MACHINE AS ONE PLAYER IN INDIAN COWRY BOARD GAME: ADVANCED PLAYING STRATEGIES

Pouyan Davoudian
Department of Studies in Computer Science, University of Mysore, India.

P. Nagabhushan
Indian Institute of Information Technology, Allahabad, India.
Department of Studies in Computer Science, University of Mysore, India

ABSTRACT

Cowry game, also known as Chowka Bhara, is an ancient board game originated in India. It is a 2-4 player strategic race game which contains an element of chance due to the throw of special dice (cowry shells). This game involves reasoning under uncertainty and stochasticity, and the decisions have to be made based on incomplete knowledge. Therefore, it may be regarded as a suitable domain for exploring various approaches in real-time strategic decision-making. This research investigates the potential for an artificially intelligent Cowry game player, capable of playing on a high-performing level against human opponents. We propose and implement a few advanced playing strategies for the game, and analyze their effectiveness based on different performance measures. We also provide an experimental comparison between the proposed advanced strategies and the basic strategies that we developed in our earlier work. The analysis of playing strategies presented in this paper can be used for evolving more sophisticated strategies, and may eventually lead to the development of an adaptive artificial player for Cowry game and similar race games

Keywords: Evaluation function, Strategy, Aggressive, Defensive, Decision tree, Artificial player, Race board game, Cowry shell, Chowka Bhara.

Cite this Article: Pouyan Davoudian and P. Nagabhushan, Machine as one Player in Indian Cowry Board Game: Advanced Playing Strategies, International Journal of Computer Engineering and Technology, 10(2), 2019, pp. 1-13.
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=10&IType=2

1. INTRODUCTION

The fast development of computers and interactive media over recent decades has encouraged many researchers to apply computational science in understanding and modeling complex real-life problems. Computational science can provide insights into many problems that are
too complex to be solved by analytical methods or mathematical reasoning, due to the existence of uncertainty and stochasticity during the process [1, 2].

In order to model real-life problems efficiently, they often need to be simplified and abstracted to some extent. An abstraction may yield a sufficiently accurate solution for the actual problem while reducing its complexity significantly. Games can be considered as good abstractions for real-life problems that contain uncertainty, such as partially observable or stochastic environments [2]. The profound connection between the concept of game and the notion of computation has been long recognized in academia, suggesting that games should be thought of as an appropriate and reasonable model of computation [3]. Computer games offer compelling applications for computational intelligence and machine learning, as well as problem-solving and decision-making.

Furthermore, board games are one of the first branches studied in artificial intelligence. Shannon devised a machine to play Chess as early as 1950 [4], Turing proposed his Chess-playing program in the same year [5], and Samuel developed a self-learning Checkers player in 1959 [6]. Although a substantial amount of research has been devoted to perfect information two-player games like Chess, Checkers and Go over the years, many strategic board games and most real-life problems involve imperfect information with more than two parties. Unfortunately, so far there have been fewer successes in designing computer programs to play imperfect information, multi-player, strategic board games. Such games offer the challenge of reasoning under uncertainty and stochasticity, and often require enormous sophistication for a machine to play at expert level. One game that incorporates many of these challenges is Cowry game, an ancient board game from India also known as Chowka Bhara.

1.1. Background

Cowry game is a race game of chance and strategy played by two to four players. Each player is represented by a distinct color (usually red, yellow, green and blue), and has four tokens or “pieces” of the same color. The game objective is to be the first player to move all four pieces along a specified track and into the finishing location. Players’ pieces are moved according to the throw of four cowry shells (shown in Figure 1), with the number of shells landing with their openings upwards indicating the number of squares to move.

Figure 1 Cowry shells

Cowry game is normally played on a 5x5 square board, with four starting squares at the outer middle locations on each side of the board, and a finishing square at the center. The starting squares are specially marked and also function as “safe squares”, which can simultaneously accommodate multiple pieces of different players. Figure 2 depicts the game board with safe squares and shows the path for one of the players with dotted lines.
In our previous paper [7], we presented a detailed description of Cowry game and outlined the rules of the game which are also considered for the implementation in this work. We proposed and analyzed two basic playing strategies: fast strategy (moving the foremost piece) and balanced strategy (moving the hindmost piece), and provided an experimental comparison of their performances.

In this research, we further develop two advanced playing strategies for the game: aggressive and defensive strategies. We also provide a comparative analysis of all the strategies, and experimentally show the advantage of the new strategies over those we proposed earlier. This work is aimed at improving the domain knowledge of Cowry game, which can be used to enhance its gameplay and might be the basis for a strong adaptive artificial player.

1.2. Overview
In the rest of this paper, we describe our proposed strategies and the logic behind them in Section 2. Section 3 provides details about the implementation of the strategies and highlights how they are constructed using simple decision trees. In Section 4, we propose a few performance measures to evaluate the effectiveness of different strategies. We also include a comparative analysis of all the proposed strategies and a brief discussion on the obtained experimental results. Section 5 presents the conclusion and provides guidelines for future research.

2. STRATEGIES
In our last paper [7], based on the approach introduced by Shannon [4], we identified two main categories of strategies for Cowry game:

1. Pure strategy, which always selects the same move in the same board position.
2. Mixed strategy, which does not necessarily result in the selection of the same move in the same position.
Accordingly, we developed two basic pure strategies: fast strategy (moving the foremost piece) and balanced strategy (moving the hindmost piece). We also implemented a random-move generator, which falls into the category of a mixed strategy. The random strategy merely functions as a benchmark to compare the performance of other proposed strategies in our research work.

In this section, we propose two advanced pure strategies (aggressive and defensive) and explain the logic behind them. We also introduce a hybrid strategy which is, in fact, a combination of all previously proposed pure strategies, and hence comes under the definition of a mixed strategy.

2.1. Aggressive Strategy
In an aggressive strategy, a player always gives a high priority to move the piece which can eliminate an opponent’s piece, whenever possible. Hitting the piece of an opponent has the advantage of delaying the opponent’s progress and in turn, increasing the player’s chance of winning [8]. The eliminated piece is sent back to its starting square and all the moves made by the opponent for the advancement of that piece are lost. Hitting an opponent’s piece also gives the attacker an additional turn to throw the cowry shells.

The players following an aggressive strategy often try to gain more control of the game by positioning their pieces on critical squares of the board. They also attempt to chase the pieces of other players closely, causing them to panic, and wait for the required cowry shells outcome to attack. Obviously, moving a piece out of a safe square to chase an opponent’s piece may put the attacking piece at risk of elimination. An aggressive strategy player, however, is willing to take such a risk and relies on lucky throws to make a hit, and the bonus throw to escape the probable danger zone.

2.2. Defensive Strategy
A defensive strategy is based on a preference for protecting the player’s pieces from being eliminated as much as possible. In this strategy, a player always tries to minimize the probable loss by keeping the pieces away from high risk of elimination. As proposed by [8], a piece is considered to be in danger if it can be eliminated in a single throw of cowry shells by its opponents. Hence, the players following a defensive strategy always give a high priority to move the piece that faces a serious threat, out of the danger zone and preferably to a safe square. A detailed description of the “threat” appears in Section 3.1.

In general, the winner of Cowry game is a player who can take all four pieces around the board and to the finishing square with the minimum number of moves. In this sense, a defensive strategy seems to serve the ultimate purpose of the game by decreasing the risk of piece elimination and in turn, minimizing the total number of moves required for all the pieces to finish the race.

2.3. Hybrid Strategy
As with any other strategic game, it is quite unlikely for a player in Cowry game to achieve a high winning rate simply by using a single pure strategy throughout the whole game. Expert human players frequently change their strategies at different stages of the game according to different factors, such as the progress and arrangement of their pieces, number of opponents and their strengths and weaknesses, fulfillment of certain conditions in the game, etc.

A hybrid strategy can be any combination of the strategies already described. In a game with two players, as an example, a hybrid strategy for the opening phase of the game could be as simple as follows: A player may initially start with a fast strategy to take at least one of the pieces close to the opponent’s pieces. The player may then change to an aggressive strategy to
hit the first piece of the opponent and fulfill the first-hit condition required to move into the inner squares. Upon making the first-hit, the player may shift to a defensive strategy combined with a balanced strategy to advance all the pieces close to each other into the inner squares, while preventing the opponent from gaining the first-hit privilege.

More sophisticated hybrid strategies can be formulated by taking more detailed knowledge of the game into account. However, designing the hybrid strategy is not part of this paper and considered as future work. In this article, we merely concentrate on the implementation of proposed pure strategies and analysis of their relative performances against each other.

3. IMPLEMENTATION AND ANALYSIS

A computer can play a strategic board game such as Cowry game, by considering all possible legal moves in a given position and selecting one of them. Obviously, our aim is not to design a computer program that merely plays a legal game, which is trivial. We would like to develop a skillful artificial player capable of making rational decisions, perhaps comparable to expert human players. The work [4] has been stated that in order to choose a tolerably good move out of the valid moves in any board position, a player must often perform a position evaluation to identify what moves are better than others. Such evaluations may not be perfect, but the more advanced the player the more accurate the evaluations.

The knowledge of approximate evaluation of a position can be provided to the computer using a heuristic, popularly known as the “static evaluation function”. The primary purpose of the static evaluation function is to analyze the current state of the board from a player’s perspective, and score the board positions that will result after simulating each possible move. The score should, in fact, express the relative chance of a player to win the game from that particular board position [9]. Hence, if the board presents a great advantage after making a certain move, that move should receive a higher score compared to other possible moves in that position.

Human players often have an intuitive sense to anticipate who is winning a race board game. A computer, on the other hand, should mainly rely on heuristics to recognize the balance of winning or losing, as proposed by [9]. Consequently, all the strategic knowledge of how to play the game and what positions to prefer must be incorporated in the static evaluation function by the programmer. Given a good static evaluation function, all that the computer needs to do is observing the scores of all possible moves and selecting the move with the highest score.

The input to the static evaluation function is the “knowledge” that a player possesses about the game environment, and the output is a set of scores corresponding to each possible move in the current state of the game. According to [9], we can further classify the knowledge into “internal” and “external” knowledge. Internal knowledge is what players know about the internal state and the situation of their own pieces on the board, without taking the opponents into consideration. For example, the number and formation of player’s pieces, their distances to the finishing square and safe squares, whether the first-hit condition has been fulfilled or not, etc. External knowledge, in contrast, is the information related to other players involved in the game, such as the number of opponents and the progress of their pieces, the player’s pieces under attack by the opponents, the opponents’ pieces in danger of being eliminated by the player, etc.

In order to construct the static evaluation function from a set of knowledge, a procedure is required that can efficiently group several inputs together and generate a corresponding score as an output. “Decision trees” are perhaps the simplest decision-making tool for this purpose. They have the advantage of being fast, easy to create, and simple to understand [9].

http://www.iaeme.com/IJCET/index.asp  

editor@iaeme.com
A decision tree is a set of connected decision nodes, starting with a single node called the “root”. For each decision, one of the possible outcomes is selected. Each of those choices results in other decision nodes, which in turn branch off into a set of other options. Most choices typically have only two possible outcomes, and the decisions are made according to the player’s knowledge of the game. Each “leaf” node of the tree holds a label or a score which is assigned to that specific move.

3.1. Implementation of Strategies

As discussed in our previous work [7], the fast and balanced strategies are considered as “position-based” strategies since they select a move based on the position of the piece that moves. They are merely concerned with the internal knowledge of players about their own pieces, without taking the opponents into account. In contrast, the aggressive and defensive strategies can be termed as “result-based” strategies, in the sense that the scores assigned to different moves are mainly influenced by the outcome of the moves [10]. Both of these strategies employ external knowledge of players about the opponents as well as their internal knowledge to calculate the scores.

Here we need to properly define the concept of threat or facing the danger of elimination, which was also used earlier in this article. We propose that a piece is at risk when it can be eliminated in a single throw of cowry shells in the current turn of the game by any of the opponents. In the work [11], we obtained the empirical probability for the occurrence of each outcome in a single throw of four cowry shells. Based on the obtained results (shown in Table 1), we can observe that the chance of getting any of the values 1, 2 or 3 is around 86% while the probability of throwing either a 4 or an 8 is only about 14%.

\[
P(1 \cup 2 \cup 3) = P(1) + P(2) + P(3) \approx 86\% \\
P(4 \cup 8) = P(4) + P(8) \approx 14\%
\]

Table 1 Empirical probability for the occurrence of outcomes in a throw of 4 cowry shells

| Cowry Shells Outcome | 1    | 2    | 3    | 4    | 8    |
|----------------------|------|------|------|------|------|
| Probability of Occurrence | 24.3% | 38.1% | 23.6% | 7.4%  | 6.6%  |

Accordingly, we propose that the threat can be further broken down into “low” and “high” threats. A piece encounters a high threat when it is located on a non-safe square and it is 1, 2 or 3 squares ahead of any of the opponents’ pieces, based on the path of that specific opponent. It is highly probable for such a piece to be eliminated in the current turn of the play. A low threat, however, is defined as a situation in which a player’s piece is on a non-safe square, exactly 4 or 8 squares in front of an opponent’s piece. Such a piece has relatively a better chance to survive the current turn. From this point of view, in any given board position, a piece can be in one of the following states:

- **High Threat**: on a non-safe square, with at least one opponent’s piece in close range (1, 2 or 3 squares behind).
- **Low Threat**: on a non-safe square, with at least one opponent’s piece in distant range (4 or 8 squares behind).
- **No Threat**: on a non-safe square, with no opponent’s piece in range.
- **Safe Square**: on a safe square, completely out of danger.
- **Finishing Square**: on the final square, already finished the race.
The primary goal of the defensive strategy is to safeguard the player’s pieces against the danger of impending attacks. This can be achieved by scanning the position of opponents’ pieces on the board, and finding a move which results in the least level of threat on the player’s pieces. The program needs to evaluate the current square of each piece, as well as its destination square with respect to the given cowry shells outcome. It should determine to which of the above-mentioned states these squares belong. Each state is associated with a numeric value or “point”, as shown in Table 2. The final score of each move is given by the difference between the corresponding points of current and destination squares of the piece, as follows:

\[ \text{Score (piece } i \text{)} = \text{Destination Points (piece } i \text{)} - \text{Current Points (piece } i \text{)} \]

For example, moving a highly threatened piece to the finishing square receives the maximum score \((5 - 1 = 4)\), whereas moving a piece out of a safe square onto a location exposed to a high danger of elimination gets the minimum score \((1 - 4 = -3)\). Figure 3 demonstrates the process of assigning points to the destination square at the leaf nodes of the decision tree. The points presented here are merely given for illustrative purposes. There are other factors that need to be included.

Table 2 Points for evaluation of current and destination squares

| Current Square      | Points | Destination Square | Points |
|---------------------|--------|--------------------|--------|
| High Threat         | 1      | High Threat        | 1      |
| Low Threat          | 2      | Low Threat         | 2      |
| No Threat           | 3      | No Threat          | 3      |
| Safe Square         | 4      | Safe Square        | 4      |
| –                   | –      | Finishing Square   | 5      |

Figure 3 Assigning points for the evaluation of the destination square

It may be noted that the score calculated in this manner is entirely concerned with the threat received from the opponents’ pieces. The algorithm does not see any advantage in moving a piece which can eliminate the piece of an opponent. Hence, we had to artificially include an appropriate score for hitting an opponent’s piece. The updated decision tree checks if any of the player’s pieces can perform a hit, and permits it only if it is a “safe-hit”. We define a safe-hit as the opportunity to eliminate an opponent’s piece when both of the following conditions are satisfied: (i) None of the player’s pieces should be at risk in their current position, and (ii) The attacking piece should not face a high threat after making the hit.

Moreover, the algorithm takes the progress of the player’s pieces into consideration to reduce the probable loss. For instance, if more than one piece of the player need to escape a
high-risk situation, the priority is always given to the more advanced piece. In case the player has no option but to move either of two pieces into the danger zone, the program always chooses to put the less advanced piece at risk.

While the defensive strategy aims to minimize the received threat on player’s pieces, the aggressive strategy attempts to maximize the threat posed on opponents’ pieces. The decision tree for the aggressive strategy is constructed using a similar principle. The chief difference is that the algorithm concentrates on the opponents’ pieces located in front of the player’s pieces, rather than those behind. Hitting a piece of any of the opponents always receives a high score. The more advanced the victim piece the higher the score. Thus, if more than one piece can be eliminated in a given position, the program calculates the progress of each targeted piece, as well as the overall progress of the corresponding opponent. To increase the probable gain, the program always gives preference to attack the more advanced pieces of the stronger opponents.

4. EXPERIMENTATION AND RESULTS

In this section, the experiments conducted on the proposed pure strategies are presented and the obtained results are evaluated. However, in order to compare the records of different strategies and to analyze how well a particular strategy plays, we first need to define a few performance measures.

4.1. Performance Measures

There are several methods to quantify the effectiveness of a strategy. “Winning percentage” is perhaps the most common method to assess the performance of a player in competitive race games. It is defined as the number of games a player has won divided by the total number of games played. A higher winning percentage over a series of many games is often considered as an overall stronger level of play.

“Expected number of moves” required for winning games can also serve as a suitable measure to analyze the performance of a player. It is basically the number of moves taken by a player on average to finish games in the long run. A smaller expected number of moves is a key indicator of a better performance.

“Margin of victory” is a statistic based on the difference between the progress of the winning player and the progress of second-place winner in a game. It is, in fact, the remaining distance for the runner-up opponent to finish the race when the player wins. For the particular case of Cowry game, it can be calculated by identifying the most advanced opponent as soon as the player finishes the game:

\[
\text{Margin of Victory} = 1 - \text{Progress of the Most Advanced Opponent}
\]

Where,

\[
\text{Progress} = \frac{1}{4 \times 24} \sum_{i=1}^{4} \text{Distance Covered (piece } i \text{)}
\]

The margin of victory is an efficient method to determine how significant the victory was by a specific player. A smaller margin of victory generally means that the winner succeeded in a close and competitive game, whereas a larger margin of victory often denotes a definite and one-sided victory.
4.2. Testing Performance of Strategies

The program is constructed to play quickly enough so that several thousands of fully automated games can be executed within a reasonable amount of time. All actions that the players take and all the relevant statistics are stored in machine-readable log files for further analysis. The program is also designed to facilitate the gameplay against humans, although such games are much slower and only a few dozen test games could be performed. Based on the approach introduced in [8], we conducted the experiments in three distinct phases:

1. Testing the independent performance of each one of the proposed pure strategies against three automatic random players for 10,000 games.
2. Testing the relative performances of all pure strategies competing against each other in the same game for 10,000 runs.
3. Testing the performance of each pure strategy in a two-player game against inexperienced human opponents for 50 trials.

4.2.1. Performance of Each Pure Strategy against Random Players

In the first phase, we observed the performance of each pure strategy individually in numerous four-player games, when the other three players were merely making randomly chosen legal moves at each turn. We evaluated the performance of each strategy according to the performance measures proposed earlier. The results of these tests are given in Table 3. The performance of the random strategy against three other random players is also included, which can be viewed as a baseline for the comparison of other strategies. From this table, it can be observed that all four players have an equal chance of winning (25% wins) when they all play randomly.

Table 3 Performance of each strategy against 3 random players

| Strategy | Winning Percentage | Expected No. of Moves | Margin of Victory |
|----------|--------------------|-----------------------|------------------|
| Random   | 25 ± 3 %           | 103                   | 24 %             |
| Fast     | 51 ± 3 %           | 78                    | 41 %             |
| Balanced | 57 ± 3 %           | 71                    | 42 %             |
| Aggressive | 76 ± 3 %        | 64                    | 62 %             |
| Defensive | 83 ± 3 %           | 62                    | 67 %             |

The fast strategy (51% wins) and the balanced strategy (57% wins) performed considerably better than random players, despite having no knowledge of the opponents’ pieces on the board. As expected, the aggressive strategy (76% wins) and the defensive strategy (83% wins) displayed comparatively better results by considering more detailed knowledge of the game, such as hits, attacks, and threats. Figure 4 gives a graphical view of test results. It can be observed that the defensive strategy always outperformed all other strategies.
Similar results are obtained from other performance measures. The defensive strategy had the best performance and took the least number of moves (62 moves) to win against all random players. The aggressive strategy (64 moves) exhibited a relatively close performance. The balanced strategy (71 moves) followed by the fast strategy (78 moves) took more number of moves on average to finish games. Nevertheless, it was a great step-up as compared to a random player (103 moves). Figure 5 illustrates the expected number of moves required by each strategy for winning games against three random players. For the sake of comparison, it may be noted that the theoretical minimum expected number of moves to finish a typical Cowry game was calculated in our previous work [11] to be around 38 moves.

Figure 6 shows the performance of different strategies regarding the margin of victory. The large margin of victory associated with the defensive (67%) and aggressive (62%) strategies reveals their overwhelming advantage over random players. In the majority of the games, these strategies achieved glorious victories while most of the random players’ pieces were not even able to reach the inner squares. The balanced (42%) and fast (41%) strategies performed almost equally in this case. Although these strategies experienced comparatively lower levels of victory, their performances against random players were still satisfactory.
4.2.2. Performance of All Pure Strategies Competing against Each Other

To determine the rank of several competitors in a two-player game like Chess, the fairest method is perhaps a “round-robin” tournament. Each player plays every other player, and the final rank is decided by the score. In a four-player Cowry game, however, we can directly determine the rank of four players, from strongest to weakest, by allowing them to compete against each other in the same game.

Therefore, in the second phase of experiments, we tested the relative performances of different strategies in a large number of four-player games, when each player was following one of the four proposed pure strategies. We examined how many times each strategy could win against the other three strategies, and calculated its winning percentage. Table 4 and Figure 7 highlight the obtained test results.

**Table 4** Performance of all strategies against each other in the same game

| Strategy   | Fast      | Balanced  | Aggressive | Defensive |
|------------|-----------|-----------|------------|-----------|
| Winning Percentage | 13.4 ± 1 % | 15.7 ± 1 % | 33.1 ± 1 % | 37.8 ± 1 % |

**Figure 6** Margin of victory for each strategy against 3 random players

**Figure 7** Winning percentage of all strategies against each other in the same game
It can be observed that the defensive strategy (37.8% wins) once again managed to outperform other strategies, and the aggressive strategy (33.1% wins) performed considerably better than the balanced (15.7% wins) and fast (13.4% wins) strategies. These results clearly substantiate the comparative ordering of the strategies outlined in the previous phase:

\[ \text{Defensive} \succ \text{Aggressive} \succ \text{Balanced} \succ \text{Fast} \succ \text{Random} \]

Where the operator \( \succ \) denotes a higher performance.

4.2.3. Performance of Each Pure Strategy against Human Players

We conducted the final phase of experiments between each one of the pure strategies and a novice human player. We state the obtained results in Table 5. We found that all pure strategies performed poorly and the human players defeated each strategy at least 97% of the times. This test indicates that the proposed pure strategies are certainly no match for a human opponent.

Table 5 Performance of strategies against human players

| Strategy   | Fast | Balanced | Aggressive | Defensive |
|------------|------|----------|------------|-----------|
| Winning Percentage | 0.4 ± 1 % | 0.7 ± 1 % | 1.5 ± 1 % | 2.9 ± 1 % |

By observing a few game runs, the human players were able to somewhat identify the playstyle of the computer, and hence predict its future moves to some extent. This clearly explains the utility of hybrid strategies, since a hybrid strategy aims to randomize the piece selection process by changing the player’s strategy at different stages of the game.

5. CONCLUSIONS AND FUTURE WORK

In this research, we presented a brief overview of Cowry game and recapitulated the design of two basic pure strategies (fast and balanced) that we had developed in our earlier work. Accordingly, we proposed and analyzed two advanced playing strategies (aggressive and defensive), and described the fundamental idea behind each of them. Furthermore, we provided an in-depth report of how these strategies were implemented using simple decision trees. In order to assess the strengths and weaknesses of different strategies, we proposed a few performance measures. We then conducted various types of experiments to evaluate the individual performance of each pure strategy against three random players, as well as their relative performances competing against each other in a game.

We observed that all pure strategies consistently outperformed the random strategy. The defensive and aggressive strategies utilized more knowledge of the game, and hence performed significantly better than the balanced and fast strategies. Our experiments, however, revealed that none of the proposed pure strategies alone could gain any notable success against human players. This clearly justifies the requirement for using different combinations of pure strategies at different stages of the game, which can give rise to a hybrid strategy.

While this paper merely explored and evaluated the pure strategies, formulation and analysis of different hybrid strategies might be a suitable topic for future work. The approach introduced in [4] demonstrates how this can be achieved by dividing the game into several phases, such as the opening, the middle game, and the end game. Accordingly, different hybrid strategies should be devised, due to the changes in strategic aims of different phases.

It is possible, in principle, to improve the gameplay by employing more sophisticated decision-making techniques rather than decision trees. Hence, further research can be
conducted to construct more complex evaluation functions, which assign weights to different factors involved in a strategy. The performance of the strategy then can be controlled by altering these weights [4, 8]. Moreover, the main weakness of our current design is that the machine will not learn from the previous games, and hence cannot improve its play. Perhaps the most efficient method to overcome this drawback is to apply a self-play learning approach, which changes the factors and weights of the evaluation function based on the statistics of games the machine has played.

REFERENCES

[1] R. Oka, The Real World Computing Program, Artificial Intelligence Review, 8, 1994, pp. 105-111.
[2] Franz Hahn, Evaluating Heuristics in the Game Risk: An Artificial Intelligence Perspective, B.Sc. Thesis, Maastricht Univ., Netherlands, 2010.
[3] Robert A. Hearn, Erik D. Demaine, Games, Puzzles, and Computation, Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, 2006.
[4] Claude E. Shannon, Programming a Computer for Playing Chess, Philosophical Magazine, 41(314), 1950, pp. 265-275.
[5] Alan Turing, Computing Machinery and Intelligence, Mind, LIX (236), 1950, pp. 433-460.
[6] Arthur L. Samuel, Some Studies in Machine Learning Using the Game of Checkers, IBM Journal of Research and Development, 3, 1959, pp. 210-229.
[7] Pouyan Davoudian and P. Nagabhushan, Machine as One Player in Indian Cowry Board Game: Basic Playing Strategies, International Journal of Computer Engineering and Technology, 10(1), 2019, pp. 174-183.
[8] Faisal Alvi, Moataz Ahmed, Complexity Analysis and Playing Strategies for Ludo and its Variant Race Games, IEEE Conference on Computational Intelligence and Games (CIG’11), 2011.
[9] I. Millington, J. Funge, Artificial Intelligence for Games, 2nd Ed., Morgan Kaufmann Publishers, San Francisco, USA, 2009, pp. 667-670.
[10] Rensink A. et al., Ludo: A Case Study for Graph Transformation Tools, Applications of Graph Transformations with Industrial Relevance (AGTIVE’07), Lecture Notes in Computer Science, Springer Verlag, 5088, Berlin, Heidelberg, 2008, pp. 493-513.
[11] P. Nagabhushan, Pouyan Davoudian, Machine as One Player in Indian Cowry Board Game: Strategies and Analysis of Randomness Model for Playing, International Journal on Recent and Innovation Trends in Computing and Communication, 5(2), 2017, pp. 57-67.