ISSN (Print): 2476-8316
ISSN (Online): 2635-3490
Dutse Journal of Pure and Applied Sciences (DUJOPAS), *Vol. 6 No. 2 June 2020*

Evaluating Artificial Intelligence Search Algorithms on Shiva Dara Game

***Kabir Umar[1], AbdulHakim Ibrahim[2]**

[1,2] Faculty of Computer Science and Information Technology,
Bayero University Kano,
Nigeria

[2] Katsina State Institute of Technology and Management,
Katsina State, Nigeria.

Email: ukabir.se@buk.edu.ng

Abstract

Popular games, such as Chess, Tic-Tac-Toe, and Go, have been formulated as search problems, and consequently many research works have been conducted concerning these games with numerous artificial intelligence (AI) search algorithms been used as the underlying search algorithms for solving such games. Despite the popularity of the ancient Daragame within African communities and its two distinct actions of Go (positioning) and Chess (movement) the game attracts less research efforts; as such no search algorithm was evaluated using the Dara game. The work adopts a mathematical formulation of Shiva Dara (Dara) using a five tuples Aisearch problem formulation model. Minimax and greedy with the same heuristic function are generic search algorithms used to develop agents that play the implemented model (game). The agents were set to play against one another to evaluatethe performance of the underlying search algorithms. The results show that the proposed model is biasedto the opponent player, and the greedy agent defeated the minimax agent and is more optimal. We consider evaluating other search algorithms or hybridization of the two algorithms in subsequent work.

Keywords: Artificial intelligence, combinatorial game, minimax, greedy.

INTRODUCTION

Generally, Before attaining game solution (goal), there is a need for formal formulation of the game, which may be based on theoretical computer science, graph model, mathematical formulation, etc.(Sato, Anada& Tsutsumi, 2015). Formulation of game as an AI search problem requires explicitly defining five tuples which are initial state of the game, actions, transitional

**Author for Correspondence*

model, final state, and cost path (AbdulHakim & Umar, 2019). Recently Dara has been formulated as an AI Search problem, which is a combinatorial game that falls under the Shiva family of games. Members of this game family have common characteristics of been 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 (Garg, Songara& Maheshwari, 2017), and movable variant of Tic-Tac-Toe (Therabytesapps, 2014).

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(Chetachi & Nwachukwu, 2016; Ndukwe & Nwulu, 2014). 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)(Silver et al., 2017). There are numbers of AI-related works to both game families, Mancala variant of Dara (Awale) tends to attract more research effort than the Shiva variant (Kabir & AbdulHakim, 2019). 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 tic-tac-toe and seems to be an extension of movable Tictactoe in size (Therabytesapps, 2014). The research work tends to evolve with the heuristic-based agents that will play the formulatedgame of Dara (Shiva variant), the game serves as test-bed to evaluate the AI search algorithms (Minimax and greedy).

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 Chess game(Ciancarini & Favini, 2010; Silver et al., 2017). 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 Tictactoe, Go, Chess (Baier & Winands, 2013; Lanctot et al., 2017), and Blind Chess in the work of Ciancarini and Favini(2010), but yet to be evaluated using Dara Shiva variant. Even though the techniques performed well with varying performance in popular and common games like Chess and Go, there are some recorded defeats by other techniques when it comes to playing less popular games that might be more complex than the popular games (Silva, Lee, Julian & Andy, 2016). 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 model (game of Dara).

RELATED WORKS CONCERNING GAMES

AI Research work in games takes many perspectives, some researchers devote their efforts to formulate new games, and AI agents for the games(Ndukwe & Nwulu, 2014; Ozcan & Hulagu, 2017; Victor & Chaimowicz, 2017). Some research works use existing games to test for the new setup of AI techniques (Ciancarini & Favini, 2010; Ozcan& Hulagu, 2017; Sethy, Patel& Padmanabhan, 2015), another perspective is enhancing the existing AI techniques which was depicted in the subsequent efforts of improving AlphaGo to AlphaZero (Silver et al., 2017).

The category of games that attracts much research effort with the basis from mathematics are the two players, zero-sum, perfect information, turn-based board games, the popular games of chess, checkers, Tictactoe and Go fall under this category. Garg et al. (2017) characterized Tictactoe 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 the four aforementioned games have been achieved, the gap is now for enhancing the existing agents(Garg et al., 2017; Silver et al., 2017). Related works concerning some popular combinatorial games reviewed in the subsequent paragraphs.

- **Go**

Go is an ancient combinatorial game originated from china 3000 years back, played on a standard board of size 18 rows X 18 columns giving rise to 19 X 19 intersections. The game starts with an empty board, to capture opponent piece(s). The size of the Go board contributes to its complexity, and paves for being the master challenge after chess has been solved (Lee, 2019). A board with a smaller size exists like 15 X 15, which was used for playing Gomoku games as reflected in(Tang et al., 2017). Gomoku games are played with the same rules as Go, but the goal for the games differ in that player 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, many Go-playing agents evolved with AlphaGo of Deep Mind in 2016 been the 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 person of Lee Sedol was defeated by AlphaGo(Oh et al., 2017).

- **Chess**

Chess is the ancient combinatorial game, which has been used for illustration in some popular AI textbooks, with muchresearch works devoted to this game. The board of chess has 8 X 8 squares, with initially positioned pieces in the first two and last two rows, players aim to checkmate the opponent's king player. Spawn, queen, king, bishops are some labels for playing pieces in chess, and special privileges are given to some playing pieces (Kowalski & Kisielewicz, 2017). Research in chess span some variants like Japanese chess (shogi)(Silver et al., 2017), Kerspgiel is a variant of chess popularly known as blind chess(Ciancarini & Favini, 2010). After much research effort on the game of chess, a milestone was achieved in 1996 where an intelligent agent named Deep Blue was able to defeat 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(Silver et al., 2017)

- **Tictactoe**

Tictactoe is a trivial combinatorial game, played on a 3 X 3 playing. 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 movable variant of Tictactoe is been played on the same board with normal Tictactoe and the game starts with an empty board, each of the players is given three playing pieces, which are positioned on the board alternatively. The game may progress to the movement phase where players move their pieces. Tictactoe ends when either player achieves the goal(Therabytesapps, 2014). Phantom Tictactoe is ablind variant of tic-tac-toethat has an abstraction of the opponent's board status from each other. The players observe rules of playing vanilla Tictactoe only that the game is played blindly (Cermák, Bošanský& Pechoucek, 2017; Jiri, Branislav& Viliam, 2017). Generally, algorithms for winning Tictactoe optimally were obtained; hence the game of Tictactoe is considered a solved game.

PROPOSED MODEL FOR DARA GAME

This study sought to evaluate two AI search algorithms usingthe adapted model of *Dara* from the work of AbdulHakim and Umar (2019). However,there are some preprocesses that needed to be prepared. The study process started with adapting a game model, transforming the adapted model into a computer model, implementation of greedy and minimax search-based agents, and evaluation of the implemented agents. The phases of activities for this study are depicted diagrammatically in Figure 1.

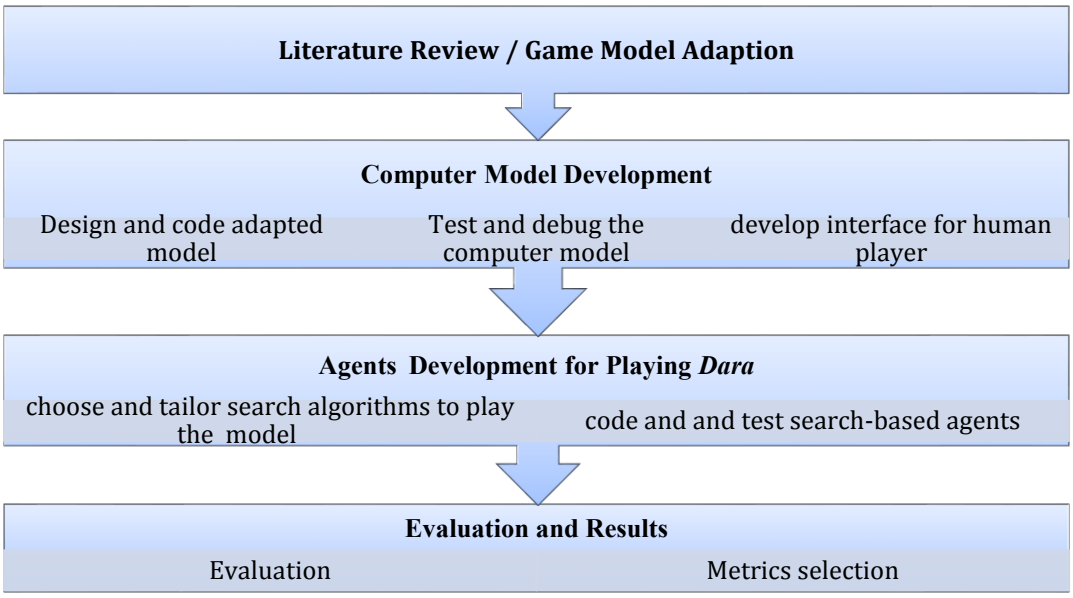

Figure 1: Overview of Proposed Research Methodology

Literature Review and Game Model Adaption

This is the initial step in conducting this study. The main purpose of this phase is to have an in-depth understanding of the problem domain (artificial intelligence in combinatorial games). Dara game can be modeled as an AI search problem, which is made up of a set of five tuples listed below(Russell & Norvig, 2010).

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

The phase also provides state-of-the-art in Dara, with the recent analytical model in the work ofAbdulHakim and Umar (2019) adapted for this study.

Computer Model Development

During this phase, an object-oriented software development paradigm was used to transform the adopted model into a computer model using Java programming language. Methods and attributes of classes were used to represent the game's characteristics, rules, states, transition, etc.

The computer model was verified to ensure the computer model is an accurate replica of the analytical model. The model was validated to ensure conformance with the real world game via human player interface incorporated into the game. Bugs were identified and fixed accordingly.

Agents Development for Playing Dara

Minimax and greedy are the search algorithms adopted and tailored to play the model. The agents were implemented using java (OOP) on Netbeans IDE, a simple heuristic function as in the adapted model was used for both agents. Algorithms 1 and 2 represent the two underlying search algorithms for the agents. The agents were tested by playing against a human player.

Algorithm 1: Greedy Search Algorithm

1. start
2. Set bestValue = - -∞
3. Set choosenAction = empty
4. fetch all the available actions from the board
5. repeat through step X for each of the available action
6. if the score of simulating an action on the board >= bestValue
    - update bestValue, choosenAction with current score and action respectively.
7. return choosenAction for agent to execute.
8. End

Algorithm 2: Minimax Search Algorithm

1. start
2. if current state is terminal state or depth of the tree is 0,
    a. evaluate state
    b. return score
3. if the player is the first player
    a. set bestScore to least value (--∞)
    b. create a list of successive states from current state
    c. for each member of a list of successive states repeat step i through iii
        i. reduce the depth of the tree by 1
        ii. set value to the return value from recursive call to the algorithm
        iii. update bestScore = MAX (bestScore, value)
    d. return bestValue
4. else // opponent player
    a. set bestScore to peak (+∞)
    b. create a list of successive state from current state
    c. for each member of the above list repeat step i through iii
        i. reduce the depth of the tree by 1
        ii. set value to the return value from recursive call to the algorithm
        iii. update bestScore = MIN (bestScore, value)
    d. return bestValue
5. finish

Evaluation

Newly developed agents are evaluated against existing agents (if any) (Baier & Winands, 2013; Ciancarini & Favini, 2010), against newly developed agent (Justesen, Tobias & Togelius, 2017;) against Human or hybrid (Silver et al., 2017). Both agents based on minimax and greedy, search algorithms are set to play five games against each other. This study evaluates greedy and minimax in terms of space complexity(number of turns) and optimality (win count).

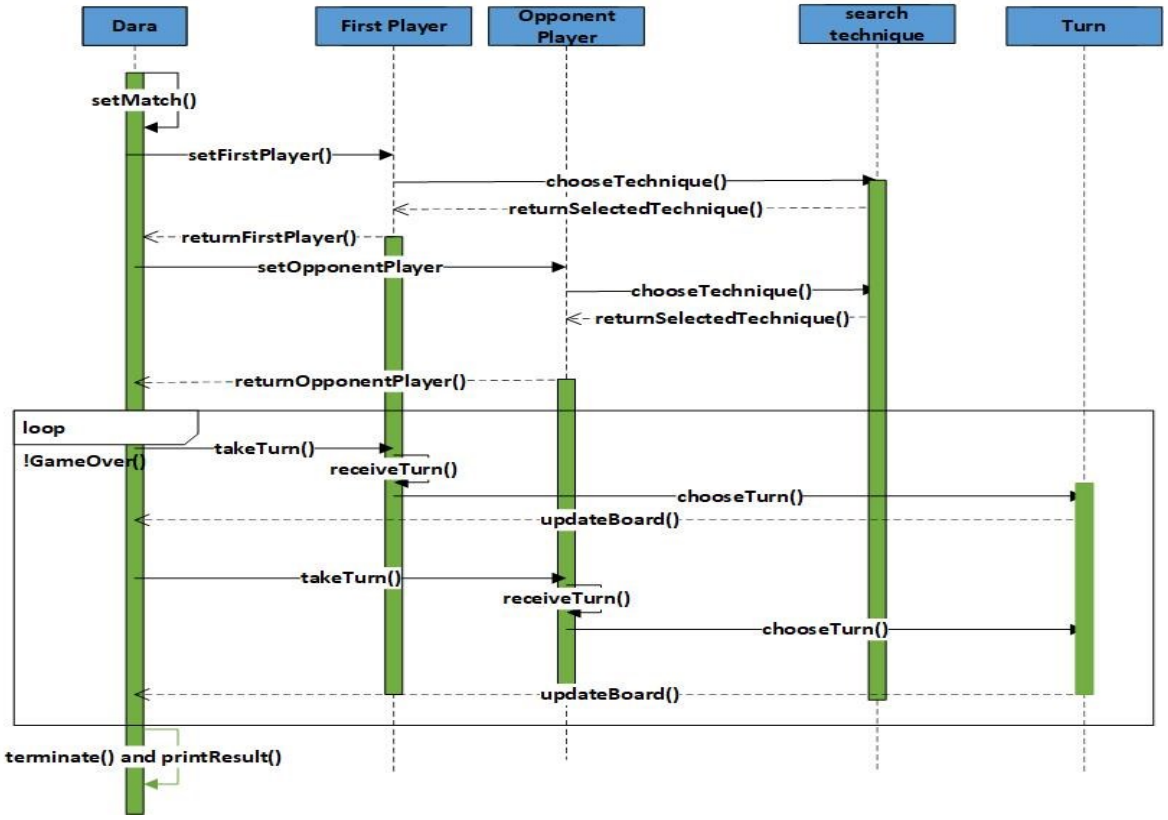

Figure 2: System Sequence Diagram

RESULTS AND DISCUSSION

This section comprises the results for the tournament of games of the agents with one another using the *Dara* game. Besides, we present the results of matches against self for each of the agents and discussion.

Results

The results of the matches are given in subsequent tables. X pieces and O pieces represent the number of pieces for the first and second playersat the end of the match respectively. Player with more than two pieces at the end of a match is the winner.

A. GREEDY BASED AGENT

In this section, we report the results of Greedy based agent as the first player, playing the game against other players.

Greedy Versus Greedy

The first experiment is a match between two players, both applying the greedy algorithm, five (5) games were played and the results of five matches is given in table 1.

Table 1: Greedy versus Greedy

	X pieces	O pieces	Turns	Time (s)
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

Table 1 shows constant winning of opponent player with four pieces against two pieces at a cost of twenty four (24) turns. Figure 3 provides a graphical representation of the results in table 1.

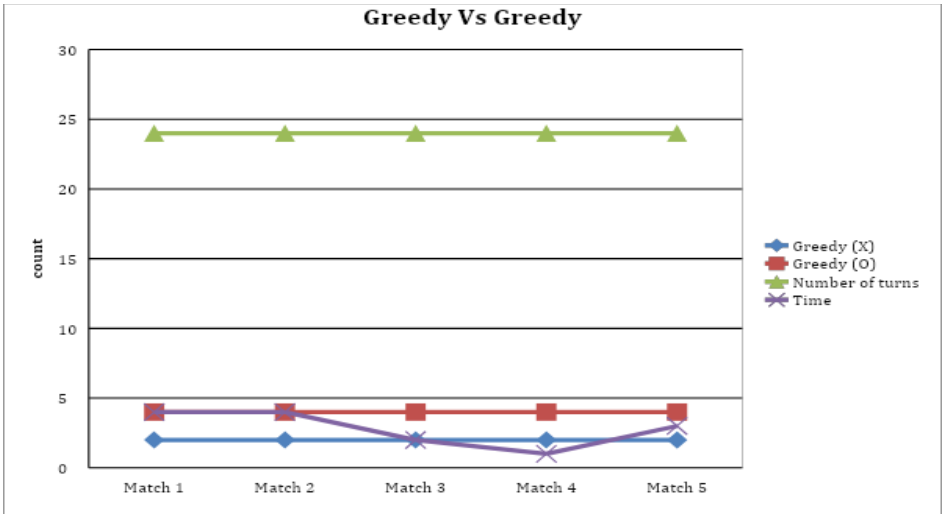

Figure 3: Greedy (X) Vs. Greedy (O)

Greedy Versus Minimax

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

Table 2: Greedy Versus Minimax

	X pieces	O pieces	Turns	Time(s)
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

Table 2 shows a constant winning of the first player with eight pieces against the opponent player in average of eleven turns. Figure 4 is a graphical representation of table 2.

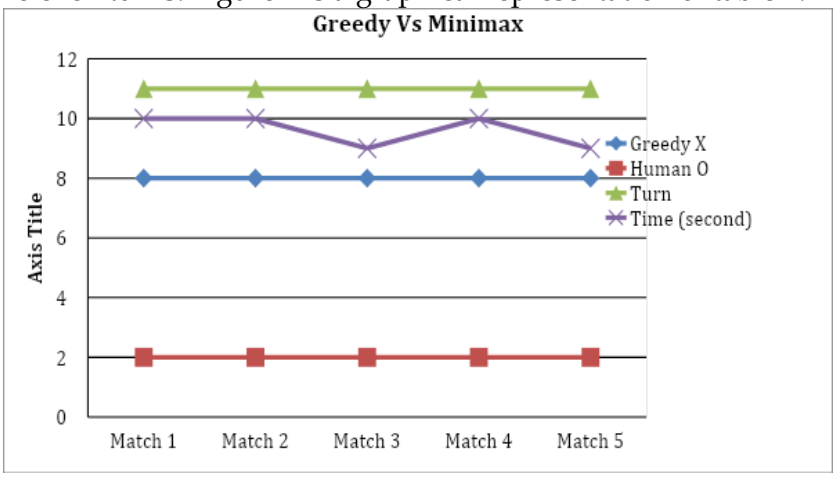

Figure 4: Greedy (X) Vs.Minimax (O)

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.

Table 3: Greedy versus Human (Novice)

	X pieces	O pieces	Turns	Time (s)
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

Table 3 shows a 4-1score with X player having four wins and opponent player one win, each much has a varying number of turns. Figure 5 is a graphical representation of table 3.

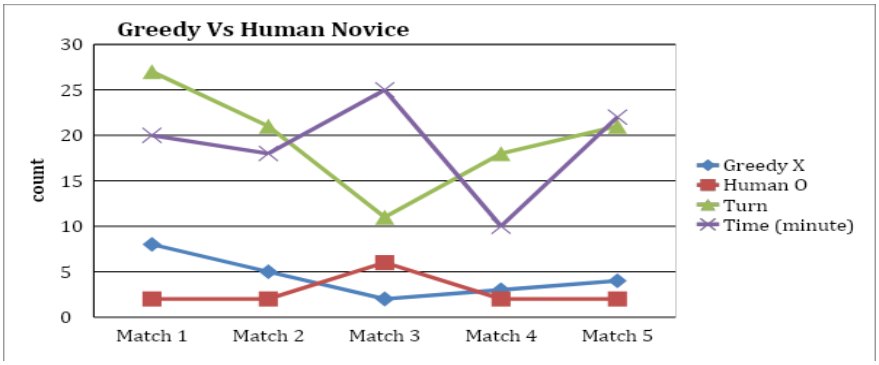

Figure 5: Greedy (X) Vs. Human (O)

MINIMAX BASED AGENT

Minimax with depth level 3 was set to play against the evolved agents; we recorded the results in the tables below.

I. Minimax Versus Minimax

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

Table 4: Minimax versus Minimax

	X pieces	O pieces	Turns	Time (s)
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 at 26 turns. The result obtained from the table shows that the game is biased to the opponent player, when both players apply the minimax algorithm. The graphical representation of the table above is given in Figure 6.

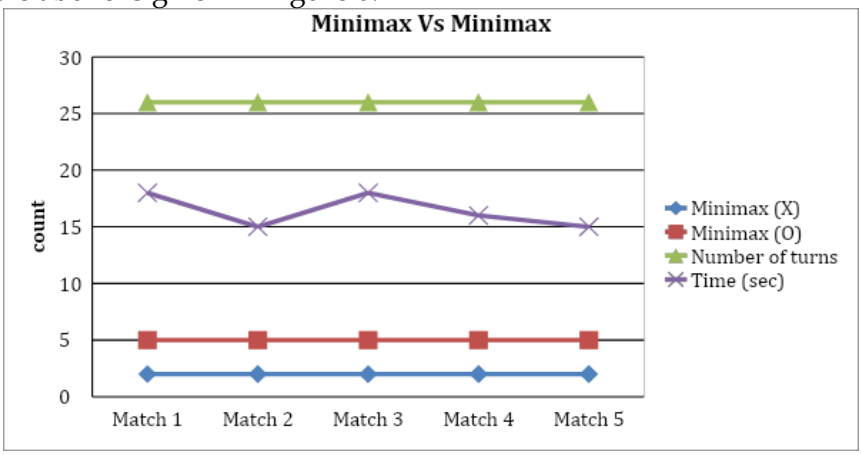

Figure 6: Minimax (X) vs. Minimax (O)

Minimax Versus Greedy

The second experiment was between minimax and greedy based agent, where the minimax agent goes first and greedy based agent played second. Table 5 represents the result of five matches.

Table 5: Minimax versus Greedy

	X pieces	O pieces	Turn	Time (s)

Match 1	2	10	12	09
Match 2	2	10	12	10
Match 3	2	10	12	10
Match 4	2	10	12	12
Match 5	2	10	12	09

The table shows a constant winning by the opponent player (greedy) with improved success compared (10 pieces) with matches in table 2 were greedy plays first in 12 turns. This shows that the game favors the opponent player. Figure 7 represents the results in table 5.

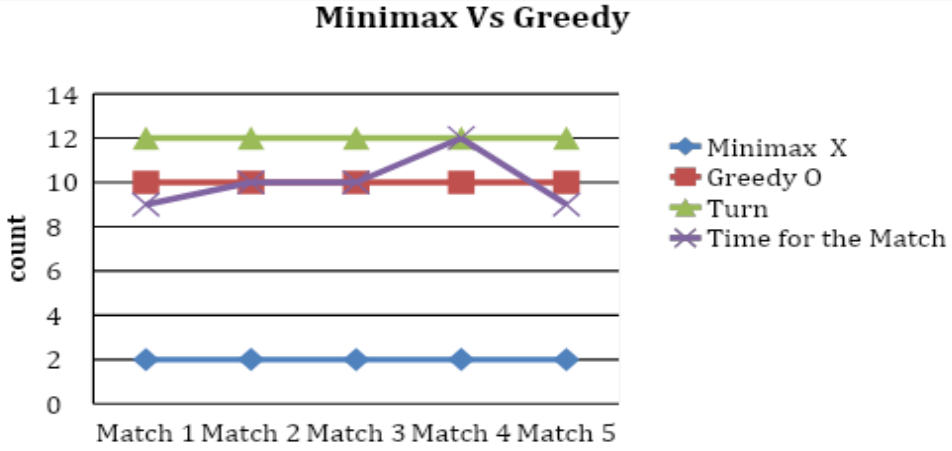

Figure 7: Minimax (X) Vs. Greedy (O)

Discussion

The most optimal match is noted when greedy defeat minimax as presented in table 2 in eleven (11) turns in all the five matches. Results presented show that while playing against self, greedy based agents find a more optimal solution (24 turns) thanminimax based agents (26 turns).
The results show that the game is biased to the opponent player when both players employ the same search algorithm, and this can further be depicted in the improvement of the number of pieces captured by minimax as an opponent player in table 2 compared with the result obtained as the first player in table 5.

CONCLUSION

We adopt and implement the model of Dara that serves as a test-bed for evaluating AI search algorithms(greedy and minimax). Agents based on the greedy and minimax search algorithms were developed and participated in a tournament. The results of ourstudy show that the game is biasedto the opponent player, with greedy-basedagents having the best performance. The outcome of matches against human players shows that none of the agents attained novice expertise. The results of the evaluations reported should serve as a benchmark for subsequent evaluations of search algorithms using the game of Dara.Weintend to incorporate learning techniques in the evolved agents in our subsequent work.

REFERENCES

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).

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)* (pp. 1–8). IEEE.

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).

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), 161–165. Retrieved from www.jsaer.com

Ciancarini, P., & Favini, G. P. (2010). Monte Carlo tree search in Kriegspiel. *Elsevier Artificial Intelligence*, *174*(11), 670–684. https://doi.org/10.1016/j.artint.2010.04.017

Garg, S., Songara, D., & Maheshwari, S. (2017). The winning strategy of Tic Tac Toe Game model by using Theoretical Computer Science. In *2017 International Conference on Computer, Communications and Electronics, (Comptelix)* (pp. 89–95). IEEE. https://doi.org/10.1109/COMPTELIX.2017.8003944

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* (pp. 937–942). AAAI Press. Retrieved from https://www.ijcai.org/proceedings/2017/130%0D

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

Kabir, U., & AbdulHakim, I. (2019). Towards Search Algorithms for Combinatorial Games. In *Proceedings of the 4th YUMSCIC*.

Kowalski, J., & Kisielewicz, A. (2017). Evaluating Chess-like Games Using Generated Natural Language Descriptions. In *Advances in Computer Games* (pp. 127–139). Springer. Cham. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-71649-7_11#citeas

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Perolat, J., … Graepel, T. (2017). A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. *Advances in Neural Information Processing Systems 30*, (Nips). Retrieved from http://arxiv.org/abs/1711.00832

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

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

Oh, C., Lee, T., Kim, Y., Park, S., Kwon, S. bom, & Suh, B. (2017). Us vs. Them: Understanding Artificial Intelligence Technophobia over the Google DeepMind Challenge Match. In

Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17 (pp. 2523–2534). ACM. https://doi.org/10.1145/3025453.3025539

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).

Russell, J. S., & Norvig, P. (2010). *Artificial Intelligence A Modern Approach* (Third Edit). Upper Saddle River, New Jersey 07458: Prentice-Hall Series in Artificial Intelligence.

Sato, M., Anada, K., & Tsutsumi, M. (2015). A Mathematical Formulation based on the Connectedness of stones for the Game of Go. *ACIS International Journal of Computer and Information Science*, *16*(4), 1–11. Retrieved from www.acisinternational.org

Sethy, H., Patel, A., & Padmanabhan, V. (2015). Real-Time Strategy Games: A Reinforcement Learning Approach. *Procedia Computer Science*, *54*, 257–264. https://doi.org/10.1016/j.procs.2015.06.030

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *AeXiv Preprint ArXiv:1712.01815*, 1–19. https://doi.org/10.1002/acn3.501

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., … Abbeel, P. (2017). Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *NIPS 2017*. Retrieved from https://arxiv.org/abs/1611.04717

Therabytesapps. (2014). Tic Tac Toe Movable. Retrieved June 7, 2018, from m.slideme.org/user/therabytesapps

Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., & Gershman, S. J. (2017). Human Learning in Atari. In *2017 AAAI Spring Symposium Series on Science of Intelligence: Computational Principles of Natural and Artificial Intelligence*. Retrieved from http://gershmanlab.webfactional.com/pubs/Tsividis

Victor, do N. S., & Chaimowicz, L. (2017). MOBA: a New Arena for Game AI. *Arxiv Preprint ArXiv:1705.10443*, 1–8. Retrieved from http://arxiv.org/abs/1705.10443