Machine as One Player in Indian Cowry Board Game:Strategies and Analysis of Randomness Model for Playing

Dr. P. Nagabhushan
Professor,Dept. of Studies in Computer Science
University of Mysore
Mysore, India
pnagabhushan@hotmail.com

Pouyan Davoudian
Research Scholar, Dept. of Studies in Computer Science
University of Mysore
Mysore, India
pouyan.davoudian@gmail.com

Abstract—Cowry game is an ancient board game originated in India. It is a game of chance and strategy with the objective of moving players' *pieces* through a specified path into a final location, according to the roll of special dice (cowry shells). This game involves decision-making under uncertainty and fuzziness with more than two parties; hence it can serve as an excellent example to apply methods and concepts for automating resource management and real-time strategic decisions. This research is aimed at evaluating the complexity of Cowry game and proposing heuristics and strategies that could be the basis of an adaptive artificial player to maximize its chances of winning the game. The main objective for considering machine as one player in Cowry game is to automate different strategies and to develop a machine player which is capable of real-time decision-making under interaction with live opponents. In this paper, we formulate several playing strategies and provide theoretical measures for comparison of these strategies. However, the main focus of this work is on analysis of playing randomly which involves no decision-making or intelligence. By applying this approach, we entirely concentrate on designing the game interface and validating the correctness of our implementation. Furthermore, the enhanced knowledge base resulting from analyzing the performance of the random strategy can be used for understanding the scenarios to be taken care of while evolving other types of strategies.

Keywords—*artificial intelligence; board game; cowry shell; game theory; machine learning; probability; race game; randomness; strategy.*

*****

I. Introduction

Researchers in the fields of artificial intelligence (AI) and machine learning have always been interested in the concept of how computers can learn or be programmed to do the tasks that humans do. As is cited in [17], tasks that are exceptionally easy for computers, like complicated mathematical computation and huge data storage, are very difficult for humans. However, tasks that humans do well by nature, such as carrying on a conversation, reasoning and problem solving, are much more difficult for a computer. One difficult task that AI researchers have been interested in for a very long time is "game playing". Furthermore, one of the earliest applications of AI to computer games was as opponents in simulated versions of common board games.

A "board game" [1] is a game that involves the movement of *tokens* or *pieces* round a pre-marked surface or board, in accordance with a set of rules. There are many categories of board games. "Race games" [2] are a classificationof board games with the objective of being the first player to move all one's pieces around a predefined track and into a final location. Race games often involve an element of chance, due to a randomizing device (such as dice) to determinehow far to move pieces.

Race games can be classified based on their ratio of luck to skill, as proposed by [2]. "Simple" race games such as *Snakes and Ladders* require no decisions by the players and are decided purely by luck. Each player is represented by only one piece and a single die is rolled to determine random movement of a player' piece, hence the outcome of the game is decided mainly by chance."Complex" race games are a combination of luck and skill. Players often have more than one piece to move, thus choices of move can put a player in advantageous positions. Complex race games generally retain an element of chance, but skill plays a greater role in determining the outcome. According to [2],most of modern complex race games such as*Ludo*, *Parcheesi* and *Trouble*primarily derive from India's *Pachisi* [3] and *Chaupur* [4].

A. *Indian Cowry Board Game*

Cowry game, also known as *ChowkaBhara* [5], is another race board game originating in India. It is one of the oldest board games extant, still being played in certain parts of India. There are references of this game in some ancient Indian epics like the *Mahabharata*. Like many such games there are regional variations for Cowry game and according to [5] it may also be called as Chowkabara, Chakaara, Chaukabara,Pagde, Pagdi, Katte Mane, AshtaChemma, Daayam, Thaayam, Kavidikali, etc.

It is a two to four player board game where playing pieces are moved according to the roll of special dice, "cowry shells" as shown in Fig. 1. A player wins by moving all of his pieces around the board and into a final location before his opponents. Obstacles to this objective includeshared paths with opponent pieces, unlucky die rolls and getting hit by opponent pieces.

Cowry game is a combination of strategy and chance (from throwing cowry shells). The cowry shells may decide the winner in a single game; however, over a series of many games, the stronger player will achieve a better record. Therefore, according to [6], records of games between players can be considered as a good measure of "skill" which refers to intelligence and real-time strategic decision-making. With each throw of cowry shells, players have to choose from different alternatives for moving their pieces and also predict probable counter-moves by the opponents.

Figure 1. Cowry Shells

B. *Overview*

In this work, we undertake a fundamental study of Cowry game in order to gain a better understanding of the game and its complexity. The approach introduced in [24] relies on the idea that by analyzing the complexity of Cowry game we can show that it is more complex than *Chess* due to the presence of an element of chance, and is comparable to the complexity of *Backgammon* [6] or *Ludo* [7], which probably indicates that Cowry game has some strategic variety and is not a trivial game.

Accordingly, we propose several types of strategies for Cowry game and briefly describe their features. Furthermore, some theoretical measures to analyze and compare the performance of different types of strategies are introduced. However, in this paper we merely explore the most basic type of strategy which is playing randomly. A "random strategy" is no strategy at all, since players move their pieces completely at random and hence requires no decisions by the players. The absence of decision-making for the players could give us the opportunity to concentrate entirely on the design and implementation of the game and to validate the correctness of our implementation and also to ensure that all of the game rules are applied properly.

We give a theoretical analysis of the random strategy based on "expected number of moves" and experimental results are also included to verify the theoretical results. The authors of [24] point out that the enhanced knowledge base resulting from analysis of the random strategy can be used to generate other types of strategies and may also serve as a baseline comparison in our future work. In this paper, we give a brief explanation about implementation of the basic version of Cowry game which has a text-based user interface, allowing users to interact with the program through a command line. Design and implementation of the graphical user interface (GUI) for the game is not a part of this paper and is considered as a future work.

The rest of the paper is organized as follows: In section 2, we briefly discuss the basics of game theory required for better understanding of this work, followed by the motivations for choosing Cowry game as well as some of its real-world applications. Furthermore, several related scientific supporting areas are classified. Section 3 gives a description of Cowry game and a brief summary of its rules is presented. In section 4, we propose several types of strategies in Cowry game, and in section 5, we provide a theoretical analysis of the random strategy.We discuss the implementation of the game and experimentations on the random strategy in section 6. Section 7 describes future research directions and concludes the paper.

II. BACKGROUND

In this section we formally define a few important game theory terms required for understanding and analysis of Cowry game. The motivations and some real-world applications of the game as well as a brief classification of relevant fields of research are presented.

A. *Game Theory Terminology*

"Game theory"[8] is a mathematical discipline concerned with the analysis of strategies for dealing with competitive games where the outcome of a player's choice depends on the actions of other players. It may have a very weak application to real-time computer games, but most of the terminology used in "turn-based strategy" games [9] is derived from it.

This section will introduce some important terms in game theory and allow us to understand a turn-based AI. In the textbook [11], the authors explore that game theory classifies games according to the number of players, the kinds of goal those players have, and the information each player has about the game.

- Number of Players

Almost all the board games that inspired turn-based AI algorithms have only two players, as proposed by [11]. Therefore, most of the well-known algorithms in this field only consider two players in their most basic form. It might be possible to modify some of these algorithms to work with more number of players, but in general, there is no routine algorithm that has been used for "multi-player" games. In a competitive game with two players, the strong player will usually win. However, the introduction of a third player doesn't guarantee that the strongest player will win.

- The Goal of the Game

The ultimate goal that a player aims to achieve in most strategy games is to win. In a "zero-sum" game a player wins if all his opponents lose. As is widely cited in [11], in a zero-sum game it doesn't matter if a player tries to win or if he tries to make his opponents lose; the outcome is the same. However, in a "non-zero-sum" game players could all win or all lose, thus a player would want to focus on his own winning, rather than his opponents losing.

- Information

The authors of [11] point out that in games like *Chess* both players have all the information about the state of the game. They know the result of every move and they can predict the possible options for the next moves. This type of game is called "perfect information". Although a player cannot be sure which move his opponent will choose to make, he has complete knowledge of every move his opponent could possibly make and the effects it would have.

Games like*Backgammon*, *Ludo* or Cowry game contain an element of chance such as dice. A player has no information in advance of his dice roll, hence cannot predict what moves he will be allowed to make. Similarly, he has no information what moves his opponent can play, because he cannot predict his opponent's dice roll. This kind of game is called "imperfect information". Finding the best move in such games is more difficult and may involve estimating probabilities by the opponents.

- Applying Algorithms

The most popular and advanced algorithms for turn-based games, as is widely cited in [11], are designed to work with two-player, zero-sum, perfect information games. But many turn-based computer games such as Cowry game are more complex, involving more players and imperfect information. AI researchers believe that it is possible to consider some of these algorithms in their most common form: for two-player, perfect information games, and then just fine-tune certain aspects of them to be adapted for the desired type of game.

- Strategy

According to game theory, "strategy" [10] refers to any of the alternatives a player can choose in a situation where the outcome depends not only on his own actions but also on the actions of other players.

However, as [10] point out, the concept of strategy should not be confused with that of a "move". A move is defined as an action taken by a player at some point during the play of a game (e.g. in Cowry game, moving a player's piece to a certain square). On the other hand, a player's strategy is a complete algorithm for playing the game, determining the action the player will take at any stage throughout the game.

In Cowry game, like most other board games, a player cannot always win by following only a single strategy throughout the whole game. The strategy that is used in the beginning of a game should be different from the strategy used in the end of the game. Different strategies can be applied based on the situation of the board, the progress of the player and the opponents, number of pieces in the game, etc.

B. *Motivations and Applications*

As mentioned earlier, an important application of AI in the gaming industry is board games. Some games such as *Chess*, *Checkers* or *Backgammon* have been solved to an extent, where an AI can win against expert human players. However, many imperfect information games with multiple players, such as Cowry game, have seen less development in terms of strong AI yet.While a great deal of research has gone into perfect information two-player board games, many games and most real-world problems involve decision-making under uncertainty or fuzziness with more than two parties.

Imperfect information games such as Cowry game can have practical importance, including most negotiations, auctions, and many applications in information security and physical battles. At first sight, Cowry game might not seem suitable as an example for real-world decisions, such as business or military decisions. However, in the thesis [26], the author demonstrates that the decisions a player has to make in games such as Cowry game are very similar to the decisions a manager had to make in real-world,

- Both are acting in a hostile, uncertain and fuzzy environment, where opponents want to get the best of them.
- Both need an abstraction of reality to make a decision, because enumerating all possible outcomes is impractical.

There are many situations that arise in Cowry game which can be mapped to real-life scenarios. For instance, there are many day-to-day situations in which a person wishes to achieve a goal, however due to lack of proper resources he may fail. This is the same constraint a player in Cowry game often faces due to unlucky die rolls or threats from opponents.

Similarly, there are real-life situations in which a person may get some unexpected extra resources, however he may not be able to keep these resources and opportunities for a long time and needs to utilize them within a short time limit. The same scenario can be mapped to a typical Cowry game when a player gets lucky die rolls or eliminates an opponent piece which can result in additional bonus turn to play, but he is not permitted to save this bonus turn for future and has to play it immediately in his turn.

Therefore, we may consider Cowry game as an excellent example to apply methods and concepts for automatingtimely resource management and real-world strategic decisions.

C. *Relevant Fields of Research*

Despite our extensive search in the literature, we could not find any scientific paper to tackle the problem of Cowry game. However, we could identify the following supporting areas and we believe that studying and analyzing some recent papers in these supporting areas could give us a solid foundation for solving the problem of Cowry game. Thus, we have classified the related work into following categories,

- Perfect Information and Imperfect Information Games [12], [13], [14], [15]
- Two-player and Multi-player Games [16], [17], [18]
- Monte Carlo Methods [19], [20], [21], [22]
- Strategies and Heuristics in AI Games [23], [24], [25]
- Fuzzy Decision-making under Risk and Uncertainty [26], [27], [28]

Ludo board game [7], a derivative of *Pachisi* [3], has been the topic of several research papers, one of which is particularly interesting for this work. In their research paper, Alvi& Ahmed [24] conducted a fundamental study on complexity analysis and playing strategies for *Ludo* and its variant race games. The article evaluated the state-space complexity of *Ludo*, and formulated and analyzed strategies based on some basic moves. The authors also provided an experimental comparison of pure and mixed versions of these strategies. We hereby acknowledge that some of the ideas, methods and analysis approaches used in our work have been adopted from the mentioned research paper [24], and have been modified to be applied for the analysis of Cowry game.

III. GAME DESCRIPTION AND RULES

Cowry game has several intricate rules which need to be followed. Although there are several variations of this game, the rules presented in this section are considered for our standard implementation.

Cowry game normally has a 5×5 square board and four players, but it is also possible to increase the number of squares to any odd number squared (for example, 7×7 or 9×9). Assuming the size of the board is $N \times N$ (with N being odd), then each player will have $N-1$ pieces. In this work we have considered and implemented the 5×5 version of the game which is shown in Fig. 2.

The outer middle squares on each side of the board are the starting squares for each player, and also function as "safe" squares.Each player starts on his own starting square and moves around the board in an anticlockwise direction in outer squares and a clockwise direction in inner squares. The path of each player is different because each player starts on a different square and moves into the inner squares at a different position on the board. The path for one of the players (South) is shown in Fig. 2 with dotted lines.

Each player is represented by one of the colors, say, *Red* (South), *Green* (East), *Blue* (North) and *Yellow* (West), and has four pieces. Each player takes a turn to play, and turns rotate according to the cyclic order of players. Four cowry shells are used as dice (see Fig. 1). They are thrown and the number of shells that lie with their openings upwards indicates the number of squares a player should move.

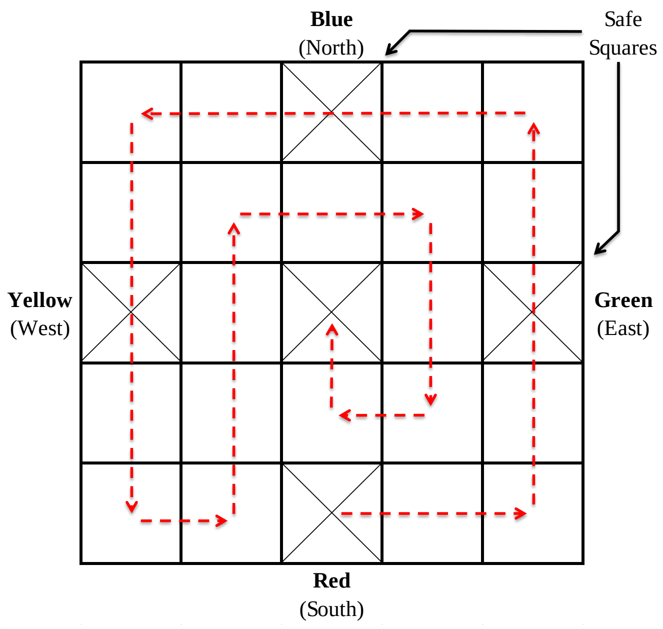

Figure 2. Board for Cowry Game with Safe Squares

As demonstrated in [5], the mouth of the shell landing upwards is considered to be of value 1 and downwards is of value 0. However, if every shell shows a value of 0, then the value is considered to be 8. Therefore, the possible values are 1, 2, 3, 4 and 8. Getting 4 or 8 gives the player an additional turn, which can continue until that player gets a number other than 4 or 8 (namely 1, 2, or 3).

For example, if a player throws a 4, he will get another chance to throw the cowry shells. If on the second turn he throws a 3, then the player can move one of his pieces 4 squares forward and one of his other pieces 3 squares forward. He can of course choose to move the same piece 7 squares forward in any order (either 4+3 or 3+4). This argument can be extended to the player getting 3 or more consecutive turns.

Pieces of two different players cannot exist in the same square, other than a “safe” square, which are marked with an X in Fig. 2. For a 5×5 board this is simply the starting squares of each of the players and the central square of the board. However, for higher dimension boards, more safe squares can be added symmetrically across the board.

So if a piece of *Red* player lands on the same square of a piece of *Blue* player, then *Red* player has “hit” *Blue* player. *Blue* player’s piece is returned to its starting square and this piece needs to start over. When a player hits an opponent’s piece, he will get an additional turn to throw the cowry shells.

For a player’s piece to progress into the inner squares, he should have hit at least one of his opponent’s pieces. When one of the player’s pieces has hit one of the opponent’s, all his other pieces will be eligible to enter the inner squares too. In case a player cannot move any of his pieces because he has not hit any of his opponent’s, the player will lose that turn. However, it should be mentioned that forfeiting the turn voluntarily is not allowed in Cowry game (unlike *Pachisi* [3]).

The goal (central square) can only be reached by a direct throw. If a player throws a number larger than that needed to reach the goal, he must move another piece or wait till his next turn. For example, if a piece is 3 squares away from the goal and the player throws a 4, then that piece cannot be moved. If that is the only piece left for the player to move, the player will lose his turn.

It is possible for a player to have two of his pieces in the same square, as is described in [5]. This is called a “double”. There are some specific rules regarding doubles. For instance, it is not possible for a single piece to hit a double, or a player cannot move past an opponent’s double for one move. However, in this paper for the sake of simplicity we have not considered the concept of “doubles” for implementation and analysis of the basic version of the game.

IV. STRATEGIES

As with any game, an interesting question from the point of view of formal analysis is to determine strategies for playing that are likely to win the game. In Cowry game, after a player has thrown the cowry shells in his move, he has four options to move his pieces. Therefore, the game-tree branching factor for Cowry game is 20 corresponding to 4 pieces × 5 possible cowry shell values for each player. In this section we introduce several strategies that a player may choose during the game.

It may be relevant to mention here that the types of moves and strategies applied in most race board games are almost similar. As an example, Alvi& Ahmed [24] give pretty straightforward definitions of different moves and strategies in their research work on analysis of *Ludo* board game, which are also applicable for analysis of Cowry board game. We hereby acknowledge that some of the following strategies are partially adopted from their research work [24].

A. *Random*

In a “random strategy”, a player chooses to play his pieces completely at random during the entire game. A player may try random moves in situations that he cannot see any advantage in moving a particular piece. As proposed by [24], although playing randomly is the most disadvantageous strategy and has little usefulness for winning games, it can be used as a benchmark to compare performance of other strategies.

Since the random strategy has absolutely no pattern for playing, it would be impossible to anticipate the next move of a player who is playing randomly. Therefore, this strategy may sometimes be used by expert players in certain stages of the game to confuse their opponents. We believe that the same approach can be employed while designing advanced AI players to confuse their opponents, especially when other AI players are trying to use machine learning approaches to detect their strategy.

B. *Aggressive*

An “aggressive strategy” is based on a preference for moving a piece which can attack and hit the piece of another player whenever possible throughout the game. However, an aggressive move is not just limited to elimination of an opponent piece, but also to chase opponent pieces in an attempt tocause threat and panic for other opponents. The authors of [24] point out that an aggressive move, if leads to hit a piece, gives a definite advantage to the aggressive player over the attacked player in the sense that the eliminated piece has to be returned to its starting square and start over. The aggressive player will also get an additional turn to throw the cowry shells.

C. *Defensive*

In a “defensive strategy”, a player shows a strong tendency to move his pieces into safe squares and keep them away from the danger of an impending attack or elimination during the entire game. However, the idea of being in danger of elimination needs elaboration, as is cited in [24].

We can propose that in Cowry game, based on the probability of occurrence for different cowry values (see Table I)obtainedempirically in the thesis [33], a piece is in danger of being eliminated when it is 1, 2 or 3 squares away from an opponent piece chasing it. This distance can be called as a "danger zone". Therefore, in a defensive strategy, a player always moves his piece if it is within the danger zone of an opponent piece, and prefers to move it into a safe square whenever possible.

The strength of a defensive strategy, if successful, is in minimizing the number of times the player pieces are being eliminated and sent back, which gives an advantage to the player against all opponents in terms of the minimum number of moves required to finish the game.

D. Move-first

A"move-first strategy" gives preference to move the foremost piece. In other words, a player always chooses to play a piece which has moved the maximum number of squares in its path around the board. This strategy is based on the idea that the loss of a piece that has advanced the most in the game (i.e. the first piece) would be the most expensive for a player in terms of the additional number of moves required, hence it must be moved first and sent to the final location.

The work [24] has been stated that a move-first strategy simply alters the arrangement of a player's pieces throughout the game and therefore may not offer any significant advantage over other players.However, the fact that a player always moves the most advanced piececan result in moving only one piece at a time and preferably keeping other pieces in safe squares, which in turns can reduce the risk of piece elimination.

E. Move-last

In a "move-last strategy" high priority is given to move the hindmost piece, that is, a player prefers to play the piece which has moved the minimum number of squares from its starting square. This strategy can result in a "balanced" movement of all pieces, keeping all of a player's pieces close to each other and moving them together as a colony.

A move-last strategy may give an advantage to a player in the sense that a cautious opponent prefers not to chase or hit a piece which is moving in a colony, because such an attempt could put his own piece in danger of elimination by other members of the colony.

F. Mixed

As mentioned earlier, in most strategic board games, a player cannot always win by following a single strategy throughout the entire game. The approach introduced in [24] relies on the idea that it could be advantageous if a player chooses to play different types of moves at different stages in the game. Certain types of moves might be preferable according to the situation of the board, the progress of the player and the opponents, number of pieces in the game, etc. For example, a player may play any combination of *defensive*, *aggressive*, *move-first*, *move-last* and *random* moves, which could result in a "mixed strategy".

TABLE I. PROBABILITY OF OCCURRENCE OF COWRY VALUES

Cowry Values	1	2	3	4	8
Probability of Occurrence	24.3 %	38.1 %	23.6 %	7.4 %	6.6 %

V. ANALYSIS

In this section we introduce a theoretical approach for analysis and comparison of different strategies. The work [24] has opened up the issue that by using "expected values" of cowry shells and "average piece movement" over a large number of games, we can have a measure to analyze the performance of different strategies.

A. Assumptions

In Cowry game we consider all pieces of a player to be identical, since unlike *Chess*, there is no difference between pieces of the same player. Therefore, a board setting in which a player's pieces are on locations (4, 1, 3, 7) is identical to another setting in which same player's pieces are on locations (7, 3, 4, 1).

For the purpose of analysis, we assume that each player plays a single throw of cowry shells during his turn in the game. Hence, a "move" is defined as the movement of a player's piece resulting from a single throw of cowry shells. However, in an actual Cowry game, a player will get an additional turn to throw the cowry shells if he gets 4 or 8 or if he hits an opponent's piece. These additional turns are not considered in the following theoretical analysis.

In an actual Cowry game with four players, after one player wins the game, other players will continue playing to identify the second and third winners as well. In such situations, the remaining players often need to change their strategies in order to become second or third winners, because removing the first winner with four pieces from the game could indicate less threats, more safe space, and easier decisions for other players. However, since the aim of this paper is to analyze the performance of the random strategy and all four AI players are using the same strategy, it would be redundant to continue the game after the first winner is identified. Hence, we stop the game as soon as one player wins and we perform our statistical analysis based on the movements of the same winner.

Due to non-symmetrical shape of cowry shells (see Fig. 1), the probability of occurrence for different cowry values (1, 2, 3, 4 or 8) is not equal. Therefore,in the thesis [33], we tried to estimate the empirical probability of occurrence for each value. The result for this experimental observation is given in Table I. It is clear that the obtained probability distribution is not uniform. A more detailed explanation on non-uniform random number generation appears in the Appendix.

B. Types of Randomness

As discussed earlier, the main theme of our current paper is to analyze the performance of randomness as a model for playing. It is worth mentioning that we could identify three types of randomness in the game,

1) Randomness due to throw of cowry shells:
Although the probability of occurrence for each cowry value has been obtained empirically, a player cannot have a certain prediction about the outcome in advance. Hence, we have absolutely no control over this type of natural randomness and it acts as an unpredictable constraint for all the players.

2) Randomness in selection of pieces by opponents:
The fact that an intelligent opponent may choose different strategies during different stages of the game, suggests that it is very difficult to predict which one of the pieces may be selected by the opponent to play the next move.

Therefore, the strategy that the opponent is following is unknown from AI point of view, and can be considered as another type of randomness. However, it might be possible for an advanced AI player to be trained using machine learning techniques, in order to find some patterns to recognize the opponent's strategy and accordingly to predict his probable next moves. Dealing with this concept is beyond the scope of this paper and can be considered as a future work.

3) Randomness in selection of player's own pieces: The idea behind random playing is to select one of the player's pieces completely at random, without any reasoning or following any pattern. In this paper, we are merely dealing with this type of randomness and all theoretical and experimental analysis performed in this work is based on random selection of pieces for all AI players.

C. *Expected Number of Moves*

By definition presented in [29], the "expected value" of a random variable is the weighted mean of all its possible values. In other words, each possible value of the random variable is multiplied by its probability of occurring, and the summation of all the resulting products generates the expected value. Assumethat random variable X can take value x_1 with probability p_1, value x_2 with probability p_2, and so on, up to value x_k with probability p_k. Then the expected value of a random variable X is given by,

$$\mathrm{E}[X] = x_1p_1 + x_2p_2 + \dots + x_kp_k$$

If the probability of occurring for all outcomes x_i are equal (i.e. $p_1 = p_2 = ... = p_k$), then the expected value is the simple average. However, according to [29], if all outcomesare not equally probable and some outcomes $x_i$are more likely than the others, then the expected value turns into the weighted average, as is the case in throwing four cowry shells shown in Table I.

Suppose X represents the outcome of a throw of four cowry shells. The possible values for X are 1, 2, 3, 4 and 8 with the probabilities 0.243, 0.381, 0.236, 0.074 and 0.066 respectively. The expectation of X is given by,

$$\mathrm{E}[X] = 1(0.243) + 2(0.381) + 3(0.236) + 4(0.074) + 8(0.066)$$

$$\therefore \mathrm{E}[cowry] = 2.537$$

So the expected value for a single throw of four cowry shells is 2.537. To complete an entire path around the board, a piece will require the following expected minimum number of moves,

$$\text{Total number of squares} \div \mathrm{E}[cowry] = 24 \div 2.537 \approx 9.46$$

Similarly, a player with 4 pieces will take an expected minimum number of $4 \times 9.46 \approx 37.84$ moves to finish the game. Therefore, the expected minimum ply-length of a game with 4 players is $4 \times 37.84 \approx 151.36$ moves.

Therefore, a trivial lower-bound can be established by considering the expected minimum number of moves for a player. This expected minimum value is based on the assumption that the piece doesn't get hit by another piece.

However in an actual Cowry game, a piece may get hit and sent back to its starting square several times, hence the practical average number of moves for a player to complete the game could be much higher.

While the total number of games won by a player in the long run is considered to be the initial measure of performance, it can be observed that the expected number of moves could also serve as a good measure to analyze the performance of a player or to compare the performance of different strategies. In general, a smaller expected number of moves to finish the game can be a definitive indicator of a better performance.

Let us assume that a player will require n expected moves to complete all his pieces' path around the board. Clearly, the player with the minimum value of n would be the expected winner. Hence, as proposed by [24], to minimize n, a player needs to follow one of the these approaches whenever possible: (a) minimize the number of times that his own pieces get hit and sent back, which can be achieved by following a defensive strategy, or (b) maximize the number of times that he attacks and hits other players, by following an aggressive strategy.

Following other strategies could also have some advantages which need elaboration. However, in this research work we have only implemented and analyzed the most basic type of strategy, that is, the random strategy. Implementation, analysis and comparison of other types of strategies introduced in this paper will be explored in our future work.

As the Cowry game always needs to have a winner, in a game with four AI players all of them playing randomly, one of them will definitely win the game. This clearly indicates that even in an actual Cowry game, a player playing randomly has a chance, no matter how small, to win the game. Intuitively, it can be argued that the expected number of moves in playing randomly should be much higher than any other type of strategy; hence having any basic strategy is better than playing randomly.

However, we believe that by analyzing random moves of AI players we can identify some obvious mistakes that players commit during the game-play. The enhanced knowledge base resulting from analyzing these mistakes can then be used for evolving better strategies and improving their performance.

VI. IMPLEMENTATION AND EXPERIMENTATION

In this section we briefly describe the implementation and the process of creation and interactions between required data structures for the basic version of the game. Furthermore, we state several phases of experiments conducted on the random strategy as well as the obtained results.

A. *Implementation*

The squares on the Cowry game board can be classified into two types: safe squares and non-safe squares (see Fig. 2). A safe square is a square that can accommodate multiple pieces of different players simultaneously without being hit or sent back. On the other hand, only one piece can be placed on a non-safe square at a time.

The approach introduced in [24] proposes that for the purpose of implementation, instead of defining several different safe squares, only one imaginary single safe square can be defined for all four starting squares of players as well as the goal (central square).

	Outer Squares																Inner Squares								Goal
Positions on Board	42	43	44	34	24	14	04	03	02	01	00	10	20	30	40	41	31	21	11	12	13	23	33	32	22
Red (South) Path	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Positions on Board	02	01	00	10	20	30	40	41	42	43	44	34	24	14	04	03	13	23	33	32	31	21	11	12	22
Blue (North) Path	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Figure 3. Lookup Tables to Store the Paths for *Red* and *Blue* Players

The game is set up by designing data structures representing entities involved in a typical Cowry game. A 5×5 two-dimensional array is applied for representing the game board, as shown in Fig. 4.As mentioned earlier, each player has a different path around the board because they have different starting squares and they move into inner squares at different positions. Hence, we need to store the sequence of squares (paths) for each player as well as a method to map them into the squares on a common board.

As an example, the sequence of squares for *Red* (South) player is shown on the board in Fig. 5. To get the path for*Green* (East) player, we just need to rotate the board by 90 degrees in anticlockwise direction. However, based on our analysis, the idea of rotating the board and obtaining the positions of all different pieces for every move appears to be computationally expensive and not feasible.

As an alternative approach, four lookup tables are designed to store the sequence of squares for different players, by applying four simple arrays of size 25. Each array index indicates the position with respect to the player's path, and the value stored in that array index represents the corresponding square with respect to the common board. Fig. 3 illustrates the lookup tables for *Red* (South) and *Blue* (North) players.

Four two-dimensional arrays are designed as "table of pieces" to store the current and destination positions of pieces for each player. Table V and Table VI are examples of table of pieces for *Red* and *Blue* players.

We could verify that all the designed data structures interact with each other accurately.All the generated random cowry values, the pieces that have been selected randomly to make the moves as well as all other statistics of the games (such as number of squares traveled by each piece, number of winnings for each player, etc.) are automatically being recorded as log files for further processing.

	0	1	2	3	4
0	00	01	02	03	04
1	10	11	12	13	14
2	20	21	22	23	24
3	30	31	32	33	34
4	40	41	42	43	44

Figure 4. 2D Array Representing the Game Board

B. *Number of Experiments*

Initially we tested our game setup by running 50 games with two AI players using pure random strategy. This was done to ensure that our game setup is correct and that each random player wins approximately equal number of times. Later we extended the number of AI players to four and we could verify that each random player wins 25.0±1.0% of the games.

The obtained results stabilized at nearly 1000 games. Running a higher number of games did not change the variation in results significantly, suggesting that 1000 trials was an adequate number to observe reliable trends considering only the random strategy. However, more number of game runs would be advisable for analysis of other types of strategies.

C. *Testing the Performance of Random Strategy*

The first phase of our experimentation was to ensure that all AI players have equal chance of winning when they all follow pure random strategy. This was achieved by running 1000 games with all four AI players playing randomly. We state the results in Table II and this test clearly demonstrates that each random player wins 25.0±1.0% of the games, hence the probability of winning for each random player is approximately equal.

Furthermore, we randomly altered the order of players starting the game to check whether there could be any correlation between the player taking the first turn and the player who wins the game. The obtained results in Table II verifies that considering the total number of games won by any random player, approximately 25.0±2.0% of times that particular player had taken the first turn. Therefore, the results were unchanged when the player order was changed, suggesting that there was no significant bias towards the player taking the first turn.

Home Square:0

Safe Squares: 0, 4, 8, 12

Outer Squares: 0 to 15

Inner Squares: 16 to 23

Central Square: 24

			Blue (North)			
	10	9	8	7	6	
	11	18	19	20	5	
Yellow (West)	12	17	24	21	4	Green (East)
	13	16	23	22	3	
	14	15	0	1	2	
			Red (South)			

Figure 5. The Path for*Red* (South) Player

TABLE II. PROBABILITY OF WINNING FOR AI RANDOM PLAYERS

1000 Games withRandom Players	Red (South)	Green (East)	Blue (North)	Yellow (West)	Total
No. of times the player has won the game	249	252	251	248	1000
No. of times the winner has started the game	63	64	62	61	–
Percentage of winning for the player	24.9%	25.2%	25.1%	24.8%	25.0 ± 1.0%
Percentage of taking the first turn when the player has won	25.3%	25.4%	24.7%	24.6%	25.0 ± 2.0%

According to the "law of large numbers" [30], the average of the results obtained from a large number of trials should be close to the expected value, and will tend to become closer as more trials are performed. Therefore, the purpose of our next phase of experimentation was to show that for 1000 game runs, the average value obtained from throwing four cowry shells converges to the expected value which was calculated theoretically in section 5.

TABLE III. AVERAGE VALUE OF THROWING 4 COWRY SHELLS

No. of Games	Winner	Sum ofCowry Values for the Winner	Total No. of Throws by the Winner	Average Cowry Value for Each Game
1	Green	134	51	2.627
2	Red	121	47	2.574
3	Yellow	118	43	2.744
4	Green	126	52	2.423
5	Blue	115	48	2.396
⋮	⋮	⋮	⋮	⋮
1000	Red	131	49	2.673
		Sum of All Cowry Values in 1000 Games	**Total No. of Throws in 1000 Games**	**Average Cowry Value in 1000 Games**
		125724	49548	2.537

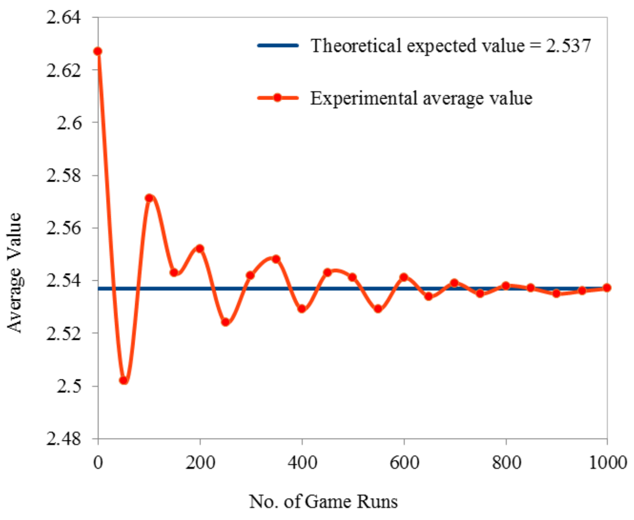

Figure 6. Average Value of Throwing 4 Cowry Shells

Table III states the summation of all cowry values obtained by a player to win a game as well as the total number of throws by the winner. Clearly, the ratio between these two values gives the average cowry value for a single game. We tabulated the results for 1000 game runs and finally calculated an overall average. Fig. 6 illustrates that as the number of throws increases, the average of all values approaches 2.537 which is the calculated theoretical expected value.

We performed the next phase of experimentation to obtain a practical average number of moves for an AI random player to win the game. We then compared the test results with the expected minimum number of moves calculated in section 5. Table IV gives a view of test results and states the practical average number of moves for an AI player to win against other AI players when all of them are playing randomly.

In general, a smaller number of moves to finish the game could be an indicator of a better performance. However, it can be observed from Fig. 7 that random players take much higher number of moves to win compared to the theoretical expected minimum number of moves, due to wrong moves and losing many opportunities. More specifically, a random player requires around 70% more number of moves on average to complete the game in comparison to the calculated minimum value, which implies that the random strategy performs very poorly in actual games.

TABLE IV. AVERAGE NO. OF MOVES WITH 4RANDOM PLAYERS

No. of Games	Practical Average No. of Moves for the Winner	Theoretical Expected Minimum No. of Moves
1	64.31	37.84
2	66.85	37.84
3	65.11	37.84
4	63.71	37.84
5	66.44	37.84
⋮	⋮	⋮
1000	63.58	37.84
	Practical Average No. ofMoves in 1000 Games	**Theoretical Expected Minimum No. of Moves**
	64.78	37.84

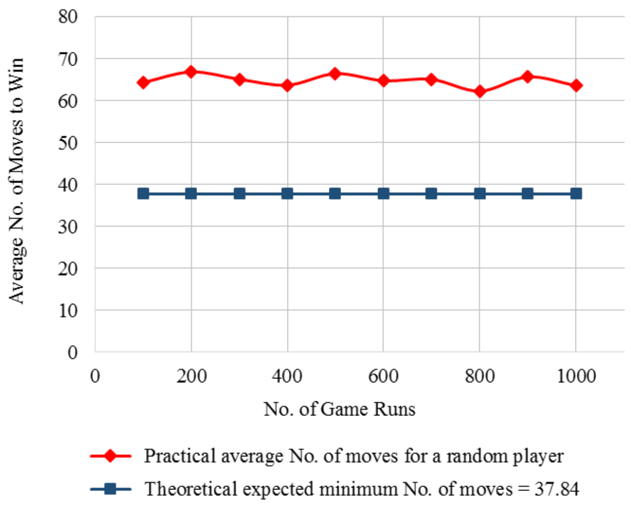

Figure 7. Average No. of Moves with 4Random Players

As discussed earlier, by analyzing the moves of AI random players we can identify some obvious mistakes they commit during the game-play. To demonstrate some of these mistakes, it may be relevant to give an actual example of a configuration in a game between two AI random players by considering the set up shown in Fig. 8.

Our implemented program is responsible to generate a random cowry value for the players and also to choose one of the pieces randomly to play that move. In case the selected piece cannot legally make the move due to the rules of the game, the program should backtrack and select another random piece to move.

Let us assume that both the players are eligible to move to inner squares. It is *Red*'s turn to play and the cowry value generated randomly is 3. The positions of *Red* pieces are given in Table V.

At first, *Red* player randomly selects R4 to play the cowry value 3. Obviously, it is an invalid move and will be rejected by the program because R4 cannot move to a square occupied with another *Red* piece. Hence the program backtracks and selects another random piece R2, which only escapes from the danger zone of one opponent piece B1, and enters the danger zone of another piece B4. Hence this move won't have any significant advantage for the player.

The best move in this configuration would be moving R1 to inner squares, escaping an advanced piece from the danger of elimination by B3 and also threatening the opponent piece B2 which is just one square away from the goal.

Now let us assume *Blue* is to play in the same configuration and the generated cowry value is 2. Table VI shows the positions of *Blue* pieces.

In order to play the cowry value 2, *Blue* player randomly selects B2 which is again identified by the program to be an invalid move because this piece only requires 1 to reach the goal. In the next attempt, B4 is randomly selected. Clearly, this would be an obvious mistake, moving a piece out of a safe square and placing it in the danger zone of R2 chasing it.

Moving B1 to a safe square and escaping from the danger zone of R4 was definitely a better choice. However, the strongest move in this set up would be hitting an advanced opponent piece R1 with B3, which could also give *Blue* player an additional turn to play.

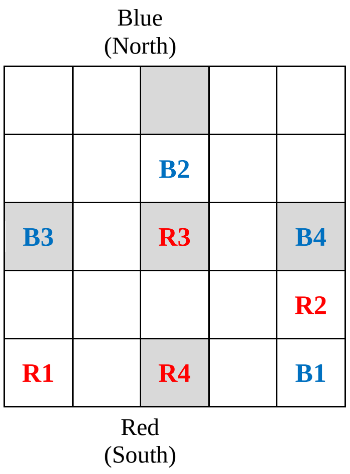

Figure 8. Example Board Setup between 2 Random AI Players

TABLE V. TABLE OF PIECES FOR *RED* PLAYER

Piece	Current Position w.r.t Player's Path	Current Position w.r.t the Board	Destination Position w.r.t Player's Path	Destination Position w.r.t the Board
R1	14	40	14 + 3 = 17	21
R2	3	34	3 + 3 = 6	04
R3	24	22	24 + 3 = 27	–
R4	0	42	0 + 3 = 3	34

- R1: Escape & Attack (exits the danger zone of B3, and attacks B2)
- R2: Under Attack (exits the danger zone of B1, but enters the danger zone of B4)
- R3: Invalid Move (already reached the goal)
- R4: Invalid Move (cannot hit another Red's piece)

TABLE VI. TABLE OF PIECES FOR *BLUE* PLAYER

Piece	Current Position w.r.t Player's Path	Current Position w.r.t the Board	Destination Position w.r.t Player's Path	Destination Position w.r.t the Board
B1	10	44	10 + 2 = 12	24
B2	23	12	23 + 2 = 25	–
B3	4	20	4 + 2 = 6	40
B4	12	24	12 + 2 = 14	04

- B1: Escape (exits the danger zone of R4, and enters a safe square)
- B2: Invalid Move (cannot move beyond the goal)
- B3: Hit (hits the opponent piece R1)
- B4: Under Attack (exits a safe square, and enters the danger zone of R2)

As the above example demonstrates, random playing could result in some obvious mistakes and losing many opportunities throughout the game. Some of the typical mistakes committed by AI random players includes but not limited to the following,

- Missing the chance to reach the goal
- Missing the chance to hit an opponent piece
- Missing the chance to attack (chase) an opponent piece
- Missing the chance to escape into a safe square
- Moving out of a safe square into a danger zone
- Missing the chance to escape from a danger zone
- Moving into a danger zone

During the experimentation phase with four AI random players, we could identify and classify a vast number of such wrong decisions due to "blind random selection" of pieces, suggesting the poor performance of the random strategy in an actual Cowry games. However, a careful analysis of these wrong decisions and attempting to avoid them can lead us to generate more advanced strategies in our future work.

In the last phase of our experimentation in this paper, we conducted several test games between one AI player which was playing randomly against a novice-level human player who was not specialized in any type of strategy and was playing purely based on intuition.It was clear that the AI random player had a very little chance of winning against a human player. However, despite its poor performance and losing many opportunities, the AI random player was successful to hit the human player a few times.

Our analysis shows that it was mainly because of human player mistakes in placing his pieces in risky positions, due to the fact that the human player was ignoring the danger from an opponent who was playing randomly without any intelligence. This test candemonstrate the possible effectiveness of the random strategy for confusing an intelligent player, in the sense thatit is impossible to anticipate the next move of the AI player who is playing randomly without any pattern.

Table VII gives the detailed result of 5 sample games and Fig. 9 illustrates the poor performance of an AI random player against a beginner human player, based on the total number of moves to finish the game.

We shall mention that our basic implementation of the game only has a text-based user interface, and although it allows human interactions with the program through a command line, it is not capable of showing the movements of pieces graphically on the screen. Design and implementation of the graphical user interface (GUI) for the game is considered as a future work.

TABLE VII. STATISTICS OF HUMAN PLAYER VS. AI PLAYING RANDOMLY

	Game 1		**Game 2**		**Game 3**		**Game 4**		**Game 5**	
	Human	AI	Human	AI	Human	AI	Human	AI	Human	AI
Status	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost
Total No. of Moves	42	71	44	73	40	75	45	78	41	76
Total No. of Squares	108	184	112	179	102	188	110	191	105	185
Average Cowry Value	2.57	2.59	2.54	2.45	2.55	2.51	2.44	2.45	2.56	2.43
No. of Hits	7	1	9	2	6	1	8	2	5	1
Lost the Chance to Hit	0	5	1	6	0	4	0	5	1	6

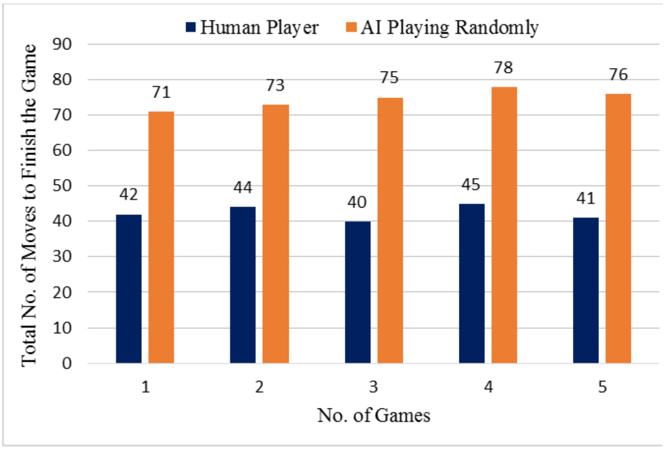

Figure 9. Performance of Human Player vs. AI Playing Randomly

VII. CONCLUSIONS AND FUTURE WORK

In this research we introduced and conducted a fundamental study of Cowry game which is a race game from India, and we investigated the potential of a Cowry game AI. We discussed that the complexity of Cowry game is more than games like *Chess*due to the presence of an element of chance, suggesting that Cowry game is more challenging to solve compared to perfect information games. Accordingly, we proposed several playing strategies in the game: *random*, *aggressive*, *defensive*, *move-first*, *move-last* and *mixed* playing strategies.

The main aim of this paper was to analyze the performance of randomness as a model for playing. A random strategy involves no decision-making for the players; hence we could entirely concentrate on the design and implementation of the game and to validate the correctness of our implementation and also to ensure that all of the game rules are applied properly.

Furthermore, we believe that the analysis of random playing can be used for evolving better strategies in future work and also can serve as a benchmark against which to compare other strategies. Theoretical and experimental results presented in this research show that the random strategy performs poorly based on the average number of moves to win the game.

The basic implementation of the game had a text-based user interface and was not capable of showing the game board and movements of pieces graphically on the screen. Design and implementation of the graphical user interface which could facilitate human interactions would be our next phase of work.

For future work, we also consider the implementation, analysis and comparison of other types of strategies presented in this work. This can be achieved by formulating different evaluation functions. Using machine learning approaches, the evaluation functions may be adjusted after several game runs, and as proposed by [24], evolutionary algorithms may also be used to improve the evaluation functions after successive generations. Considering variations in learning modes such as learning by playing against an expert human player and learning by observation can also result in the discovery of better strategies and improved game-play.

APPENDIX

The "empirical probability" or "experimental probability" of an event, according to [31], is defined as the ratio of the number of results in which a certain event occurs to the total number of trials, obtained from experience and observation. In other words, empirical probability determines probabilities, not in a theoretical sample space but in an actual experiment.

As is widely cited in the thesis [33], we performed the experiment of throwing four cowry shells together 5000 times and tabulated the probability of occurrence for each value. The probability of occurrence and cumulative probability for each cowry value is shown in Table VIII.

Clearly the obtained probability distribution is non-uniform.Therefore, the problem in simulation of throwing four cowry shells is the generation of pseudo-random numbers that are following our obtained empirical distribution.

Uniform random number generators are used so frequently that almost every computer programming language include functions or library routines that provide random number generators. However, there are no fast practical methods of generating non-uniform random numbers on the computer, except via the uniform random numbers.

TABLE VIII. EMPIRICAL PROBABILITY OF OCCURRENCE

Sub-intervals	0 – 1	1 – 2	2 – 3	3 – 4	4 – 5
Cowry Values	1	2	3	4	8
Probability of Occurrence	0.243	0.381	0.236	0.074	0.066
Cumulative Probabilities	0.243	0.624	0.860	0.934	1.000

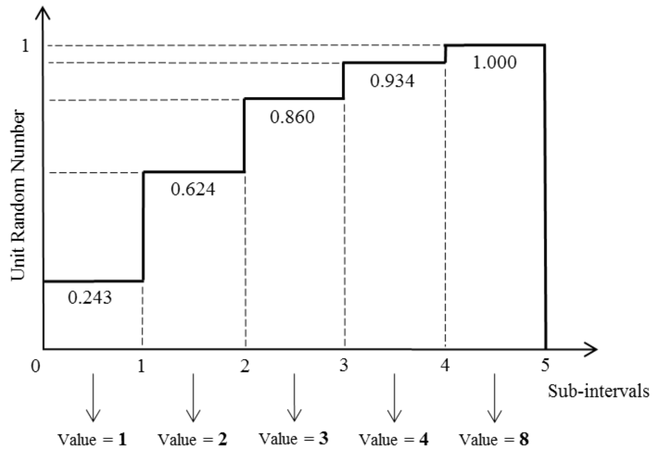

Figure 10. Mapping a Unit Random Number to Represent a Cowry Value

Most algorithms are based on a pseudo-random number generator that produces numbers that are uniformly distributed in the interval [0,1). These random numbers are then transformed via some algorithm to create a new random number having the required probability distribution. There are many techniques and tricks for converting uniform random numbers into our desired non-uniform random numbers, among which we have experimented the following two most commonly used methods,

- Inverse Transformation Method
- Acceptance-Rejection Method

After careful experimentation on both of the above mentioned methods in the thesis [33], we observed that the performance and complexity of the "inverse transformation method"[32] is better for the simulation of throwing four cowry shells in our game. Therefore, a uniform random number in the interval [0,1) is generated, then by using the inverse transformation method the generated unit random number is mapped to one of the five sub-intervals shown in Fig. 10, where each sub-interval represents a cowry value.

REFERENCES

[1] "Board game", (Accessed: 07-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Board_game

[2] "Race game", (Accessed: 07-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Race_game#cite_note-1

[3] "Pachisi", (Accessed: 25-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Pachisi

[4] "Chaupar", (Accessed: 25-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Chaupar

[5] "Chowka bhara", (Accessed: 25-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Chowka_bhara

[6] "Backgammon", (Accessed: 07-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Backgammon

[7] "Ludo (Board Game)", (Accessed: 07-Nov-2016). [Online]. Available: https://en.wikipedia.org/wiki/Ludo_(board_game)

[8] "Game theory", (Accessed: 13-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Game_theory

[9] "Turn-based strategy", (Accessed: 13-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Turn-based_strategy

[10] "Strategy (game theory)", (Accessed: 13-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Strategy_(game_theory)

[11] I. Millington, J. Funge, "Artificial intelligence for games", 2nd ed., Morgan Kaufmann Publishers, 2009, pp. 667–670.

[12] D. Koller, A. Pfeffer, "Generating and solving imperfect information games", Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 1185–1192.

[13] D. Koller, N. Megiddoy, B. von Stengel, "Fast algorithms for finding randomized strategies in game trees", Proceedings of the 26th Annual ACM Symposium on the Theory of Computing, 1994, pp. 750–759.

[14] M. Zinkevich, M. Bowling, M. Johanson, C. Piccione, "Regret minimization in games with incomplete information", Proceedings of the 21th Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, 2007.

[15] T. Sandholm, "The state of solving large incomplete information games, and application to Poker", Association for the Advancement of Artificial Intelligence, 2010.

[16] C. A. Luckhardt, K. B. Irani, "An algorithmic solution of N-person games", Proceedings of the 15th National Conference on Artificial Intelligence, Philadelphia, PA, 1986.

[17] N. Sturtevant, "Multi-player games: algorithms and approaches", PhD thesis, Dept. of Computer Science, Univ. of California, Los Angeles, 2003.

[18] J. P. Zagal, J. Rick, I. His, "Collaborative games: lessons learned from board games", Simulation and Gaming, vol. 37, No. 1, Sage Publications, 2006, pp. 24–40.

[19] Browne, Powley, Whitehouse, Lucas, Cowling, Rohlfshagen, Tavener, Perez, Samothrakis, Colton, "A survey of Monte Carlo tree search methods", IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, No. 1, 2012.

[20] J. Long, N. Sturtevant, M. Buro, T. Furtak, "Understanding the success of perfect information Monte Carlo sampling in game tree search", Proceedings of the 24th AAAI Conference on Artificial Intelligence, Atlanta, Georgia, USA, 2010.

[21] T. Furtak, M. Buro, "Recursive Monte Carlo search for imperfect information games", IEEE Conference on Computational Intelligence and Games (CIG), 2013, pp. 1–8.

[22] P. Ciancarini, G. Favini, "Monte Carlo tree search techniques in the game of Kriegspiel", Artificial Intelligence, vol. 174, 2010, pp. 670–684.

[23] F. Aiolli, C. Palazzi, "Enhancing artificial intelligence in games by learning the opponent's playing style", International Federation for Information Processing, vol. 279, 2008, pp. 1–10.

[24] F. Alvi, M. Ahmed, "Complexity analysis and playing strategies for Ludo and its variant race games", IEEE Conference on Computational Intelligence and Games (CIG), 2011.

[25] Rensink etal., "Ludo: a case study for graph transformation tools", AGTIVE, vol. 5088, 2008, pp. 493–513.

[26] Bachelorarbeit, "Decision-making in hostile environments with incomplete information", PhD thesis, Bamberg Univ., Germany, 2010.

[27] F. Oderanti, "Fuzzy decision-making system and the dynamics of business games", PhD thesis, Heriot-Watt Univ., Edinburgh, United Kingdom, 2011.

[28] Caird-Daley, D. Harris, K. Bessell, M. Lowe, "Training decision-making using serious games", Human Factors Integration Defence Technology Centre, 2007.

[29] "Expected value", (Accessed: 22-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Expected_value

[30] "Law of large numbers", (Accessed: 22-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Law_of_large_numbers

[31] "Empirical probability", (Accessed: 28-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Empirical_probability

[32] "Inverse transform sampling", (Accessed: 28-Dec-2016). [Online]. Available: https://en.wikipedia.org/wiki/Inverse_transform_sampling

[33] P. Davoudian, Ranjitha R.Y. , "On simulating the throw of cowry shells for Chowka Bhara and intelligent decision-making", MSc thesis, Dept. of Computer Science, Univ. of Mysore, India, 2013, unpublished.