Evaluating Artificial Intelligence Search Algorithms on Shiva Dara Game.

***Yahaya Saidu[1], Siyani Ezra Dogo[2], Anas I. Usman[3], John S. Wejin[4] and Tirmizi Mohammed[5]**
[1, 2, 3, 4, 5] Department of Mathematical Sciences, Taraba State University, Jalingo, Nigeria
*yahayasaidu17@tsuniversity.edu.ng)

ARTICLE INFO
History:
Received: 1st May, 2022
Accepted: 5th June, 2022
Received in revised form:
10th July, 2022
Published: 28th August, 2022

Keywords

Artificial intelligence, combinatorial game, minimax, greedy, Dara.

ABSTRACT
In the 21st Century, the role of Artificial Intelligence (AI) has increased rapidly in the gaming industry. Popular games, such as Chess, Tic-Tac-Toe, and Go, have already been formulated as search problems, and consequently, many research works have been conducted concerning these games with numerous AI search algorithms being used as the underlying search algorithm for solving such games. However, despite the popularity of the ancient *Dara* game within African communities and its two distinct actions of Go (positioning) and Chess (movement) the game attracts less research effort and seems not formulated as an AI Search problem; as such, no search algorithm was evaluated using *Dara* game. In this paper, a problem formulation of *DARA* as a formal AI Game defined by five tuples is proposed. Minimax and greedy algorithms with the same heuristic function and generic search patterns were used to develop agents that play the proposed formulation (game). The agents were set to play against one another to experiment with the performance of the underlying search algorithms. The result shows that the proposed game is biased toward the opponent player, and the greedy-based agent defeated the minimax agent. This proves that AI search problems can be effectively modeled using the DARA game.

1.0 Introduction

Generally, prior to attaining a game solution (goal), there is a need for formal formulation of the game, which may be based on theoretical computer science, graph model [1], mathematical formulation, etc. Formulation of a game as an AI search problem requires explicitly defining five tuples which are initial state of the game, actions, transitional model, final state, and cost path [2]. The yet to be formulated AI Search Problem Dara game is a combinatorial game that falls under the Shiva family of games. Members of this game family have a common characteristic of being board games, with two players aiming at having three of their playing pieces arranged orthogonally. Dara varies from the members of the family in the sense that the game ends when any of the two players' captures ten opponent's pieces or blocks all opponent's liberties, whereas the other games of the family end with first three-in-row, just like in Tic-Tac-Toe [3] and movable variant of Tic-Tac-Toe [4].

The name Dara is also used synonymously to Awale game that falls under Mancala family of games, Mancala is a generic name given to 2 X N board games, mostly played in African countries, with sowing as the basic action [5, 6]. In contrast, the Shiva variant of Dara is played on 5 X 6 boards with two distinct basic actions of the game of Go (positioning), and Chess (movement) [7]. There are number of AI-related works for both game families. To be more specific, Mancala variant of Dara (Awale) tends to be ahead of the Shiva variant, with two noted works devoted to Awale as in [5, 6]. Both researchers used the same Dara of the Mancala family. However, the Dara adopted for this research belongs to the Shiva family, which differs from that of the Mancala family. The game is similar to the movable variant of the tic-tac-toe and seems to be an extension of the movable Tic-tac-toe in size [4]. The research work tends to formulate the traditional game of Dara (Shiva variant) to serve as an additional test-bed for AI techniques.

Monte Carlo Tree Search (MCTS), Minimax, and greedy are some of the most common generic search algorithms in evolving with AI agents for playing games, the techniques have been used across different types of games. Minimax was noted to be the underlying search algorithm used in developing Deep Blue, which mark the first computer program to master the Chess game [7, 8]. Recently, literature shows how MCTS has been used to evolve with playing agents for different categories of games, with AlphaGo being the latest breakthrough. Consequently, agents developed are mostly used to evaluate the performance of their underlying search algorithms or AI techniques. MCTS, Minimax algorithms and greedy had been evaluated using Tic-tac-toe, Go, Chess [9, 10], and Blind Chess in the work of Ciancarini and Favini [8] but yet to be evaluated using Dara Shiva variant. Even though the techniques performed well with varying performances in popular and common games like Chess and Go, there are several recorded defeats by other techniques when it comes to playing less popular games that might be more complex than the popular games [11]. Moreover, these techniques are rarely evaluated using unpopular traditional games.

Hence, this research work tends to evaluate the performance of minimax and greedy search algorithms using the proposed model (game of Dara).

2.0 Related Work

AI Research work in games takes many perspectives, some researchers devote their efforts to formulating new games [5, 7, 12], and AI agents for the games [13]. Some research works use existing games to test for new setups of AI techniques [8, 13, 14], while another perspective is enhancing the existing AI techniques which can be depicted in the subsequent efforts of improving AlphaGo to AlphaZero [7].

A type of game that attracts much research effort with a basis on mathematics are the two players, zero-sum, perfect information, turn-based board games, and the popular games of chess, checkers, Tic-tac-toe, and Go fall under this category. The games were also characterized to be turn-based games; with players alternating moves. The research by Garg *et al.* [3] characterized Tic-tac-toe as a solved game due to its small board size, with a known optimal strategy for winning the game. AI agents capable of defeating humans in all four aforementioned games have been achieved, but the gap is now for enhancing the existing agents [3, 7]. Related works with respect to some popular combinatorial games reviewed form the subsequent paragraphs.

Go

Go is an ancient game that originated in China 3000 years back, played on a standard board of size 18 rows × 18 columns giving rise to 19 × 19 intersections. The game starts with an empty board, with two players alternatively dropping (positioning) pieces until both players exhaust their pieces, or either player has given up. The goal of players in Go is to capture opponent piece(s) by surrounding the piece(s) or to conquer a large territory of the board. The size of the Go board contributes to its complexity and paves way for being the master challenge after chess has been solved [15]. Board with smaller size exists like 15 × 15, which was used for playing Gomoku games as reflected in [16]. Gomoku games are played with the same rule as Go in which two players alternate dropping of playing pieces, but the goal for the games differ in that players in Gomoku aspire to have five pieces in a row, column or diagonal, and is theoretically found to favor the first player. The first attempt to design artificial intelligence capable of playing Go was in 1970. Several Go-playing agents have evolved with AlphaGo of Deep Mind in 2016 becoming a notable breakthrough in the world of AI. The work expired the speculation that no computer program can defeat humans in playing the game of Go when the world champion in the person of Lee Sedol was defeated by AlphaGo [17].

Chess

Chess is an ancient combinatorial game, which has received great popularity among game developers and researchers. The chess board has 8 × 8 squares, with initially positioned pieces in the first two and last two rows, the players alternate turn by moving playing pieces to checkmate the opponent's king player. Pawn, queen, king, bishops are some labels for playing pieces in chess, and special privileges are given to some playing pieces [7]. Research in chess span many variants like Japanese chess (shogi). Shogi is played on a similar chess board, but more complex than chess in terms of computational complexity. It is played on a larger board, and any captured opponent piece changes sides, and may subsequently be dropped anywhere on the board [7]. Kerspgiel is a variant of chess popularly known as blind chess; the game is played on a board of chess with players having partial information about the current state. Players will only know the location of the playing pieces on the board and will blindly try to move a piece, and a message from a referee will determine legal moves for players [8]. After much research effort on the game of chess, a milestone was achieved in 1996 when an intelligent agent named Deep Blue was able to defeat the human champion player Garry Kasparov in a chess match. Subsequent works with increasing successes were conducted, with Stockfish being the state-of-art chess-specific AI playing agent, which was defeated by domain-independent agent AlphaZero as presented in Silver *et al.* [7].

Tic-Tac-Toe

Tictactoe is a trivial combinatorial game, played on a 3 × 3 playing space by two players alternating turns. The game is simple with a search complexity of 9! The goal of players is to arrange three of their playing pieces orthogonally or diagonally, the first player to achieve the goal wins the game and the game terminates. Sometimes all the nine cells may be occupied without a winner, and in that case, the game ends in a tie (draw) [3, 13]. The movable Tic-tac-toe is been played on the same board as a normal Tic-tac-toe and the game starts with an empty board. Each of the players is given three playing pieces which are positioned on the board. Alternatively, players have three in a row, column, or diagonal while positioning. There will be three empty cells for the players to move their pieces alternatively after positioning six pieces on the board, and each player aims at arranging all his pieces in a row, column, or diagonal. Movable Tic-tac-toe ends when either player wins the game, that is to say, there is no draw [4]. The next variant is the imperfect blind Tictactoe (Phantom Tictactoe), this variant is an imperfect information game, with the abstraction of the opponent's board status from each other. The players observe the rules of playing vanilla Tic-tac-toe only that the game is played blindly. If a player tries to place his stone in a position that is occupied by the

opponent's stone the player learns this information and plays again [18, 19]. Generally, algorithms for always winning Tictactoe or having a draw at the end of the game were discovered, hence the game of Tictactoe is considered a solved game.

Mancala Games

An online-published dissertation for Master graduate harmonized the game naming convention; that game-to-name relationship is many to many relationships, the work also discussed five members of the Mancala game family and serves as an important source for this section. All five games were characterized to be board games on two, three, or four rows of n columns with n ≥6. The games are common in African countries and some parts of the Middle East.

Wari is a Mancala game played on a 2 × 6 rectangular board with a store at either end. Initially, each of the pits fills with four pieces, and two players alternate turn with sowing as their basic actions. Awari adheres to the same rules as Wari but is formulated with few restrictions to evolve with a computerized version of Wari, the credit for this contribution goes to Herik as cited by Rovaris [20]. Oware was also a variant of Wari whose name was rooted in the work of Rovaris [20], the computerized version was implemented by Rovaris. Another variant played in India is Vai Lung Thlan, with initially five pieces in each of the 2 × 6 pits. The game ends when a player captures all the pieces of another player, or when a player has thirty pieces in his store. Ovalhu also refers to Dakun played on a 2 × 8 board initially filled with eight pieces per pit. The last mancala considered in the work of Rovaris [20] is Kalah, which is composed of a 2 × 6 board initially configured with four pieces per pit. There are two types of capture in Kalah; Place Capture and Store Capture.

The main differences among the five aforementioned games arise from the rules of playing each game. Chetachi and Nwachukwu [6] presented another Mancala game named Awale with the same board configuration as Kalah presented by Rovaris [20]; the same game named Dara by Ndukwe and Nwulu [5].

3.0 Proposed Model for Dara Game

Dara game can be modeled as an AI search problem, which is made up of a set of five tuples listed below [2].

- S_0 is the initial state of the game
- $\sum$ is the set of operators/actions
- ∂ is the transition function between states (QX$\sum$)
- F is the set of final states
- C is the path cost.

A detailed description of the above Model for the DARA game can be found in a paper titled On the Formulation of Shiva Dara Game as a Test-bed for Evaluating AI Search Algorithms [21].

4.0 Implementation of AI Agents for Playing Dara Game

Minimax and greedy are the search algorithms adopted and tailored to play the proposed formulated game. The model and algorithms are implemented using java (object-oriented programming (OOP) paradigm) on Netbeans IDE. We use a simple heuristic function proposed based on the common strategy for playing the game of Dara for both algorithms. Algorithms 1 and 2 represent the two algorithms.

Algorithm 1: Greedy Search Algorithm

```
FUNCTION GREEDY (board, player)
bestValue ←  -∞
choosenAction ←0
A ← availableActions(Board)
for all m in M do
      if EVALUATE(m, board, player) ≥ bestValue then
bestValue← EVALUATE (m, board);
choosenAction ← m
       end if
end for
return choosenAction
end function
FUNCTION EVALUATE (move, board, player,
opponent)
Nx← piecesForX
N0← piecesForO
       c← 1
                if (No==2 || Nx==2)
                    c=Math.log(Nx);
                end if
return 10-(pow((Nx-2),c)-(No-2));
   end function
```

5.0 Experiments

Newly developed agents are evaluated against existing agents (if any) [8, 9], against a newly developed agent [22, 23], against humans [17] or hybrid of [7]. Each of the two agents based on minimax and greedy search algorithms is set to play five games against self, then five games against human (novice), and five games against each other. Both agents play as first players as well as opponent players. The graphical presentation of the results of the experiments, and their discussions in the subsequent sections. Java programming language was used to implement the proposed model and the agents, and NetBeans IDE 8.0.2 is the IDE used for this experiment. In all the experiments, a DELL laptop of 64 GHz Intel Processor, with 4GB of RAM and 500 G Hard Disk was used.

Algorithm 2: Minimax Search Algorithm

```
FUNCTION MINIMAX (node, player, depth)
if node is terminal OR depth = 0
      return EVALUATE (board, player)
end if
if TURNPLAYER(player) = player
bestValue ← -∞
children ←  CHILDREN(board)
for all child in children do
val←  MINIMAX(child, player, depth-1)
bestValue ←  MAX(bestValue, val)
end for
return bestValue
else
bestValue ← +∞
children ←  CHILDREN(board)
for all child in children do
val←  MINIMAX(child, player, depth-1)
bestValue ←  MIN(bestValue, val)
end for
return bestValue
end if
end function

FUNCTION CHILDREN (board)
M ←AVAILABLEACTION(board)
      for all m in M do
        m← DOACTIONS (board,m)
       add child to Children
        end for
return children
end function

FUNCTION EVALUATE (move, board, player,
opponent)
Nx← piecesForX,  N0← piecesForO,   c← 1
         if (No==2 || Nx==2)
                    c=Math.log(Nx);
                end if
return 10-(pow((Nx-2),c)-(No-2));
    end function
```

3.0 Results and Discussion

This section comprises the results for the tournament of games of the agents (based on the greedy search algorithms discussed in the previous section) with one another using the *Dara* game. In addition, the results from the played matches against self for each of the agents are also presented.

3.1 Greedy Based Agent

In this section, the results of Greedy based agents as first players, playing the game against other evolved agents and against human players are presented.

Greedy Versus Greedy

The first experiment is a match between two players, each applying the greedy algorithm, five (5) games were played and the results of the five matches are given in Table 1, the experiment find out if the game is biased towards a specific player. Figure 1 provides the graphical representation of the results in Table 1.

Table 1: Greedy versus Greedy

	No. of X pieces	**No. of O pieces**	**Number of turns**	**Time**
Match 1	2	4	24	4
Match 2	2	4	24	4
Match 3	2	4	24	2
Match 4	2	4	24	1
Match 5	2	4	24	3

Greedy Versus Minimax

The second experiment was between Greedy (first player) against the minimax agent, for the minimax a depth level of three for searching was used in the game tree. Table 2 represents the results of five matches played. Note: Figure 2 is a graphical representation of Table 2.

Table 2: Greedy Versus Minimax

	Greedy (X) pieces	**Minimax (O) pieces**	**Number of turns**	**Time**
Match 1	8	2	11	10
Match 2	8	2	11	10
Match 3	8	2	11	09
Match 4	8	2	11	10
Match 5	8	2	11	08

Greedy Versus Human (Novice)

A novice player, who was introduced to the game for the first time, was set to be the opponent player for the five matches and the results are presented in table 3. Figure 3 provides the graphical representation of the results in Table 3. The x-axis represents the matches played and the y-axis represents the counts for pieces and time.

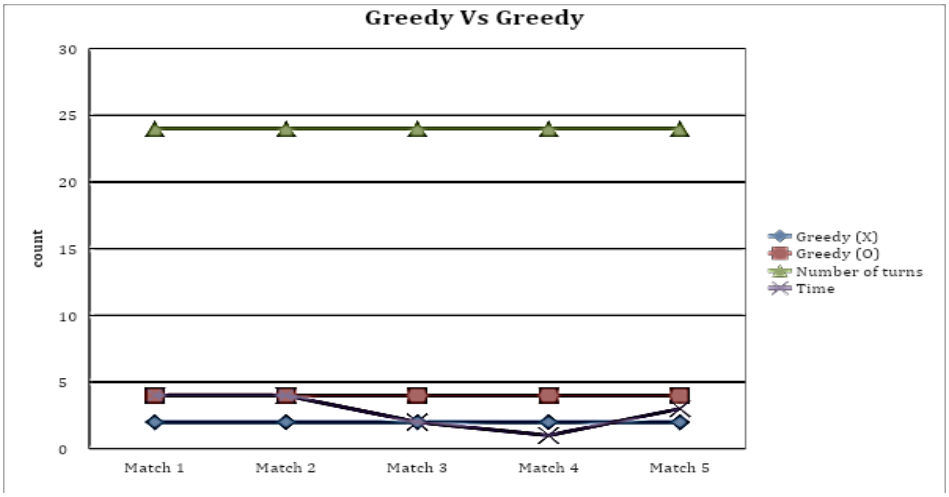

Figure 1. Greedy Vs Greedy

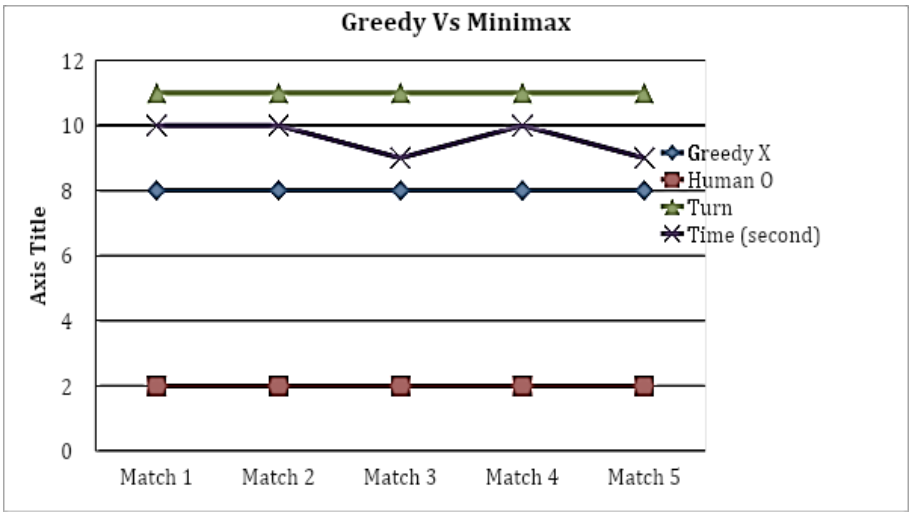

Figure 2. Greedy (X) Vs Minimax

Table 3. Greedy versus Human (Novice)

	No. of X pieces	No. of O pieces	No. of turns	Time
Match 1	8	2	27	20
Match 2	5	2	21	18
Match 3	2	6	11	25
Match 4	3	2	18	10
Match 5	4	2	21	22

3.2 Minimax Based Agent

Minimax with depth level 3 was set to play against the evolved agents.

Minimax Versus Minimax

The first experiment was a match between two players, each adopting a minimax algorithm, five (5) games were played and the results of the five matches is given in Table 4, experiment was intended to find out if the game is biased towards a specific player while adopting minimax as an underlying search algorithm.

Table 4. Minimax versus Minimax

	No. of X pieces	No. of O pieces	No. of turns	Time
Match 1	2	5	26	18
Match 2	2	5	26	15
Match 3	2	5	26	18
Match 4	2	5	26	16
Match 5	2	5	26	15

The result in Table 4 shows a dominant winning for the opponent player over the first player. The game is biased toward the opponent player, that is to say, the game favors the opponent player when both players apply the minimax algorithm. There are constant twenty-six turns for each player before the terminal state. A graphical representation of the table above is given in Figure 4.

The constant number of turns (25 turns) compared with the number of turns for match 3 of table 4 shows the need for improvement to achieve an optimal solution.

Minimax Versus Greedy

The second experiment was between minimax and greedy based agent, where the minimax agent goes first and the greedy-based agent played second. Table 5 represents the result of five matches. The Table shows a constant winning by the opponent player (greedy) with improved success compared with matches in Table 2 where greedy plays first. This shows that the game favors the opponent player. Figure 5 represents the results in Table 5.

Table 5. Minimax versus Greedy

	Minimax X	Greedy O	Turn	Time for the Match
Match 1	2	10	12	9
Match 2	2	10	12	10
Match 3	2	10	12	10
Match 4	2	10	12	12
Match 5	2	10	12	09

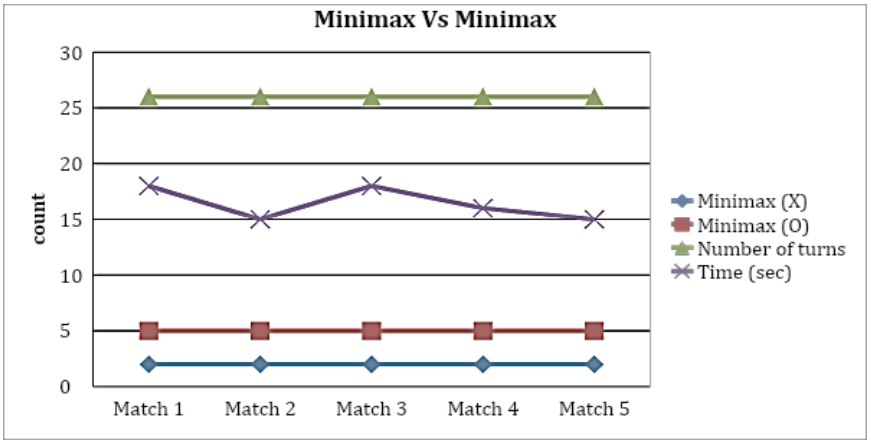

Figure 4. Constant Winning of Minimax (O) Player Minimax First (X) Player.

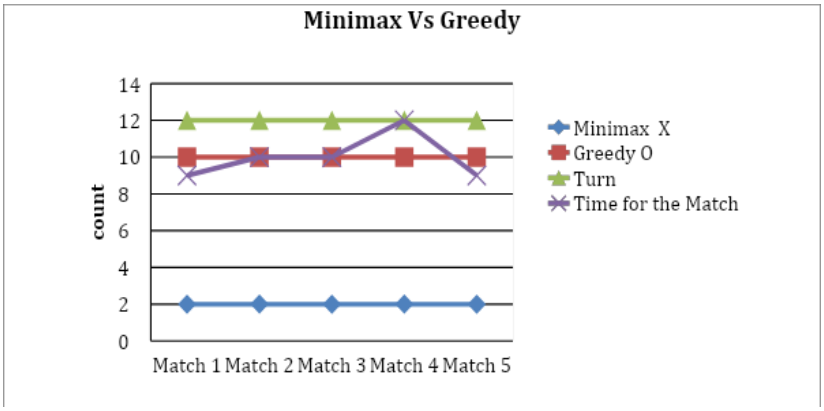

Figure 5. Minimax Vs Greedy

In presentation, the results from the tournament showed that for both players employing greedy or minimax search algorithm, the game tends to be biased toward the opponent player and this can further be depicted in the improvement of the number of pieces captured by minimax as an opponent player (Table 2) compared with the result obtained as the first player (Table 5) .

The most optimal match is noted when the game is played against humans, with a human (expert) agent winning the game in eleven (11) turns, which shows none of the evolved agents is as optimal as a human (intermediate) player in the game of Dara. However, the speed at which they both played the game is convincing compared with the speed of humans at all levels of expertise. Results presented also showed that while in self-play mode greedy-based agents find the more optimal solution (24 turns) than a minimax-based agent (26 turns), even though minimax-based agents take a longer time to choose action. The defeat of minimax by greedy in eleven (11) turns in table 2 which is the same number of turns taken in the most optimal match against a human expert in match 3 (Table 3) did not invalidate the latter as the most optimal considering the ratio of playing pieces left at the termination of the game.

Conclusion

A proposed Model of Dara that will serve as a new environment for testing AI search techniques (Search Algorithms) has been proposed in this paper. We implement the model using JAVA programming language, with two agents based on the suggested algorithms for playing the game; in addition, an interface for the human player was developed. The agents have successfully played the game and participated in the tournament of matches. The result from the tournament shows that the game is biased toward the opponent player, with greedy base agents having the best performance. The outcome of matches against human players shows that none of the agents attained novice expertise. The game is to provide an environment for testing other AI search algorithms, and the result of the evaluation reported should serve as a benchmark for subsequent evaluations of search algorithms using the game of Dara.

The research work achieves its aim, yet some of the areas that require additional research effort. Design of a new Dara 5 × 6 Test-bed board two-step complexity game with two distinct strategic actions of Go and Chess performed sequentially, will be a good green research area. This may inspire variants of Dara with larger board sizes such as that of chess (8 × 8), connect4 (7 × 6), or Go (19 ×19). Further evaluation with other AI Search algorithms apart from greedy and minimax, or possibly their hybrid version using the implemented game (suggested search algorithms are MCTS, variants of Minimax) will be explored. Develop a GUI for the game to make it user-friendly, for fun and entertainment will be undertaken**.**

References

[1] Kim, H. *et al.* (2015). A Mathematical Formulation based on the Connectedeness of stones for the Game of Go', *ACIS International Journal of Computer and Information Science*, 16(4), pp. 1–11. Available at: www.acisinternational.org.

[2] Russell, J. S. and Norvig, P. (2010). Aritficial Intelligence A Modern Approach. Third Edit. Upper Saddle River, New Jersey 07458: Prentice Hall Series in Artificial Intelligence.

[3] Garg, S., Songara, D. & Maheshwari, S. (2017) 'The winning strategy of Tic TacToe Game model by using Theoretical Computer Science', in *2017 International Conference on Computer, Communications and Electronics, (Comptelix)*. IEEE, pp. 89–95. doi: 10.1109/COMPTELIX.2017.8003944.

[4] Therabytesapps (2014). *Tic Tac Toe Movable, Tic Tac Toe Movable| SlideMe LLC*. Available at: m.slideme.org/user/therabytesapps (Accessed: 7 June 2018).

[5] Ndukwe, I. G. & Nwulu, E. (2014). Computerized Board Game : Dara', *International Journal of Advance Research in Computer Science*, 5(7), pp. 59–62.

[6] Chetachi, A. C., & Nwachukwu, E. O. (2016) 'Research Article A Hybridize Heuristic Based Method in Evolving Mancala Game / Awale', *Journal of Scientific and Engineering Research*, 3(5), pp. 161–165. Available at: www.jsaer.com.

[7] Silver, D. *et al.* (2017) 'Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm', *aeXiv preprint arXiv:1712.01815*, pp. 1–19. doi: 10.1002/acn3.501.

[8] Ciancarini, P. & Favini, G. P. (2010) 'Monte Carlo tree search in Kriegspiel', *Elsevier Artificial Intelligence*. Elsevier B.V., 174(11), pp. 670–684. doi: 10.1016/j.artint.2010.04.017.

[9] Baier, H. & Winands, M. H. M. (2013). 'Monte-Carlo tree search and minimax hybrids with heuristic evaluation functions', in *2013 IEEE Conference on Computational Intelligence in Games (CIG)*. IEEE, pp. 1–8.

[10] Lanctot, M. *et al.* (2017) 'A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning', *Advances in Neural Information Processing Systems 30*, (Nips). Available at: http://arxiv.org/abs/1711.00832.

[11] Silver, D. *et al.* (2016) 'Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm', *aeXiv preprint arXiv:1712.01815*, pp. 1–19. doi: 10.1002/acn3.501.

[12] Silva, M. P., Silva, V. D. N. & Chaimowicz, L. (2017) 'Dynamic difficulty adjustment on MOBA games', *Entertainment Computing*.doi: 10.1016/j.entcom.2016.10.002.

[13] Ozcan, E. & Hulagu, B. (2017) 'A Simple Intelligent Agent for Playing Abalone Game : ABLA', in *Proceedings of the 13th Turkish Symposium on Artificial Intelligence and Neural Networks*, pp. 281–290.

[14] Sethy, H., Patel, A. & Padmanabhan, V. (2015). Real Time Strategy Games: A Reinforcement Learning Approach', *Procedia Computer Science*. Elsevier Masson SAS, 54, pp. 257–264. doi: 10.1016/j.procs.2015.06.030.

[15] Lee, C. (2019) 'The Game of Go: Bounded Rationality and Artificial Intelligence'.

[16] Tang, H. *et al.* (2017) 'Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning', in *NIPS 2017*. Available at: https://arxiv.org/abs/1611.04717.

[17] Hölldobler, S., & Tigunova, A. (2017). Lessons Learned from AlphaGo. *YSIP*.

[18] Cermák, J., Bošanský, B. & Pechoucek, M. (2017). Combining Incremental Strategy Generation and Branch and Bound Search for Computing Maxmin Strategies in Imperfect Recall Games', in *Workshop on Computer Poker and Imperfect Information Games at the Thirty-First AAAI Conference on Artificial Intelligence.*, pp. 902–910.

[19] Jiri, C., Branislav, B. & Viliam, L. (2017) 'An Algorithm for Constructing and Solving Imperfect Recall Abstractions of Large Extensive-Form Games', in *Proceedings of the Twenty-Six International Joint Conference on Artificial Intelligence*. AAAI Press, pp. 937–942. Available at: https://www.ijcai.org/proceedings/2017/130%0D.

[20] Rovaris, G. (2016) *Design of Artificial Intelligence for Mancala Games*. Politecnico Di Milano.

[21] AbdulHakim, I. & Umar, K. (2019). On the Formulation of Shiva Dara Game as a Testbed for Evaluating Artificial Intelligence Search Algorithms', in *Proceedings of the 4th YUMSCIC*, pp. 416–422.

[22] Justesen, N., Tobias, M. & Togelius, J. (2017) 'Online Evolution for Multi-Action Adversarial Games', in *EuropianComference on the Applications of Evolutionary Computation*. Springer. Cham, pp. 590–603. doi:https://doi.org/10.1007/978-3-319-31204-0_38.

[23] Tsividis, P. A. *et al.* (2017) 'Human Learning in Atari', in *2017 AAAI Spring Symposium Series on Science of Intelligence: Computational Principles of Natural and Artificial Intelligence*. Available at: http://gershmanlab.webfactional.com/pubs/Tsividis.