end-game played on a plank with 12 pits and 48 seeds; the goal of the game is to collect 25 seeds before the other player does. In this paper, we illustrate the importance of problem domain representation, using our own Awari playing program: *Ayo*. We use a Genetic Algorithm to optimize the weights of the feature evaluation function of Ayo. We play Ayo against a commercially available Awari player, then compare Ayo's results to the results achieved by an older Awari player; one that uses a 7-levels deep mini-max search. Ayo, with a 5-levels deep mini-max search, returns better results, due to better more intelligent representation of the state space.**

I. INTRODUCTION

Awari is a very old game that seems to have originated from Ethiopia, and gradually spread to all of Africa and beyond. The Masai say that the Awari was invented by Sindillo, the son of the first man, Maitoumbe, and was originally called Geshe. It is believed to be about 3500 years old [1].

Awari is a game that requires calculation and strategy, with the aim of capturing seeds while keeping to the rules, agreed by the players [13, 16]. Due to the number of strategies and amount of calculation involved, Awari has captured the attention of many Artificial Intelligence researchers and computer scientists [3, 4].

There are up to ten different types of Awari shareware softwares available in the market at the moment. Probably, the most popular software is "Awale" developed by Didier *et al.* of Myriad Software [5].

The aim of this paper is to illustrate the advantage of using more features within the evaluation function rather than increasing the search depth in an endgame such as Awari. Most of the previous work on Awari focuses on implementing improved search techniques, which are sometimes augmented with databases [12].

A Genetic Algorithm (GA) was used to evolve a new Awari player called "Ayo", which uses a mini-max search. A special tournament selection technique made it possible to construct a fitness evaluation mechanism that measures the ability of one player relative to a pool of other players in a population of players. Ayo produced better results with depth 5 searches than Davis *et al.* player, which used a depth 7 search [5]. The principle established in this paper is applicable to other endgames like chess and checkers.

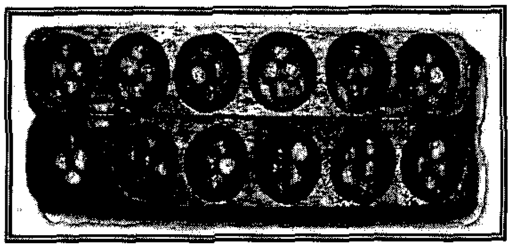
Figure. 1 The Awari board

A. What is Awari?

Awari is a game played by two players. It is traditionally played on a plank of wood with twelve pits. The pits are arranged into two rows (north and south) of six pits each. It has 48 seeds, as shown in Fig.1 [1, 14].

The objective of the game is to capture as many seeds as possible. Each seed is worth one victory point, and when one of the players gets at least 25 seeds he wins.

To start, four seeds are placed in each of the twelve pits on the board. The player on the north side must then choose one of the pits on his side of the board. The player scoops up all four seeds and sows them in a counter-clockwise fashion around the board. This is done by placing one seed in the pit to the right of the original pit, then one seed in the pit to the right of that one, and so on until there are no more seeds left in the player's hand. When a player gets to the last pit on his side, he simply places a seed in his opponent's pit and keeps going counter-clockwise. If a player has enough seeds to go all the way around the board (generally known as *kroo* move), he skips the pit he started with.

Seeds are captured when the last seed sown is placed in an opponent's pit with one or two seeds. In addition, if the previous pit has two or three seeds in it after sowing, those seeds are collected as well. And so on, up to five pits' worth, therefore it is possible to totally wipe out an opponent's side of seeds this way. During the game, players alternate turns; a player never takes two turns in a row.

B. Related Work

Awari is one of the games that were played in the Computer Olympiad [1, 5] until 2000, when the Olympiad was discontinued due to lack of interested participants. The Computer Olympiad is a multi-games event in which all of the participants are computer programs. The purpose of the games is to find the most competitive programs in a number of categories including chess.

In 1990, van der Meulen *et al.* constructed an artificial Awari player called "Lithidion" [2]. Lithidion used an alpha-beta search algorithm and an endgame database containing the game theoretic values of each position with 13 seeds or less.

In 1991, Lithidion performance was improved with a pn-search and a larger database containing the game theoretic values of each position with 17 seeds or less [1]. Van der Meulen *et al.* improved Lithidion performance in 1992 with the use of an Opening Book. These enhancements in the performance of Lithidion enabled it to win and defend the gold medal title in the Computer Olympiad in 1990, 1991 and 1992. After 1992 it was retired.

Lincke developed an Awari player known as "Marvin" [17], which automatically constructs opening books.

Marvin uses a hybrid method called *drop-out expansion* that can mix depth-first and breadth-first search techniques according to the setting of a single parameter.

Van der Goot designed an algorithm to construct a large endgame database for the game of Awari known as "SoftWari" [1]. The algorithm does not employ *reverse moves*, but instead takes multiple passes over the database until internal consistency is achieved. The Awari database contained 52 billion positions, representing 4.5% of the total state space of the game.

Marvin and SoftWari played against each other in the Awari competition held in the 2000 Computer Olympiad [13, 15, 18]. In an eight game tournament, Marvin won the first game, while SoftWari won the second game. The next five games were played so perfectly that they resulted in draws. In the final game Marvin made 13 errors but still won the game and the tournament.

All the work reported thus far focuses on search techniques and database utilization. Generally speaking, these previous players gave little importance to the evaluation function. In contrast, van Rijswijck proposed a GA-based technique that mines endgame databases for relevant features useful in the construction of evaluation functions [3]. The evaluation function was applied to a decision tree in which each node represents a binary decision based on atomic features, where the atomic features describe the current board position. Each board position is statistically mapped to one unique class whose evaluation values are not learned by game playing but extracted directly from the database. The class mapping is done using the evolved decision tree, which allows the evaluation value to be computed quickly.

Davis *et al.* designed an Awari player, which uses a simple evaluation function [5]. The evaluation function is a linear combination of features that represent the current game position. Each feature has a weight associated with it. This weight is evolved using an Evolution Strategy. The output of the evaluation function is used in a mini-max search to determine the next move to be made by the player.

Davis et al. evolved the weights of the evaluation function for a 7-levels deep mini-max search tree. They played each member of the population against every member of the population for 250 generations. The evaluation function used for deciding the next move took into account the number of pits vulnerable to having 2 or 3 seeds captured in the next move, and the current scores of both players. The performance of this program was measured by playing against "Awale". The program beat Awale 5 - 0 at the initiation level, 5 - 0 at the beginner level, 3 - 2 at the amateur level, but lost 0 - 5 in the master level.

II. METHODOLOGY

A. Overview

The name of our Awari player is "Ayo". This name was chosen based on what the game is generally known as in Nigeria. Ayo is designed to be able to compete and win against "Awale".

Ayo uses a mini-max search which employs an evaluation function to asses all possible future moves. Our aim is to prove that increasing the number of features in the evaluation function leads to a reduction in the mini-max search depth. Thereby reducing the response time, the CPU usage, and the amount of memory required during evaluation. The evaluation function used by Ayo is:

$$f = \sum_{i=1}^{12} w_i \times a_i \tag{1}$$

where:

w1..w12	The weights of each term of *f*. They range between [0,1].
a1	The number of pits that the opponent can use to capture 2 seeds. Range: 0 – 6.
a2	The number of pits that the opponent can use to capture 3 seeds. Range: 0 – 6.
a3	The number of pits that Ayo can use to capture 2 seeds. Range: 0 – 6.
a4	The number of pits that Ayo can use to

	capture 3 seeds. Range: 0 – 6.
a5	The number of pits on the opponent's side with enough seeds to reach to Ayo's side. Range: 0 – 6.
a6	The number of pits on Ayo's side with enough seeds to reach the opponent's side. Range: 0 – 6.
a7	The number of pits with more than 12 seeds on the opponent's side. Range: 0 – 6.
a8	The number of pits with more than 12 seeds on Ayo's side. Range: 0 – 6.
a9	The current score of the opponent. Range: 0 – 48.
a10	The current score of Ayo. Range: 0 – 48.
a11	The number of empty pits on the opponent's side. Range: 0 – 6.
a12	The number of empty pits on Ayo's side. Range: 0 – 6.

The evaluation function has twelve features, the first six (a1 - a6) are identical to those used by Davis *et al.* [5], and the rest are added to enhance Ayo's performance.

Using a mini-max search tree, all the possible moves are outlined to the required depth, then the evaluation function evaluates each terminal node.

Each feature has a degree of importance, reflected in the value of the weight associated with it. These feature weights are unknown and there is no mathematical method of calculating them. We use a GA to evolve these weights.

B. Genetic Algorithm Operation

The primary function of our genetic algorithm is to evolve the feature weights of the evaluation function. Figure 2 provides an outline of the operation of the GA.

1) Problem Encoding: A binary encoding scheme was selected for chromosomes. Each chromosome consists of 48 bits, 4 bits for each weight – see Figure 3.

Fifty chromosomes are randomly created to construct the first population. The population size is constant throughout the run.

2) Fitness Evaluation: The fitness of each chromosome is evaluated using a special form of tournament selection which assesses the fitness of each individual relative to its peers. More specifically, twenty percent of the population (ten chromosomes) is chosen randomly to form a *fitness set.* Next, each chromosome in the population plays twice against every chromosome in the fitness set, once as north and once as south. The winning player is awarded 2 points for a win, 1 point for a draw and zero points for a loss. The sum of points awarded to a chromosome in those twenty games is equal to the fitness of that chromosome.

3) Selection and Elitism: Copies of the best ten percent of the population are placed without changes in the *elitism set.* Elitism ensures that the best chromosomes will not be destroyed during crossover and mutation.

The selection process is then implemented. Fitness proportional selection (with replacement) is used to select chromosomes for the *mating pool.* The size of the mating pool equals ninety percent of the population size.

4) Crossover: Single-point crossover is used with probability of 0.5.

5) Mutation: Bit-flip mutation was used with probability of 0.001.

The chromosomes resulting from crossover and mutation are then combined with the elitism set to construct the next generation.

6) *Termination Criteria*: The GA runs for one hundred generations. Next, the weights extracted from the best chromosome in the last generation are used for the fitness function employed by Ayo.

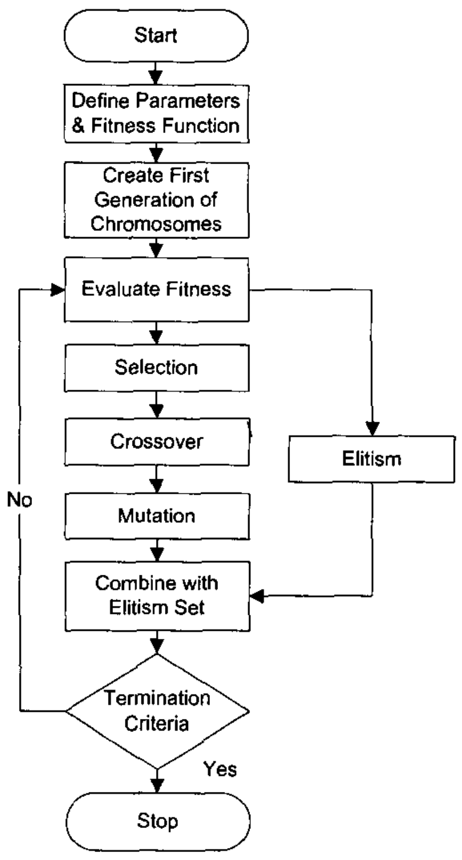

Figure. 2 GA operation

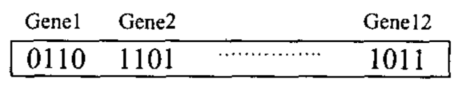

Figure. 3 The chromosome structure

III. RESULT & DISCUSSION

The GA is used to evolve the weights for the evaluation function used in the mini-max search by Ayo. Ayo was evolved with a 3 and 5 level deep mini-max search.

A. Ayo with a mini-max search of depth 3

The values of the optimized weights for a depth 3 mini-max search are listed in Table 1. This table indicates that the number of empty pits on the opponent's side has the highest weight of all the features. This feature is the starting point for executing many seed-capturing maneuvers and is indicative of Ayo's ability to attack the opponent in the future.

TABLE I. THE WEIGHTS OF THE FEATURES IN THE EVALUATION FUNCTION AS GENERATED BY THE GA FOR MINI-MAX SEARCH OF DEPTH 3

Feature	Weight
The number of empty pits on the opponent's side. Range: 0 - 6.	1.00
The number of empty pits on Ayo's side. Range: 0 – 6.	0.67
The number of pits that Ayo can use to capture 2 seeds. Range: 0 - 6.	0.93
The number of pits that Ayo can use to capture 3 seeds. Range: 0 - 6.	0.93
The number of pits that has more than 12 seeds on the opponent's side. Range: 0 - 6.	0.06
The number of pits with more than 12 seeds on Ayo's side. Range: 0 - 6.	0.93
The number of pits on the opponent's side with enough seeds to reach Ayo's side: Range: 0 - 6.	0.13
The number of pits on Ayo's side with enough seeds to reach the opponent's side. Range: 0 - 6.	0.87
The current score of the opponent. Range: 0 - 48.	0.13
The current score of Ayo. Range: 0 - 48.	0.60
The number of pits that the opponent can use to capture 2 seeds. Range: 0 - 6.	0.06
The number of pits that the opponent can use to capture 3 seeds. Range: 0 - 6.	0.20

The number of empty pits on Ayo's side has a weight of 0.67. Ayo's ability to defend itself is highly affected by this feature.

The numbers of seeds that Ayo can use to capture 2 or 3 seeds both have a weight of 0.93. This makes sense since capturing seeds is the main aim of the game.

The number of pits with more than twelve seeds, a feature of relevance to the *kroo* strategy has a weight of 0.93.

The number of pits with enough seeds to reach the other side is given a weight of 0.87. This feature is very important because it assists Ayo identifying its offensive strength.

The evaluation function employed by Ayo consists of other features such as the scores of both players at each instant. Ayo's score was given a higher weight (0.6) than its opponent's score (0.13).

Finally, the numbers of pits that the opponent can use to capture 2 or 3 seeds are assigned 0.67 and 0.20 weights respectively. Both features affect Ayo's ability to be defeated and reflect its bias towards attack rather than defense.

To evaluate the performance of Ayo, Ayo played 10 games at each level of the four levels of "Awale". The results are presented in table 2.

Ayo won all the games at the initiation level; at the beginner level Ayo won 5 of the 10 games and drew the remaining 5 games. Ayo portrays its ability to use *Kroo* moves by turning around 3 games to win them at the amateur level; it won 4, drew 3 and lost 3 games. Ayo was not able to win any game at the grand master level but it was able to capture a decent average of 16 seeds in the 10 games.

Davis *et al.* [5] developed an evaluation function that focuses on the number of seeds that could be captured by both players and their score. Their results (against Awale) are shown in table 3 below.

Comparing the results of Davis *et al.* (table 3) to Ayo's results, shows that the enhancement of the evaluation function makes up for the low search depth of Ayo.

TABLE II. AYO'S RESULTS FOR MINI-MAX SEARCH OF DEPTH OF 3

Level	Average Moves in Game (SD*)	Average Ayo' s Score (SD)	Average Awale's Score (SD)	Overall Score (%) W:D:L
Initiation	49.9 (0.33)	26.9 (0.05)	11.2 (0.35)	100:0:0
Beginner	102 (0.49)	23.2 (0.13)	16.6 (0.42)	50:50:0
Amateur	68.4 (0.54)	23.7 (0.18)	20.2 (0.31)	40:30:30
Grand Master	49.4 (0.16)	16.4 (0.15)	29.8 (0.05)	0:0:100

*SD stands for Standard Deviation

TABLE III. RESULTS OF DAVIS *ET AL.* FOR MINI-MAX SEARCH OF DEPTH OF 7

Level	Average Moves in Game (SD)	Average Score (SD)	Average Awale's Score (SD)	Overall Score (%) W:D:L
Initiation	47.40 (16.64)	29.80 (2.28)	2.80 (2.59)	100:0:0
Beginner	55.80 (22.74)	26.20 (4.44)	7.80 (3.03)	100:0:0
Amateur	108.20 (35.35)	24.20 (10.01)	16.80 (8.93)	60:20:20
Grand Master	80.00 (5.48)	4.40 (0.55)	26.68 (1.64)	0:0:100

Table 2 and 3 show very little difference in the result between Davis *et al.* depth 7 mini-max search and Ayo depth 3 mini-max search. The results at other levels exhibit the same similarity in terms of average scores. The most significant result is that of the grand master level, where

Davis *et al.* player captured an average of 4.40 seeds, while Ayo captured 16.4 seeds, on average.

B. Ayo with a mini-max search of depth 5

The result of the Ayo player with depth of 5 is depicted in the table 5, where table 4 contains the list of values of the weights used to achieve the results in table 5. The results depicted in table 5 show improvements in the number of wins per level.

TABLE IV. THE WEIGHTS OF THE FEATURES IN THE EVALUATION FUNCTION AS GENERATED BY THE GA FOR MINI-MAX SEARCH OF DEPTH 5

Feature	Weight
The number of empty pits on the opponent's side. Range: 0 – 6.	0.00
The number of empty pits on Ayo's side. Range: 0 – 6.	0.80
The number of pits that Ayo can use to capture 2 seeds. Range: 0 - 6.	0.06
The number of pits that Ayo can use to capture 3 seeds. Range: 0 - 6.	0.00
The number of pits that has more than 12 seeds on the opponent's side. Range: 0 - 6.	0.00
The number of pits with more than 12 seeds on Ayo's side. Range: 0 - 6.	0.20
The number of pits on the opponent's side with enough seeds to reach Ayo's side. Range: 0 - 6.	0.87
The number of pits on Ayo's side with enough seeds to reach the opponent's side. Range: 0 - 6.	0.60
The current score of the opponent. Range: 0 - 48.	0.73
The current score of Ayo. Range: 0 - 48.	0.93
The number of pits that the opponent can use to capture 2 seeds. Range: 0 - 6.	0.80
The number of pits that the opponent can use to capture 3 seeds. Range: 0 - 6.	1.00

TABLE V. AYO'S RESULTS FOR MINI-MAX SEARCH OF DEPTH OF 5

Level	Average Moves in Game (SD)	Average Ayo's Score (SD)	Average Awale's Score (SD)	Overall Score (%) W:D:L
Initiation	53.6 (0.27)	26.5 (0.07)	8.3 (0.53)	100:0:0
Beginner	121.6 (0.45)	26.3 (0.09)	11.7 (0.34)	100:0:0
Amateur	140 (0.47)	24.8 (0.40)	15.5 (0.40)	70:10:20
Grand Master	51.1 (0.19)	6.4 (0.27)	26.5 (0.10)	0:0:100

At the initiation level, Ayo won all the games. At the beginner level, Ayo won all the games. At the amateur level, Ayo won seven games (10% *more* than Davis' *et al.* player), drew one and lost one game. Finally, at the grand master level, Ayo lost all the games (but still managed to return a higher score than Davis' *et al.* player). Hence, what is significant is that, in comparison to Davis *et al.* player (with 7 levels of search depth), Ayo with 5 levels of search depth, still managed to win more games and collect more points.

IV. CONCLUSIONS & RECOMMENDATIONS

Problem solving is one of the main purposes of Artificial Intelligence research. Problem Solving consists of two sub-processes: choosing an appropriate representation and performing a search. Knowledge representation should include analysis, conceptualization and formalization. A good knowledge representation may considerably reduce the amount of search needed to solve a problem, while a poor representation may render solving the problem impossible. Therefore, choosing a representation should have a high priority in problem solving [10].

In this paper, we illustrate the importance of problem domain representation, using our own Awari playing program: *Ayo*. We use a Genetic Algorithm to optimize the weights of the feature evaluation function of Ayo. We play Ayo against a commercially available Awari player, then compare Ayo's results to the results achieved by an older Awari player; one that uses a 7-levels deep mini-max search. Ayo, with a 5-levels deep mini-max search, returns better results, due to better more intelligent representation of the state space.

Game-players that execute shallower searches require fewer resources and run faster than game-players that require deeper searches. Hence, it should be possible, in principle, to speed-up many two-player endgames, such as chess and checkers by including better more intelligent features of the current state of the game, rather than executing ever deeper searches of future moves trees. After all, the best chess players in the world have very limited search capabilities compared to even the humble PC!

REFERENCES

[1] The university of Alberta computer Awari Group, http://www.cs.ualberta .ca/~awari/.

[2] V. Allis, M. van der Muellen, and J. van den Herik, "Proof-number Search", *Artificial Intelligence*, vol. 66, pp.91-124, 1994.

[3] J. Van der Rijswijck, "Learning from Perfection: A Data Mining Approach to Evaluation Function Learning in Awari". *In proceedings of the 2nd International Conference (CG-00),* Hamamatsu, vol. 2063, pp.115-132, 2001.

[4] J. W. Romein and H. E. Bal, "Awari is Solved". *Journal of the International Computer Games Association (ICGA)*, vol. 25, pp.162-165, 2002.

[5] J. E. Davis and G. Kendall, "An Investigation, using Co-Evolution, to evolve an Awari Player". *In proceedings of Congress on Evolutionary Computation (CEC2002)*, Hawaii, pp.1408-1413, 2002.

[6] C.R. Darwin, *The Origin of Species*, Vol.11, Ch. 4, P.F. Collier & Son, New York, 1909.

[7] J. Cogleyd, "Designing, implementing and optimizing an object-oriented chess system using a genetic algorithm in

Java and its critical evaluation ", M:S thesis, The Open University, London, England, 2001.

[8] A. S. Fraenkel, "Combinatorial Games: Selected Bibliography with a Succinct Gourmet Introduction", *Games of No Chance*, vol. 29, MSRI Publications, Cambridge University Press, pp.493-537, 1996.

[9] M. Page-Jones, *Fundamentals of Object-Oriented Design in UML*, pp. 13,33 Dorset House Publishing, New York, 2000.

[10] S. Russell and P. Norvig, *Artificial Intelligence*, New Jersey: Prentice Hall, Ch. 3-5, pp.55-149, 1995.

[11] P. S. Wang, *Standard C++ with Object-Oriented Programming*, Pacific Grove: Kent State University, 2001.

[12] Awari Statistics, http://awari.cs.vu.nl/statistics.html.

[13] The rules of Awari, http://www.cs.ualberta.ca/~awari/rules.html.

[14] Awari-a game from Africa, http://www.guidezone.skl.com/ liz_africa_games.htm.

[15] computer Olympiad 2000-Awari, http://www.dcs.qmul.ac.uk/ ~olympiad/awariresult.htm.

[16] Oware by many other names, http://www.svn.net/rkovach/ oware/owarname.htm.

[17] Thomas R. Lincke, Strategies for the Automatic Construction of Opening Books: Computers and Games 2000: 74-86.

[18] Mind sport Olympiad, Lonndon, 2000, http://www.msoworld.com/history2000/jpg/c-omput.htm.