A REFINEMENT-BASED HEURISTIC METHOD FOR DECISION MAKING IN THE CONTEXT OF *AYO* GAME

BY

AKINYEMI, Ibidapo Olawole
(CUPG040055)
B.Sc (Mathematical Sciences (Computer Science Option)), University of Agriculture, Abeokuta, 1997
M.Sc. (Computer Science), University of Agriculture, Abeokuta, 2004

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES, SCHOOL OF NATURAL AND APPLIED SCIENCES, COLLEGE OF SCIENCE AND TECHNOLOGY, COVENANT UNIVERSITY, OTA, OGUN STATE, NIGERIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DOCTOR OF PHILOSOPHY DEGREE IN COMPUTER SCIENCE

SEPTEMBER, 2012

CERTIFICATION

This is to certify that this thesis is an original research work undertaken by **Ibidapo Olawole Akinyemi** and approved by:

1. Name: **Professor Harrison O. D. Longe**

   Supervisor

   Signature / Date:..... 14/11/12

2. Name: **Professor Ezekiel F. Adebiyi**

   Co-Supervisor

   Signature / Date:.....

3. Name: **Professor Charles K. Ayo**

   Head of Department

   Signature / Date:..... 27/12/12

4. Name: **Professor J. O. A. Ayeni**

   External Examiner

   Signature / Date:..... 15/11/12

DECLARATION

It is hereby declared that this research was undertaken by Ibidapo Olawole Akinyemi. The thesis is based on his original study in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ota, under the supervision of Professor H. O. D. Longe and Professor E. F. Adebiyi. Ideas and views of this research work are products of the original research undertaken by Ibidapo Olawole Akinyemi, and the views of other researchers have been duly expressed and acknowledged.

Professor H. O. D. Longe

Supervisor

Signature / Date..... 14/4/12

Professor E. F. Adebiyi

Co-Supervisor

Signature / Date..... 23-5-12

DEDICATION

This thesis is dedicated to my late father, Elder Pa Michael Akinyemi Olasemojo.

ACKNOWLEDGEMENTS

I am deeply indebted to God, the author of wisdom and understanding for His faithfulness and generous endowments of grace that saw me through my doctoral studies. My deep and sincere appreciation goes to the Chancellor, Covenant University, Dr. David O. Oyedepo and the members of the Board of Regents of Covenant University for the vision and mission of the school. Also, special thanks to the management staff of the University: the Vice Chancellor, the Registrar, the Deans of the Colleges, the Heads of Departments and other principal officers for their commitment to the pursuit of excellence.

My earnest appreciation goes to my supervisor, Professor Harrison O. D. Longe in his capacity as my supervisor who provided the research direction and qualitative guidance for the work. I am deeply appreciative of Professor Ezekiel F. Adebiyi, my co-supervisor, for his critical review of the work and useful suggestions through his insistence on quality till the completion of this work.

I shall forever be grateful to my late father, Elder Pa Michael Akinyemi Olasemojo for giving me a solid educational foundation. His belief in education for success has kept me resilient in pursuing the best of education despite all the financial constraints. May his soul rest in perfect peace. I appreciate my caring mother, Mrs. Janet Olauli Akinyemi for enduring all the hardship she went through right from my childhood till date in order to see that her children are successful. You are indeed a mother. I must not fail to appreciate my uncle, Mr. Samuel Omopariola Odimayo for his words of encouragement

and for constantly motivating me in the pursuit of excellence even though I did not get a clear picture of his messages at the initial stage of my undergraduate days. You are really a role model.

I want to specially thank Professor Charles K. Ayo for all his fatherly advice, support and encouragements. I thank Professor O.O. Olugbara for all his contributions to this Ph.D work. I appreciate the contributions of Professor T. O. Adewoye most especially for the provision of a number of materials for studying the concept of Completely Determined Game. I thank all my friends and colleagues in the Department of Computer and Information Sciences for their support and encouragement all through the course of this work.

Finally, this acknowledgment would not be complete if I fail to appreciate my loving, ever caring, prayerful, hardworking, resilient and encouraging wife, Mrs. Caroline Olasumbo Akinyemi, who has been a source of encouragement throughout all the period of my postgraduate studies. She has been a balm to my bones. I sincerely thank you for taking good care of me and our children. You will surely live to reap the fruits of your labour in Jesus name. My children, Modupe, Abimbola, Omolola and Omotola, I appreciate all of you for your prayers. I remain grateful to Pastor & Mrs. J. O. Odufejo for all the supports given during my programme, God will open heaven for your family in all areas of your needs. I appreciate Mrs. J. M. Osinowo for all her words of encouragement and support. Mr. Iseoluwa Elkanah, Pastor Ogungbemi, Chief Sanya, and a host of others that I could not mention, I thank you all.

ABSTRACT

Games of strategy, such as chess have served as a convenient test of skills at devising efficient search algorithms, formalizing knowledge, and bringing the power of computation to bear on "intractable" problems. Generally, minimax search has been the fundamental concept of obtaining solution to game problems. However, there are a number of limitations associated with using minimax search in order to offer solution to *Ayo* game. Among these limitations are: (i.) improper design of a suitable evaluator for moves before the moves are made, and (ii.) inability to select a correct move without assuming that players will play optimally. This study investigated the extent to which the knowledge of minimax search technique could be enhanced with a refinement-based heuristic method for playing *Ayo* game. This is complemented by the CDG (an end game strategy) for generating procedures such that only good moves are generated at any instance of playing *Ayo* game by taking cognizance of the opponent strategy of play. The study was motivated by the need to advance the African board game – *Ayo* – to see how it could be made to be played by humans across the globe, by creating both theoretical and product-oriented framework. This framework provides local *Ayo* game promotion initiatives in accordance with state-of-the-art practices in the global game playing domain. In order to accomplish this arduous task, both theoretical and empirical approaches were used. The theoretical approach reveals some mathematical properties of *Ayo* game with specific emphasis on the CDG as an end game strategy and means of obtaining the minimal and maximal CDG configurations. Similarly, a theoretical analysis of the minimax search was given and was enhanced with the Refinement-based heuristics. For the empirical approach, we simulated *Ayo* game playing on a digital

computer and studied the behaviour of the various heuristic metrics used and compared the play strategies of the simulation with AWALE (the world known *Ayo* game playing standard software). Furthermore, empirical judgment was carried out on how experts play *Ayo* game as a means of evaluating the performance of the heuristics used to evolve the *Ayo* player in the simulation which gives room for statistical interpretation. This projects novel means of solving the problem of decision making in move selections in computer game playing of *Ayo* game. The study shows how an indigenous game like *Ayo* can generate integer sequence, and consequently obtain some self-replicating patterns that repeat themselves at different iterations. More importantly, the study gives an efficient and usable operation support tools in the prototype simulation of *Ayo* game playing that has improvement over Awale.

CONTENTS

CHAPTER 2: LITERATURE REVIEW

CHAPTER 3: *AYO* GAME

CHAPTER 4: HEURISTIC DECISION MAKING SYSTEM FOR PLAYING *AYO* GAME

LIST OF FIGURES

LIST OF TABLES

LIST OF NOTATIONS

CDG .....Completely Determined Game

GAs.....Genetic Algorithms

RBH.....Refinement – Based Heuristic

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND INFORMATION

Game playing has a long history within Artificial Intelligence (AI) research, and it has been a very popular machine learning research domain of AI. Basically, games have existed among many ancient peoples and are known in all contemporary human cultures. It has been suggested that the playing of games is one of the keys to defining the characteristics of man. The knowledge of game playing strategies that is gained through interaction with entertaining games such as; chess, poker, tic-tac-toe, and so on has found relevance in many real life applications. Examples of these include politics i.e. political game (Ajayi, 2007), the competition between firms, the conflict between government and labour, the fight to get bills passed from congress, the power of the judiciary, war and peace negotiations between countries, and many more, all of which are instances of game playing in action. Similarly, there are biological games, such as the competition between species, where natural selection can be modeled as a game played between genes (Smith, 1982).

The major goal in defining and examining game scenarios is to find good strategies as solution to the game. The solution is a recommendation to the players on how to play the game, and it is given as a tuple of strategies. This has given birth to what is called game-playing in computer science, which has been well studied as a formal intelligent task in AI. Game design is one of the challenging problem areas of AI research. Games

provide a useful domain to study machine learning and other AI techniques while providing a structured problem space in which optimization algorithms can be applied to search for solution. A lot of game-playing programs have been developed in the past decades among which Samuel's checkers program (Samuel, 1959; 1967) and Tesauro's TD-gammon (Tesauro, 1995) were important breakthrough.

Ayo game is the most popular board game among the Yorubas who are mainly in the south-western states of Nigeria and parts of Republic of Benin. *Ayo* is a member of a family of board games called Mancala, which is widespread in the tropical and sub-tropical regions of Asia, Africa, and the adjacent Islands (Adewoye and Awoniyi, 1985). It is a game of perfect information known as combinatorial games (Fraenkel, 1996). It is a two – player game, with no hidden information, no chance move, a restricted outcome (win, lose and draw) and with each player moving across the board. Several methods have been explored to solve *Ayo* game, but due to its complexity and irregular patterns as the play of the game progresses, the exact solution for the game has not hitherto been found. In view of this, the game has captured the attention of many AI researchers, mathematicians and computer scientists. In this work, effort was made to show how a refinement-based heuristic machine learning approach can assist the minimax search algorithm to produce an *Ayo* player that can play at a reasonable level.

Refinement is a mapping that accepts a set of moves and then evaluates each move and returns a move with the best advantage. In order to classify *Ayo* game strategies (moves), a refinement technique is pertinent, which was incorporated into the minimax

search algorithm so as to enhance the evaluation and selection of move processes in *Ayo* game playing.

1.2 STATEMENT OF THE RESEARCH PROBLEM

The context of this research is computer–game playing. Game playing is an exciting exercise and its notion has been developed as a model of computation (Olugbara, *et al.*, 2006). Generally, minimax search has been the fundamental concept of obtaining solution to game problems. However, there are a number of limitations associated with using minimax search. These are:

i. improper design of a suitable evaluator for moves before the moves are made, and

ii. inability to select a correct move without assuming that players will play optimally.

It is our credence that eliminating these limitations would improve the playing of *Ayo* game. For example, in a game scenario, a player can be irrational in move selection as a play strategy like bluffing as applicable in *Ayo*. Bluffing is a powerful play strategy and is defined as the ability to tradeoff invaluable seed(s) so as to gain advantage. Hence, it involves sacrificing immediate reward to obtain a greater reward in the long term. But two important factors that must be taken into consideration when bluffing are:

i. when to bluff, and

ii. the number of seeds (i.e tradeoff seeds) to sacrifice.

Consequently, the effect of a single move can be so large that it becomes incalculable for human in competitive situations. In order to construct an efficient evaluator for the minimax search, a refinement-based heuristic technique is imperative. This consideration will lead us to giving answers to the following research questions as searchlight for the research work.

1. Under what conditions can minimax search improve the computer-game playing?
2. How can refinement-based heuristic approach be used to assist minimax search algorithm to produce an *Ayo* player that can play at a reasonable level?
3. How can these search methods be implemented efficiently?
4. How can a winning strategy in the playing of *Ayo* be obtained?

1.3 AIM AND OBJECTIVES OF THE RESEARCH

The aim of this research work is to evolve a machine-*Ayo* game player that can emulate human expertise in Ayo game playing in order to deepen the understanding of human intelligent processes through computer simulations.

The specific objectives of this research are to:

i. develop and describe Completely Determined Game (CDG) as an endgame winning strategy and present some mathematical characterizations of the CDG strategy that can guarantee a player winning in any tournament; and

ii. simulate the playing of *Ayo* game by applying a refinement-based heuristic method for generating procedures such that only good moves are generated at any instance of play.

1.4 RESEARCH METHODOLOGY

In an attempt to accomplish the stated objectives of this research work, both theoretical and empirical approach was used. The theoretical approach reveals some mathematical properties of *Ayo* game with specific emphasis on the CDG as an end game strategy and means of obtaining the minimal and maximal CDG configurations. Moreover, a theoretical analysis of the minimax search was given which was enhanced with the Refinement-based heuristics. But due to the fact that most game-tree search algorithms are associated with high complexity, theoretical approach alone may not be sufficient to obtain true solution for game problems hence, an empirical judgment is required.

For the empirical approach, we simulated *Ayo* game playing on a digital computer and studied the behaviour of the various heuristic metrics (Canberra, Angular, and Correlation (Kristof,2004)) used and compare the play strategies of the simulation with AWALE (the world standard known as at today (Didier & Olivvier, (1996)). Furthermore, we carried out empirical judgment on how experts play *Ayo* game as a means of evaluating the performance of the heuristics used to evolve the *Ayo* player in the simulation which gives room for statistical interpretation.

For clarity, we separated the descriptions of our theoretical and empirical approach. The theoretical research is described in chapters three and four while the detailed empirical approach is given in chapter five.

The work is implemented in C++.

1.5 MOTIVATION FOR THE WORK

Games have long been seen as human universals, and for some thousands of years humans have created board games and competed against each other to see who has the strongest mind. With the invention of computers, scientists in the field of Computer Science and Artificial Intelligence have tried to develop computer programs which are capable of winning against a human player. A number of successes have been recorded for a list of games. Notable among the board games is chess, which has been revolutionalised by the Internet as players play each other across the globe. In the same notion, the motivation for this work stems from the desire to bring the global awareness of Ayo game to a level comparable to that enjoyed by other regular games

1.6 CONTRIBUTION TO KNOWLEDGE

The research provides transference of skills in the design of human computer interfaces in game playing domain. The contribution of this research work is in two folds:

i. The study presents how a subgame of an indigenous game called *Ayo* can generate integer sequence, and consequently obtain some self-replicating patterns

that repeat themselves at different iterations which we hope will attract the attention of number theorists in academic arena on further studies of the characteristics of the game.

ii. The study investigates the extent to which the knowledge of minimax search algorithm could be enhanced with refinement-based heuristic algorithm vis-à-vis the concept of CDG for the purposes of creating procedures such that only best possible move is generated under a given set of play conditions, while incorporating the opponent's game strategy in its decision making process

1.7 THESIS ORGANIZATION

This thesis is organized into six chapters. Chapter one gives a general introduction to the study. It presents the background information to the study and also highlights the statement of the problem solved. The aim and objectives achieved by the work are also presented. Furthermore, the chapter presents the methodology with which the objectives were achieved. The motivation for embarking on the study as well as the contribution made is also presented in this chapter.

Chapter two presents a critical review of relevant literature for the study while chapter three presents a detailed study of *Ayo* game. In chapter four, details of proposed game architecture for the prototype simulation, its design as well as experimental tests and results are presented. Chapter five contains the results of the evaluation carried out on the developed prototype simulation and chapter six presents the summary of the work, the conclusion, as well as the opportunities for future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 OVERVIEW OF GAME THEORY

Game theory is a branch of applied mathematics that studies situations where intelligent and rational players need to choose actions that maximize their returns (Anthony, 2004). It is the study of intelligent decision making in a situation where the gain (or loss) depend not just on what is done, but what others do (Adeday*o*, 2005). Game theory provides analytical tools for examining strategic interactions among two or more participants (Smith, 2003). It is concerned with decision making in organizations where the outcome depends on the decisions of two or more autonomous players, one of which may be nature itself, and where no single decision maker has full control over the outcomes (Kreeps, 1990). From the definitions above one could therefore see game theory as a "science of strategic decision making in situations where parties compete, and possibly cooperate, to influence the outcomes of the parties' interaction to each party's advantage". Basically, game theory aims to find the optimal solutions to situations of conflict and cooperation under the assumption that players are instrumentally rational and act in their own best interests. It offers an interesting perspective on the nature of strategic selection in both familiar and unusual circumstances.

Game theory was conceived in the seventeenth century by mathematicians that attempted to solve the gambling problems of idle French nobility (Colman, 1982). In 1713, an Englishman, James Waldegrave, wrote a letter in which he solved a version of the card game *Ie Her* with a mixed strategy based on minimax. A century later, in 1838, Antoine Augustin Cournot published "Researches into the Mathematical Principles of the Theory of Wealth," which dealt with the dynamics of an economic duopoly seen as a game. Game theory in the modern era was ushered in with the publication in 1913, by the German mathematician, Ernst Zermelo, where he proved that every competitive two-person game possesses a best strategy for both players, provided both players have complete information about each other's intentions and preferences. Zermelo's theorem was quickly followed by others, most notably is the minimax theorem, which states that there exists a strategy for each player in a competitive situation such that none of the players regrets their choice of strategy when the game is over. The minimax theorem then became the fundamental theorem of game theory, although its genesis predated Zermelo by two centuries. The minimax theorem was later proved for the general case in December 1926, by the Hungerian mathematician, John Von Neumann. The complicated proof, published in 1928 by (Von Neumann, 1928) was subsequently modified in 1937 (Von Neumann, 1937). This made the development of game theory attributed to John Von Neumann. Von Neumann's work culminated in a fundamental book on game theory written in 1944, in collaboration with Oskar Morgenstern entitled "*Theory of Games and Economic Behaviour"* (Von Neumann and Morgenstern, 1953)

Nash (1951), succeeded in generalizing the minimax theorem by proving that every competitive game possesses at least one equilibrium point in both mixed and pure strategies, and named the equilibrium points as "Nash Equilibrium" that represents the solution for the game. Nash's solution established game theory as a glamorous academic pursuit, and has expanded dramatically. In 1953, Harold Kuhn, removed the two-person zero-sum restriction from Zermelo's theorem, by replacing the concept of best individual strategy with that of the Nash Equilibrium. He proved that every *n*-person game of perfect information has equilibrium in pure strategies and, as part of that proof, introduced the notion of sub-games. This too became an important stepping-stone to later developments in game theory.

Games have existed among many ancient peoples and are known in all contemporary human cultures. It has been suggested that the playing of games is one of the keys defining characteristics of man. Games elicit a strong imaginative response, and thus have come to occupy a prominent place among the metaphors, which have been employed for human life. A game is a mathematical entity consisting of a set of players, a set of moves (strategies) available to players, and a set of pa*yo*ffs specified for each combination of strategies (Njoku, 2004).

Game occurs in diverse ways. For example, entertaining games, such as, chess, poker, tic-tac-toe, bridge, computer game, and so on are known today. Moreover, there is a vast area of economic games (Myerson, 1991; Kreeps, 1990), political games (Ordeshook, 1996; Shubik, 1982; Taylor, 1995; Ajayi, 2007). The competition between firms, the

conflict between government and labour, the fight to get bills from congress, the power of the judiciary, war and peace negotiations between countries, and so on, all provide examples of games in action. There are also psychological games played on a personal level, where the weapons are words, and the payoffs are good or bad feelings (Berne, 1964). There are biological games, the competition between species, where natural selection can be modeled as a game played between genes (Smith, 1982). The major goal in defining and examining game scenarios is to find good strategies as solutions to the game. The solution is a recommendation to the players on how to play the game, and is given as a tuple of strategies. This has given birth to what is called game-playing in computer science, which has been well studied as an intelligent task in AI.

The basic constituents of any game are its participating autonomous decision makers, called "players". A player can be an individual person, an organization, or nature itself (Anthony, 2004). A game must have two or more players, one of which may be nature. The total number of players may be large, but must be finite and a priori deterministic. An outcome is the result of a complete set of strategic selection by all the players in a game and it is assumed that players have consistent preferences among the possibilities. Similarly, it is equally assumed that individuals are capable of arranging these possible outcomes in some order of preference.

There are a few basic elements, which are common to all games, such as:

i. **Specific objectives:** For a game condition to exist there must be a clear objective and the possibility of winning or losing, or at least, of something of value being at stake.

ii. **Rules**: A game must possess rules, delineating the powers and limitations of players, though the rules may not be completely known to the players.

iii. **Visualization**: A game must be **visualizable,** that is, it must be possible to picture what is going on, and must possess certain simplicity or elegance.

iv. **Playability:** Finally, a game must be playable. It must have manageable mechanics of play..

In order to play a game, a player must possess two characteristics (Kreeps, 1990): interest in the objectives of play, and sufficient intelligence to understand the consequences of possible lines of play (though not necessarily fully). More than one type of intelligence may be required. All games require at least some degree of abstract intelligence, while many also require sophistication, judgment (particularly where the human factor is important), creativity, or a combination of these. Computers have come to possess impressive abstract abilities, particularly in game playing, to which considerable effort has been devoted by researchers in artificial intelligence.

On the account of the preceding paragraph, there have been several research works on computer game-playing and many successes have been recorded. Nowadays, computers are able to play chess and other games (like checkers, Othello, and backgammon) at world-champion level. Moreover, a number of game problems have been solved by computers (e.g connect-four and awari). This study attempts to model the method of

playing *Ayo* game with a machine learning technique, develop, and describe a winning strategy that can always guarantee a player winning in any competition.

2.1.1 Game Classification

There are three categories of games; game of skill, game of chance, and game of strategy (Anthony, 2004). Games of skill are one-player games whose defining property is the existence of a single player who has complete control over all the outcomes. Games of chance are one-player games against nature. Unlike games of skill, the player does not control the outcomes completely and strategic selections do not lead inexorably to certain outcomes. The outcomes of a game of chance depend partly on the players' choices and partly on nature, which is the second player. Games of chance are further categorized as either involving risk or involving uncertainty. Games of strategy are games involving two or more players not including nature, each of which has partial control over the outcomes. They are games involving uncertainty since the players cannot assign probabilities to each other's choices (Colman, 1982). They can be sub-divided into two-player games and multi-player games. Within each of the two sub-divisions, there are three further sub-categories depending on the way in which the payoff functions are related to one another – whether the player's interests are completely coincident, completely conflicting, or partially coincident or partially conflicting. The game of strategy in which the player's interests coincide are called cooperative games of strategy while the one in which the player's interests are conflicting (i. e. strictly competitive games) are known as Zero-sum games of strategy. They are so called because the payoffs always add up to zero for each outcome of a fair

game, or to another constant if the game is biased. Games of strategy in which the interests of players are neither fully conflicting nor fully coincident are called mixed-motive strategy. The general taxonomy of the game theory can be found in (Anthony, 2004)

2.2 GAME CONCEPTS

A game is a mathematical entity consisting of a set of players, a set of moves (strategies) available to the players, and a set of payoffs specified for each combination of strategies. Generally, players are assumed to be intelligent and rational, that is consistently pursuing the goal of maximizing their own payoffs. Basically, a game can be defined as $G = \{N, S_i, U_i\}$, where N is the number of players, each player ***i*** has a possible set of strategies S_i, and $U_i(s_i, s_j)$ is a function that gives a payoff for every strategy profile (s_i, s_j) consisting of player i's own strategy and strategies of other player $s_i = (s_1, s_2, \ldots, s_{-1}, s_{i+1}, \ldots s_n)$. This notion spawns a number of ways for which game could be represented depending on the particular game strategy.

2.2.1 Game Representations and Optimal Strategies

There are two main mathematical forms used in the study of games: the strategic (or normal) form, and the extensive form (Ferguson, 1967). Before going into the details of these forms, the following definitions are imminent (Adedayo, 2005):

i. ***Player or Decision Maker*:** This is an active participant in a game. Player can be individual, company, team or inanimate object. This is a person in the committee who makes the final choice among alternatives. A decision maker

is therefore a player in the game. A game with two decision makers or players is called a two-person game while that with more than two is called a multi-player game.

ii. ***Moves*:** This is either a decision by a player or a chance event. Each player knows the moves available to other players.

iii. ***Games*:** This is a sequence of moves that are defined by a set of rules that governs the players' moves.

iv. ***Conflict*:** A state of affair in which two or more parties claim what they cannot have simultaneously e.g value conflict, conflict of interest.

v. ***Strategy*:** This is the description of the decision that a player will make at all possible situations arising in the game. It can also be regarded as any plan that a player has the chance of selecting.

vi. ***Payoffs*:** They are numerical values (or returns) received by players at the end of a game. They are associated with combinations of actions taken by the player. If a game has a random outcome we talk of expected payoff.

vii. ***Minimax*:** This is the minimum of the maximum payoff a player can get from possible strategies available to him.

viii. ***Maximin*:** This is the maximum of the minimum of payoff

ix. ***Saddle Point*:** This is the point of intersection of the maximin and minimax strategies. A game that has a saddle point is said to be stable.

2.2.1.1 Strategic (or Normal) Form Game

One of the basic ways of describing a game is called the strategic (or normal) form. A game in normal form is a table, comprising of tuples of pure strategies, specifying payoffs for each player resulting from a combination of strategies. Solutions for game in this form are found using methods that check for saddle points (maximin criterion), dominance, pure (deterministic), mixed, and equilibrium strategies (Njoku, 2004). A game in strategic form is said to be **zero-sum** if the sum of the p*ayo*ffs to the players is zero no matter what actions are chosen by the players. That is, the game is zero-sum if

$$\sum_{i=1}^{n} f_1(a_1, a_2, ..., a_n) = 0, \ \forall \ \ a_1 \in A_1, a_2 \in A_2, ..., a_n \in A_n .$$

Two-person Zero-sum game is a game that deals with competitive or conflict situations involving two players such that the sum of wins and losses on each set of the strategies in the game is Zero (Adedayo, 2005). This means that whatever player one wins, is what player two losses, and vice versa. Two-person Zero-sum game can be represented by a matrix of the form (Ayeni and Longe, 1985):

$$A = \begin{pmatrix} a_{11}, a_{12}, & \dots & a_{1n} \\ . & & \\ . & & \\ . & & \\ a_{m1}, a_{m2}, & \dots & a_{mn} \end{pmatrix} \quad where \ \ a_{ij} = A(x_i, \ y_j)$$

i.e each player has a utility for each (x_i, y_j) pair of actions with player **I** having $A_1(x_i, y_j)$ utility and player **II** having $A_2(x_i, y_j)$. The predominant solution concept for normal form games is the Nash Equilibrium (NE). This is a strategy profile in which each player plays a best response to the play of others, and no player gets better pa*yo*ff by unilaterally changing his strategy.

2.2.1.2 Extensive-form Game

An extensive-form game is an explicit description of the sequential structure of decision problems encountered by the players in a strategic situation. The model gives rise to game solutions in which each player can consider plan of actions not only at the beginning of the game, but also at any point of time at which decision has to be made.

The extensive-form of a game is a mathematical model of the game built on the basic notions of position and move (Ferguson, 1967). It is a game in which players move sequentially and the order of move matters. That is, it is a multi-stage game in which players take turn instead of making a simultaneous move. The Prisoner's Dilemma (PD) is an example of a simultaneous game. The extensive-form of a game conveys more information about the game. It tells exactly which player should move, when, what are the choices, the outcomes, the information of the players at every stage, and the pa*yo*ff each player receives when called upon to move. The foundation of extensive form game is centered on three concepts namely game tree, chance move, and information set.

2.2.1.3 Game Tree

A convenient representation of a game in extensive form is by the use of a tree. A tree is a graph where any two nodes are connected by exactly one path. Generally, a tree with n-nodes has n-1 arcs (or paths). A game tree models the behaviour of a two-player game. The vertices are often referred to as ***nodes*** and the edges as ***branches***. In extensive form of a game, play starts at the initial vertex and continues along one of the paths,

eventually ending in one of the terminal vertices. At the end of the vertices, the rule of the game specifies the pa*yo*ff.

A game tree algorithm computes the root successor with the highest pa*yo*ff for the current player or the minimax value of a game tree from which the best move can easily be inferred. A large number of game playing algorithms use game tree to represent game positions and moves. Nodes of the tree are game positions and the root node corresponds to the current game position. Branches of a node represent legal moves from the position represented by the node and a leaf node has no successor. Generally, a leaf can be evaluated as a win, lose, draw or a specific score value accordingly, using the rules of the game. The total number of nodes in game tree (i.e. size of the tree) is approximately $\mathbf{W^D}$, where **W** stands for branching factor and **D** is the average game length. The value 2.8×10^{11} is approximately the state-space complexity of *Ayo* (Donkers *et al.*, 2003). The problem is that no practical algorithm can manage a full tree due to time demand and memory limitation and consequently, true minimax search is conspicuously expensive for some games like *Ayo*.

2.3 HEURISTIC SEARCH

More often than not, game trees for board games are usually very large thereby making it infeasible for humans and computers to determine optimal moves, except for the last moves of a game when the subgame that has to be solved is sufficiently small. The impossibility of finding a game-theoretic solution for chess, checkers, or go has stimulated the development of a range of heuristic methods to replace the optimal

strategy. A heuristic search method can be seen as a procedure taking advantage of the problem structure in order to identify a good solution within a reasonable amount of computing time (Jacques and Daniel, 2001). The primary aspect of most AI players is the search algorithm, which is used to evaluate a board state based on a prediction of future moves from that state (Jack, 2009). In order to play a game perfectly, that is, using optimal strategy, it is not necessary to determine a complete solution that specifies a move for all nodes in the tree where the player is to move (Donkers, 2003). The task of game-playing algorithm is to provide the best move, according to the optimal strategy when the opponent has already moved. This task is equivalent to determining the minimax value of the subgame. In research on computer game playing, many algorithms have been developed that determine the minimax value of a game tree without determining a complete solution. Some examples are α-β search (Knuth and Moore, 1975), SSS* search (Stockman, 1979), Conspiracy-Number search (McAllster, 1988), Proof-Number search (Allis *et al.*, 1994), and MTD(*f*) (Plaat, 1996) and many more.

2.3.1 Minimax Search

The most widely used heuristic search technique is to determine the best move according to the optimal strategy for a reduced game and use the result as an approximation of the optimal strategy for the complete game. In computer-game playing, a reduced game is called the search tree and the maximum length of the search is called the search depth. Algorithms that determine the score of a reduced game are normally called (Minimax) game-tree search algorithms (Plaat, 1996). The actual size of the search tree depends on

the available resources such as time, memory, processor speed, and the number of processors. It also depends on the ability of the search algorithm to prune part of the search tree. This pruning of the search tree is of major importance to practical computer game-playing because the more that can be pruned, the larger (deeper and wider) the search tree can be (Marsland, 1983).

Minimax search is a game-playing search algorithm which is used for selecting the best choices of action in a game (or situation) where two players are working towards mutually exclusive goals, by acting on the same set of perfect information about the outcome of the situation. It is specifically applied in searching game trees to determine the best move for the current player of a game. It uses the simple principle that at each move, the moving player will choose the best move available to him. It is a recursive algorithm that takes arguments as the current la*yo*ut of the game board, the depth being searched, and the maximum depth to be searched. It returns the chosen move, and the score that such a move might lead to. The algorithm has two parts: the base case and the calls minimax on the boards resulting from each possible move at the current level. The base case of the algorithm evaluates the score for the given board. This occurs in two situations. First, the board will be evaluated if the current depth of the search is the maximum desired depth. Secondly, the board will be evaluated if there are no possible moves for either player, signaling a possible end to the game. Should there be moves for the current player and the maximum depth has not yet been searched, minimax generates possible moves at that level, and applies them in turn to the current board, calling

minimax on the resulting board. Each returned score is compared to the current best. If the returned score is more advantageous for the current player, that score replaces the best score and the move that generated it replaces the best move.

In games where opponents alternate taking turns which affect a board position (chess, checkers, etc.), a given position can be thought of as a node on a tree. All positions reachable from a given position are, therefore, children nodes on the game tree. Minimax recursively evaluates positions on a tree with the intent of selecting the best move for a given player in a given position. In order for a minimax routine to work, a function that maps a board position into a ``score" is needed. In two-player games, often this evaluation function returns real values between -1 and 1. A value of -1 means that one side has won outright, 0 indicates an even position, and 1 is returned when the other side has achieved victory. This is often referred to as a zero-sum.

2.3.2 Alpha-Beta (α-β) search

Alpha-Beta (α-β) search is a method that reduces the number of nodes explored in Minimax strategy. It reduces the time required for the search and it must be restricted so that no time is wasted searching moves that are obviously bad for the current player. The exact implementation of alpha-beta keeps track of the best move for each side as it moves throughout the tree and proceeds in the same (preorder) way as for the minimax

algorithm. For the MIN nodes, the score computed starts with +infinity and decreases with time. For MAX nodes, scores computed starts with -infinity and increase with time. The efficiency of the α-β procedure depends on the order in which successors of a node are examined. At a MIN node, consideration would always be given to the nodes in order from low to high score and at a MAX node the nodes in order from high to low score. In general, it can be shown that in the most favourable circumstances, the alpha-beta search opens as many leaves as minimax on a game tree with double on its depth. An alpha-beta algorithm consists of two functions: **evaluatemin** and **evaluatemax**. If calling from the **MIN** nodes, function **evaluatemin** is used; while beginning from a **max** node, function **evaluatemax** should be required.

2.4 MACHINE LEARNING

Machine learning (ML) is a set of tools that, broadly speaking, allows us to "teach" computers how to perform tasks by providing examples of how they should be done. ML is an area of AI concerned with the development of techniques which allow computers to "learn" (http://en.wikipedia.org/wiki/Machine_learning). Learning, like intelligence, covers such a broad range of processes that it is difficult to define precisely. A dictionary definition includes phrases such as "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience," and "modification of a behavioural tendency by experience". Learning is an intrinsic part of intelligent behaviour, and it can be defined as the ability to improve on past knowledge due to present experiences, leading to better decisions and more accurate problem solving (Njoku, 2002). ML can be

applied to problems concerned with optimization, concept formulation, pattern recognition, automatic classification, scientific discovery, and automatic programming. There are two machine learning approaches that have been used in commercial computer games with some degree of success. These are Artificial Neural Networks (ANNs) and Genetic Algorithms (GAs).

2.4.1 Neural Networks

A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in two ways (Haykins, 1994). They are as follows;

1. A neural network acquires knowledge through learning.
2. A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled.

An artificial neural network is an information-processing system that has certain performance characteristics in common with biological neural networks (Fausett, 1994).

Artificial Neural networks have been developed as generalizations of mathematical models of human cognition or neural biology, based on the following assumptions:

1. Information processing occurs in simple elements called neurons
2. Signals are passed between neurons over connection links
3. Each of these connections has an associated weight which alters the signal
4. Each neuron has an activation function to determine its output signal.

An artificial neural network is characterised by (a) the pattern of connections between neurons, i.e. the architecture, (b) the method of determining the weights on the connections (Training and Learning algorithm) and (c) the activation function.

2.4.1.1 The Biological Neuron

A biological neuron has three types of components that are of particular interest in understanding an artificial neuron: its *dendrites*, *soma*, and *axon*, all of which are shown in Figure 2.1.

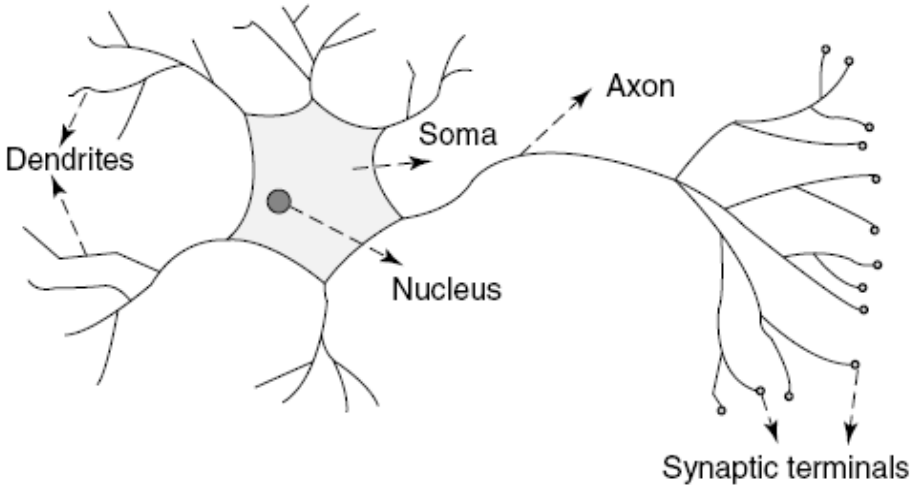

Figure 2.1: A Biological Neuron (Source: Ajith (2005))

The Dendrites receive signals from other neurons (synapses). These signals are electric impulses that are transmitted across a synaptic gap by means of a chemical process. This chemical process modifies the incoming signal. The Soma is the cell body. Its main function is to sum the incoming signals that it receives from the many dendrites connected to it. When sufficient input is received, the cell fires, and sends a signal up the *axon*. The Axon propagates the signal, if the cell fires, to the many synapses that are connected to the dendrites of other neurons.

2.4.1.2 The Artificial Neuron

The artificial neuron structure is composed of (i) n *inputs*, where n is an integer number, (ii) an *activation function* and (iii) an *output*. Each one of the inputs has a weight value associated with it and it is these weight values that determine the overall activity of the neural network. Thus when the inputs enter the neuron, their values are multiplied by their respective weights. Then the activation function sums all these weight-adjusted inputs to give an activation value (usually a floating point number). If this value is above a certain threshold the neuron outputs this value, otherwise the neuron outputs a zero value. The neurons that *receive* inputs from or *give* outputs to an external source are called *input* and *output* neurons respectively.

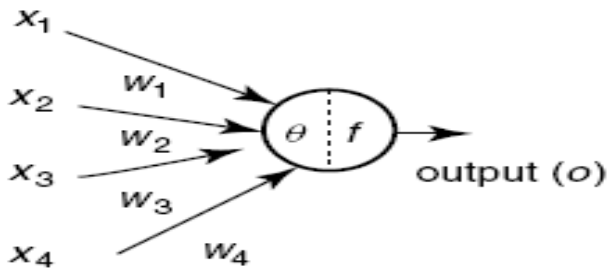

Figure 2.2: Architecture of an artificial neuron

Thus the artificial neuron resembles the biological neuron in that (i) the inputs represent the dendrites and the weights represent the chemical process that occurs when transferring the signal across the synaptic gap, (ii) the activation function represents the soma and (iii) the output represents the axon. As shown in figure 2.2 above, the signal flow from inputs $x_1, \ldots, x_n$ is considered to be unidirectional, which are indicated by arrows, as is a neuron's output signal flow (O). The neuron output signal O is given by the following relationship:

$$O = f(net) = f\left(\sum_{j=1}^{n} w_j x_j\right)$$

where $\boldsymbol{w_j}$ is the weight vector, and the function ***f(net)*** is referred to as an activation (transfer) function. The variable *net* is defined as a scalar product of the weight and input vectors,

$$net = w^{\mathrm{T}} x = w_1 x_1 + \cdots + w_n x_n$$

where **T** is the transpose of a matrix, and, in the simplest case, the output value ***O*** is computed as;

$$O = f(net) = \begin{cases} 1 & if\ w^{\mathrm{T}}x \geqslant \theta \\ 0 & otherwise \end{cases}$$

where $\boldsymbol{\theta}$ is called the threshold level; and this type of node is called a *linear threshold unit*.

It is often convenient to visualise neurons as arranged in layers, with the neurons in the same layer behaving in the same manner. The key factor determining the behaviour of a neuron is its activation function. Within each layer, all the neurons typically have the same activation function and the same pattern of connections to other neurons. Typically, there are three categories of layers, which are Input Layer, Hidden Layer and Output layer.

- **Input Layer:** The neurons in the input layer do not have neurons attached to their inputs. Instead, each of these neurons has only one input from an external source. In addition, the inputs are not weighted and so are not acted upon by the activation function. In essence each neuron receives one input from an external source and passes this value directly to the nodes in the next layer.

- **Hidden Layer:** The neurons in the hidden layer receive inputs from the neurons in the previous input/hidden layer. These inputs are multiplied by their respective weights, summed together and then presented to the activation function which decides if the neuron should fire or not. There can be many hidden layers present in a neural network although for most problems, one hidden layer is sufficient.

- **Output Layer:** The neurons in the output layer are similar to the neurons in a hidden layer except that their outputs do not act as inputs to other neurons. Their outputs however represent the output of the entire network.

2.4.1.3 Activation Function

The same activation function is typically used by all the neurons in any particular layer of the network. However, this condition is not required. In multi-layer neural networks, the activation used is usually non-linear, in comparison with the step or binary activation function used in single layer networks. This is because feeding a signal through two or more layers using linear functions is the same as feeding it through one layer. The two functions that are mainly used in neural networks are the ***Step*** function and the ***Sigmoid*** function (S-shaped curves) which represent linear and non-linear functions respectively.

Figure 2.3 shows the three most common activation functions, which are binary step, binary sigmoid, and bipolar sigmoid functions respectively.

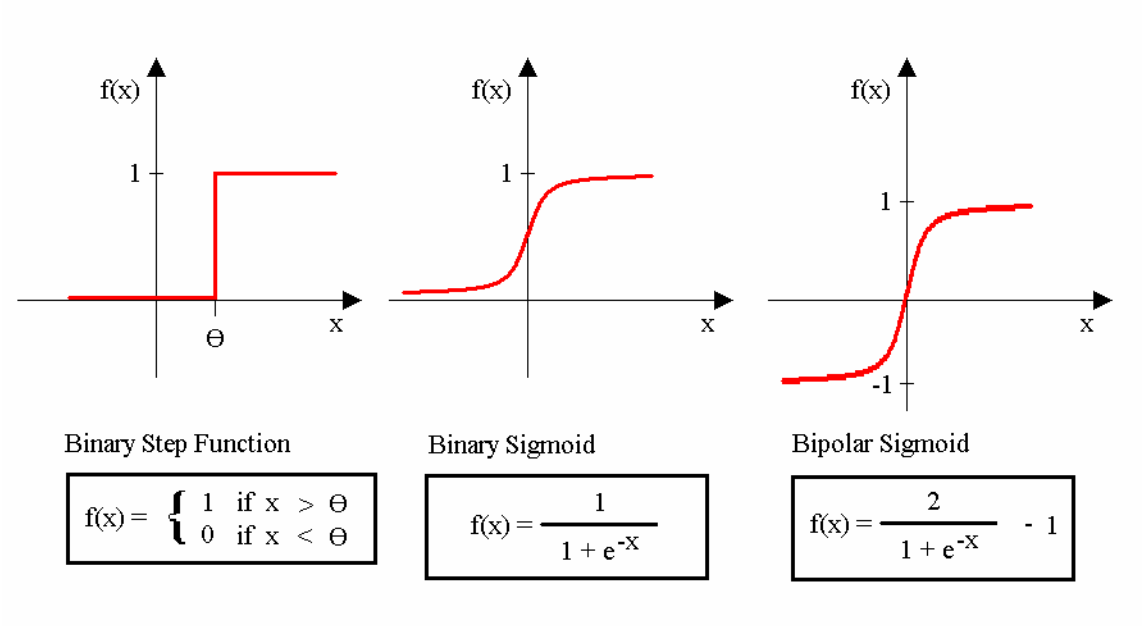

Figure 2.3: The Common Activation Functions Used in Artificial Neural Networks.

2.4.2 Genetic Algorithm

Genetic Algorithm (GA) is an optimization technique that was formulated during the early years of the 1970's by (Holland, 1975). It is a stochastic search algorithm based on the mechanics of natural selection and population genetics. Genetic algorithms are patterned after natural genetic operators that enable biological populations to effectively and robustly adapt to their environment and to changes in their environment. GA, as stated and demonstrated by (Holland, 1975) is theoretically and empirically proven to provide robust search in complex spaces. The GA performs its search, balancing the need to retain population diversity 'exploration', so that potentially important information is not lost, with the need to focus on fit portions of the population 'exploitation'. Reproduction in GA theory, as in biology, is defined as the process of reproducing offspring. However, mating may occur between any two classifiers due to their androgynous nature.

Over the years, GA has been used to solve a wide range of search, optimization and machine learning problems. As the name indicates, genetic algorithm attempts to solve problems in a fashion similar to the way in which human genetic processes seem to operate. GA starts with an initial population of chromosomes chosen at random. In contrast to other search techniques, GA has no need for auxiliary information about features of search space. They only require the value of an application-dependent objective function to be associated with an individual chromosome. Each member of the initial population must be evaluated using this function. Objective function associates a numerical value (also called "fitness" value) with a chromosome, which serves as some measure of "goodness" of a chromosome. That is, a measure of how well the chromosome fits the search space or solves the problem at hand that is to be maximized. Further steps of GA are repeated iteratively until either all chromosomes have the gene-associated fitness value (convergence conditions) or the desired number of iterations is reached. Each iterations of the algorithm consist of two basic steps**: "Selection**" and **"Recombination**".

GAs are used in solving problems in the areas of cellular automata, fuzzy logic, image registration (Grefenstette and Fitzpatrick, 1985), communications network configuration, simulation modeling and optimization (Azadivar and Tompkins, 1999), timetabling (Horman, 1998), multiobjective workforce scheduling, time constraint scheduling of limited resources, and combinatorial optimization. The most widely studied combinatorial task is traveling salesman problem (Goldberg and Lingle, 1985). Bin packing problems are also widely studied (Davis , 1985). They have been utilized in

playing games such as SimCity, SimEarth; in biology, chemistry and medicine; circuitry design and computer engineering; network routing for the telephone company; to detect computer viruses; for military artificial intelligence applications; military guidance and deciphering applications; art and music. GAs have been shown to be able to out-perform conventional optimisation techniques of difficult, discontinuous, multimodal, noisy functions (DeJong, 1975).

2.4.2.1 Outline of the Basic Genetic Algorithm

1. **[Start]** Generate random population of *n*-chromosomes (suitable for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population.
3. **[New population]** Create a new population by repeating the following steps until the new population is complete
    i. **[selection]** select two parent chromosomes from a population according to their fitness (the better the fitness, the higher the chances to be selected)
    ii. **[Crossover]** with a crossover probability, crossover the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    iii. **[Mutation]** with a population probability, mutate new offspring at each locus (positions in chromosomes).
    iv. **[Accepting]** place new offspring in a new population.
4. **[Replace]** Use newly generated population for a further run of the algorithm.

5. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population.
6. **[Loop]** Go to step 2.

The main advantage of genetic algorithms over other optimization methods is that there is no need to provide a particular algorithm to solve a given problem. It only needs a fitness function to evaluate the quality of different solutions (Kosmas and Donald, 1996). Also since it is an implicitly parallel technique, it can be implemented very effectively on powerful parallel computers to solve exceptionally demanding large-scale problems.

2.5 MACHINE LEARNING IN GAMES

The history of the interaction of machine learning and computer game-playing dated back to the early days of AI, when Arthur Samuel worked on his famous checker-playing program, pioneering many machine and game-playing techniques, Samuel (1959, 1967). Game, whether created for entertainment, simulation, or education, provides great opportunity for machine learning (Bowling *et al,* 2006). Game-playing is a very popular machine learning research domain for AI. A lot of game-playing programs have been developed in the past decades. Many of the game playing programs that have been developed are highly dependent on knowledge to increase the accuracy of their evaluation function. Samuel's checkers program (Samuel, 1959; 1967) and Tesauro's TD-gammon (Tesauro, 1995) were important breakthrough. Samuel's

checkers program was the first successful checkers learning program which was able to defeat amateur players. He used a search procedure which was suggested by Shannon in 1950, called minimax. In 1995, Tesauro presented a game-playing program called TD-Gammon, which was able to compete with the world's strongest backgammon players. It was trained by playing against itself and learning on the outcome of those games. It scored board positions by using neural networks as its evaluation function. Learning algorithms can be divided into three groups, that is, supervised learning, unsupervised learning and reinforcement learning.

2.5.1 Supervised Learning

Supervised learning is a machine learning technique for creating a function from training data. The training data consist of pairs of input vectors and desired outputs. The task of the supervised learner is to predict the value of the function for any valid input object. Supervised learning occurs when a neural network is trained by giving it examples of the task we want it to learn, i.e, learning with a teacher. The way this is done is by providing a set of pairs of patterns where the first pattern of each pair is an input pattern and the second pattern is the output pattern that the network should produce for that input. The difference in the output between the actual output and the desired output is used to determine the changes in the weights of the network.

2.5.2 Unsupervised Learning

Unsupervised learning on the other hand, has no target output given by an external supervisor. The learning takes place in a self-organising manner. Generally speaking, unsupervised learning algorithms attempt to extract common sets of features in the input data. An advantage of these learning algorithms is their ability to correctly cluster input patterns with missing or erroneous data. The system can use the extracted features it has learned from the training data to reconstruct structured patterns from corrupted input data. This invariance of the system allows for more robust processing recognition tasks.

2.5.3 Reinforcement Learning

In reinforcement learning, an agent can improve its performance by using the feedback it gets from the environment. This environmental feedback is called the reward signal. With reinforcement learning, the program receives feedback just like the supervised learning. Reinforcement learning differs from supervised learning in the way of how an error in the output is treated. With supervised learning, the feedback information is what exact output is needed. The feedback with reinforcement learning only contains information on how good the actual output was. By trial-and-error, the agent learns to act in order to receive maximum reward.

CHAPTER THREE

AYO GAME

3.1 HISTORICAL BACKGROUND

Around the world, various versions of mancala games (to which Ayo belongs) have been observed dating back to the Empire Age of ancient Egypt (Murray 1952). Players take turns of harvesting seeds (or seeds) by moving around the board according to various rules. Ayo game is the most popular board game among the Yorubas that occupy the entire south-western states of Nigeria and parts of the Republic of Benin. Ayo game is one of the oldest games of strategy ever known. It is a game of perfect information known as combinatorial games (Fraenkel, 1996). It is a two–player game, with no hidden information, no chance move, a restricted outcome (win, lose and draw) and with each player moving across the board. Ayo is a game of strategy, which has been shown to be of great use in solving human psychological related problems due to its attributes (Ayeni & Longe, 1985). Ayo is a game that requires rigorous calculations and strategies, with the aim of capturing as many seeds as possible (Olugbara *et al*., 2006).

Studies in anthropology indicate that Ayo and Islam appear to have the same home. Of course, the game was not Ayo originally. It was called Mancala, a word of Arabic origin, from the verb nagala – 'to move'. Murray, in his book titled "A history of board games other than Chess" claim that the name Mancala originally referred to a popular game in Egypt which was played on a board containing two rows of holes in which the counters

(seeds) are arranged and moved (Murray, 1952). However, anthropologists now use the term Mancala for any similar game played on a board in which the pattern of lines and cells is the same as for Ayo, but there could be less or more than six holes in each row.

There is a record (perhaps the earliest record of the existence) of Mancala in Egypt in the Empire age (about 1580 – 1150 B.C.), a later appearance in Ceylon during the early centuries A.D., and in Arabia before the time of Mohammed. It is generally believed that Mancala, in its various versions, spread from these parts to the rest of the world. It is therefore very likely that our forefathers brought Ayo (or Mancala) with them in their migration from the northern to eastern parts of Africa, and from Arabia.

It is to be expected that Mancala spread to the other parts of the world as well. In fact, today, Mancala is found all over the world especially in tropical and sub-tropical regions of Asia, Africa, and the adjacent Islands. There is a strong evidence that African slaves (mainly from West Africa) took Mancala with them to the West Indies, America and other places to which the slave trade extended. It is significant that Mancala is popular mainly in these countries to which the culture and religion of Islam have extended; and this would perhaps explain why Mancala is not popular in the non-Islamic parts of Europe.

The game is known by many different names in Nigeria and other parts of the world, (Adewoye, 1990; Daoud, *et al*., 2004) such as; Ayo (by the Yorubas), Darra (by the Hausas), Okwe (by the Ibos), Igori (by the Igaras and Igbiras), Dagh (by the Tiv or

Nushi tribe), Whyo (by the Ibiobios), Lok (by the Jabas around Zaria), Obridjie (by the Arochukwu and Abiribi), Makwini (in Kukurukuland), Ndim-ndum (by the Ekon tribe in Ikom), Gifia (in Calabar area), and Ogiarise (in Benin), Kate (in Gabon), Songo (in Cameroon), Adjito (in Dahomey), Adji (in Togo), Wari (in Ghana and West Indies), Aware (in Ivory coast), Warre (in Sierra Leone), Awari (in Dutch), and many more as this is by no means complete. All these games as they may be called are generally classified into a family of Mancala game.

Ayo game like every other board games, is seen to consists of a coherent series of consecutive movements ("moves") of physical pointers ('counters') along co-ordinates defined in a space ('board') which, for that specific purpose, is set apart (i. e bounded and restructured) in such a way that formal and explicit rules define the movement of individual pointers as well as their interaction. By implication in the context of this interaction, the players are defined as opponents in a struggle (Biusbergen, 2001). Two persons play Ayo at a time with the board put in between the players. A typical Ayo game board and players playing the game of Ayo is shown in figure 3.1. The board is a plank of wood consisting of two rows of six pits belonging to either row and each of the pits contains four seeds of the plant "caeselpinia crista" (Odeleye, 1977) such that a total of forty-eight seeds are contained in a board at the start of the game. Often, there are two extra hollows normally placed centrally at the end of each rows of the board. These are called "seed bags" that are used to store the captured seeds by each of the players. As the game progresses, each pit can contain any number of seeds or no seed at all. Just like any other game, the ultimate objective of the game is to capture more seeds than the

opponent to emerge as winner or have equal number of seeds as a draw game. As a seed is captured, it is removed from the board and put in the seed bag and plays no further part, other than being used to evaluate the current game position.

Figure 3.1: Players *of Ayo Game*

Various methods have been explored to solve the game's problem, but due to its complexity and irregular patterns as the play of the game progresses, the exact solution for the game has not hitherto been found. In view of this, the game has captured the attention of many Artificial Intelligence (AI) researchers, mathematicians and computer scientists. Ayeni *et al*., (1985), opined that solving problem of *Ayo* game as a linear programming problem could be expensive but most suitable for myopic decision. It is therefore, inadequate for futuristic (or hyper-myopic) decision, which is of paramount interest in Move-and-Capture Games (MCG) such as Ayo. Retrograde Analysis (RA)

has been used to solve Awari (Romein and Bal, 2002) by searching the entire state space on a parallel computer with 144 processors. As reported by the researchers, the state space contains 889,063,398,406 positions and was searched using RA. It was admitted by the authors that no single computer has enough processing power and memory to search the state space, but even on a modern parallel computer, the problem was extremely challenging (Romein and Bal, 2002). This is a major drawback alluded to RA as no such method could easily be implemented on a small memory device like wireless handset for playing Ayo game. Furthermore, both endgame database and retrograde approaches can be very expensive to implement. The hybrid of co-evolution and minimax has been investigated for evolving Awari player (Davis and Kendall, 2002), but such method cannot suggest a best move until after learning. Consequently, it has poor starting ability and suffer from overlay delay to suggest a move. The first problem posed by minimax search is how an evaluation function is developed and applied to the game tree, since computer game programs are differentiated by the quality of the evaluation. The work of Davis and Kendal (2002) uses evolutionary strategy to evolve a simple evaluation function and the output of the function is then used in a minimax search. They reported that their implementation took about one minute to suggest a move for a search depth of seven, but this is not too efficient for a computer that is expected to play at a faster speed.

3.2 THE PLAY STRATEGY

The first player (which can be any of the players) picks up all the seeds in any of non-empty pits p_1, p_2, …, p_6 on his/her side (home) of the board and deposits one seed at a time into the pits in an anticlockwise direction until all the seeds are deposited. This is called "sowing" the seeds. When the player reaches the end of a row, sowing continues in an anticlockwise direction in the other row. When a player picks a pit with so many seeds (twelve or more) such that the player passes completely around the board, the originating pit is skipped and the seed is played in the next pit. This means that the originating pit is always left empty at the end of the turn. If the last seed is sown in the opponent row and the pit concerned finishes with two or more seeds, those seeds are captured in a clockwise direction until either a hollow does not have two or three seeds in it, or the end of the opponent row is reached. An illustration of players making moves to capture seeds is shown in figure 3.2, where when player 1 plays the seed in the fifth pit. The player captures the seeds in pit 4, 3 and 2 of player 2 as they result in 3, 2 and 3, making a total of eight seeds being captured. On the other hand, if player 2 plays the seed in the sixth hollow of his own, captures the seeds in pit 1 and 2 of player 1, making a total of 4 seeds captured. The play progresses this way until the end is reached when a total number of seeds on the board will be equal to or less than five seeds.

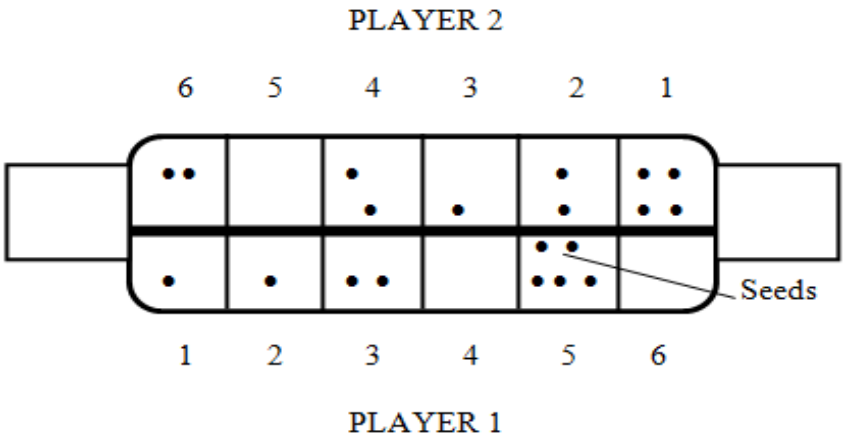

Figure 3.2: A *Capture Strategy Move in Ayo Game*

Each player stores the seeds he captures in his hand (the non-playing hand or in receptacles which are sometimes provided for this purpose at two opposite ends of the board. At the end of the game, each player lays the seeds he captured in the hole in fours on his side, in preparation for the next game. Obviously, the player who is unable to fill up all his holes in fours, captures fewer seeds and is therefore the loser.

One critical rule called the 'golden rule' (Adewoye, 1990) for capturing seeds is that a player must ensure his opponent has at least one seed in a pit with which to play after he had captured the rest of the seeds. Any player who, in a bid to capture, contravenes this rule is automatically disallowed from making the capture and so takes nothing even if the content of the holes are technically ripe for capture. As shown in Figure 3.3 below, if player 1 plays the seeds in the fourth hole, which contains seven seeds, all the seeds on player 2 sides qualify for capture.

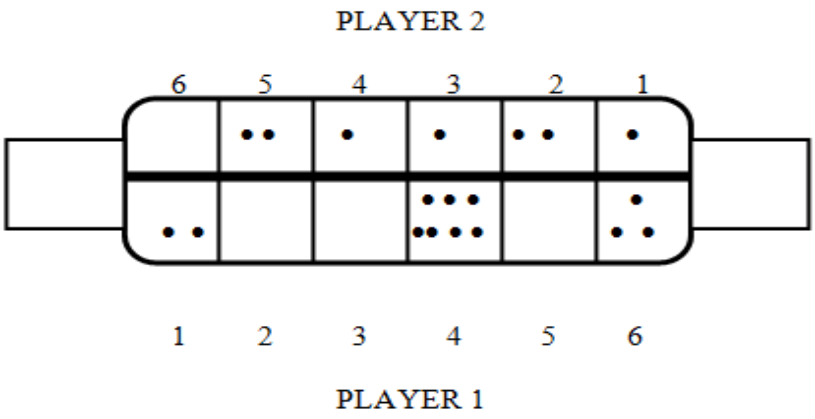

Figure 3.3: A *Golden Rule Capture Strategy in Ayo Game*

By this rule, player 1 is not allowed any capture, although the move is allowed. Therefore, a wise player 1 will rather play (or move) the three seeds in hole six to capture seven seeds, while player 2 can play the seeds in his fifth hole to capture three seeds.

Sometimes, towards the end of a game, when there are only a few seeds to capture, a player is forced to give up one or more of his seeds for capture in order not to break the golden rule. What is still considered a conjecture till date with this rule is that there is still no consensus as to what to do if a player finds it impossible to play a seed into his opponent's side. Sometimes, the player who caused the stalemate forfeit all the remaining seeds to his opponent; at other times, the remaining seeds are shared out, while in some cases, the game is declared void and cancelled. Our investigations in this work overcome this chaos and foster a promising solution that will disallow any player from exploiting the delicate nature of the 'end-game' to create a stalemate.

The step by step instructions for playing the game can be given as a set of rules thus:

1. The game commences with players selecting who is north or south and who starts first. A player selects a non-empty pit on his side and sow the seeds from that pit around the board, dropping one at a time, counter-clockwise into each pit
2. If a player chooses a pit with enough seeds to completely go around the board (i.e. pit with 12 or more seeds, usually called ***odu*** or **kroo**), the original pit is skipped and left empty
3. If the last seed is dropped into a pit on the opponent's side, leaving that pit with 2 or 3 seeds, the player captures all the seeds in that pit. The capture continues with consecutive previous pits on that side, which also contain 2 or 3 seeds
4. If all the opponent pits are empty, the player must make a move that will give his opponent a move, this is called "Golden rule". If no such move can be made, the player captures all the remaining seeds on the board, ending the game. If no move is possible, the winner is the person with the greater number of captured seeds
5. If by making a move, a player can capture all the seeds on opponent's side of the board, no capture is allowed for the player
6. The game is over when one player has captured 25 or more seeds, or both players have taken 24 seeds each (a draw) or when fewer seeds (say 3) circulate endlessly on board

In the play of the game, there are different strategies and variations of them as listed in the following section.

3.2.1 *Odu* (or Kroo)

A very unique strategy (that is, an **offensive** strategy) of play in the game of Ayo is the accumulation of more than twelve seeds in a pit; this is called ***Odu***, to allow the seeds to go round the board and end on the opponent's side. Because ***Odu*** has to go round the board and further go into the opponent's area, it needs a lot of seeds. The sixth bin (the farthest on your right) needs from 12 to 16 seeds to count as ***Odu***. The first pit (farthest left) needs 17 to 21 seeds. So it takes some time to build ***Odu***, though it is often possible to build up two at once. It is easier to build ***Odu*** in the sixth pit, of course not only does it need fewer seeds, but must also be fed frequently. Compare that with the first pit, which is rarely fed - it's up to the opponent to build one if possible. Since there will always be at least one empty bin when the opponent takes his turn, then there is every possibility to learn to defend against ***Odu***. We use the game configuration below for an illustration.

Figure 3.4: *Formation of* ***Odu***

3.2.2 Blocking

Blocking is a way of making sure that the target end-space of an ***Odu*** cannot be captured - this is the first defence a beginner learns, though probably applied to direct attacks rather than **'Odu'** attacks. For example, given the game configuration in figure 3.5 below, player 1 will have to protect (or defend) his seeds in pits 1, 2, and 3, which could be captured by player 2 if he plays from his pit 4 with 5 seeds. In order to do this, player 1

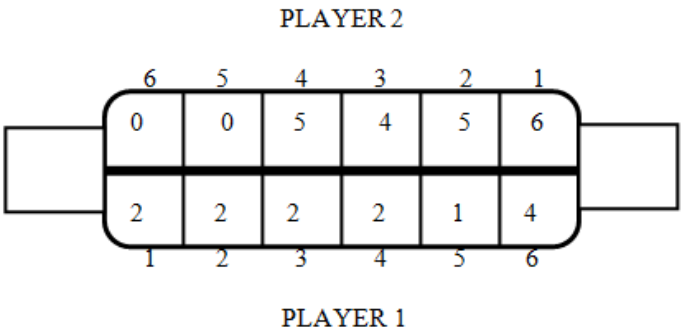

Figure 3.5: A Game Configuration Showing Blocking Strategy

will have to play from his pit 1 with 2 seeds so as to make his pits 2 and 3 to have 3 seeds each as shown in figure 3.6

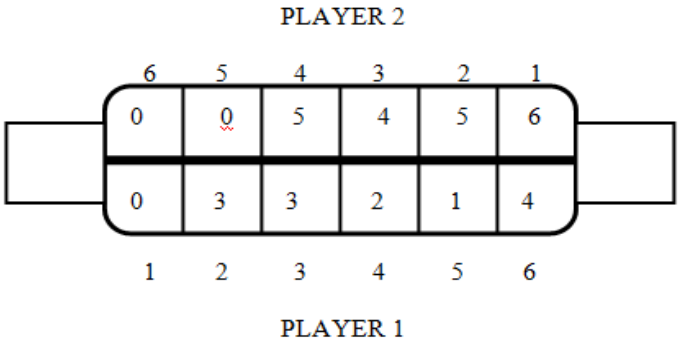

Figure 3.6: A Blocking Configuration

3.2.3 Pressure

This is trying to deny all other moves on an opponent's side, making him use the ***Odu*** prematurely or prevent a mature ***Odu*** from capturing some seeds. This is done largely by making moves so as to keep the seeds on player 1 side of the board as much as possible. The following configuration gives an illustration.

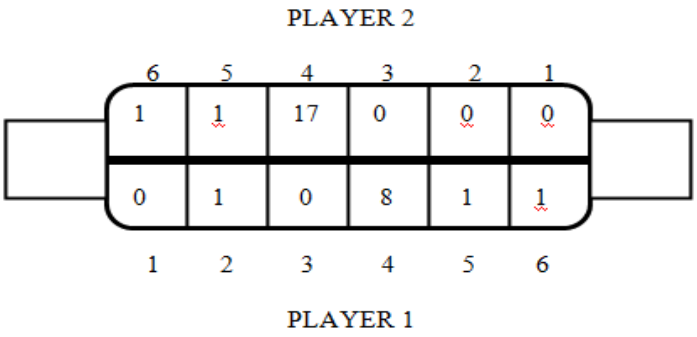

Figure 3.7: A Game Configuration Showing Prevention of the use of ***Odu***

From the configuration above, player 2 already has a mature ***Odu*** in his pit 4 with 17 seeds, which of course would not want to play because it will end on the pit 4 of player 1 thereby capturing no seed. Player 2 would rather want to play from pit 5 so as to have 2 seeds in his pit 6 so that at his other turn he could capture 2 seeds from pit 2 of player 2. But it is apparent that player 2 will not allow him to capture seeds, but rather would play from pit 2 so as to ensure that player 2 put more seeds on his side in order to acquire more seeds. On the contrary, if it is the turn of player 1 to move first, he would want to move from his pit number 4 thereby allowing him to capture four seeds and render the ***Odu*** in player 2 pit useless.

3.2.4 Overloading

It is the act of sowing so many seeds into the opponent's ***Odu*** so that it loops around again and ends on his own side.

3.2.5 Attack

This is a means of trying to leave the opponent with pits vulnerable for capturing. That is, more than one pit with 1 or 2 seeds. So that the player has to leave at least one pit vulnerable. This can be achieved in several ways and it depends greatly on the disposition of the seeds.

3.2.6 Counterattacking

It involves setting up of some tactics to take advantage of the seeds that must be placed on the players' empty pits when trying to attack with ***Odu***, thus capturing some seeds. Of course, one has to take his sowing into consideration when setting up counterattack – all the pits on the player 1 side will have more seeds once the ***Odu*** **is** used.

3.2.7 The Endgame

Ayo is concluded when there are four or fewer seeds circulating endlessly around the board. The seeds that remain become part of the seeds of the player on whose side they are. This is one of the lapses discovered in the 'awale'. This weakness is being taken care of by the concept of Completely Determined Game (CDG) proposed and described

in section 2.6 in this work, which further enhanced the endgame strategy of play of the Ayo game, although a game can be concluded if one of the players voluntarily resigns.

3.2.7.1 Completely Determined Game (CDG)

CDG is a game configuration in *Ayo* where there exists a series of move and capture strategies for a player 1 (henceforth referred to as player X for the purpose of our analysis) on the action of player 2 (again henceforth referred to as player Y) such that the move process continues for player X until only one seed remains on the board and the seed belongs to player Y (see Adewoye (1990) and Adebiyi (1999 for a detailed description). As the concept of the CDG is defined, its usefulness for ending a game should be apparent, and can be configured (or arranged) as the number of seeds on the board reduces to twenty-one (21).

In Ayo game, a CDG play exists if the configuration of seeds on the board satisfies the two conditions given below:

i. All the holes on the side of player Y (the first player) are empty except the first hole that contains only one seed; and
ii. As player Y moves, player X takes a move that results in a capture of two seeds. This strategy of move and capture continues until player X has captured all the seeds except one seed that is left for player Y to ensure that the golden rule is obeyed.

Figure 3.8 below depicts the initial configuration of player X and Y for the CDG

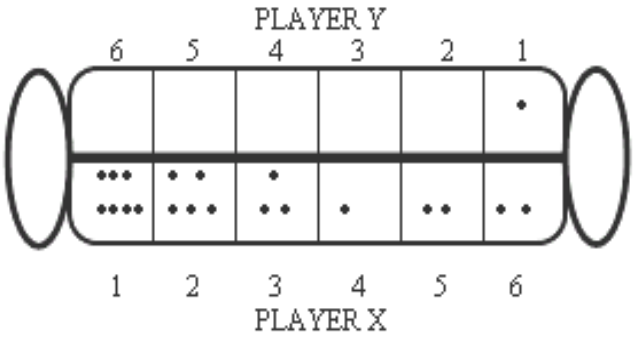

Figure 3.8: Initial configuration of CDG in Ayo Game

A soon as CDG play occurs at any instance of play, the game could be regarded as ended because one player may capture all the seeds on the board except for the one to be left in the opponent's pit as described above. The task of solving the CDG is to find the correct order of moves that causes all seeds on the board to be captured as soon as a CDG is arranged in the play. The basic goal is to give an effective and efficient algorithm that ensures perfect distribution of seeds in the CDG configuration such that a particular move on the CDG can result in another CDG configuration, and so on until the game ends. Table 3.1 below depicts the reducibility of the CDG configuration in a bottom-up manner.

To appropriately solve the concept of CDG in *Ayo* game, it is worthy of note that for every number of seeds that are in play, only one position is appropriate for a valid move otherwise the anticipated total seeds to be captured will not be possible. In any case, if there is more than one pit that has enough seeds to capture seeds for which the CDG strategy must hold, then it is the pit at the rightmost part that must be selected as the best

move. If the leftmost pit is selected, the seeds in the rightmost pit will increase by one, hence that pit cannot be used again for the CDG strategy.

Table 3.1: Pattern Spaces for CDG

CDG	X_1	X_2	X_3	X_4	X_5	X_6	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
1	0	0	0	0	0	2	1	0	0	0	0	0
2	0	0	0	0	3	1	0	1	0	0	0	0
3	0	0	0	0	3	1	1	0	0	0	0	0
4	0	0	0	4	2	0	0	1	0	0	0	0
5	0	0	0	4	2	0	1	0	0	0	0	0
6	0	0	0	4	2	2	0	1	0	0	0	0
7	0	0	0	4	2	2	1	0	0	0	0	0
8	0	0	5	3	1	1	0	1	0	0	0	0
9	0	0	5	3	1	1	1	0	0	0	0	0
10	0	6	4	2	0	0	0	1	0	0	0	0
11	0	6	4	2	0	0	1	0	0	0	0	0
12	0	6	4	2	0	2	0	1	0	0	0	0
13	0	6	4	2	0	2	1	0	0	0	0	0
14	0	6	4	2	3	1	0	1	0	0	0	0
15	0	6	4	2	3	1	1	0	0	0	0	0
16	7	5	3	1	2	0	0	1	0	0	0	0
17	7	5	3	1	2	0	1	0	0	0	0	0
18	7	5	3	1	2	2	0	1	0	0	0	0
19	7	5	3	1	2	2	1	0	0	0	0	0

3.3 KROO AND CDG

There are two most powerful strategies available in Ayo game: accumulate **Odu** (Kroo) and plan for CDG. The rationale for the choice of CDG culminates from the fact that it is more powerful than **Kroo**. While the formation of **Kroo** can be prevented or ruined once formed, the CDG cannot be rendered ineffective once set as complemented by the refinement-based heuristic technique proposed in this work. In addition, the maximum number of seeds that can be captured by a **Kroo** cannot exceed 15 at the instance of formulation, but CDG can capture up to a maximum of 20 seeds unpreventable.

3.4 COMPUTATIONAL ANALYSIS OF *AYO* CONFIGURATION

Ayo is a two-person, zero-sum game (that is, a game played by two persons and that which one player wins is that which the other player loses), and the solution of a two-person zero-sum game is a pair of strategies in equilibrium. Generally, the approach to analyzing the state of *Ayo* game requires looking many steps ahead in order to decide which pit to play from in the course of playing the game (Longe and Ayeni, 1991). The reward associated with the winning of a position while it is played is called a payoff. For the purpose of analysis, a one-step look ahead is used, which involves the net gain of a player if he chooses to play from a particular pit, subject to a probable response of the opponent. Consequently, the outcome of the game results in a payoff matrix of size 6 x 6, where the rows represent the pits of which the first player can play from and the columns represent the pits of which the second player can play from.

If the pits are represented as $x_1, x_2, \ldots, x_6$ and $y_1, y_2, \ldots, y_6$ for players X and Y respectively, the payoff matrix element a_{ij} is defined as:

$$a_{ij} = x_i - y_j \setminus x_i \qquad (1)$$

where x_i is the winning of player X if he distributes seeds from pit X_i and y_j is the winning of player Y if he distributes seeds from pit Y_j given that X has distributed seeds from pit X_i and this yields a matrix of the form

		Y					
		y_1	y_2	.	.	.	y_6
	x_1	a_{11}	a_{12}	.	.	.	a_{16}
X	x_2	a_{21}	a_{22}	.	.	.	a_{26}
	.						
	.						
	x_6	a_{61}	a_{62}	.	.	.	a_{66}

3.4.1 Algebraic Characterization of CDG

A mathematical property that can give a direct insight into the complexity of *Ayo* game is the number of possible positions on the board. A position in Ayo consists of a certain distribution of seeds in the pits and it includes the captured seeds in the store or kept by the players if there are no stores. In addition, a position includes the knowledge, which a player applies to move next. The number of possible positions is a function of the number of pits (and stores) and the number of seeds. Details can be found in Donkers *et al* (2002). For the purpose of the characterization of CDG, the following propositions are presented:

Proposition I

Let $A, B \subset \Re^n$ be sets of CDG-vectors and $x = (x_1, x_2, \ldots, x_{i-1}, ..., x_n) \in \mathrm{A}$,

Suppose $k \in Z^+$ is the least number such that $\mathrm{x_k} = 0$. If T is a transformation that acts on the components $\mathrm{x_i}$ of its argument x, then $\mathrm{T: A \rightarrow B}$ could be defined by:

$$y_i = \begin{cases} x_i, \text{if } i > k \\ k+1, \text{if } i = k \\ x_i - 1, \text{otherwise} \end{cases}$$

Then T is invertible with inverse T^1 acting on $T(x) \in B$ as follows.

$$x_i = \begin{cases} y_i, \text{if } i > k \\ 0, \text{if } i = k \\ y_i + 1, \text{otherwise} \end{cases}$$

Proof:

To establish the validity of the proposition, it is sufficient to verify that:

$$T^1(T(x)) = x \text{ and } T(T^1(y)) = y$$

The application of these transformations follows that

$$T^1(T(x)) = T^1(x_1 - 1, x_2 - 1, \ldots, x_{k-1} - 1, k+1, x_{k+1}, \ldots, x_n)$$

$$= T^1(y_1, y_2, \ldots y_{k-1}, y_k, y_{k+1}, \ldots, y_n)$$

$$= (y_1 + 1, y_2 + 1, \ldots, y_{k-1} + 1, 0, y_{k+1}, \ldots, y_n)$$

$$= (x_1, x_2, .., x_k, \ldots, x_n)$$

$$= x$$

Similarly,

$$T(T^1(y)) = T((y_1 + 1, y_2 + 1, \ldots, y_{k-1} + 1, 0, y_{k+1}, \ldots, y_n))$$

$$= T(x_1, x_2, \ldots, x_{k-1}, x_k, x_{k+1}, \ldots, x_n)$$

$$= (x_1 - 1, x_2 - 1, \ldots, x_{k-1} - 1, k+1, x_{k+1}, \ldots, x_n)$$

$$= (y_1, y_2, \ldots\ y_k, \ldots, y_n)$$

$$= y.$$

3.4.2 Vector Reducibility in Space

Let $(i, x) \in \Re \times \Re^n$, $x = (x_1, x_2, \ldots, x_{i-1}, \ldots, x_n)$, $n > 1$, and consider the following transformation T^*, which does nothing to the prefixes of component x_i but initializes x_i to zero and increases each postfixes of x_i by unity.

$T^*: \Re \times \Re^n \rightarrow \Re^n$, defined by the specification

$$T^*(i, x) = \begin{cases} (x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, x_{i+2}, \ldots, x_{n+1}), \text{if } x_i = i + 1 \\ x, \ otherwise \end{cases}$$

If for some $T^*(i, x) \neq x$, then x is said to be reduced to T*(i, x) by T^*.

Clearly, the domain of T has no fixed point and consequently if T^* is taken as a seed sowing function in *Ayo* game, it implies that the game can have more than one equilibrium at a time. Selecting an optimum strategy from a list of equilibra points is the goal of the vector reducibility in space, which is formalized for this class of *Ayo* game.

3.4.3 The Morphology of CDG

The trivial CDG-vector $x_i = (0, 0, 0, .., 2)$ is called minima CDG-vector. Generally, the CDG-vector $(y_1, y_2, \ldots, y_n) \in \Re^n$ is called maxima CDG-vector if for every vector $x = (x_1, x_2, \ldots, x_n) \in \Re^n$, $\| y \|_1 - \| x \|_1 \geq 0$

The significance of maxima and minima CDG-vector enables us to define a transformation function between vectors. While it is easier to determine minima CDG-vector for any game of size ***n***, the problem of computing maxima CDG-vector is a little bit intensive. Since the problem of determining maximal CDG-vector is to solve the

optimization problem: $\max \sum_{i=1}^{n} x_i$ subject to $x_i \geq 0$, then the interest would be on how to efficiently compute the maxima CDG-vectors. To achieve this purpose, we propose algorithm I and implemented it.

Algorithm I

Input n the size of the board

Output $x = (x_n, x_{n-1}, \ldots, x_2, x_1)$

Step 0: $x_n = n+1, d_n = 2, j = n-1$

If $j<1$ stop otherwise go to step 1

Step 1: $x_j = (j+1)\left\lfloor \frac{(j+1)-(x_{j+1}-d_{j+1})}{j+1} \right\rfloor + (x_{j+1}-d_{j+1}$

If $j=1$, stop otherwise, go to step 2

Step 2: $d_j = d_{j+1} + 2\left\lfloor \frac{(j+1)-(x_{j+1}-d_{j+1})}{(j+1)} \right\rfloor$

$j = j-1$

Go to step 1

END Algorithm I

The reducibility of the maximal CGD for n = 6 shown in Table 3.1 above, while a sample implementation of the above algorithm (see figure 3.9 for the C code) gives the maximal CDG-vectors in Table 3.2 below for any board size of ***2n*** pits.

```c
#include<stdio.h>

void main()
{
	int flr=0, n=6, x[7], d[7], j, i;

	x[6]=n+1;
	d[6]=2;
	printf("x[%d] = %d,\n\n",6,x[6]);

	for(i=n-1; i>=1; i--)
	{

		//printf("x[%d] = %d\n",6,x[6]);
		j=i;
		//printf("j = %d\n",j);

		flr = ((j+1) - (x[j+1] - d[j+1])) / (j+1);
		//printf("floor = %d\n",flr);

		x[j] = ((j+1) * flr) + (x[j+1] - d[j+1]);

		//if(i==1) return;

		d[j] = d[j+1] + (2 * flr);

		printf("x[%d]: %d \n\n",j,x[j]);
		//printf("d[%d]: %d \n\n",j,d[j]);

	 }

}
```

Figure 3.9: C – Code for Implementing Algorithm 1

Table 3.2: Maximal CDGs for Board Size of ***2n*** Pits

	Maximal CDG-vectors
n	x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 . . . ……..x_{16}……..x_n
1	2
2	1 3
3	2 2 4
4	1 1 3 5
5	1 3 2 4 6
6	2 2 1 3 5 7
7	1 3 4 2 4 6 8
8	2 2 3 1 3 5 7 9
9	1 3 1 5 2 4 6 8 10
10	1 1 4 4 1 3 5 7 9 11
11	2 2 2 2 6 2 4 6 8 10 12
12	1 1 1 1 5 1 3 5 7 9 11 13
13	1 3 2 4 3 7 2 4 6 8 10 12 14
14	2 2 1 3 2 6 1 3 5 7 9 11 13 15
15	1 3 1 5 6 4 8 2 4 6 8 10 12 14 16
16	1 1 4 4 5 3 7 1 3 5 7 9 11 13 15 17
17	2 2 2 2 3 1 5 9 2 4 6 8 10 12 14 16 18
18	1 1 3 5 1 7 4 8 1 3 5 7 9 11 13 15 17 19
19	1 3 4 2 5 5 2 6 10 2 4 6 8 10 12 14 16 18 20
20	2 2 3 1 4 4 1 5 9 1 3 5 7 9 11 13 15 17 19 21
21	1 3 3 3 1 1 7 3 7 11 2 4 6 8 10 12 14 16 18 20 22
22	1 1 3 5 5 7 6 2 6 10 1 3 5 7 9 11 13 15 17 19 21 23
23	1 3 4 2 2 4 3 9 4 8 12 2 4 6 8 10 12 14 16 18 20 22 24
24	2 2 3 1 1 3 2 8 3 7 11 1 3 5 7 9 11 13 15 17 19 21 23 25
25	1 1 1 1 3 7 8 6 1 5 9 13 2 4 6 8 10 12 14 16 18 20 22 24 26
26	1 3 2 4 1 5 6 4 10 4 8 12 1 3 5 7 9 11 13 15 17 19 21 23 25 27
.	.

Proposition II

The finite set $A \subset \Re^n$ of CDG-vector is enumerable.

Proof:

Let $B \subset N$ and $x_1, x_2, ..., x_n \in A$, where | A | = m. We prove that there exists a mapping from A to B. If T in proposition I is recursively applied m-times to x, that is $T^m = T^{m-1}T$, then

$$\| x_n \|_1 = \sum_{i=1}^{k-1} (x_i - 1) + (k+1) + \sum_{j=k+1}^{n} x_i$$

$$d = x_1 + x_2 + ... + x_n + 2$$

$$= 2 + \sum_{i=1}^{n} | x_i |$$

$$= 2 + \| x^{(m-1)} \|$$

The $\iota^1 -$norms of the m CDG-vectors can be computed as:

$$\| x^{(1)} \|_1 = \| (0,0,0,,...,2 \|_1 = 1(2)$$
$$\| x^{(2)} \|_1 = \| x^{(1)} \|_1 + 2 = 4 = 2(2)$$
$$\| x^{(3)} \|_1 = \| x^{(2)} \|_1 + 2 = 6 = 3(2)$$
$$...$$
$$\| x^{(m-1)} \|_1 = \| x^{(m-2)} \|_1 + 2 = 2m - 2 = (m-1)(2)$$
$$\| x^{(m)} \|_1 = \| x^{(m-1)} \|_1 + 2 = 2m = m(2)$$

Let $f : A \to B$ be defined $\forall x \in A$ by $f(x) = \frac{1}{2} \| x \|$

Then **f** is a mapping that associates **A** to **B** and consequently, **A** is enumerable.

Proposition III

Let $x^{(m)}$ be the maxima element of the CDG A, the cardinality of A is

$$| \mathrm{A} | = \frac{1}{2} \| \mathrm{x}^{(\mathrm{m})} \|$$

Proof:

By proposition II,

$$\| \mathrm{x}^{(\mathrm{m})} \| = 2 | \mathrm{A} | .$$

Hence

$$| \mathrm{A} | = \frac{1}{2} \| \mathrm{x}^{(\mathrm{m})} \|$$

3.4.4 Nearness of a Game from a CDG Play

Determining nearness from a CDG-vector is the problem of finding a neighbourhood of **A** with the least distance from **A**. This is essentially the same as determining nearest neighbour from a CDG play. When a game is near its end, it is natural to frequently find out when a CDG play is close by. It is equally reasonable to find out the closeness of a game to CDG play even when the game just begins. Suppose $S = (s_1, s_2, ...s_n)$ is a sequence of move patterns (or strategies), the goal is to guess correctly, which strategy $s_k, k = l(1)n$ is the best play. A simple algorithm for accomplishing this task based on remainder vectors is suggested in Adewoye (1990).

The algorithm emphasizes that a remainder vector $r = (r_1, r_2, ..., r_n)$ computed for each move pattern and the pattern whose remainder vector has at least $l^1 - norm$ is the best

move. The task is synonymous with solving the positive definite constraint optimization problem:

$$\text{Minimax} \sum_{i=1}^{n} r_i, \; r_i > 0$$

Formally, this is called the remainder vector of an arbitrary vector $y = (y_1, y_2, ..., y_n)$, if the m-times **T** transformation applied to **y** gives **r**, that is, $T^m(n) = r$. if **r** = 0, then **y** is a CDG-vector. Remainder vectors provide a powerful strategy for playing *Ayo* and help to suggest a not-so-obvious option. Computing a remainder vector is essentially peeping into the future moves to determine a more futuristic strategy. This is the kind of thing an experienced player would like to do when the game is close to the end.

Let $y = (y_1, y_2, ..., y_n)$ be an arbitrary vector, $y_i \leq i+1, \forall i = l(1)n$ and suppose $\lfloor x \rfloor$ is the floor function of the real number **x**, the following recast algorithm of Adewoye (1990) will compute the remainder vector **r** in linear time.

Algorithm CDGNearness(y)

Initialization

$$i = n, g_i = 0, T_i = \lfloor y_i /(i+1), \rfloor$$
$$ri = (y_i + (r_i) \bmod (i+1);$$

Begin

While (i > 1)

{

i = $i - 1$;

$G_i = G_{i+1} + T_{i+1}$,

$T_i = \lfloor (y_i + G_i /(i+1 \rfloor$;

$R_i = (y_i + G_i) \bmod (i+1)$;

}

return || r ||;

END CDGNearness

Theorem 1

The payoff matrix of any CDG configuration does not always have a saddle point.

***Proof*:** The payoff matrix of Ayo at any stage of the game can be represented as:

$a_{ij} = x_i - y_j \setminus x_i$ as in (1) above.

This results in a payoff matrix *P*(M) of size 6 x 6 of the form:

$$P(M) = \begin{pmatrix} x_1 - y_{11} & x_1 - y_{12} & x_1 - y_{13} & x_1 - y_{14} & x_1 - y_{15} & x_1 - y_{16} \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ x_6 - y_{61} & x_6 - y_{62} & x_6 - y_{63} & x_6 - y_{64} & x_6 - y_{65} & x_6 - y_{66} \end{pmatrix} \quad (2)$$

where

$x_i \geq 0 \quad i = 1, 2, ..., m; 1 \leq m \leq 6$

$y_{ij} \geq 0 \quad j = 1, 2, .. , n; \; 1 \leq n \leq 6, \quad i = 1, 2, ..., m; 1 \leq m \leq 6$.

To prove the theorem, we consider the payoff matrix in the CGD 7 of Table 3.1 for the game configuration in Figure 3.8 given that player Y played first. The configuration becomes:

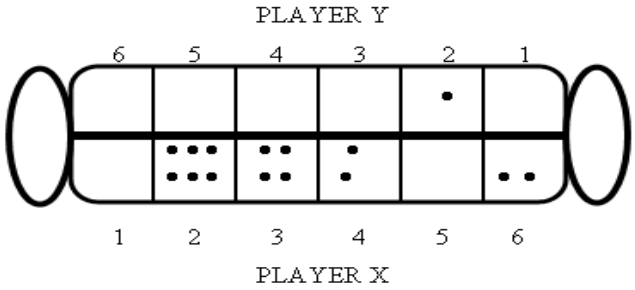

Figure 3.10: Dual Equilibrium in CDG Configuration in *Ayo* Game

The resulting payoff matrix is given as:

$$: \quad \begin{pmatrix} 0 & -2 & 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \end{pmatrix}$$
$$(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$

Figure 3.11: Payoff Matrix

The payoff matrix above has no saddle point (i.e point at which maximin equals minimax), since the maximum of the minimum of the rows is -2 and the minimum of the maximum of the columns is 0, while the payoff matrix exists. The strategies that is open to player X and Y via the reduced game payoff matrix are X_0 = (1, 0, 0, 0, 0, 0) and Y_0 = (0, ½, 0, 0, 0, ½). This completes the proof. Hence, we conclude that a CDG of any configuration has a payoff matrix, but without a saddle point.

***Theorem 2*:** All maximal CDG-vectors derive an *M* x *N* matrix

Proof: To establish the validity of the theorem, it is sufficient to show that

$$A(n) = x_1 + x_2 + \ldots + x_n$$

Let *n* be the board size of *Ayo* and let *m* be the number of blocks of arithmetic sequences that appear in the final CDG-vector $y_1, y_2, \ldots, y_n$ as in Table 3.2. We give an example of *n* = 16 in the step-by-step implementation of algorithm 1 to illustrate the proof as shown in Figure 3.12 below.

$$
\begin{array}{cc}
j: & \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{array} \\
\begin{pmatrix}
-13_1 & -11_1 & -9_1 & -7_1 & -5_1 & -3_1 & -1_1 & 1_1 & 3_1 & 5_1 & 7_1 & 9_1 & 11_1 & 13_1 & 15_1 & 17_1 \\
-4_1 & -2_1 & 0_1 & 2_1 & 4_1 & 6_1 & 8_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2_1 & 0_1 & 2_1 & 4_1 & 6_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1_1 & 1_1 & 3_1 & 5_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0_2 & 2_2 & 4_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1_3 & 3_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \\
x_j: & \begin{array}{cccccccccccccccc} 1 & 1 & 4 & 4 & 5 & 3 & 7 & 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \end{array}
\end{array}
$$

Figure 3.12: Matrix Computation

The arithmetic sequence of the first block is 17, 15, 13, …., 3, 1, that is, $x_{16}, \ldots, x_8$. But x_7, as part of the sequence, has a value of -1. Since an entry in CDG-vector must be positive, then the second block begins in the 7^{th} column. Computing the x_7, from the algorithm, we have x_7 to be 7.

Let the first column begin in column *n* such that $k_n = 1$, written as subscript. Similarly, let **k** be the smallest positive integer such that $k(j+1) - 1 > 0$, and call it k_j. such that $k_7 = 1$, then entries of the second column are determined as $(j + 1)_{kj}$, $(j - 1)_{kj}$, $(j - 3)_{kj}$ and so on. Now, summing up the values column by column of the entries, we obtain an n-vector $y_1, y_2, \ldots, y_n$. To obtain the r-th row for which $k_j = 2, 3, \ldots$, we first sum column by column in all the entries of rows with $k_n = 1$, so as to obtain an n-vector $y_1, y_2, \ldots, y_n$.. If $\Sigma x_{i's} = x_j$ for all j, then r-th row does not exist, otherwise, let $k_j = 2, 3, \ldots$, for the r-th row. Note that when computing the value of the $x_{j's}$, it is taken that $a_{(kj)} = a(kj)$. For example, $4_2 = 4(2) = 8$.

Now, for $A(n) = x_1 + x_2 + \ldots + x_n$, we sum up the entries of the matrix in Figure 3 row-wise and then add up the x_j. From the proof, we obtain a 7x16 matrix in Figure 3.12, indicating that the number of blocks when n = 16 is 7 by taking the blocks as $x_{16} - x_8$, $x_7 - x_6$, x_5, x_4, x_3, x_2, and x_1 as shown in table 3.3 below.

Table 3.3: Blocks of sequences in Maximal CDG Vectors

	Maximal CDG-vectors																									
n	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	.	.	.	.....	x_{16}	.....												x_n
1	2																									
2	1	3																								
3	2	2	4																							
4	1	1	3	5																						
5	1	3	2	4	6																					
6	2	2	1	3	5	7																				
7	1	3	4	2	4	6	8																			
8	2	2	3	1	3	5	7	9																		
9	1	3	1	5	2	4	6	8	10																	
10	1	1	4	4	1	3	5	7	9	11																
11	2	2	2	2	6	2	4	6	8	10	12															
12	1	1	1	1	5	1	3	5	7	9	11	13														
13	1	3	2	4	3	7	2	4	6	8	10	12	14													
14	2	2	1	3	2	6	1	3	5	7	9	11	13	15												
15	1	3	1	5	6	4	8	2	4	6	8	10	12	14	16											
16	1	1	4	4	5	3	7	1	3	5	7	9	11	13	15	17										
17	2	2	2	2	3	1	5	9	2	4	6	8	10	12	14	16	18									
18	1	1	3	5	1	7	4	8	1	3	5	7	9	11	13	15	17	19								
19	1	3	4	2	5	5	2	6	10	2	4	6	8	10	12	14	16	18	20							
20	2	2	3	1	4	4	1	5	9	1	3	5	7	9	11	13	15	17	19	21						
21	1	3	3	3	1	1	7	3	7	11	2	4	6	8	10	12	14	16	18	20	22					
22	1	1	3	5	5	7	6	2	6	10	1	3	5	7	9	11	13	15	17	19	21	23				
23	1	3	4	2	2	4	3	9	4	8	12	2	4	6	8	10	12	14	16	18	20	22	24			
24	2	2	3	1	1	3	2	8	3	7	11	1	3	5	7	9	11	13	15	17	19	21	23	25		
25	1	1	1	1	3	7	8	6	1	5	9	13	2	4	6	8	10	12	14	16	18	20	22	24	26	
26	1	3	2	4	1	5	6	4	10	4	8	12	1	3	5	7	9	11	13	15	17	19	21	23	25	27
.	.																									
.	.																									
.	.																									

To support the claim above, we present the following proposition.

***Proposition IV*:** The sum A(n) is given by

$$A(n) = \sum_{j=1}^{n} j(d_j - d_{j+1}) \text{ (by using algorithm I above)}$$

Proof: We shall show that

$$A(n) = \sum_{j=1}^{n} 2jk_j \text{ where } k_j = \left\lfloor \frac{(j+1) - (x_{j+1} - d_{j+1})}{j+1} \right\rfloor$$

From algorithm I, and $\forall\ j < n$, we have

$$d_j = d_{j+1} + 2k_j \text{ and}$$
$$x_j = (j+1)k_j + x_{j+1} - d_{j+1} \quad (*)$$

Thus,

$$x_j = (j+1)k_j + x_{j+1} - d_{j+1}$$

$$= (j+1)k_j + (j+2-2)k_{j+1} + x_{j+2} - 2d_{j+2} \quad (\textit{if } j+2 \le n)$$

$$= (j+1)k_j + (j+2-2)k_{j+1} + (j+3-2*2)k_{j+2} + x_{j+3} - 3d_{j+3} \quad (\textit{if } j+3 \le n)$$

$$= \sum_{r=0}^{n-j-1} (j+1-r)k_{j+r} + x_n - (n-j)d_n \quad (\textit{when } (*) \textit{ is repeatedly applied})$$

$$= \sum_{r=0}^{n-j-1} (j+1-r)k_{j+r} + (n+1) - 2(n-j) \quad (\text{since } x_n = n+1 \text{ and } d_n = 2)$$

This expression yields an n X n matrix, say M = (m_{ij}), such that $x_j = \sum_{i=1}^{n} m_{ij}$ as shown below.

-n+3	. . .	n-3	n-1	n+1
$(-n+4)k_{n-1}$	. . .	$(n-2)k_{n-1}$	nk_{n-1}	0
$(-n+5)k_{n-2}$	. . .	$(n-1)k_{n-2}$	0	0
.	.	0	0	0
.	.	.	.	.
.	.	.	.	.
$2k_1$	0 . . .	0	0	0

Now, by summing the entries of M row by row, we have that $A(n)=\sum_{j=1}^{n} 2jk_j$ ■

Surprisingly, we noticed that if we remain on a row of Table 3.3 above and approach from the right, we have a sequence of numbers with common difference of 2, which changes as we hit the 'wall'. These 'walls' form an interesting staircase patterns that are grouped into blocks of sequences as shown in Table 3.3 above.

From the staircase pattern, we conjecture thus:

Conjecture 1: For all *n>0,*

$$f(n)=\begin{cases}\left\lfloor \frac{1}{3}(n+5)\right\rfloor + C, & n<25\\ \left\lfloor \frac{1}{4}(n-25)\right\rfloor +10, & otherwise\end{cases}$$

where $C=\begin{cases}1 & ;\ n=15\\ -1 & ;\ n=1,\ 2,\ or\ 4\\ 0 & ;\ otherwise\end{cases}$, n is the board size, *s(n)* is the sum of seeds of the maximal CDG, and *f(n)* is the number of blocks of arithmetic sequences that appear in the maximal CDG in Table 3.3, in line with the implementation of algorithm I.

From the conjecture, we derive that $g(n)=\begin{cases}\left\lfloor \frac{1}{3}(n+5)\right\rfloor, & n<25\\ \left\lfloor \frac{1}{4}(n+15)\right\rfloor, & otherwise\end{cases}$ for generating an integer sequence *g(n)* in Table 3.4 below:

Table 3.4: Computational Result for Conjecture 1

n	*s(n)*	*f(n)*	*g(n)*	*n*	*s(n)*	*f(n)*	*g(n)*	*n*	*s(n)*	*f(n)*	*g(n)*
1	2	1	2	18	130	7	7	35	440	12	12
2	4	1	2	19	148	8	8	36	452	12	12
3	8	2	2	20	152	8	8	37	496	13	13
4	10	2	3	21	172	8	8	38	508	13	13
5	16	3	3	22	196	9	9	39	538	13	13
6	20	3	3	23	208	9	9	40	568	13	13
7	28	4	4	24	212	9	9	41	610	14	14
8	32	4	4	25	238	10	10	42	620	14	14
9	40	4	4	26	256	10	10	43	646	14	14
10	46	5	5	27	272	10	10	44	670	14	14
11	56	5	5	28	280	10	10	45	716	15	15
12	58	5	5	29	320	11	11	46	730	15	15
13	76	6	6	30	328	11	11	47	778	15	15
14	80	6	6	31	358	11	11	48	820	15	15
15	100	7	6	32	370	11	11	49	838	16	16
16	106	7	7	33	400	12	12	50	868	16	16
17	116	7	7	34	416	12	12				

From our conjecture, the number 25 indicates that at any instance of play in the game of *Ayo*, the moment a player captures 25 seeds, the player emerges the winner. But on the contrary, a stalemate (or draw) may arise, that is, each player captures 24 seeds or the same number of seeds lesser than 24. Having this in mind, it is then important to obtain a general term for all numbers within the context of integer sequence that could range

from 1 to infinity, even though the total number of seeds on board is not more than 48, which this work was able to achieve.

3.5 SEEDS CAPTURING, PERIODIC PATTERN AND SEQUENCES

It has been shown by Broline and Loeb (1995) that a number of seeds could be captured (harvested) in an unlimited pits and that as the number *n* pits increases, the sum of seeds on the board s(*n*) also increases as n^2/pi + O(*n*), as shown in Table 3.4 above. In the work, an arithmetic sequence whose common difference is 2 and whose largest term is n (the number of pits) were sum. This sum counts the number of seeds in pits n/2 to n. Next, there is an arithmetic sequence whose largest term is approximately n/2 and whose common difference is 4. This accounts for (approximately) the seeds in pits 3n/8 to n. Next, there is an arithmetic sequence whose largest term is approximately 3n/8 and whose common difference is 6. This argument is continued to get the asymptotic results, but it was noted that this asymptotic result varies widely from precise results for specific numbers.

Now, let us revisit the play and seed capturing strategy of the CDG earlier discussed in section 3.2.7.1, where the initial game distribution is shown in figure 3.8 above. After the move by player Y (the first player to move), player X captures two seeds by playing from pit 6. Since there is only one positional move that is appropriate for a valid CDG move, otherwise the anticipated total seeds to be captured will not be possible. If there is more than one pit that have enough seeds to capture seeds for which the CDG strategy must hold, then it is the pit at the rightmost pit that must be selected as the best move.

For example, the play and capturing strategy is given in Table 3.5 in bottom-up manner. That is, at n = 21 (total seeds on board), there are 20 seeds available for player X and only 1 for player Y that has only one option of play. When player Y moves form his own pit 1 to pit 2 (that is Y_1 and Y_2 respectively), the game configuration gives the n = 20 arrangement such that when player Y plays, the arrangement of n = 19 is obtained. This process is continued till the end as shown in Table 3.5.

Table 3.5: Play and Capturing Strategy in CDG

n	X_1	X_2	X_3	X_4	X_5	X_6	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
1	0	0	0	0	0	0	1	0	0	0	0	0
2	0	0	0	0	0	2	0	1	0	0	0	0
3	0	0	0	0	0	2	1	0	0	0	0	0
4	0	0	0	0	3	1	0	1	0	0	0	0
5	0	0	0	0	3	1	1	0	0	0	0	0
6	0	0	0	4	2	0	0	1	0	0	0	0
7	0	0	0	4	2	0	1	0	0	0	0	0
8	0	0	0	4	2	2	0	1	0	0	0	0
9	0	0	0	4	2	2	1	0	0	0	0	0
10	0	0	5	3	1	1	0	1	0	0	0	0
11	0	0	5	3	1	1	1	0	0	0	0	0
12	0	6	4	2	0	0	0	1	0	0	0	0
13	0	6	4	2	0	0	1	0	0	0	0	0
14	0	6	4	2	0	2	0	1	0	0	0	0
15	0	6	4	2	0	2	1	0	0	0	0	0
16	0	6	4	2	3	1	0	1	0	0	0	0
17	0	6	4	2	3	1	1	0	0	0	0	0
18	7	5	3	1	2	0	0	1	0	0	0	0
19	7	5	3	1	2	0	1	0	0	0	0	0
20	7	5	3	1	2	2	0	1	0	0	0	0
21	7	5	3	1	2	2	1	0	0	0	0	0

Following this description, we obtain a capturing sequence given in Table 3.6 below. For example, if we consider a game configuration of the form 6, 4, 2, 3, 1, 1; such that one seed belongs to player Y and the remaining 16 seeds belong to player X as shown in Figure 3.13 below.

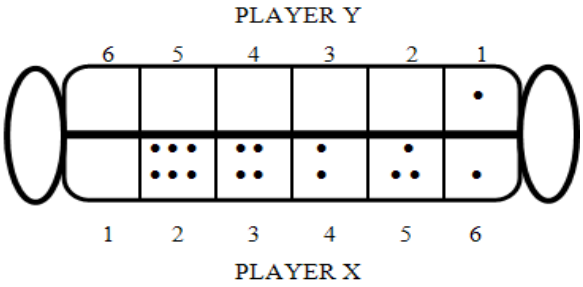

Figure 3.13: A Typical CDG Configuration for Capturing Sequence in *Ayo* Game

From this configuration, as soon as player Y moves from pit 1 to pit 2, player X must move from pit 5 that has 3 seeds to capture 2 seeds from pit 2 of pit Y. The capturing sequence here is 1,3. Now, there would be 2 seeds available in pit 6 of player X such that when player Y moves from his pit 1 to pit 2, the capturing sequence again give 1, 2 thereby making the sequence to be 1, 3, 1, 2. Continuing this way, the entire capturing sequence becomes 1,3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1. Table 3.6 below depicts the distribution of seeds and the capturing sequence for up to 21 seeds such that player X has 20 seeds and player Y has only 1 seed to obey the CDG rule.

Table 3.6: Distribution of Seeds and the Capturing Sequence for up to 21 Seeds

n	X_1	X_2	X_3	X_4	X_5	X_6	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Capturing Sequence
1	0	0	0	0	0	0	1	0	0	0	0	0	1
2	0	0	0	0	0	2	0	1	0	0	0	0	2,1
3	0	0	0	0	0	2	1	0	0	0	0	0	1,2,1
4	0	0	0	0	3	1	0	1	0	0	0	0	3,1,2,1
5	0	0	0	0	3	1	1	0	0	0	0	0	1,3,1,2,1
6	0	0	0	4	2	0	0	1	0	0	0	0	4,1,3,1,2,1
7	0	0	0	4	2	0	1	0	0	0	0	0	1,4,1,3,1,2,1
8	0	0	0	4	2	2	0	1	0	0	0	0	2,1,4,1,3,1,2,1
9	0	0	0	4	2	2	1	0	0	0	0	0	1,2,1,4,1,3,1,2,1
10	0	0	5	3	1	1	0	1	0	0	0	0	5,1,2,1,4,1,3,1,2,1
11	0	0	5	3	1	1	1	0	0	0	0	0	1,5,1,2,1,4,1,3,1,2,1
12	0	6	4	2	0	0	0	1	0	0	0	0	6,1,5,1,2,1,4,1,3,1,2,1
13	0	6	4	2	0	0	1	0	0	0	0	0	1,6,1,5,1,2,1,4,1,3,1,2,1
14	0	6	4	2	0	2	0	1	0	0	0	0	2,1,6,1,5,1,2,1,4,1,3,1,2,1
15	0	6	4	2	0	2	1	0	0	0	0	0	1,2,1,6,1,5,1,2,1,4,1,3,1,2,1
16	0	6	4	2	3	1	0	1	0	0	0	0	3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1
17	0	6	4	2	3	1	1	0	0	0	0	0	1,3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1
18	7	5	3	1	2	0	0	1	0	0	0	0	4,1,3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1
19	7	5	3	1	2	0	1	0	0	0	0	0	1,4,1,3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1
20	7	5	3	1	2	2	0	1	0	0	0	0	5,1,4,1,3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1
21	7	5	3	1	2	2	1	0	0	0	0	0	1,5,1,4,1,3,1,2,1,6,1,5,1,2,1,4,1,3,1,2,1

From Table 3.6, it was noticed that when the game states for which proper maximal CDG occurs for 6 pits, 5 pit, through to 1 pit is given as n = 1, 3, 5, 9, 11, 17, 21. This interestingly, corresponds to the sequence A007952 in the online Encyclopedia of Integer Sequence.

Again, there could be a self-replicating pattern while distributing seeds in *Ayo* game, which is referred to as matching groups Bruhn (2005); Bouchet,(2005). A matching group is made up of successive pits in *Ayo* whose numbers of seeds are given by the sequence n, $n - 1$, . . ., 2, 1. A typical move will result in replicating the initial pattern

with a right-shift by distributing seeds from the pit having the largest number of seeds, and placing one seed in each succeeding pit as in figure 3.14.

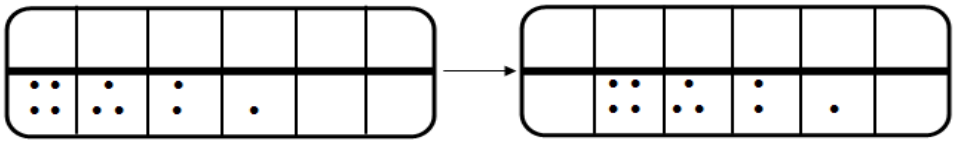

Figure 3.14: Matching Group in Ayo

If the distribution is left uninterrupted, repeated application will allow it to propagate in this way as far as needed. For example, considering a board configuration that has 10 seeds as 5, 3, and 2, a number of distributions could be made and obtain a distribution that results in a matching group as shown below.

[5, 3, 2]→[4, 3, 1, 1, 1]→[4, 2, 2, 2]→[3, 3, 3, 1] → [4, 4, 2]→[5, 3, 1, 1]→[4, 2, 2, 1, 1] →

→[3, 3, 2, 2]→[4, 3, 3]→[4, 4, 1, 1]→[5, 2, 2, 1]→[3, 3, 2, 1, 1]→[4, 3, 2, 1]

In the same light, there could be some configurations that will result in the initial pattern but not necessarily a matching group after some periods. In other words, the matching group rule can be used to produce periodic behaviour thereby showing that some number of seeds exhibits certain periodicity. Here, the initial configuration of seeds will follow the matching group rule and return to the initial state after some iterations, which gives its period. For example, if there are four (4) seeds arranged as 2, 1, 1, one will only obtain the initial configuration after 3 iterations, thereby making number 4 to be a period 3 system as shown in Figure 3.15 below.

Figure 3.15: A Period 3 System

Sequel to the above, some numbers of seeds and their corresponding pattern behaviour is presented as shown in Table 3.7 below.

Table 3.7: Pattern Behaviour of up to 21 Seeds in *Ayo*

Total Seeds	Pattern Behaviour
1	Matching Group
2	Period 2
3	Matching Group
4	Period 3
5	Period 3
6	Matching Group
7	Period 4
8	Period 4
9	Period 4
10	Matching Group
11	Period 5
12	Period 5
13	Period 5
14	Period 5
16	Matching Group
17	Period 6
18	Period 6
19	Period 6
20	Period 6
21	Matching Group

Again, it could be observed that the numbers of seeds that lead to matching groups are 1, 3, 6, 10, 15, 21, and so on (see Table 3.7), which is represented as a triangular sequence A000217 in the online Encyclopedia of Integer Sequence as shown in Figure 3.16, and can generally be obtained as [n(n+1)] /2 as shown in Table 3.8.

1

1+2=**3**

(1+2)+3=**6**

(1+2+3)+4=**10**

(1+2+3+4)+5=**15**

(1+2+3+4+5)+6=**21**

…..

Figure 3.16: Triangular Sequence from Matching Group Numbers in *Ayo*

Table 3.8: Generation of n^{th} term for Matching Group Numbers in *Ayo*

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
[n(n+1)]/2	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	153	171	190	210

CHAPTER FOUR

HEURISTIC DECISION MAKING SYSTEM FOR PLAYING *AYO* GAME

4.1 GAME TREE SEARCH AND HEURISTIC EVALUATION

A *search algorithm* takes a problem (i.e., a game) as input and returns a solution in the form of an action sequence (i.e., a move sequence) (Russell and Norvig, 1995). Many tree-search algorithms were developed in the last century. For tree-search-based optimization problems, the A* algorithm (Hart *et al*, 1968) is one of the standard algorithms. For two-player games, the foundation of most algorithms is Minimax (von Neumann, 1928). Minimax has been improved into a more powerful algorithm: αβ (Knuth and Moore, 1975). There are many variants of the αβ algorithm. Amongst them, one of the most successful is the iterative deepening principal variation search (PVS) (Marsland, 1983), which is nearly identical to nega-scout (Reinefeld, 1983). They form the basis of the best programs in many two-player games, such as chess (Marsland and Bj¨ornsson, 2001). Other αβ variants are MTD(*f*) (Plaat, 1996) and Realization-Probability Search (Tsuruoka *et al*, 2002). Several other algorithms exist for tree-search, for instance, Proof-Number (PN) search and its variants (Allis *et al.,* 1994; Winands and Bj¨ornsson, 2008), B* (Berliner, 1979), SSS* (Stockman, 1979), and λ-search (Thomsen, 2000). Most of these algorithms rely on a *positional evaluation function*. This function computes a heuristic value for a board position at leaf nodes. The resulting value can be interpreted in three different ways (Donkers, 2003): (1) as a *prediction* of

the game-theoretic value, (2) as an estimate of the *probability* to win, or (3) as a measure of the *profitability* of the position.

4.1.1 Position Evaluation in *Ayo*

Generally, in any *Ayo* game configuration, each player has a maximum of 6 pit choices to move from and each of them has its own corresponding game value (payoff) associated to it. Basically, a game tree algorithm computes the root successor with the highest payoff for the current player or the minimax value of a game tree for inferring best move.

The traditional AI approach to game programming is a brute-force search, which is based on defining a game as a kind of search problem with the following components:

i. The initial state: the board position and an indication of which player is to move
ii. A set of operators: what are the legal moves a player can make?
iii. A terminal test: is the game over?
iv. A utility function: a numeric value for the outcome of a game

In *Ayo* game playing, instead of determining the exact utility for a state, it is advisable to get it approximated by an evaluation function. An evaluation function, also known as static evaluator, is a heuristic function used to estimate the value or "goodness" of a game position. The evaluation function is where the domain experts' knowledge resides and it is the most important element of intelligence in evolving a game player and not necessarily the search depth. Here, heuristics are used to reliably evaluate a game

position so that a game tree search can be cut off at a certain depth. Typically, it is the construction of such an evaluation function that is the problem of building a strong game playing engine for most games. Generally, the number of plies that can be looked ahead is an important factor in the accuracy of the evaluation function – searching deeper into the game tree (usually) means that the estimated odds of winning the game are closer to the true odds. Using good heuristics as ingredients for an accurate enough evaluation function together with a search engine performing deep searches form the core of most computer programs for two-player games with perfect information. Well known examples of games in which this approach has been successfully applied are chess, checkers, shogi, othello, awari, gomoku and nine men's morris (Russel and Norvig, 1995).

4.2 MATHEMATICAL DESCRIPTION OF COMPLEXITY OF *AYO* GAME

Ayo is a count-and-capture, two-persons-zero-sum strategic board game that is rapidly becoming popular around the world. *Ayo* belongs to the family of board games called Mancala such as Awale, Owari and Awari. *Ayo* is one of the oldest known combinatorial games of perfect information. Two persons play *Ayo* turn-by-turn at a time with the board put in between the players. The rules of the game and its strategies have been previously discussed in chapter three. The main goal of the game is for a player to capture as many seeds as possible.

In *Ayo* game a position consists of a certain distribution of seeds in the pits and it includes the captured seeds in the stores or kept by the players if there are no stores. The

number of possible positions is a function of the number of pits (and stores) and the number of seeds. Generally, for mancala family, this number has been derived from basic combinatorics and is given by Donkers *et al.*, (2002) as:

$$P = K\binom{n+m-1}{m} \qquad (1)$$

where **k** is the number of players, ***m*** is the total number of seeds and ***n*** is either the total number of pits and stores together or the total number of pits incremented with the number of players if no stores are present. The number P increases very rapidly with increasing number of pits and seeds. For example, given the initial board state of Ayo as shown in Figure 4.1 below, the value of *P* could be computed. **K** should equal two, since there are only 2 players playing, *n* must equal 14 because there are 12 pits plus 2 stores.

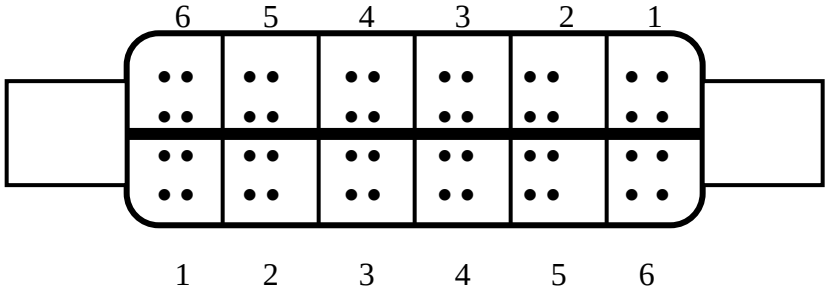

Figure 4.1: Initial Board Configuration before the Start of Game Play

m must equal 48 since there are 4 marbles in each of the 12 pits. Hence, the outcome of this is 1.313244 x 10^{14}. This number represents the number of positions possible, or the possible number of moves that can be made. Of course, it would be hard to think of all of the possible moves that a player could make, and it would be hard to see what would happen as the outcome of this game.

The process of seed sowing in *Ayo* game describes a linear motion that can repeat θ times, called θ cycles, usually θ = 0, 1, 2, and so on. When θ = 0, the distance travelled is less than n – k, but when θ > 0, the distance travelled is greater than n – k and this is called kroo (Daoud, *et al.*, 2004) or **Odu** (Broline & Loeb 1995; Odeleye 1977). Let (i, j) denote a pair of positions that both players (North and South) can scoop or drop seeds during the sowing assignment. This coordinate denotes player's pit ***i*** and opponent's pit ***j***. The value $\mathbf{p_x}$ is the number of seeds in **x** and $p_x \varepsilon S$, where S is the set of seeds, for which $i = 0(1)n/k - 2$, and $j = (n/k - 1)(1)(n - k - 1)$.

An equivalence relationship exists between the features p_x, i and j according to the following formula:

$$< (P_{x,}(\frac{n}{k-2})\theta) > \quad = \quad \begin{cases} \{(i,j):(p_{x,}(n-k-1)\theta) \sim (i,j)\}, x = i \\ \{(i,j):(p_{x,}n-k+(n-k-1)\theta) \sim (i,j)\}, x = j \end{cases} \qquad (2)$$

By equivalence relation, we mean a measure of likeness or similarity of objects. A relation ~ on a set **S** is called equivalence if it has the following three essential properties:

i. Reflexive; for each a in S, a ~ a
ii. Symmetric; if a ~ b, then b ~ a
iii. Transitive; if a ~ b and b ~ c, then a ~ c.

If ~ is an equivalence relation on set S, the equivalence class of each element a in S, denoted by <a> is the set of elements to which a is related and it is given as:

$<a> = \{x \mid a \sim x\}$

Using the symmetric property of ~ and for a given coordinate (i, j), it follows from equation (2) that:

$$p_x = \begin{cases} j - i + (n - k - 1)\theta, & x = i \\ i - j + n - k + (n - k - 1)\theta, & x = j \end{cases} \qquad (3)$$

The value p_x is equivalent to the linear distance travelled during the sowing assignment.

4.3 THE REFINEMENT–BASED HEURISTIC STRATEGY

The Refinement-Based Heuristic (RBH) strategy utilized the minimax concept as it was considered to enhance minimax algorithm to evolve an Ayo game player. The idea of minimax algorithm is such that for every Two-Person-Zero-Sum (TPZS) game like Ayo, two players (Max and Min) choose a legal move turn-by-turn and each tends to maximize his advantage to the detriment of the opponent. Max player tries as much as possible to increase the minimum value of the game, while Min tends to decrease its maximum value at a node as both players play towards optimality. This process is achieved using the stockman equality (Bruin *et al.*, 1994):

$$Score(p) = \begin{cases} \max\{ f(c) \mid c \ \ is\ a\ child\ \ node\ of\ \ n\} - f(n), \\ if\ \ n\ is\ a\ \min\ node \\ \\ \min\{ f(c) \mid c \ \ is\ a\ child\ \ node\ of\ \ n\} + f(n), \\ if\ \ n\ is\ a\ \max\ node \end{cases} \qquad (4)$$

Specifically, a node *n* in Ayo game tree G with game value f(G) is called feasible if f(*n*) = f(G) and *n* is the immediate child of root node *r*. In addition, if *n* gives the player the best reward, it is called a best node. The number in the nodes is the scores and the scores of the leaves follow from the rule of Ayo game which could be negative, suggesting highest payoff for Max, zero for equal payoff or positive for highest payoff for Min.

Payoff simply mean the value that would be attributed to the action of making a move from a particular pit. Example of a best node represented by bold dashes line and circle for the game configuration in Figure (4.2a) is shown in Figure (4.2b) below by using equation 4.

North					
N_6	N_5	N_4	N_3	N_2	N_1
0	0	1	0	1	1
0	4	2	1	3	1
S_1	S_2	S_3	S_4	S_5	S_6
South					

Figure 4.2a: A Typical *Ayo* Game Configuration

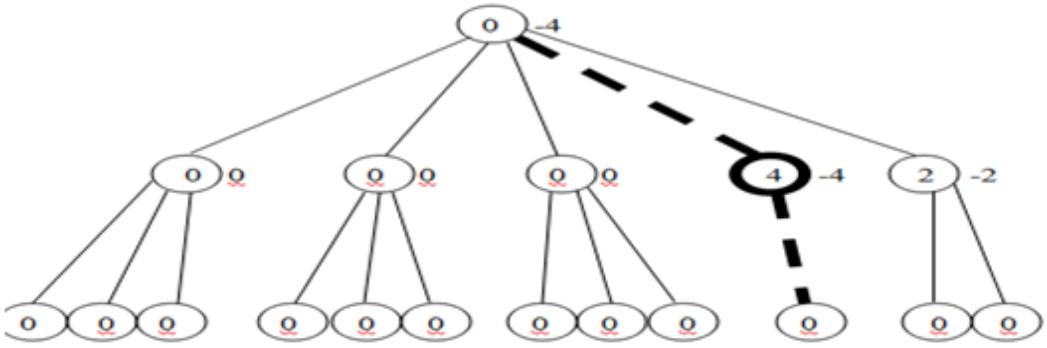

Figure 4.2b: Game Tree for a Typical *Ayo* Game Configuration

The RBH algorithm has three main components: (1) build game tree, (2) compute game value and (3) suggest best move after refining feasible moves. The "buildTree" procedure constructs a game tree in top-down fashion using breath-first traversal and nodes are evaluated as fan out is made to all nodes adjacent to their immediate parents. The "computeValue" procedure computes the game value bottom-up using Equation (4) above, and "predictMove" uses move refinement procedure (MRP) to predict the best

move. A refinement is a mapping that accepts a set of moves and then evaluates each move and returns a move with the best advantage.

The MRP uses a 2-ply look-ahead that is represented in a hyper-myopic rule as: Given a game state, let move [k] = {m_1, m_2, …, m_k} be a set of k feasible moves. m_k is called the head and m_1 the tail. A move is protected if it is not vulnerable to being forfeited when the opponent plays

(1) If k=1, then select the only available move and stop

(2) If tail/head is not protected for South/North player respectively, then select it, else select a move with the highest mobility strength.

In order to allow for mobility, bluffing and measure the correctness of move selection, the game strategies are simply regarded as episodes. The similarity between the i^{th} target episode x_i and the source episodes y_j of the j^{th} class is computed and the largest similarity measure is returned. The target episode with the maximum similarity greater than or equal to a given threshold value (called bluffing threshold) and a game value less or equal to the tradeoff seed is selected. Mobility is a process that allow for further game move no matter the pressure from the opponent while bluffing is the ability to tradeoff some seeds so as to gain an advantage of capturing more seeds in the nearest future. The similarity between two episodes x_i and y_j is calculated using the refined product-moment metric for linear correlation coefficient given as:.

$$Corr(x_i, y_j) = 1 - \frac{\sum_{k=1}^{m}(x_{ik} - \overline{x_i})(y_{jk} - \overline{y_j})}{\sqrt{\sum_{k=1}^{m}(x_{ik} - \overline{x_i})^2 \sum_{k=1}^{m}(y_{jk} - \overline{y_j})^2}} \qquad (5)$$

Other metrics used are the Canberra and angular metrics. But while this is done, it was observed that none of these metrics could be used in isolation. Nevertheless, the correlation metric was preferred because it suggests best moves faster and allow for bluffing better than the other two metrics (i. e. angular and canberra) as could be seen in the experimental results.

4.4 THE GAME ARCHITECTURE

The game-playing engine in this work is written in C++ and can be divided into three conceptual layers: the Game-Agent Interface, the Game-Play Interface and the Game-Logic Interface as shown in Figure 4.3. The Game-Agent interface handles external communications and manages the flow of the game by interacting with and executing command requests from the game player (for suggesting best move) in the simulation process. It queries the Game-Play interface for all intelligent behaviour regarding the game. It also includes a game parser for building a compact internal representation for referencing needed by the Game-Play interface. The parser converts moves sent from the game player into the internal form. Upon receiving a message, the agent saves the description in the message to a file where appropriate evaluation of action (or move) to be taken is carried out.

The Game-Logic interface encapsulates the state space of the game, provides information about available moves, and tells how a state changes when a move is made and whether the state is terminal and its goal value. It provides a well-defined interface for the Game Controller. Once initialized by the Game Agent layer, it spawns an

external process for translating the previously saved game description into C code. The generated code is compiled into a library responsible for all game-specific state-space manipulations.

The Game Play Interface is the main AI part of the agent responsible for its move decisions. The design for the play logic – called Game Players – uses a well-defined interface allowing different Game Player implementations to conveniently plug into the layer and use its services. We have experimented with three different heuristic metrics (Angular, Canberra, and Correlation) and carried out detailed performance analysis of these metrics relative to move selection (see sections 5.2.1 and 5.2.2).

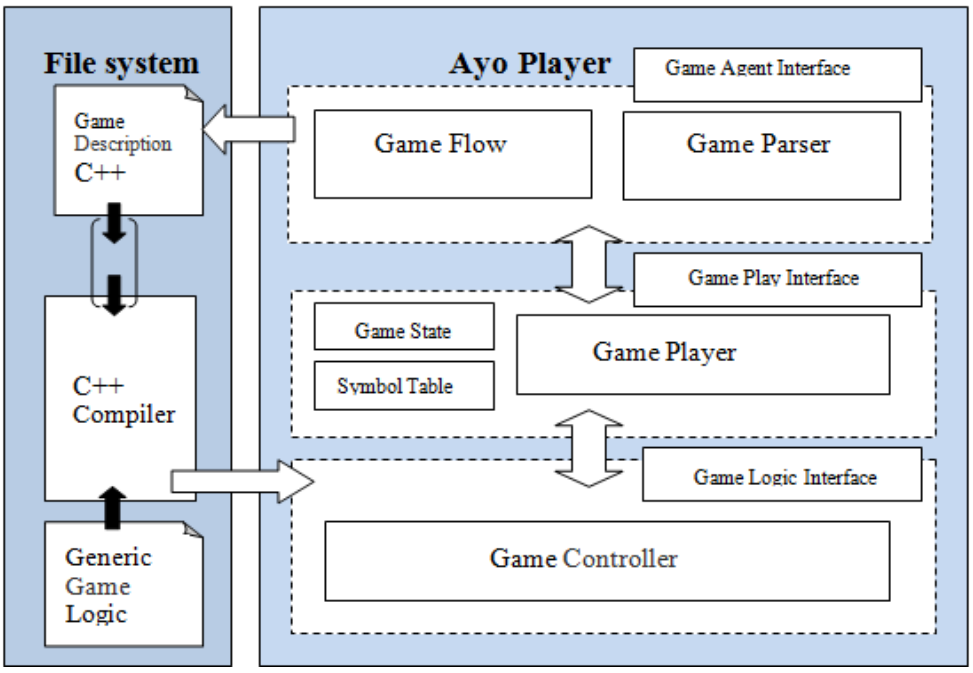

Figure 4.3: The Game Architecture

4.5 THE PROTOTYPE SIMULATION OF *AYO* GAME

Following the architecture described in section 4.5 above, the interface for the prototype simulation of the *Ayo* game is designed using C++ builder and it is divided into three parts; the menu bar, the game position evaluation part, and the game play interface. The menu bar has five other options, which are controlled by the game agent and the game logic. They are; retry, cancel, payoff, the south/north pit, and close. The game agent controls the retry, cancel, south/north pit and close, while the payoff is controlled by the logic. The game position evaluation part is made up of the move gain and game value for each of the pits denoted as S_1, S_2, . . ., S_6. The heuristic metrics (i.e. Canberra, Correlation and Angular) are used to evaluate the respective game values for each position by using a 4 ply look-ahead and thereby suggests the best move for the player as soon as the payoff button is clicked in the course of the game play. Best move refers to the best pit to play from by taking cognizance of the opponent reaction (play). The game play interface is a typical board representation of *Ayo* game with four seeds in each pit on either side of the board at the start of the game with store for each player. A typical screenshot of the simulated *Ayo* player is shown in Figure 4.4 below.

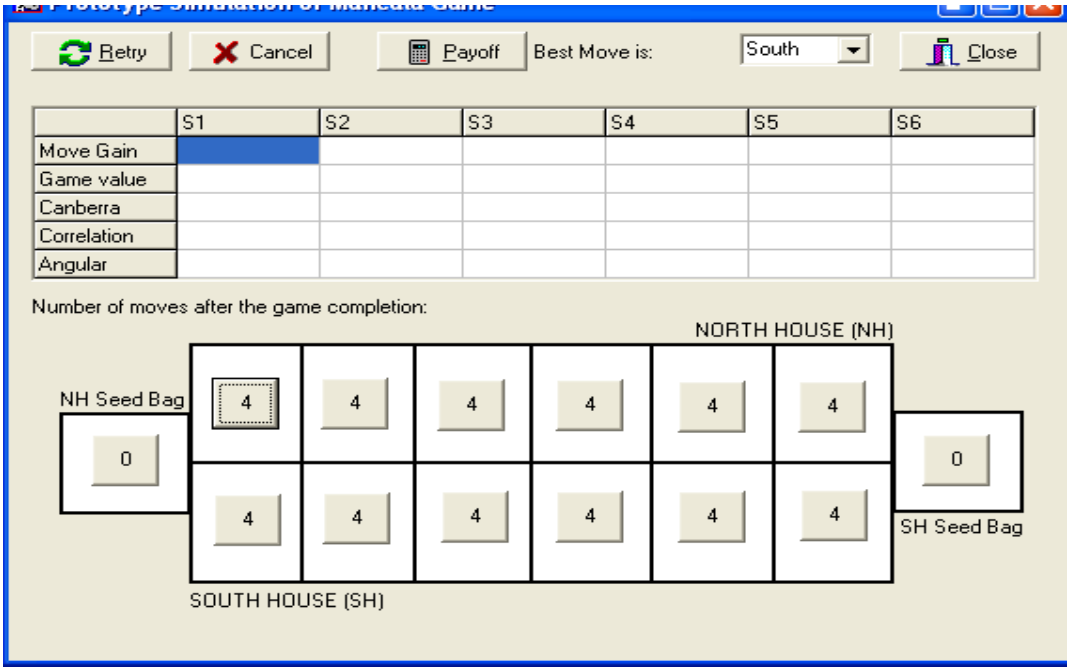

Figure 4.4: Screenshot of the *Ayo* Game Simulation

To play the game, the player would have to choose a pit (south or north), but by default, south is chosen. Any of the players (south or north) could start the game first. We have experimented with several cases whereby the prototype simulation started the game first while playing with Awale and a human expert player and as well played second. When playing the game, as soon as it is the turn developed application to play, payoff menu (button) on the interface is clicked, this spawns the game evaluation section and the respective game values are computed. Suggestion is then made on the best pit to move from. This is indicated in front of the payoff button as shown in Figure 4.5 .

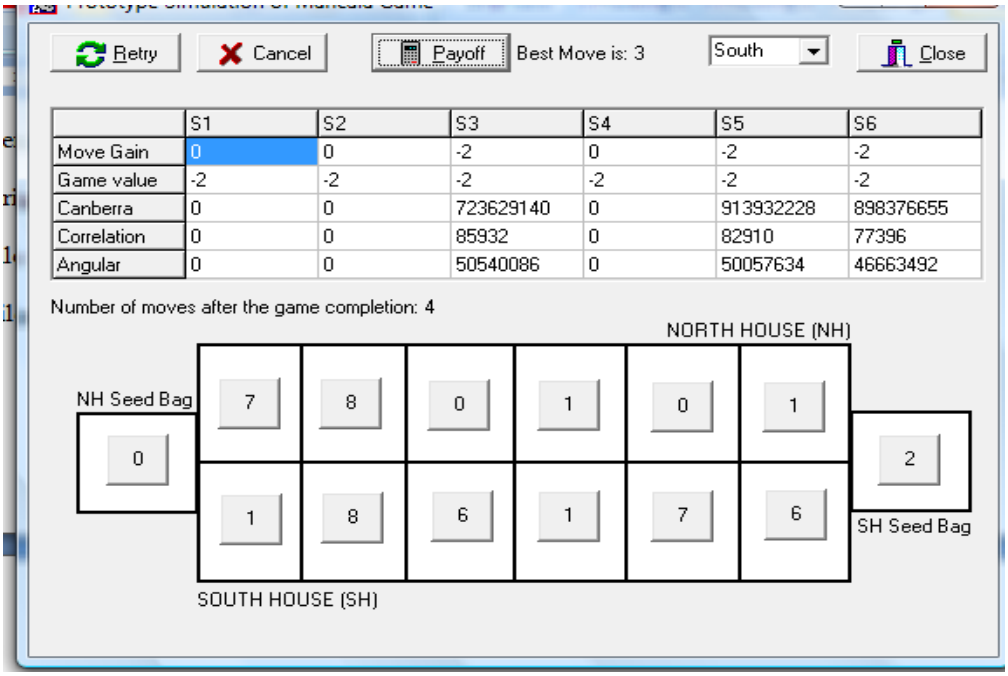

Figure 4.5: A Typical Screenshot of *Ayo* Simulation Showing Best Move

As seen in Figure 4.5, the payoff has suggested that move should be made from pit three (3) that has six (6) seeds; this gives rise to the capture of 2 seeds.

Apparently, there could be a situation when the player makes mistake in playing from a wrong pit as suggested, when this happens, the cancel button is simply clicked, and a dialogue box appears which requests if the operation is to be canceled. As soon as click 'yes' is clicked on, the former action is reversed hence the player could play the right suggestion. A typical screenshot for cancelation of an operation is shown in Figure 4.6 below. In between the game evaluation part and the game play interface is a mechanism that registers the number of moves taken for a particular game play.

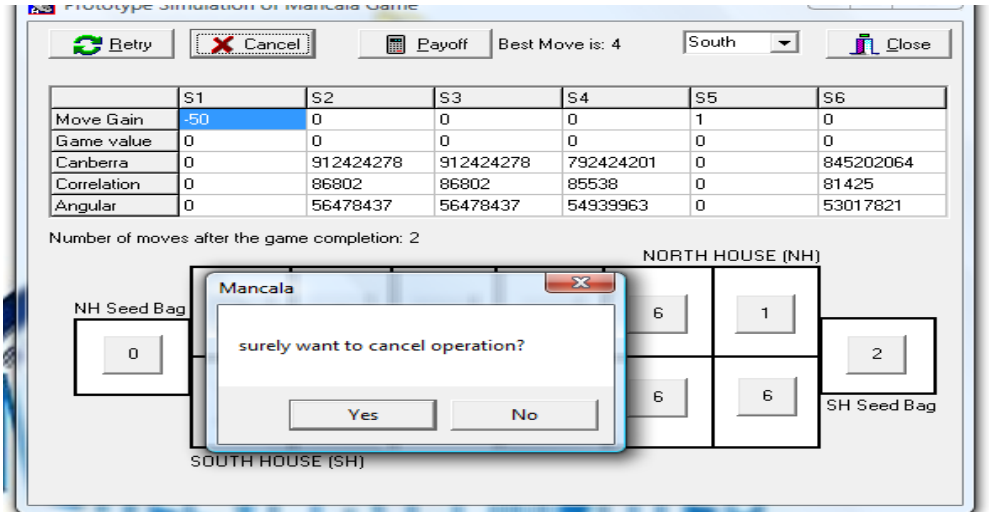

Figure 4.6: Simulation Screenshot Showing Cancellation of Operation

As soon as the game is over and the player is still interested in another play, the retry button is clicked. A dialogue box appears that request if the person would want to reset the board (see Figure 4.7). When the 'yes' button is selected, the initial game configuration of *Ayo* is displayed with four seeds on each pit.

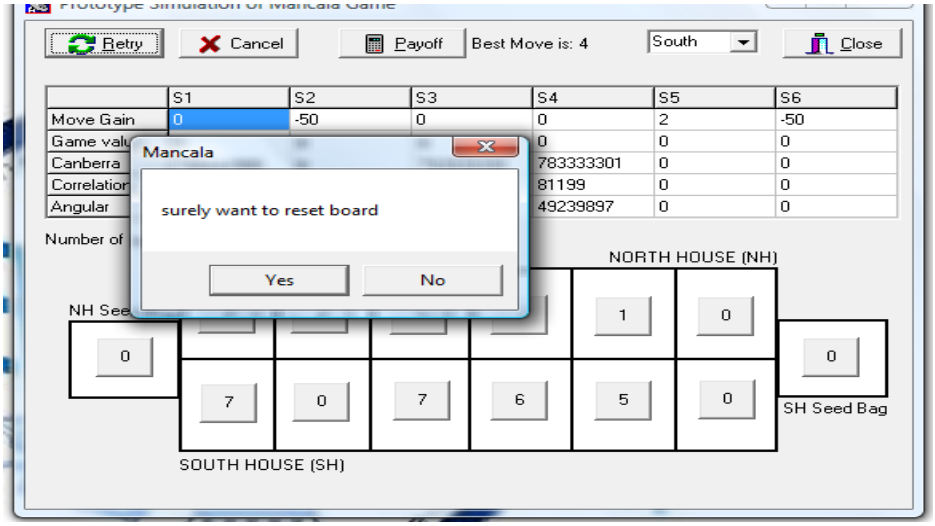

Figure 4.7: Simulation Screenshot Showing Board Reset

4.6 EXPERIMENTAL TESTS, DISCUSSION AND RESULTS

We implemented the Refinement-based Heuristic Decision making system to evolve *Ayo* player on a PC with Microsoft Windows XP Professional operating system and Pentium (R) 4, 3.00GHz, 80GB hard disk with 1GB RAM. The performance of the RBH was evaluated by playing series of games with Awale shareware (simply referred to as Awale). The Awale is still at the moment the benchmark program generally accepted within academic community. In order to test the prototype application, we registered to play with Awale at its various available levels, that is, Initiation, Beginner, Amateur and Ground master. Subsequently, some human players (experts and novice) of *Ayo* game were contacted and the developed application was made to play with them (see section 5.2.1 for detail report on the performance evaluation). The results obtained from six of the series of game played at each level having allowed each player to start thrice are recorded in Table 5.1 using the playing rules and the strategies described in sections 3.1 and 3.2.

The move-by-move account of a typical game play for two different games tournament played between RBH (south (S) player) and Awale grandmaster (north (N) player), when each of them starts first is represented in figures 4.8 and 4.9 below with the pits numbered from left to right. Appendix A presents the move recorded in the game scenario for the various game tournaments and arranged linearly in the form $nP_i(s)$ as shown in Figure 4.10.

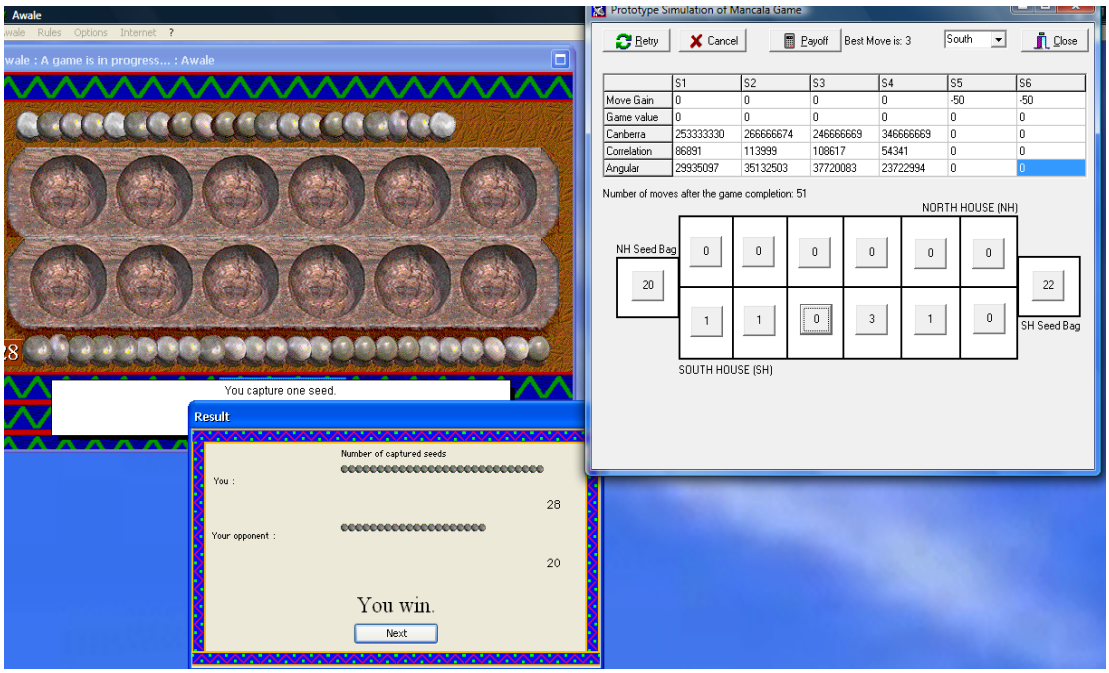

Figure 4.8: Screenshot Showing Complete Game Play With Total Seeds Capture and Number of Moves When RBH Starts First.

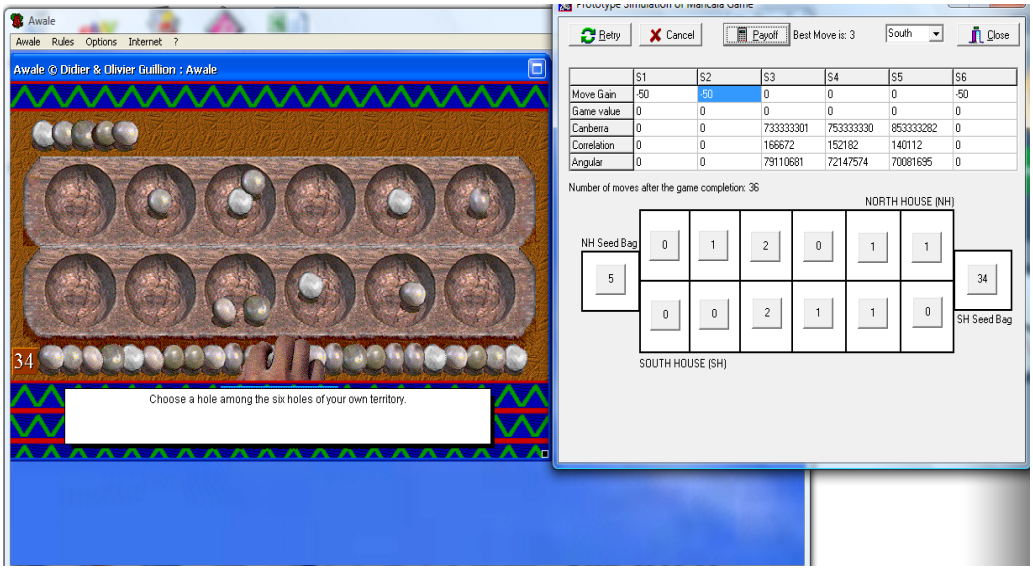

Figure 4.9: Screenshot Showing Complete Game Play With Total Seeds Capture and Number of Moves When Awale Starts First.

The statement $nP_i(s)$ means that a player *P* moves ***n*** seeds from pit **i** to capture **s** seeds. If a player does not capture any seed, the bracket term is dropped. All the moves for each game are arranged in a single sequence of statements. RBH won both games by capturing 28 seeds in Game 1 (RBH started first) against the 20 seeds captured by Awale in 51 moves, and 34 seeds in Game 2 (Awale started first against the 5 capture by Awale in 36 moves.

Game 1: RBH Started First

$4S_6, 4N_6, 4S_5, 6N_2, 6S_1, 7N_1, 1S_1, 8N_3(2), 2S_4(2), 3N_6, 2S_1, 1N_1, 9S_3, 2N_2, 8S_4(2), 11N_4,$
$3S_5(3), 1N_6(2), 1S_4, 4N_3, 1S_{3,\ 1}N_6(2), 1S_4, 11N_5, 1S_1, 2N_4, 1S_3, 1N_3, 5S_6(4), 2N_2, 2S_4,$
$2N_3, 1S_6, 2N_6, 1S_1, 1N_5, 17S_2(2), 9N_1(6), 6S_5(3), 3N_6, 3S_1, 2N_2, 1S_4, 4N_3, 1S_1, 7N_4(2),$
$5S_2(2), 2N_5(2), 2S_4, 3N_6, 2S_5, 1N_1, 5S_3(2), 1N_1, 1S_2, 1N_2, 1S_1, 1N_3, 1S_2, 1N_4, 2S_3, 1N_5,$
$2S_4, 1N_6, 3S_5, 1N_1, 1S_1\ 2N_2, 1S_2, 1N_4, 1S_3, 1N_5, 1S_4, 1N_3, 1S_5, 1N_4, 10S_6, 2N_6(4), 1S_4,$
$1N_2, 1S_3, 1N_1, 1S_4, 2N_3, 2S_5, 1N_2, 1S_6(2), 3N_5, 1S_1, 1N_3, 2S_2, 1N_6, 1S_1, 3N_4, 1S_1, 1N_5,$
$2S_2(6), 2N_6.$

Game 2: Awale Started First

$4N_2, 4S_1, 5N_5, 1S_1, 4N_1, 6S_4, 7N_4, 5S_6(3), 2N_1, 7S_3(2), 1N_1, 1S_1, 6N_2, 1S_1, 1N_5, 1S_6,$
$1N_4, 2S_4, 1N_5, 1S_6(2), 11N_3, 1S_6(2), 1N_5, 1S_1, 12N_6(2), 1S_6(2), 2N_2, 2S_4, 2N_3, 2S_3, 4N_4,$
$1S_1, 3N_5, 1S_4, 2N_6, 1S_6, 1N_1, 2S_1, 1N_2, 19S_2(9), 1N_5, 3S_3, 2N_6, 17S_5(10), 1N_6, 4S_4, 1N_2,$
$4S_1, 1N_3, 2S_5(2), 1N_4, 3S_2, 1N_5, 3S_3, 1N_6, 3S_4, 1N_1, 3S_5(2), 1N_1, 1S_1, 1N_2, 1S_2, 1N_3, 1S_3,$
$1N_4, 1S_4, 1N_5, 1S_5, 1N_6, 11S_6, 1N_6(3), 1S_2.$

Figure 4.10: Move-by-Move Game play result between Awale and RBH Simulation.

CHAPTER FIVE

PERFORMANCE EVALUATION OF THE HEURISTIC DECISION MAKING SYSTEM

This chapter reports the empirical evaluation of the Refinement-Based Heuristic Decision Making System. The evaluation of the RBH was undertaken so as to harvest the users' impression about its quality with a view to assessing the functionality, the effectiveness and efficiency of the game design interface either in use or in prototype, and providing suggestions for improvements.

5.1 EVALUATING THE USABILITY OF THE PROTOTYPE APPLICATION

Usability evaluation as a quality characteristic is defined by many authors and several ISO standards, e.g. ISO/IEC 9126 and ISO/IEC 9241-11, as a quality attribute that assesses how easy a product is to be used (Abran, *et al.*, 2003; Nielson and Mack, 1994; Rubin and Chisnell, 2008; Folmer and Bosch, 2004). According to Jitka *et al.*, (2011), usability is regarded as an ability of a system to fulfil all explicit (expressed) requirements and implicit user's needs in a given context of use. In a more detailed point of view, usability means that an application is useful, efficient, effective, learnable, accessible, and satisfying (Rubin and Chisnell, 2008). Usability evaluation is basically an attempt to measure the user's perception of a system after an interaction experience

as a means of assessing the human-computer interaction properties of the system (Daramola, (2009); Yngvi & Hitmal,(2009)).

Conducting an empirical usability evaluation involves making numerous decisions about the concrete design of the evaluation procedure, which would be optimal for the purposes and general context of the evaluation. In this work, the evaluation is in two folds. Firstly, a thorough evaluation of behaviour of the RBH (the evolved player) against Awale (an Expert shareware player) through a series of game play was provided (see section 5.2.1) and secondly, an empirical evidence about the importance and relative advantages of using the prototype simulation of the *Ayo* game for learning the game by any game player was provided (here, the judgment was centered on human expert players) on the following usability dimensions:

(1). Visual impression (i.e. screen display);

(2). Playability (i.e. Ease of use);

(3). Comfortability with the functionality of contents;

(4). Reliability of features to enhance learning of game playing;

(5). Users' satisfaction; and

(6). Response time for move suggestion.

5.1.1 Performance of the RBH versus Awale

In this subsection, a detailed study of the behaviour of the RBH (the evolved player) against Awale (an Expert shareware player) through a series of game played was carried out.

The setting of the game is as follows: the RBH was considered as the south player and Awale as the North player. The two players were made to start the game on two occasions at each level of Awale shareware as described in section 4.7. This implies four different game tournaments for each level. In comparing the performance of the RBH with Awale, it was observed that the RBH (evolved player) has a tendency to start well and was able to end well as a result of the endgame strategy incorporated into it. Similarly, the graphic user interface used for the design of the game makes it more user - friendly than Awale. In using the game design, unlike the Awale, the player may not have the knowledge of how to play the game but still be able to play it since the algorithm computes the game value and suggest a corresponding pit to move from to the player. Again, it was observed that in distributing the seeds in the process of play, Awale takes longer time to distribute the seeds, which was designed as the exact human-like process of play, than the RBP that makes an instant distribution since the algorithm adds up the number of seeds directed. The respective average response time for making move in playing the game is shown in Table 5.1 below. Again, since the experimentation was carried out on the same system, it was observed that Awale occupied higher memory space than the RBH algorithm as shown in Table 5.1. The graphical representation of the seeds captured by both players at different levels of play are shown in Figures 5.1 through to Figure 5.4.

Table 5.1: Results of Game Performance Evaluation for RBH and Awale

Average Response Time (per sec)		Size on Disk (MB)		Play Levels	Tournament			
							Seeds Captured	
RBP	Awale	RBP	Awale			Average Moves	RBP	Awale
1.84	2.11	2.12	3.33	Initiation	1	19	26	4
					2	18	28	11
					3	20	26	9
					4	23	30	14
					5	28	29	12
					6	17	26	17
2.21	2.63	2.12	3.33	Beginner	1	37	26	11
					2	38	25	10
					3	36	27	10
					4	39	26	16
					5	42	27	15
					6	33	26	17
2.73	2.79	2.12	3.33	Amateur	1	36	15	27
					2	41	21	18
					3	48	25	14
					4	38	20	23
					5	33	26	15
					6	41	27	19
2.95	4.13	2.12	3.33	Ground Master	1	36	14	28
					2	41	17	25
					3	41	25	15
					4	52	29	12
					5	47	27	16
					6	63	27	14

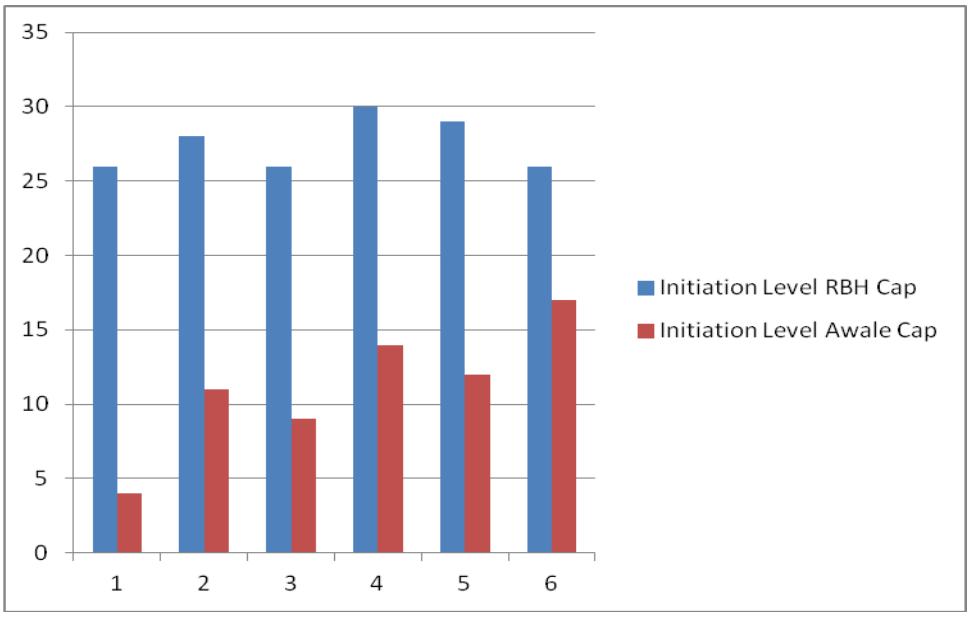

Figure 5.1: Graphical Representation of Seeds Captured Between RBH and Awale at Initiation Level

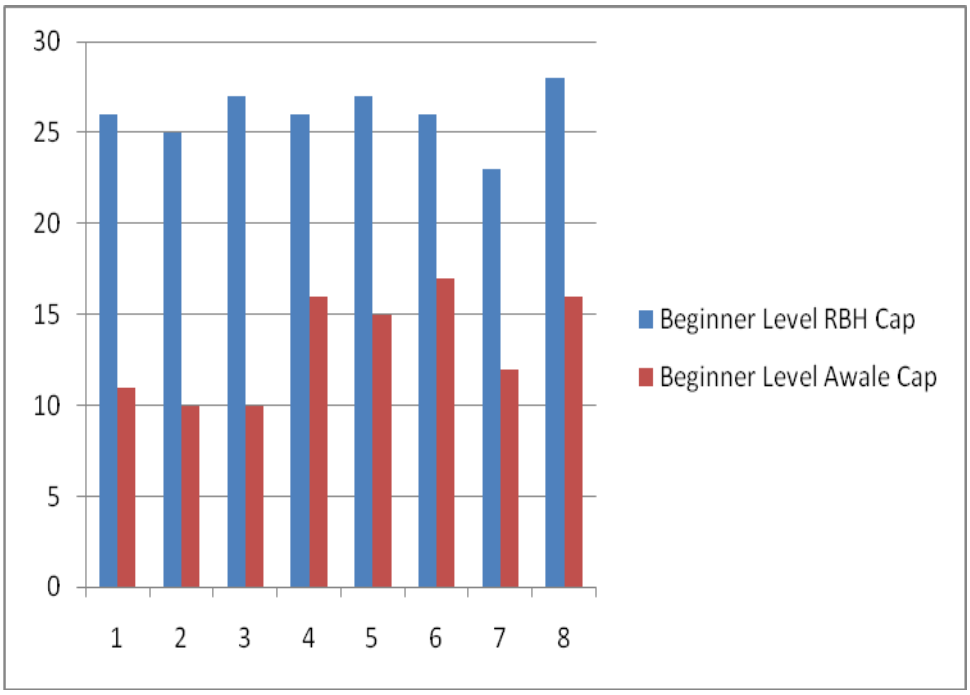

Figure 5.2: Graphical Representation of Seeds Captured Between RBH and Awale at Beginner Level

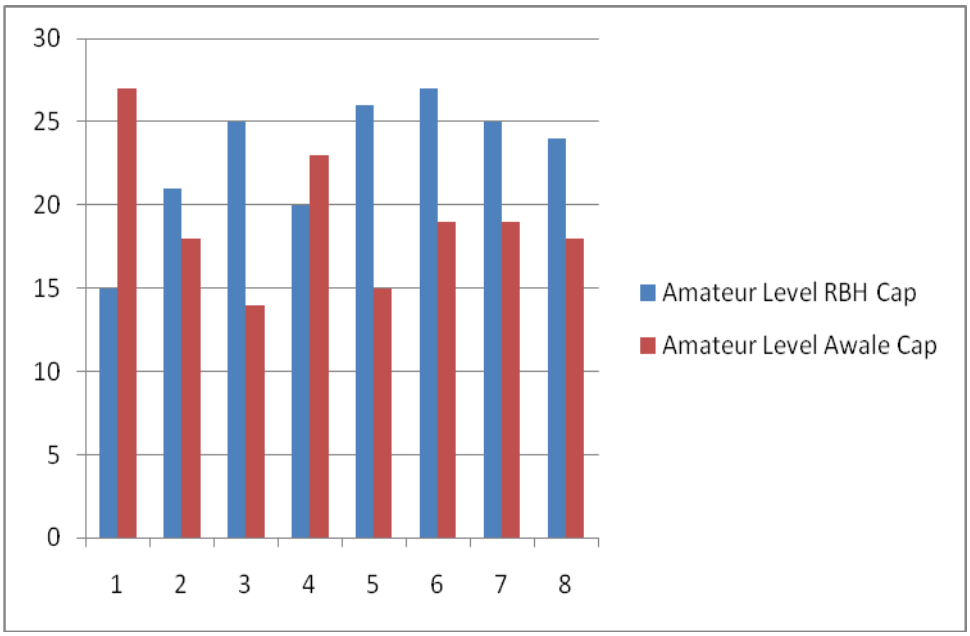

Figure 5.3: Graphical Representation of Seeds Captured Between RBH and Awale at Amateur Level

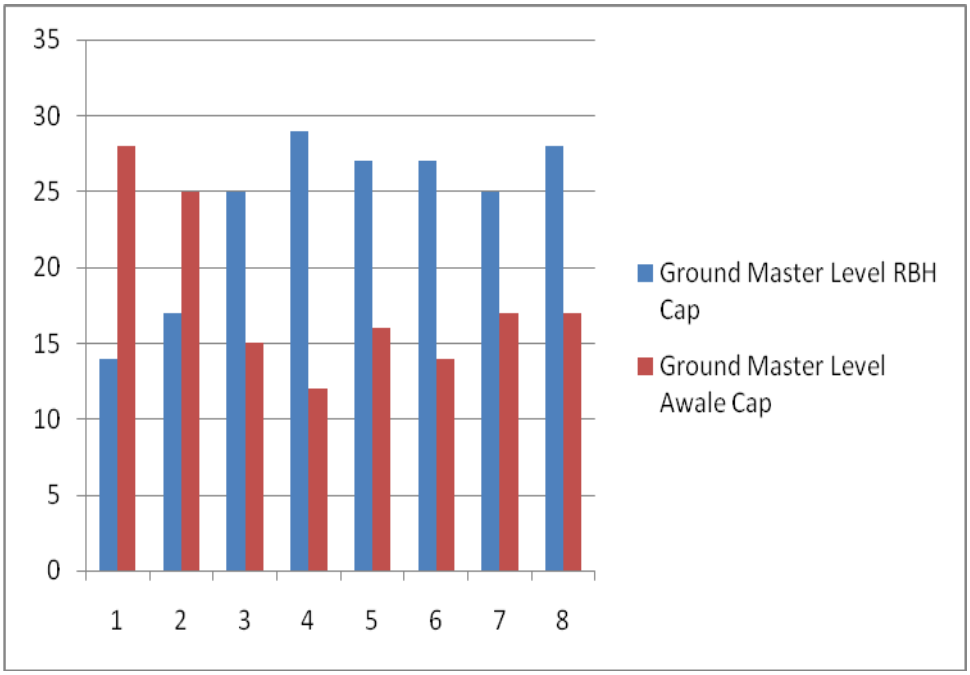

Figure 5.4: Graphical Representation of Seeds Captured Between RBH and Awale at Ground Master Level

5.1.2 Performance of the RBH versus Human Player

Again, the game setting is as stated in section 5.2.1 and the RBH was made to play with 10 human players (experts, novice and interested learners) twice, and all the players were made to start first on each game played so as to carry out a prototype usability testing in order to assess the performance of the RBH in terms of the accuracy of move suggestions, seed distributions, functionality and reliability of features, etc and hence obtain timely feedback from the players.

5.1.2.1 Experimental Design for Evaluation of the RBH

Trial experiments were undertaken by 50 human game players with 25 experts, 15 interested learners and 10 novices for the purpose of determining the extent to which the prototype application (the RBH) can be used by specified *Ayo* game players to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. All the players gave their consent to participate in the experiment and were taken through a brief tutorial (for about 10 minutes) on the usage of the application before the commencement of the experiment.

A paper-based questionnaire was designed and administered to the *Ayo* game players which were used for the assessments of players' perceptions of the prototype simulation based on each of the six (6) usability metrics stated in section 5.2 above. The questionnaire asked the players to indicate the degree of agreement with each item. The players interacted with the prototype simulation by performing a regular game playing on *Ayo* board. For the purpose of objectivity, the players in the game tournaments were made to use the

prototype simulation against the opponent to suggest the pit to move from thereby validating the correctness and fastness of move suggestions by the prototype simulation as opposed to human mental reasoning. A typical scenario of the exercise is shown in Figure 5.5 in their domain while Figure 5.6 shows the move suggested by the application as used by the players.

Figure 5.5: A Typical View of the Game Players that used the Application

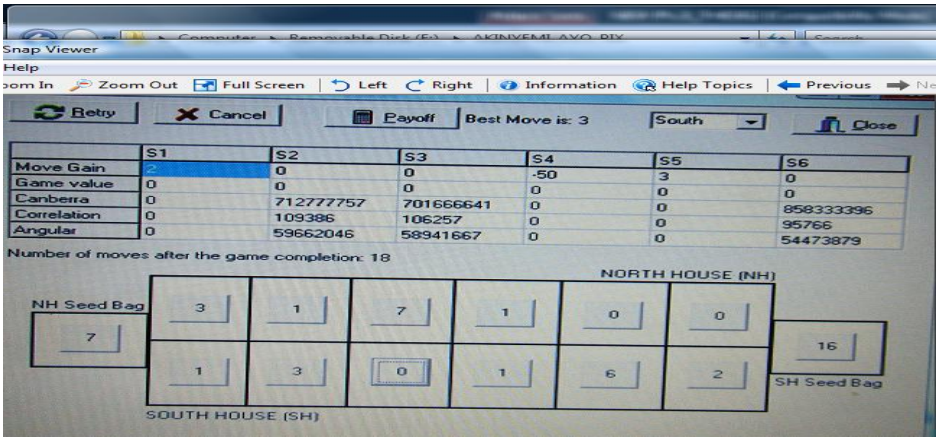

Figure 5.6: The Screen Display Showing Move Suggestion for the Players

Again, the administrator of the questionnaire only intervened when a player (especially those with little or no knowledge of computer usage) called for explanation on some of the functions of the menu items on the prototype application or could not follow the process to conclusion. The questionnaires were administered immediately after each game played to capture the players view about the *Ayo* game prototype simulation. All data were collated using a five point scale from "1", being "Strongly disagree" to "5" being "Strongly agreed" (see Appendix B for the sample questionnaire).

5.1.2.2 Data Analysis

The feedback obtained from game players through the questionnaire was analysed using Statistical Package for Social Sciences (SPSS 15.0 for Windows) to generate the frequency distribution, mean score, standard deviation, and variance for all the ratings for the prototype application based on the various usability metrics used for the evaluation of the prototype application. Table 5.2 shows the mean scores of the parameters used in the questionnaire for the evaluation. From the result, a mean score of above 4 in nine out of the 12 parameters considered from the questionnaire was obtained. Several usability studies have revealed that a system should have a mean score of 4 on a 1-5 scale and 5.6 on a 1-7 scale (Jeff and Erika, 2005). Since the adopted approach used a 1-5 scale, it is therefore sufficient to conclude that the prototype application developed for this thesis has a "Good Usability" as most users expressed satisfaction with the prototype application.

Table 5.2: Descriptive Statistical Analysis of Questionnaire Data

	Layout Fascinating	Feel Comfortable using the Prototype	Satisfy with Performance	Friendly Game Interface	Flexible interface Design	Work the way I want
Mean	4.31	4.08	4.24	4.02	4.10	4.20
Std. Deviation	0.675	1.281	1.367	0.679	0.926	0.341
Variance	0.456	1.724	1.868	0.421	0.854	0.727

	Can use the prototype without Instruction	Recover from mistakes quickly	Need more computing skills	Easy to learn	Fun to use	Suggestion wonderful and Accurate
Mean	2.42	4.58	3.44	4.17	3.55	4.17
Std. Deviation	1.495	0.579	1.388	0.947	1.313	1.147
Variance	2.234	0.318	1.926	0.896	1.725	1.726

Figure 5.7 presents a visualization of user satisfaction with the prototype application based on the ratings for the various metrics used for the evaluation in terms of percentage.

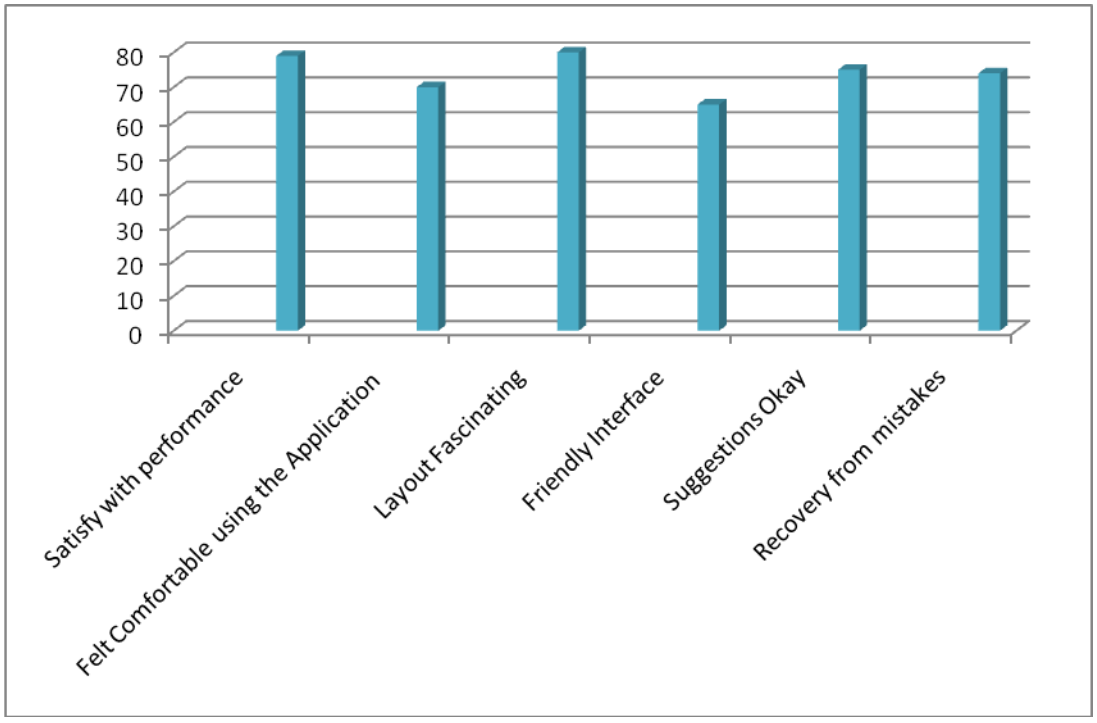

Figure 5.7: Summary of User Satisfaction with the Prototype Application

CHAPTER SIX

SUMMARY AND CONCLUSION

This chapter summarizes and discusses the contributions of the thesis, and highlights some areas for future research work.

6.1 SUMMARY

The thesis describes the Completely Determined Game (CDG) as an endgame winning strategy and presented the use of refinement-based heuristic method to evolve a machine-*Ayo* game player that can emulate human expertise in *Ayo* game playing in order to deepen the understanding of human intelligent processes through computer simulations.

We have been able to describe the combinatorial richness of the game of *Ayo* and hope that this work will attract the attention of a number theorists on further studies into the characteristics of the game and in particular the characterization of the CDG play as an endgame strategy which turned out to have solutions involving higher mathematics that generates integer sequence of the form 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7, 8, 8, 8, 9, 9, 9, 10, 10, 10, 10, 11, 11, 11, 11, 12, 12, 12, 12, and so on as conjectured.

Effort was made to successfully simulate the playing of *Ayo* game using simple heuristic approach (RBH) which is enhanced with an endgame strategy as encapsulated in the Completely Determined Game (CDG). The algorithm employed to evolve the *Ayo*

player is computationally effective and can improve AI performance and make computer players more adaptable and responsive. Similarly, it has the tendency to incorporate new play strategies in form of expert instruction and thus become sensitive to its mistakes/weaknesses.

Finally, a report of the procedure adopted for the evaluation of the RBH (Prototype application) process was presented, which showed that the prototype application had substantially favourable usability rating from users based on the empirical test conducted.

6.2 CONCLUSSION

This research work has succeeded in presenting a fascinating mathematical characterization of the CDG and provided some theoretical concepts on how an indigenous game like *Ayo* can be leveraged for the generation of integer sequence, and consequently obtain some self-replicating patterns that repeat themselves at different iterations.

The research has also provided a theoretical and product-oriented framework for refinement-based heuristic approach to evolve an *Ayo* player which turns out to be a novel means of solving the problem of decision making in move selections in computer game-playing of *Ayo* game.

In conclusively, it is hoped that if the prototype simulation could be extended with more detailed requirements, this will provide a platform for increased publicity and promotion of the local *Ayo* game as a veritable tool for economic development in the entertainment arena.

6.3 FUTURE WORK

The thesis provides a number of research opportunities in the immediate future. An open problem that is given in this work that calls for further research interest is, how do we derive an algorithm that can force the opponent into the CDG play at any instance of play? Again, the prototype application developed in this thesis lacks autonomy for performing game play like Awale. It is believed that the application could be enhanced with some Artificial Intelligence techniques like Case-Based Reasoning so as to bring up a complete game package with which people could use to learn how to play the game from which credible local *Ayo* game promotion initiatives can evolve in accordance with state-of-the-art practices in the global game playing domain.

REFERENCES

Abran, A., Khelifi, A., Suryn, W. and Seffah, A. (2003) "Usability Meanings and Interpretations in ISO Standards," *Software Quality Journal*, vol. 11, pp. 325-338

Adebiyi, E. F. (1999): A step Further Toward Improving *Ayo* Game System. *AMSE*, (2); pp 53- 69.

Adeda*yo*, O. A. (2005) Optimal Strategies in Two-Person Zero-Sum Games. A Seminar paper presented at Covenant University, Ota, Nigeria. May 25, 2005

Adewoye, T. O. and Awoniyi, S. A. (1985): Towards Obtaining Computer Programs for Playing the Africa Game of *Ayo*. In Proceedings of the International Conference on the Theory, Methods and Practice of Programming, Computer Association of Nigeria.

Adewoye T. O. (1990): On Certain Combinatorial Number Theoretic Aspects of the African Game of *Ayo*. *AMSE REVIEW*, vol. 14, no, 2, pp. 41-63, 1990

Ajayi, M. O. (2007) The Soccer Pitch and the Arena of Politics in Nigeria. A Paper Presented at the Public Lecture Delivered at Covenant University, Ota, Nigeria, on 27th February, 2007.

Ajith A. (2005). Artificial Neural Networks. Handbook of Measuring System Design, edited by Peter H. Sydenham and Richard Thorn. John Wiley & Sons, Ltd. ISBN: 0-470-02143-8.

Allis, L.V., Meulen, M. van der, and Herik, H.J. van den (1994). Proof-Number Search. *Artificial Intelligence*, Vol. 66, No. 1, pp. 91–123.

Anthony, K. (2004) Decision Making Using Game Theory: An Intr**odu**ction to Managers. Cambridge University Press.

Ayeni, J. O. A. and Longe, H. O. D. (1985): Game People Play: *Ayo*; *Int. Journal of Game Theory*, Vol 14 Iss. 4, pp. 207 – 218.

Azadivar, F. and Tompkins, G. (1999). *Simulation optimization with qualitative variables and structural model changes: A genetic algorithm approach*, European Journal of Operational Research, 113, pp 169-182.

Berliner, H.J. (1979). The B*-Tree Search Algorithm: A Best-First Proof Procedure. *Artificial Intelligence*, Vol. 12, No. 1, pp. 23–40.

Berne, E. (1964) Games People Play, Grove Press n., New York.

Bouchet,A. Owari I.Mtching Goups and Periodical Queues. Reprint 2005 http://www.rpi.edu/eglash/isges.dir/texts.dir/awari.pdf. Accesses11/09/2009.

Bowling, M., Furukranz, J., Graepel, T., and Musick, R. (2006) Machine Learning and Games. *Springer Science plus Business Media , LLC, Mach Learn* (2006),m vol. 63, pp 211 – 215.

Broline, D.M. and Loeb, D.E. (2006) The Combinatorics of Mancala-type games: *Ayo*, Tchoukaillon and 1/π. *UMAP*, J., 10(1), 1995.

Bruhn, H. (2005) Periodical States and Marching Groups in Closed Awari.

Bruin, A.D., Pijls, W. and Plaat, A. (1994) Solution Trees as a Basic for Game Tree Search, *ICCA Journal*, 17, 4, pp. 207-219.

Colman, A. M. (1982) Game Theory and Experimental Games: The Study of Strategic Interaction (Oxford, Pergamon Press).

Daramola, J. O. (2009). A Software Product Line Approach to Ontology-Based

Recommendations in E-Tourism System. Ph.D Thesis. Department of Computerand Information Sciences, Covenant Universit, Ota, Nigeria.

Davis, L. (1985). *Applying adaptive algorithms to epistatic domains.* In 9th International Joint Conference on AI, pp 162- 164, 1985.

Davis, J.E. and Kendall, G. (2002) An Investigation, using co-evolution, to evolve an Awari Player. In proceedings of Congress on Evolutionary Computation (CEC 2002), pp. 1408-1413.

Daoud, M., Kharma, N., Haidar, A. and Popoola, J. (2004) *Ayo*, the Awari Player, or How Better Representation Trumps Deeper Search, Proceedings of the 2004 IEEE Congress on Evolutionary Computation, pp. 1001-1006.

DeJong, K. (1975). *The Analysis and behaviour of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan, 1975

Didier, G. and Olivier,R. (1996) Awale Game. Available online – Subscribed and Downloaded on 12/07/2006.

Donkers, H.H.L.M., Uiterwijk, J.W.H.M. and Voogt, A.J.D.V. (2002) Mancala Games-Topics in Artificial Intelligence and Mathematics. Step by Step Proceedings of the 4th Colloquium Board Games in Academia.

Donkers, H.H.L.M. (2003) Searching with Opponents Models. Ph.D Dissertation, Universiteit Maastricht, Netherlands

Fausett, Laurene, (1994). Fundamentals of Neural Networks Architectures, Algorithms, and Applications, Prentice-Hall, Inc, 1994.

Ferguson, T. S. (1967) Mathematical Statistics – A Decision-Theoretic Approach, Academic Press, New York.

Folmer, E. and Bosch, J. (2004). "Architecting for usability: a survey," *Journal of Systems and Software*, vol. 70, pp. 61-78.

Fraenkel, A. S. (1996). "Combinatorial Games, Selected Bibliography with a Succinct Gourmet Intr**odu**ction", Games of no Chance, Vol. 29, MSRI Publications, Cambridge University Press (ed. R. Nowakowski), pp 253 – 537

Goldberg, D. and Lingle, R. (1985) *Alleles, loci and the traveling salesman problem.* In J. J. Grefenstette, editor, Proceedings of International Conference on GAs. Lawrence Erlbaum, 1985.

Grefenstette, J. J. and Fitzpatrick. J. M. (1985). *Genetic Search with Approximate Function Evaluations,* Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1985.

Hart, P. E., Nilsson, N. J. and Raphael, B. (1968). "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. SSC-4, no. 2, pp. 100–107.

Haykin, S. (1994). Neural Networks: A Comprehensive Foundation, NY: Macmillan.

Holland J. (1975). "Adaptation in Natural and Artificial Systems", Ann Arbor: The University of Michigan Press, 1975.

Horman, S. J. K. (1998) *Using Genetic Algorithms to Schedule The University of New South Wales Examination Timetable*, MSc Thesis, The University of New South Wales, 1998.

Jack, C. (2009). Applications of Artificial Intelligence and Machine Learning in Othello. TJHSST Computer Systems Lab. 2009-2010. Available online

http://www.tjhsst.edu/~rlatimer/techlab10/Per5/ThirdQuarter/ChenPosterQ3-10.pdf. Downloaded on 12/7/2010.

Jacques and Daniel (2001) Heuristic Search Methods to Combinatorial Programming Problems

Jeff Sauro and Erika Kindlund (2005), "A Method to Standardize Usability Metrics into a Single Score", *ACM, CHI,* April 2-7, Portland, Oregon, USA, 2005.

Jitka, K., Pavel, S., Martin, N., Alena, M., and Veronika, S. (2011) Methods of usability evaluation of web-based geographic information systems. *International journal of systems applications, engineering & development* Issue 1, Volume 5, 2011

Knuth, D.E. and Moore, R.W. (1975). An Analysis of Alpha-Beta Pruning. Artificial Intelligence, Vol. 6, No. 4, pp. 293–326.

Kosmas, K. and Donald, H. K.(1996). Genetic Algorithms and the Multiple Sequence Alignment Problem in Biology. Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, February, 1996, Baton Ronge, LA.

Kreeps, D. .M (1990) Game Theory and Economic Modeling, Oxford Unversity Press.

Kristof, V. L. (2004) Basic Statistics and Metrics for Sensor Analysis. Available online: http://ubicomp.lancs.ac.uk/~kristof/research/notes/basicstats/index.html. downloaded on 20/12/2005.

Longe, H. O. D. and Ayeni, J. O. A. (1991): *Ayo* Game, Some Practical Difficulties and Efficiency Considerations, *Journal of Computer Science and its Application*, a Publication of Computer Society of Nigeria, vol. 5, issue 7, 58-72.

Marsland, T.A. (1983). Relative Efficiency of Alpha-Beta Implementations. Proceedings

of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83), pp. 763–766, Karlsruhe, Germany.

Marsland, T.A. and Bj¨ornsson, Y. (2001). Variable-Depth Search. Advances in Computer Games 9 (eds. H.J. van den Herik and B. Monien), pp. 9–24. Universiteit Maastricht, Maastricht, The Netherlands.

McAllester, D.A. (1988). Conspiracy Numbers for Min-Max Search, *Artificial Intelligence 35*, 3 (1988), 287-310.

McCarthy, J. (1990) chess as the drosophila of AI. Computers, Chess, and Cognition (eds. T.A. Marsland and j. Schaeffer), pp. 227 – 237, Springer Verlag, Berling, Germany.

Murray, H.J.R. (1952) A History of Board Games other than Chess. Clarendon Press, Oxford.

Myerson, R. B. (1991): Game Theory – Analysis of Conflict, Harvard University Press.

Nash, J. (1951) Non-cooperative games, *Annals of Mathematics*, 54, pp. 286-295,

Nielsen, J., and Mack, R. L. (1994). *Usability Inspection Methods*. New York: John Wiley & Sons, 1994.

Njoku, C.N. (2004) Towards the Theory and Application of Jonda Game. *Journal of Computer Science and its Applications, vol.10 no. 1*.pp. 103 - 116

Odeleye, A. O. (1977) *Ayo*: A popular Yoruba game, Oxford University Press, Ibadan, Nigeria, 1977

Olugbara, O. O., Adewoye, T. O, and Akinyemi, I. O.(2006). An Investigation of Miinimax Search Techniques for Evolving *Ayo*/Awari Player. In Proceedings of IEEE- ICICT Conference, 10 – 12 December 2006

Plaat, A. (1996). Research Re: Search & Re-search. Ph.D. thesis, Tinbergen Institute and

Department of Computer Science, Erasmus University Rotterdam, Rotterdam, The Netherlands.

Reinefeld, A. (1983). An Improvement to the Scout Search Tree Algorithm. *ICCA Journal*, Vol. 6, No. 4, pp. 4–14.

Romein, J.W. and Bal, H.E., (2002) Notes Awari is Solved, *Journal of the ICGA*, vol. 25, pp. 162-165.

Rubin, J. and Chisnell, D. (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. 2nd edition. Indiana: Wiley Publishing, Inc., 2008.

Russell, S.J. and Norvig, P. (1995). Artificial Intelligence: A Modern Approach. Prentice- Hall, Inc., Upper Saddle River, NJ, USA.

Samuel, A.L. (1959) Some Studies in Machine Learning Using the Game of Checkers, *IBM J. of Res. And Dev*. 3, 210-229.

Samuel, A.L. (1967) Some Studies in Machine Learning Using the Game of Checkers, *IBM J. of Res. And Dev*. 3, 210-229.

Smith, J. M. (1982) Evolution and Theory of Games, Cambridge University Press.

Smith, M. S. (2003) Game Theory: Knowledge Base Easy. Downloaded on 23/06/2007. Available on-line http://www.beyondintractability.org/essay/prisoners_dilemma/

Shubik, M. (1982) Game Theory in the Social Sciences, The MIT Press.

Stockman, G.C. (1979). A Minimax Algorithm better than Alpha-Beta? Artificial Intelligence, Vol. 12, No. 2, pp. 179–196.

Taylor, A. D. (1995) Mathematics and politics – Strategy, Voting Power and Proof, Springer-Verlag, New York.

Tesauro, G. (1995) Yemporal Difference Learning and TD-gammon. *Communications of the ACM*, 38(3); 58 – 68.

Thomsen, T. (2000). Lambda-Search in Game Trees with Application to Go. ICGA Journal, Vol. 23, No. 4, pp. 203–217.

Tsuruoka, Y., Yokoyama, D., and Chikayama, T. (2002). Game-Tree Search Algorithm Based on Realization Probability. ICGA Journal, Vol. 25, No. 3, pp. 132–144.

Von Neumann, J. (1937) Ube rein Okonomisches Gleichungssytem und eine Veral-Igenemeimerung des Brouwerschen Fixpunktsatzes, in: *Menger, K. Ergebrusse eines Mathematischen Seminars* (Vienna).

Von Neumann, J. (1928). Zur Theorie der Gesellschaftsspiele. Mathematische Annalen, Vol. 100, No. 1, pp. 295–320.

Von Neumann, J. And Morgenstern, O. (1953) *Theory of Games and Economic Behaviours*. (Princeton, NJ, Princeton University Press).

Winands,M.H.M. and Bj¨ornsson, Y. (2008). Enhanced Realization Probability Search. New Mathematics and Natural Computation, Vol. 4, No. 3, pp. 329–342.

Yngvi Bj¨ornsson and Hilmar Finnsson (2009) CADIAPLAYER: A Simulation-Based General Game Player. *IEEE Transactions on Computational Intelligence and AI in Games*, VOL. 1, NO. 1, March 2009

APPENDIX A

SAMPLE GAME PLAY CONFIGURATION BETWEEN AWALE AND RBH

(THE PROTOTYPE SIMULATION)

Game 1

AWALE (North Player)

4	4	4	4	4	4
4	4	4	4	4	4

	4	5	5	5	5
5	5	5	5	4	

1	5	6	7		6
6	6	5	5		1

2	6	7	8	1	
1	8	6	6	1	2

3	7	8		1	
1	10	7	7	**2***	2

	7	8			1
2	11	8	7		

	7	8		1	
	12	9	7		

1	8	10	2		1
	12		8	1	1

1	10		4	2	3
1	13	1	1	3	3

	10		4		4
2*	13	1	1		4

1	11	1			4
1	13	1		1	4

	11	1			4
2*	13		1	1	4

RBH Starts First South Player)

4	4	4	4	4	4
4	4	4	4	4	4

4	4	5	5	5	5
4	4	4	4	4	

	4	5	6	6	6
5	5	5	5		1

1	5	6	7		7
	7	6	6	1	2

2	6	7	8	1	
	9	6	6	1	2

3	7	8		**2***	1
1	10	7	7		

	7	8			1
	12	9	7		

1	8	9	1	2	1
	12		8	1	1

2*	9	11	3	1	2
	12			2	2

1	10		4	**3***	4
1	13	1	1		4

	10		4		4
	13	1		1	4

1	11	1			4
1	13		1	1	4

1		2	1	1	5
1	14	1	1	3	5

	11	1			4
	13			2	4

2	1		1	1	5
	15	1	1	3	5

1		2	1	1	5
	15	1	1	3	5

2	1	1		1	5
	15		2	3	5

2	1		1	1	5
	15		2	3	5

2		1	2		6
	15		2	3	

2	**2***	**2***	1	2	6
	15		2	3	

2	1	2			6
	15			4	1

2		1	2		6
	15			4	1

	1	2			7
1	16			4	

2	1	2			7
	15			4	

1		2			7
	17			4	

	1	2			7
	17			4	

3	2	4	2	1	
2	1	**3***	**3***	6	2

2	1	3	1	**2***	9
1		2	2	6	2

		5	3	2	1
3	2	1			3

3	**3***	5	3	2	1
2	1				3

		6	4		1
	3	2	1		3

		5	3	2	1
	3	2	1		3

1	1	7			1
1	3	2		1	3

		6	4		1
	3	2		1	3

2	2				1
1	5	3	1	**2***	3

1	1	7			1
	4	2		1	3

3					
2*		4	2	1	4

2	2				**2***
1		4	2	1	4

1	1	5		2	5

		4		2	5

				1	
1	1	5			6

					1
1	1	5			6

				1	
1	1		1	1	7

				2*	1
1	1		1	1	7

			1		
1		1	1	1	7

				1	
1		1	1	1	7

		1			
	1	1	1	1	7

			1		
	1	1	1	1	7

	1				
		2	1	1	7

		1			
		2	1	1	7

1					
			2	2	7

	1				
			2	2	7

1				3	7

1					
				3	8

				2	
1					8

				1	1
1					9

		1	1		
	1				8

				2	
	1				9

	1		1		
		1			8

		1	1		
		1			9

1			1		
			1		8

	1		1		
			1		9

1		1			
				1	8

1			1		
				1	9

1	1				
					10

1		1			
					10

2	2	1	1	1	1
1	1	1	1		

	2	1	2		1
		1		1	

	2	1	1	1	1
		1		1	

	2	1	2	1	
			1	1	

	2	1	2		1
			1	1	

	3	2		1	
				2	

	2	1	2	1	
				2	

	3	2	1		1
					1

	3	2		1	1
					1

1		2	1		
1	1				

	3	2	1		**2***

1		3			
	2				

1		2	1		
	2				

		3			
1		1	1		

1		3			
		1	1		

1	1				
1	1	1	1		

		3			
	1	1	1		

2					
	2	1	1		

1	1				
	2	1	1		

1	1	2	2		

2					
		2	2		

1	1	2		1	1

GAME OVER

Game 2

AWALE Start First (North Player)

4	4	4	4	4	4
4	4	4	4	4	4

5	5	5	5		5
4	4	4	4	4	4

6		5	5		4
1	6	6	6	5	4

6	1	6	6	1	
	7	6	6	5	4

7	2		7	2	1
1	8	7	1	7	5

7		1	9	4	
1	8	7	1	7	

7			10	6	
1	8		2	8	1

8	1	1	11		
1	10		2	8	1

9		1	11		
	11		2	8	1

9	1		11		1
	11		2	8	

10			11		1
	11			9	1

11	1	1		1	1
1	12	1	1	10	1

12		1		1	
1	12	1	1	10	

RBH (South Player)

4	4	4	4	4	4
4	4	4	4	4	4

5	5	5	5		4
	5	5	5	5	4

6		5	5		4
	7	6	6	5	4

6	1	7	7	2	1
	7	6		6	5

7	**3***	1	8	3	2
1	8	7	1	7	

7		**2***	10	5	1
1	8		2	8	1

7			10	6	
	9		2	8	1

8	1	1	11		
	11		2	8	1

9		1	11		1
	11		2	8	

9	1		11		1
	11			9	1

10			11		**2***
	11			9	

11	1	1		1	**2***
1	12	1	1	10	

12		1		1	
	13	1	1	10	

	1	2	1	2	1
2*	14	2	2	11	1

	1	2	1	2	2*
	14	2	2	11	

	1	3	2		
	14	2	2	11	

	1	3	2		
	14	2		12	1

	2	4			
	14	2		12	1

	2	4			
	14		1	13	1

1	3				
1	15		1	13	1

1	3				
	16		1	13	1

2					
1	17		1	13	1

1					
1	17			14	1

2	18			14	1

					1
2	18			14	

				1	
2	18			14	

				1	
	19	1		14	

			1		
	19	1		14	

1	1	2*	3*	2*	2*
1		3	2	16	2

2					
1		3	2	16	2

2					
1			3	17	3

2	1		3	17	3

1	2*	2*	2*	2*	2*
3	2	1	4		5

4	2	1	4		5

				1	1
4	2	1		1	6

			1		1
4	2	1		1	6

			1		1
	3	2	1	2	6

		1			1
	3	2	1	2	6

		1			2*
	3	2	1		7

	1				
	3	2	1		7

	1				
		3	2	1	7

1					
		3	2	1	7

1					
			3	2	8

1			3	2	8

					1
1				3	9

				1	
1				3	9

				2*	1
1					10

				1	
1					10

				1	
	1				10

			1		
	1				10

			1		
		1			10

		1			
			1		10

	1				
			1		10

	1				
				1	10

1					
				1	10

1					
					11

1					11

1	1	1	1	1	1
2	1	1	1	1	

	1	1	1	1	1
3*	1	1	1	1	

	1				
		2	1	1	

Game Ended

APPENDIX B

SAMPLE QUESTIONNAIRE FOR THE USER INTERACTION

SATISFACTION WITH THE *AYO* PROTOTYPE SIMULATION

SCHOOL OF POSTGRADUATE STUDIES
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES, COVENANT UNIVERSITY OTA, OGUN STATE, NIGERIA

Questionnaire for the User Interaction Satisfaction with the *Ayo* Prototype Simulation

INTRODUCTION

The aim of this research is to obtain the views of human *Ayo* game players (experts and interested learners) with a view to measure user satisfaction and reliability of the prototype simulation of *Ayo* game.

For each question, kindly tick (√) the answer that appropriately expresses your view about using the *Ayo* game prototype design to learn how to play the game, and please answer the questions honestly and concisely as possible in the spaces provided.

Your responses shall be treated as confidential. Thank you very much**.**

SECTION A:
Demographic Data

1.) Gender: Male [ ] Female [ ]

2.) Your age range
[15 - 20] [21- 30] [31- 40] [41-50] [51 and above]

3.) What is your educational background?
PhD [ ], M.Sc. [ ] B.Sc. [ ], HND [ ] OND [ ] NCE [ ] WASCE [ ]

4.) Do you have prior knowledge/experience of using computer for playing games?

Yes [ ] No [ ]

5.) How would you rate your experience/skill in the use of computer?
[Novice] [Average] [Good] [Expert]

SECTION B:
Screen Display

1.) The visual design/layout for game play is quite fascinating.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

2.) I feel comfortable using the prototype.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

SECTION C:
Ease of Use

1.) I found the system easy to use and understand:

	1.	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

2.) Organization of information on the game interface is very clear:

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

3.) The interface is user friendly

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

4.) It requires the fewest steps possible to suggest accurate game move.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

5.) The interface design is flexible to use.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

6.) Using game prototype is effortless.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

7.) I can use game prototype without written instructions.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

8.) I did not notice any inconsistency as I used it.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

9.) Both occasional and regular users would like it.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

10.) I can recover from mistakes quickly and easily.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

11.) I can use it successfully every time.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

SECTION D:
Comfortability with the Functionality of Contents

1.) I am satisfied with the performance of the system in accomplishing my tasks

	1.	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

2.) I need more computing skills/training/time to be able to use the system.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

SECTION E:
Ease of Learning

1.) I learned to use it quickly.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

2.) I easily remember how to use it.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

3.) It is easy to learn to use it.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

4.) I quickly became skillful with it.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

SECTION F:
Users' Satisfaction

1.) I am satisfied with the game design.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

2.) I would recommend it to a friend to learn how to play the game.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

3.) It is fun to use for learning of the game.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

4.) It works the way I want it to work in respect of move suggestions.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

5.) Its move recommendations/suggestions are very wonderful.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

6.) It is pleasant to use.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

SECTION F:
Response Time

1.) The system was slow and sluggish in move suggestions.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

2.) The pace at which the system responded to seed distribution was fast and accurate.

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree