TELKOMNIKA, Vol.17, No.1, February 2019, pp.210~217
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v17i1.11590

Modelling and implementation of 9tka game with Max^N algorithm

Dina Stefani*[1], Frederikus J.[2], Irene Astuti Lazarusli[3], Samuel Lukas[4], Petrus Widjaja[5]
[1,5]Mathematics Department Faculty of Science and Technology, Universitas Pelita Harapan,
UPH Tower, Lippo Karawaci, Tangerang, 15811, Indonesia
[2,3,4]Computer Science Department Faculty of Computer Science, Universitas Pelita Harapan,
UPH Tower, Lippo Karawaci, Tangerang, 15811, Indonesia
*Corresponding author, e-mail: dina.stefani@uph.edu, irene.lazarusli@uph.edu, samuel.lukas@uph.edu, petrus.widjaja@uph.edu, erikkeren@gmail.com

Abstract

9tka is a board game created by Adam Kaluza. The game can be played with 2 up to 4 players, with the goal of conquering as many areas in the board as possible. The aim of this research is to implement the 9tka game into a digital game that can be played on a personal computer. The implementation will include the feature to play against computer players. The rules and game's play of 9tka is modelled, and then implemented using Java. The Artificial Intelligence (AI) of the computer player is implemented using the Max^N algorithm, which is an extension of the minimax algorithm. Several tests were done to gauge the robustness of the implemented AI. The experiments show that the AI is capable to make a move in time less than 541 milliseconds on average, across all types of players. Moreover, from eight respondents, the average amount of human wins is 2.25 out of 5 matches, across all types of players. This shows that the implemented AI on computer player has a better chance to defeat human opponents.

Keywords: *9tka board game, max^N algorithm*

1. Introduction

9tka("dziewiątka"="nine" in Polish) game is a board game created by Adam Kaluza and has its origins in Poland [1]. The game can be played with 2 players up to 4 players, using a 9x9 square board, 9 red counters, 18 yellow counters, 18 blue counters, 12 orange counters and 9 green counters as its components. The board is divided in 9 sections of 3x3 squares each as depicred in Figure 1. In the Initial condition of the game, one red counter (neutral counter) has to be placed randomly on each section but cannot be placed on the edge sections of the board. Other colored counters are used as the player counters. If there are two players then each has 18 counters, yellow and blue. If it is played by three players then each has 12 counters yellow, blue and orange. If four players involve then each has 9 counters, yellow, blue, orange and green.

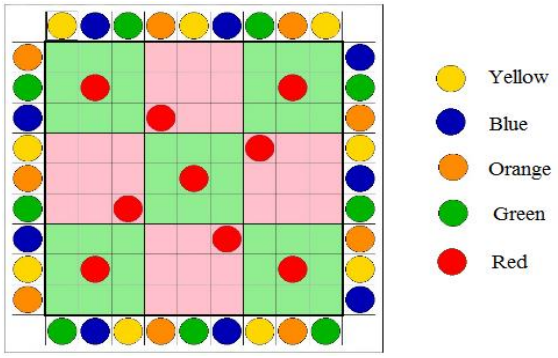


Figure 1. An example of initial condition of 9tka game played with four players

Received 22 July, 2018; Revised October 2, 2018; Accepted November 12, 2018

The game is played in 2 phases, placement and movement. In placement phase, players take turns clockwise by placing freely their counters outside the board, next to the edge fields. The placement phase ends when all slots are taken then the game continues to the movement phase. Figure 1 shows an example of initial condition after placement phase ends with four players.

During the movement phase, players take turns by picking one of their counters at the edge of the board and moving it along its row or column until it hits any other counter or hits the opposite edge of the board. If it hits the opposite edge, it will block one or two counters that are not moved yet as shown in Figure 2. If player cannot move his counter, then that player passes to move.

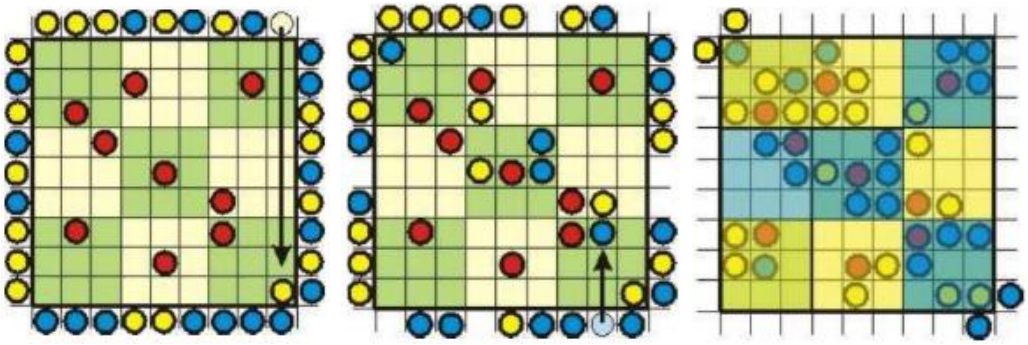


Figure 2. First player first movement, second player fourth movement and game end

The game ends when there are no more possible moves for every player. Each player scores one point for every section they conquer. A section is conquered by a player when at that section, player has more counters than that of any other player. In case of a tie, it means there is no player has more counters than that of others player, no one scores for that tie section. The winning player is the one who has the highest score. In case of a tie, the last of the tied player wins. For example, if score of players tie in a two player game, the second player wins the game. In Figure 2, the top left section is conquered by the yellow player because the yellow player has 3 counters on that section while the blue player only has 2. The yellow player has conquered totally 5 sections and the blue player has conquered 4 sections. Therefore, the yellow player wins the game.

This game is quite simple but it is considered as one of interesting logic game. From the beginning of the game, at placement phase, players should place their counters intelligently. In movement phase, players should move their counters accurately to win the game. The problem is this game played in a board and have to be played at least by two players.

The most famous algorithms in brute-force search are MiniMax [2], NegaMax [3], Alpha-Beta [4], NegaScout [5], and Principle-variation [6]. This paper discusses on how this game can be digitalized and being played by single person and computers by implementing Max^N algorithm.

Minimax Algorithm is a depth-first search algorithm designed for zero-sum games that involves 2 players. This algorithm is proven optimal strategy in Tic Tac Toe game [7]. Minimax algorithm with alpha-beta modification has a time complexity of $O(b^m)$ in the worst case, $O(b^{3m/4})$ in the average case and $O(b^{m/2})$ in the best case; where b is the branching factor and m is the maximum depth of the search tree [8]. Minimax algorithm implements a nondeterministic strength-adapted AI opponent for board games and behaves as expected [9, 10]. Researchers agree that the goal of the minimax algorithm is to find the best move that can be made by the current player, by searching the game tree. Minimax can use one of two basic strategies, either by searching the entire game tree, or by searching the tree until a predefined depth before it is evaluated [11]. The algorithm works by searching the game tree until a certain depth, and evaluating every single node, child on that depth, using an evaluation function. Then, the algorithm sets the value of the parent to be either the lowest estimated or the highest estimated

value of its children. The algorithm proceeds with this for all branches of the tree until the best move to make is "bubbled" up to the root node [12].

The Max^N algorithm is an extension of the minimax algorithm for N-player games. The algorithm works similar to the Minimax algorithm, except that the evaluation function now returns an N-tuple of values, with each component corresponds to the estimated merit of the position with respect to one of the players. Then, the value of each interior node where player *k* is to move is the entire N-tuple of the child for which the k^{th} component is a maximum [13-15].

The purpose of this paper is to model and to implement the 9tka board game into a digital game that can be played on PCs. The implemented game will include the feature to play against up to three computer players with Artificial Intelligence using the Max^N algorithm.

2. Research Methodology

First step is creating mathematical model of the game. Since the game is a board game and there are four possible players represented with the counter colour than it is perfectly modelled with the 11x11- sized matrix A = {a(i,j)} where

$$a(i,j) = \begin{cases} 0 & Empty \\ 1 & Yellow \\ 2 & Blue \\ 3 & Orange \\ 4 & Green \\ 5 & Neutral \end{cases}$$

$$i \in \{0,1,\dots,9,10\} \wedge j \in \{0,1,\dots,9,10\}, a(0,0) = a(0,10) = a(10,0) = a(10,10) = 0$$

Second step is defining the 9 sections of the board in which each section consists of 3x3 squares in the board game. These sections are represented with matrix, R = {g(p,q)}, where g(1,1) consists of the squares a(1,1), a(1,2), a(1,3), a(2,1), a(2,2), a(2,3), a(3,1), a(3,2), and a(3,3). A function of g: $R^2 \rightarrow R^2$ needs to be created that charts position (i,j) in the game board to (p,q) as g(p,q) where the values of p and q are determined by (1). Then a function of f: $R^2 \rightarrow N^+$ is created to label (p,q) by (2)

$$p = div(i,3) + 1 \quad \text{and} \quad q = div(j,3) + 1 \tag{1}$$

$$k = f(p,q) = 3(p-1) + q \tag{2}$$

Third step is entering the first phase of the game, placement phase. There are 2 types of counter placement, one for neutral counters (red counters) and another for player counters. Neutral counters are placed randomly on each of the 3x3 sections. Neutral counter placement in the matrix A={a(i,j)} uses the following 2 procedures:

```
RandomizeNeutralCounter ()
For i = 1 to 3
    For j = 1 to 3
        repeat
               Pos1 = (i-1) * 3 + 1 + random (0,2)
               Pos2 = (j-1) * 3 + 1 + random (0,2)
        Until (Pos1 ≠ 1 or Pos1 ≠ 9 OR Pos2 ≠ 1, Pos2 ≠ 9)
        PlaceCounter (Pos1, Pos2, 5)
PlaceCounter (i, j, counter)
    If (i < 0) OR (i > 10) OR (j < 0) OR (j > 10) return;
    Else IF (a(i,j) ≠ 0 ) return;
            Else a(i,j) = counter
```

Player counters are placed in the order of yellow, blue, orange, and then green by clicking the position of placement square on the board, at the respective player's turn. However, for the AI players, the position of the counter placement will be decided at random from all the available positions, with the following rules:

1. For the positions a(i, 0) or a(i, 10), the system will check whether those positions at the i^{th} row have a counter or not. If it has then the position is invalid to place the counter.
2. The same rule (1) is applyed for the positions a(0, j) or a(10, j), with those positions at the j^{th} column.
3. Each pair of the following eight pairs only has one computer counter {(1,0), (0,9)} {(1,10), (0,1)} {(9,0), (10,9)} {(9,10), (10,1)} {(0,1), (9,0)} {(10,1), (1,0)} {(0,9), (9,10)} {(10,9), (1,10)}

If any of the conditions above are not fulfilled, then the AI will check another random available position for those conditions. However, if there are no more available positions left, the AI will place the counter on the last checked position, regardless of the conditions.

Fourth step is the second phase of the game, movement phase. In movement phase, a counter at a(r_1, c_1) moves and stops at a(r_2,c_2) is defined by the following procedure.

```
MoveCounter (a(r1, c1), a(r2,c2))
    If (r1=0  or  r1 = 10) then
       If ( r1=0 ) then Up = 1 else Up = -1
       If (a(r1+Up,c1)=0) then
             r2 = r1
          Repeat
             r2 = r2 + Up
          Until (a(r2, c1) ≠ 0) OR (r2 = 0) OR (r2 = 10)
          r2 = r2 – Up
          c2 = c1
          a(r2, c2) = a(r1,c1)
          a(r1, c1) = 0
    Else
       If (c1 = 0 OR c1 = 10) then
       If (c1 = 0) then Up = 1 Else Up = -1
       If (a(r1, c1 + Up) = 0 then
          c2 = c1
          Repeat
             c2=c2+Up
          Until (a(r1, c2) ≠ 0) OR (c2 = 0) OR (c2 = 10)
          c2 = c2 – Up
          r2 = r1
          a(r2, c2) = a(r1,c1)
          a(r1, c1)  = 0
```

Fifth step is the determining the best move of AI players. AI player moves a counter a(i,j) using Max^N algorithm [16] is designed with the depth level of n + 3, where n is the number of players in the game. The evaluation value of player *m*, moving a(i,j) counter to section k on the board A, is value eva(m,i,j) stated as (7). However, it should be defined some functions. Firstly, v(m,k) to save the amount counters of player m in section k. It is defined in (3) for every counter of player m comes to section k.

$$v(m,k) = v(m,k)+1 \tag{3}$$

Secondly, c(m,k) to indicate player m takes control of section k. If the amount of player m counters is the largest in that section then c(m,k) = 1 otherwise c(m,k) = 0. It is stated in (4)

$$c(m,k) = \begin{cases} 1 & v(m,k) > v(p,k) \quad \forall p \neq m \\ 0 & others \end{cases} \tag{4}$$

Thirdly, y(k) to encourage or discourage the computer to move the counter to section k. If v(m,k) is greater than 4 then that that section is no need to be filled with any counter since it definitely belong to the player. However, if v(m,k)=0 that it should be encourage to fill that section. It is stated in (5). Value C_1 is set to 30 and C_2 equals to 15

$$y(k) = \begin{cases} C_1 - C_2 v(m,k) & 0 \leq v(m,k) \leq 4 \\ -C_2 & v(m,k) \geq 5 \end{cases} \tag{5}$$

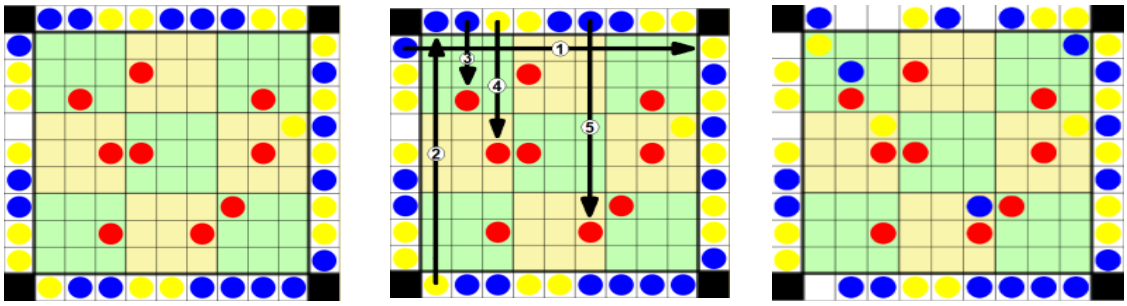
Fourthly, r(k) and q(k) stand for number of blocked opponent's counters and number of blocked computer's counters at section k.

Fiveth, eva(m,i,j) to save the evaluation value of player *m*, moving a(i,j) counter to section k. It is stated in (6).

$$eva(m,i,j) = \sum_{k=1}^{9} C_3\, c(m,k) + C_4(r(k) - q(k)) + \sum_{k=1}^{9} y(k) \quad (6)$$

Later, how can the constantas C_3 and C_4 be valued. It is clear that first consideration is by moving a(i,j) counter to section k then player can control that section. Therefore, C_3 should be the bigest. The more value r(k) and the lower value q(k) is the better, therefore C4 is the second bigest. Since the value of $y(k)\varepsilon\{-30,-15,0,15,30\}$, then C_3 is set to be 100 and C_4 is 90 experimently.

Suppose initial condition of game played with 2 players is stated in Figure 3 (a) and one possible game states at depth 5 after moving (1, 0) counter, is shown in Figure 3 (c). It is reached by moving counters in order from (1, 0), (10, 1), (0, 2), (0, 3) and (0, 6).



(a) (b) (c)

Figure 3. Sample of 5 movement (a) Initial condition
(b) 5 movement after initial condition (c) Final 5 movement

It can be seen that the computer has conquered 2 sections k = 3 and k = 8 and has 1 counter each at section k=1, k = 3 and k = 8, and has blocked 2 opponent counters at (1,10), (0,9) and no blocking computer counters. Therefore, the evaluation value eva(2,1,0) is as follows. eva(2,1,0)=100×2+90×(2-0)+15×3+30×6=605.

Another possible movement is shown on Figure 4. The game state is reached by moving counters consecutively (1, 0), (5, 0), (0, 1), (2, 0) and (0, 2). The computer has conquered 2 sections, k = 3 and k = 7, has 1 counter each at section k=1, k=3 and k=7 and has blocked 4 opponent counters and no computer counters. Therefore, the evaluation value is eva(2,1,0)=100×2+90×4+30×6+15×3=785.

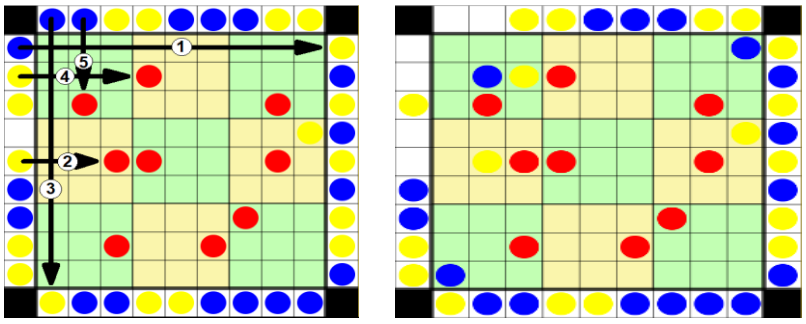


Figure 4. Another State after moving (1, 0) counter

By generating all possible movements, constructing the tree with depth 5 and using Max^N algorithm with evaluation as its heuristic value then computer will determine which counter is the best to be moved. Finally, Player m is the winner of the game in (7).

$$s(m) = \max_{k\epsilon\{1,2,3,4\}}\{s(k)\} \quad (7)$$

3. Results and Discussion

The 9tka game is implemented using the Java programming language. User must first determine the settings of the game on the start screen window, such as the number of players and which counters are controlled by the player. As seen in Figure 5.

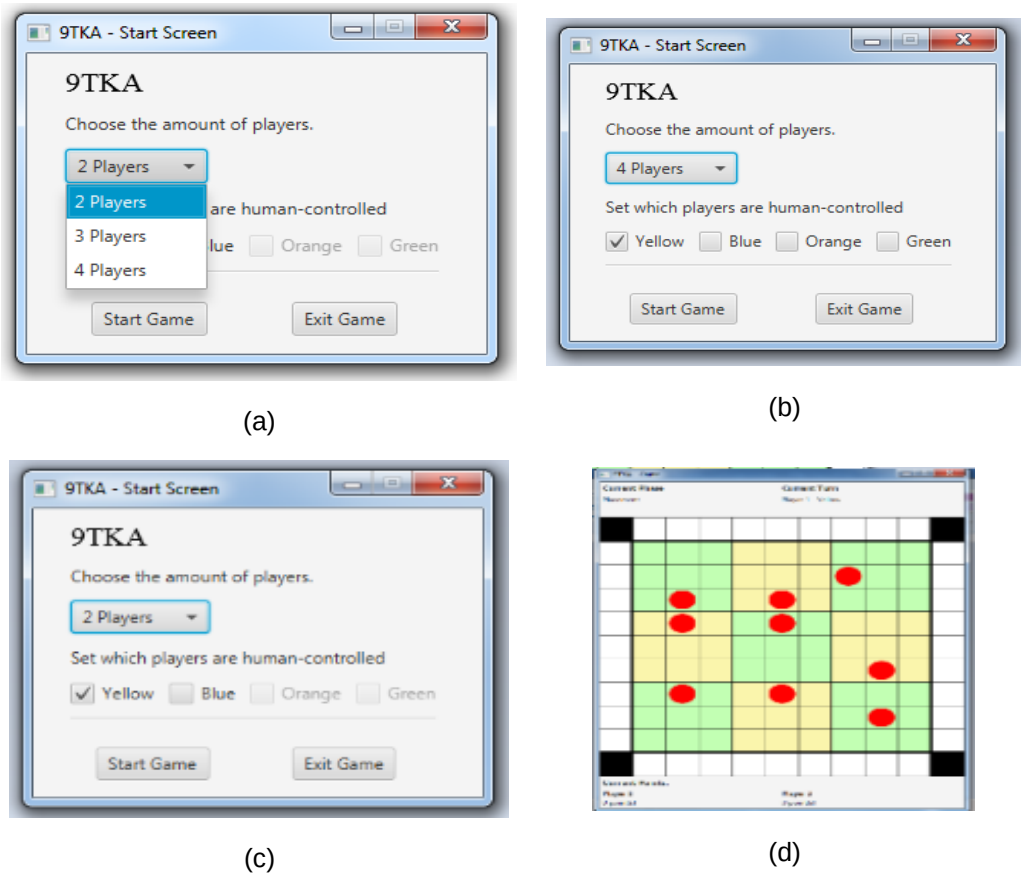


Figure 5. User interface design (a) # Players (b) 4 players (c) 2 players (d) board game

After it is done, the user clicks the Start Game button, and the main game window will appear, as shown in Figure 6 (a). Placement phase is done by clicking on the white squares at the edge side of the board to place a player's counter, Figure 6 (b). Afterwords, clicking on the counters inside the same squares to move it in the movement phase. Figures 6 (c) and 6 (d) show the examples of first movement that can be made by a player and the end of the game.

First experiment is conducted to gauge the time needed for the AI to make a move in the movement phase, in milliseconds. The test is conducted in 15 games, 5 games for each of number of player. Each game consists of 18 steps for 2 players, 12 steps for 3 players and 9 steps for 4 players. The results are depicted at Table 1. Figure 7 shows that amount of time to move a counter, drops exponentially according to number of steps and the more the number of players, the amount of time is longer to move the first and second counters. On average for two, three and four players, every move is less than 180, 270 and 541 respectively in milliseconds.

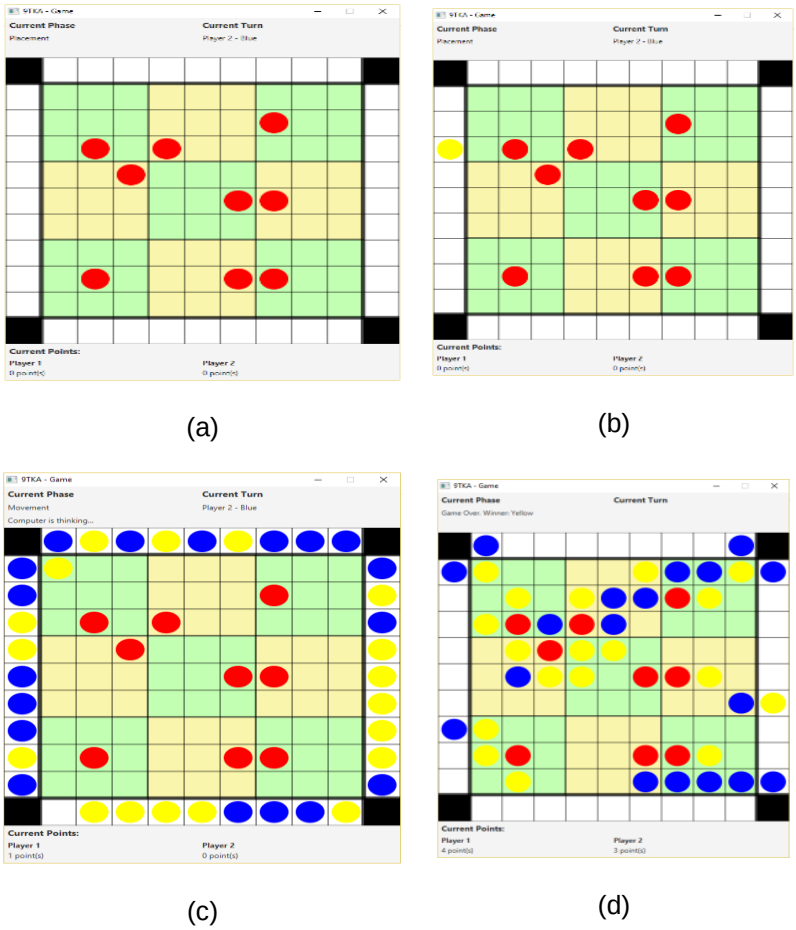


Figure 6. User interface of the game (a) Start game (b) first placement (c) first movement (d) end of the game

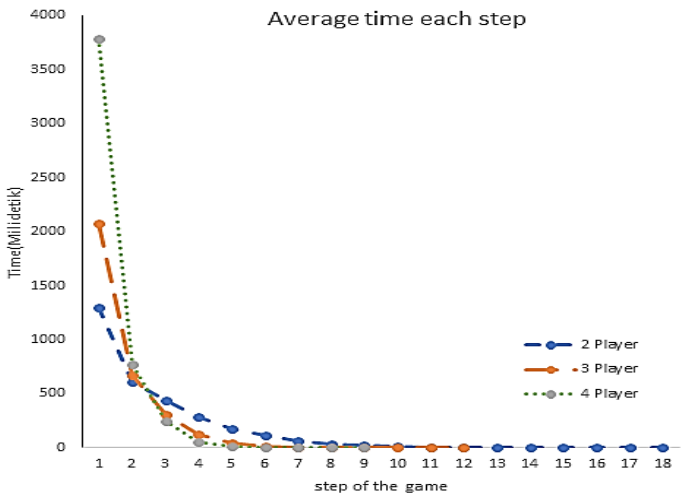


Figure 7. Respond time needed for each step toward number of players

The second experiment is done to know how well the AI of the system comparing to the human player. Eight respondents are selected to test the AI of the system. Each respondent is asked to play the implemented 9tka game 15 times, 5 games each for 2, 3 and 4 players. The

number of wins that the respondent has achieved for each game type is tabulated at Table 1. It shows that the average a human wins the game of two players is 2.25 out of 5 games. It means the probability AI of computer player winning the game is 0.55. It has bigger chance to win the game than that of the human. The more the number of computer player, the harder human can defeat the computer players.

Table 1. Number of wins for each player

Respondent	Number of players		
	2	3	4
1	4	4	1
2	2	0	1
3	1	1	0
4	2	4	2
5	2	2	1
6	3	2	1
7	1	2	2
8	3	2	2
Average	2.25	2.125	1.25
Probability of Winning	0.45	0.425	0.25

4. Conclusion

The 9tka game has successfully been implemented into a digital game and able to be played by 2, 3, and 4 players. The implemented AI is capable of making relatively good decisions in a relatively short amount of time, using the Max^N algorithm. Even though system has a few general rules at the placement phase, further research can be done to improve in placing the computer counters in placement phase. If it is possible that the evaluation function for the placement phase can be also modelled. The evaluation function for the movement phase also needs several improvements such as using pruning algorithm so that it can speed up the movement.

References

[1] A Kaluza, NR Andrés. 9 (Nine). A Game for 2 to 4 Players by Adam Kaluza. 2011.
[2] S Russell, P Norvig. Artificial intelligence: a modern approach. Prentice Hall Press. 2009: 1152.
[3] GT Heineman, G Pollice, S Selkow. Path Finding in AI in Algorithms in a Nutshell. O'Reilly Media. 2008: 213–217.
[4] D Knuth. Selected Papers on Analysis of Algorithms. California: Center for the Study of Language and Information. 2000.
[5] HJ Chang, MT Tsai, T Hsu. Game Tree Search with Adaptive Resolution. *Advances in Computer Games SE–26*. in HJ Herik, A Plaat. Springer Berlin Heidelberg. 2012; 7168: 306–319.
[6] MHM Winands, HJ van den Herik, JWHM Uiterwijk, ECD van der Werf. *Enhanced forward pruning*. *Inf. Sci. (Ny)*. 2005; 175(4): 315–329.
[7] K Kosciuk. Is Minimax Really an Optimal Strategy in Games? Zeszyty Naukowe Politechniki Bialostockiej Informatyka. 2009.
[8] AA Elnaggar, M Gadallah, MA Aziem, H El-Deeb. A Comparative Study of Game Tree Searching Methods. *International Journal of Advanced Computer Science and Applications*. 2014; 5(5): 68-77.
[9] SG Diez, J Laforge, M Saerens. Rminimax: An Optimally Randomized MINIMAX Algorithm. *IEEE Transactions on Cybernetics*. 2013; 43(1): 385-393.
[10] P Borovska, M Lazarova. *Efficiency of Parallel Minimax Algorithm for Game Tree Search*. International Conference on Computer Systems and Technologies (CompSysTech), Bulgaria. 2007: II.7-1-II.7-6.
[11] MT Jones. *Artificial Intelligence, A Systems Approach*. Infinity science Press LLC. 2010.
[12] V Öberg. Evolutionary Ai in Board Games, Bachelor Degree Project in Informatics. University of Skovde. 2015.
[13] M Fridenfalk. N-Person Minimax and Alpha-Beta Pruning. Nicograph International, Sweden. 2014: 43-52.
[14] RE Korf. Multi-player alpha-beta pruning. *Artificial Intelligence*. Elsevier. 1991; 48: 99-11.
[15] I Zuckerman, A Felner, S Kraus. *Mixing Search Strategies for Multi-Player Games*. Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, Pasadena. 2009: 646-651.
[16] NR Sturtevant. Multi-Player Games: Algorithms and Approaches. A dissertation project in Computer Science. University of Califormia, Los Angeles. 2009.