Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)
Edisi, .. Volume, ... Bulan 20.. ISSN : 2089-9033

IMPLEMENTATION OF MTD(F) ALGORITHM TO DETERMINE THE SOLUTION ON BOARD GAME 2048

Rizqiana Trinenda

Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : rizqiana.trinenda@yahoo.co.id

ABSTRACT

The 2048 game is a game created by web developers coming from Italy, namely Gabriele Cirulli in March 2014. The game is fairly easy to play because it just move left, right, up or down to merge certain numbered boxes in order to form 2048. Various algorithms have been implemented to solve the game, such as greedy algorithm, backtracking, and minimax. The result of implementation showed that greedy algorithm not optimal at completing the game 2048 because of in the 11 trials could not get the numbers 2048, but quickly in taking the step while backtracking algorithm can finish the game 2048 in the first time trial but with a time of 23 minutes 18 seconds. For minimax algorithm results are determined from the depths in the search, the more searching the more likely to be able to complete the game 2048, but it took much longer appropriate depth search. For that reason, research is still needed on other algorithms to complete the game 2048.
The algorithm that studied is MTD (f) algorithm. Based on the results of the study, analysis of system design and implementation as well as tests performed on the algorithms MTD (f) to finish the 2048 game, it could be concluded that the algorithm MTD (f) can be applied to the 2048 game, but to get the tile in 2048 in order to win the 2048 game requires search the depths of at least level three and takes about 13 minutes 59 seconds. The level of accuracy of the algorithm MTD (f) on a search with a depth of level zero, one and two is 0%, whereas on level three is 20%, level four is 30%, and level five is 60%.

Keywords: algorithm, MTD (F),minimax, game, 2048, searching

1. INTRODUCTION

The 2048 game is a game created by a web developer who comes from Italy, namely Gabriele Cirulli in March 2014. The game is quite easy to play because it just move left, right, up or down to incorporate a specific numbered squares that make up the final result in 2048 [1]. The game was released in 2048 for various platforms so that many people can play it. Today many people use various ways to finish the game. Various algorithms have been implemented to solve, such as by using a greedy algorithm, backtracking [2], and minimax [3]. The implementation of the results showed that less than optimal greedy algorithm in completing the game in 2048 because of the 11 trials could not get the numbers in 2048, but quickly in taking the step while backtracking algorithm can finish the game in 2048 in the first time trial but with a time of 23 minutes 18 seconds. For minimax algorithm results are determined from the depths in the search, the more in looking the more likely to be able to complete the game in 2048, but it took much longer appropriate depth search. It required a study of other algorithms to complete the game in 2048. The algorithm that will be examined is the algorithm MTD (f). Algorithm MTD (f) development is one of the minimax algorithm. MTD (f) stands for Memory-enhanced Test Driver with node n and value f. In his research, Aske PLAAT [4] states that the algorithm MTD (f) algorithm is the most efficient minimax than previous minimax algorithms. In some experiments computer games such as chess, othello, and checkers, algorithms MTD (f) have average performance is better than the algorithms optimization minimax such as Alpha-Beta and negascout [5], so the algorithm MTD (f) possible can be finishing the game in 2048. From the description, it will be conducted to implement the algorithm MTD (f) in 2048. It is expected to complete the game to implement the algorithm MTD (f) can be determined the ability of the algorithm MTD (f) to finish the game in 2048.

1.1 The 2048 Game

The 2048 game is a game for single player that was created in March 2014 by an Italian programmer named Gabriele Cirulli. When the first release of this game, the game went straight to number 1 of the world in just a week and more than 4 million downloads and it is a record that is quite fantastic. In fact, this game is just a project "fad" that was done by the Italian programmer who just want to try if he could create a game from scratch, and he managed to finish preparation in just one

week. The game is also not the first created, previously also there is a game called Three! which has the look and play a similar way, and also in 1024 that the way to play the game exactly the same, just with a slightly different game goal [1].



Figure 1 The 2048 Game

1.2 Rules Of The Game 2048

For the rules of how to play the game in 2048 is as follows [1]:

1) *The game is played on a board 2048 gray 4 × 4.*
2) *At the top there are several boxes with different colors that shift when the player presses the four directional keys on the keyboard.*
3) *The box can move as far as possible to meet another box or skirting boards.*
4) *If the two boxes of the same number to join the current shift, the two will merge and produce a new box number is the sum of the two numbers in the box to join earlier.*
5) *For example in Figure 2 boxes are worth 2 could be merged when a player shift to the left or right and will generate valuable box 4.*
6) *The new box is not able to join the other boxes in the same direction.*
7) *In every shift there will be a new box numbered 2 or 4 that appear randomly on the empty space on the board.*
8) *The new player wins when a 2048 numbered boxes appear on the board.*
9) *When a player can not move anymore, the game ends.*
10) *Score obtained from the merger of the box.*

1.3 *Alphabeta PrunningAlgorithm*

In minimax algorithm, a search is performed on all parts of the tree, while most trees should not be checked. Alpha-beta pruning is a modification of the minimax algorithm, which will reduce the number of nodes that are evaluated by the search tree. The search for the next node will be considered first. This algorithm will stop evaluating steps when there is at least a possibility of being discovered and proved that such a step worse than the measures examined earlier. Thus, the next step does not need to be evaluated further. With this algorithm optimization results of an algorithm will not change [5]. In this algorithm, there are two values that set, namely alpha and beta, which represents the value of max is believed to have a drink and a maximum value of which is believed min. The initial value is infinite negative alpha and early beta value is positive infinity. As a result of a recursive process, the search area will be smaller. When beta becomes smaller than alpha, will mean that the current position can not be the best outcome for both the game and the players do not need to search further. In Figure 2 can be seen pseudocode of the algorithm Alphabeta.

```
function alphabeta(node, depth, α, β, maximizingPlayer)
    if depth = 0 or node is a terminal node
        return the heuristic value of node
    if maximizingPlayer
        v := -∞
        for each child of node
            v := max(v, alphabeta(child, depth - 1, α, β, FALSE))
            α := max(α, v)
            if β ≤ α
                break (* β cut-off *)
        return v
    else
        v := ∞
        for each child of node
            v := min(v, alphabeta(child, depth - 1, α, β, TRUE))
            β := min(β, v)
            if β ≤ α
                break (* α cut-off *)
        return v
```

Figure 2 Pseudocode Alphabeta Algorithm

1.4 MTD(f) Algorithm

MTD(f) algorithm is a new Minimax optimization algorithm that is simpler and more sangkil than some of its predecessors [4]. The name of the algorithm is short for MTD (n, f), which is abbreviated from Memory-enhanced Test Driver with node n and value f. MTD is the name of a set of drivers looking for a program that uses the calling tree Minimax zero-window Alphabeta. In some experiments computer games such as chess, othello, and checkers, this algorithm has average performance is better than Negascout (variation of Alphabeta implemented in almost all of the game of chess, checkers and othello). One of the strongest chess program, Cilkchess MIT that uses parallel computing method, also using MTD (f) as a search algorithm replaces Negascout used by his predecessor chess program, StarSocrates. Algorithm MTD (f) consists of 10 lines of code alone as in Figure 3.

```
function MTDF(root, f, d)
g := f
upperBound := +∞
lowerBound := -∞
  while lowerBound < upperBound
    if g = lowerBound then
      β:= g+1
   else
      β := g
    g := AlphaBeta (root, β-1, β, d)
```

```
    if g < β then
      upperBound := g
    else
      lowerBound := g
  return g
```

Figure 3 Psuedo Code MTD(f) Algorithm

Algorithm MTD(f) calling Alpha Beta function many times by the method of zero-Alpha-Beta search window, unlike Negascout that uses wide-window search. Dialing Alpha Beta returns the limit of the value of evaluation Minimax. Limit of the value is then stored in the upperbound (upper limit) and lowerbound (lower limit), forming an interval that covers the actual value of Minimax in search of a particular depth. Positive and negative infinity stands for values outside the interval on the leaves of trees. When the upper limit and lower limit of the same value or lower limit value has exceeded the upper limit, then the value of Minimax has been found.

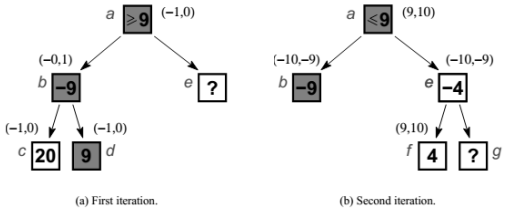


Figure 4 MTD(F) Searching Tree

2. THE CONTENT OF RESEARCH

2.1 Problem Analysis

Based on research conducted by Stephen Bandung Institute of Technology applying greedy algorithm and backtracking in the game in 2048 [2] and Vasilis Vryniotis that implement the algorithm minimax on a game in 2048 [3], it was found that the greedy algorithm less than optimal in completing the game in 2048 because of 11 times trial can not get the numbers in 2048, but quickly in taking the step while backtracking algorithm can finish the game in 2048 in the first time trial but with a time of 23 minutes 18 seconds. For minimax algorithm results are determined from the depths in the search, the more in looking the more likely to be able to complete the game in 2048, but it took much longer appropriate depth search. It required a study of other algorithms to complete the game in 2048.

2.2 Analysis of The 2048 Game

The 2048 game is a game played alone. In this final game will be done by an algorithm that will be examined, namely MTD (f) to determine a solution in resolving permaian 2048. 2048 Games played by sliding to the left, right, top and bottom. The game ends when no boxes that can be shifted or merged again. The game will be won when it managed to get a box that is worth 2048. Randomly number 2 or 4 will appear when it is shifted. If there was a box of equal value and its position adjacent the value can be combined. Entries for opponents enter the data in the form of figures 2 and 4 with the possibility of appearing to number 2 is 90% and the number 4 is 10%. As for the players enter the data in the form of direction to pan left, right, top and bottom. If one direction can not be moved so the application will not respond and the game board unchanged.

2.3 Value Evaluation

The evaluation value to the game in 2048 on research by snaking tactic [1] which can be calculated by the number of linear value on the board multiplied by the value of the geometric sequence with ratio <1. It can also be written as follows:

$$p_n \in Path_{0....N-1} \quad (1)$$

$$Score = \sum_{n=0}^{N-1} value(p_n) * r^n \quad (2)$$

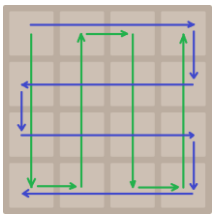


Figure 5 Snaking Tactic

2.4 Analysis of MTD(f) Algorithm

In this section we will discuss about the completion of the game in 2048 using the algorithm MTD (f). 2048 The game is played on a game board with the size 4x4 box so the total is 16. The game board is represented by a 4x4 matrix.

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

Figure 6 Board Game Matriks

MTD(f) algorithm would be implemented as a player that will maximize scores. Algorithm MTD (f) algorithm call Alphabeta pruning in the search for a solution.

1024	512	256	128
2	8	4	128
4	4	2	2
16			

Figure 7 The Initial State Board Game

Then look for the next step by using an algorithm MTDf) to obtain an evaluation value of each step. Each step has been sought can be seen in the following table:

Table 1 Left Step Search Results

Iteration	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1170,5	0	1170,5	+∞
2	1170,5	1171,5	1170,5	+∞
	1172	1171,5	1172	+∞
3	1172	1172,5	1172	+∞
	1172	1172,5	1172,5	1172

Table 2 Right Step Search Results

Iteration	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1170,5	0	1170,5	+∞
2	1170,5	1171,5	1170,5	+∞
	1172	1171,5	1172	+∞
3	1172	1172,5	1172	+∞
	1172	1172,5	1172,5	1172

Table 3 Up Step Search Results

Iteration	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1184	0	1184	+∞
2	1184	1185	1184	+∞
	1184	1185	1184	1184

Table 4 Down Step Search Results

Iteration	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1025	0	1025	+∞
2	1025	1026	1025	+∞
	1172	1026	1172	+∞
3	1172	1173	1172	+∞
	1172	1173	1172	1172

In the above table have been searching algorithms using MTDf). Variable G is the value of a function call Alphabeta search, variable beta beta value at the MTD (f) to the value upperbound on the function of the alphabet, while lowerbound and upperbound is the upper limit and lower limit of the algorithm MTD (f) in the search. Then each displacement value obtained as follows: Left = 1172, 1172 = Right, Up and Down = 1184 = 1172. From the results obtained that the next step is to up.

2.5 Testing System

Testing algorithm is performed to test the accuracy and speed of the algorithm MTD (f) which has been implemented in 2048. The game 2048 game will be played by the algorithm MTD (f) as many as 10 times each search depth level. Then count the number of games that could be reached in 2048 at each level of the depth of the search. So as to get the accuracy of the algorithm MTD (f) by comparing the number of games that could be reached in 2048 with the number of games that have been made. Search will be made until a greater degree of accuracy equal to 50%. As for the speed is calculated based on the time required to complete the game in 2048.

Table 5 Results of Testing Depth Level 0

Trial	Status	Largest Tile	Time (HH:mm:ss.SSS)	Skor
1	Failed	256	00:00:05.627	3368
2	Failed	256	00:00:05.452	3236
3	Failed	128	00:00:04.053	1932
4	Failed	256	00:00:04.695	2632
5	Failed	512	00:00:07.212	5184
6	Failed	64	00:00:01.734	668
7	Failed	256	00:00:05.787	3472
8	Failed	256	00:00:05.761	3364
9	Failed	256	00:00:06.370	3884
10	Failed	128	00:00:03.698	1796

Table 6 Results of Testing Depth Level 1

Trial	Status	Largest Tile	Time (HH:mm:ss.SSS)	Score
1	Failed	512	00:00:10.845	8548
2	Failed	256	00:00:04.725	2928
3	Failed	256	00:00:05.151	3140
4	Failed	512	00:00:09.751	7108
5	Failed	256	00:00:06.577	4200
6	Failed	512	00:00:08.955	6580
7	Failed	128	00:00:02.640	1292
8	Failed	512	00:00:08.136	5728
9	Failed	256	00:00:07.661	5012
10	Failed	128	00:00:02.824	1380

Table 7 Results of Testing Depth Level 2

Trial	Status	Largest Tile	Time (HH:mm:ss.SSS)	Score
1	Failed	1024	00:07:50.290	16404
2	Failed	1024	00:06:03.666	12588
3	Failed	512	00:01:45.126	9352
4	Failed	1024	00:04:01.744	14824
5	Failed	512	00:00:46.471	7272
6	Failed	512	00:01:18.127	8092
7	Failed	1024	00:06:11.310	16848
8	Failed	1024	00:08:14.355	18380
9	Failed	512	00:01:49.780	9204
10	Failed	512	00:01:10.856	7464

Table 8 Results of Testing Depth Level 3

Trial	Status	Largest Tile	Time (HH:mm:ss.SSS)	Score
1	Succeed	2048	00:15:20.262	24508
2	Failed	1024	00:10:42.021	17788
3	Failed	1024	00:11:36.187	16476
4	Failed	1024	00:15:19.595	9552
5	Succeed	2048	00:26:19.731	20176
6	Failed	1024	00:09:35.158	15620
7	Failed	512	00:07:35.641	8780
8	Failed	1024	00:12:23.831	12456
9	Failed	1024	00:12:21.244	12492
10	Failed	512	00:09:37.873	7968

Table 9 Results of Testing Depth Level 4

Trial	Status	Largest Tile	Time (HH:mm:ss.SSS)	Score
1	Failed	1024	00:31:49.967	16160
2	Failed	1024	00:28:05.387	17748
3	Failed	1024	00:39:40.811	12252
4	Succeed	2048	00:43:53.370	20168
5	Failed	1024	00:36:10.246	14040
6	Succeed	2048	00:33:54.834	20256

7	*Succeed*	*2048*	*00:29:55.392*	*21124*
8	*Failed*	*1024*	*00:34:29.883*	*15448*
9	*Failed*	*1024*	*00:37:18.236*	*15640*
10	*Failed*	*1024*	*00:42:23.269*	*10260*

Table 10 Results of Testing Depth Level 5

Trial	*Status*	*Largest Tile*	*Time (HH:mm:ss.SSS)*	*Score*
1	*Succeed*	*2048*	*00:57:05.272*	*20472*
2	*Succeed*	*2048*	*01:11:45.322*	*21220*
3	*Failed*	*1024*	*01:08:59.489*	*15804*
4	*Succeed*	*2048*	*01:27:40.496*	*20516*
5	*Succeed*	*2048*	*01:13:29.821*	*20304*
6	*Failed*	*1024*	*01:19:01.814*	*18708*
7	*Failed*	*1024*	*01:25:12.240*	*19948*
8	*Failed*	*1024*	*01:23:10.546*	*18412*
9	*Succeed*	*2048*	*01:32:27.933*	*20572*
10	*Succeed*	*2048*	*01:21:23.320*	*20344*

2.6 Testing Result

The results of testing that was done ten times at each depth level can be seen in the picture.

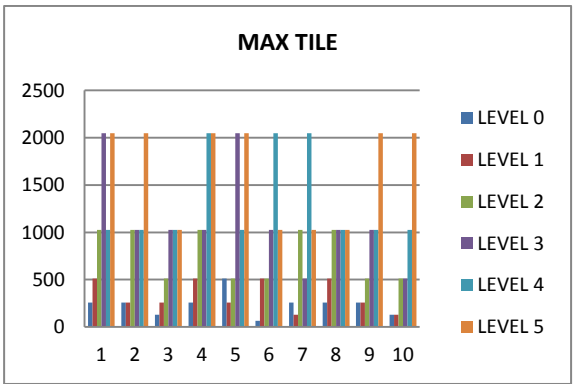


Figure 8 Testing Results Based on Max Tile

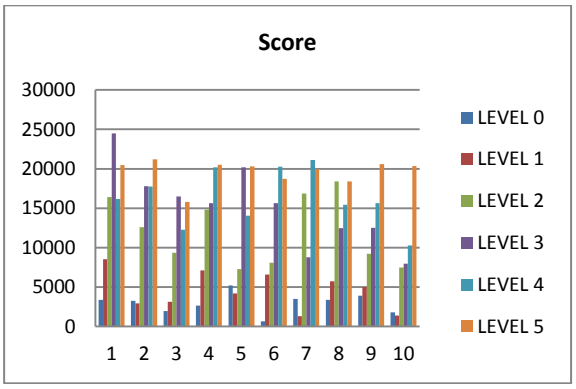


Figure 9 Testing Results Based on Score

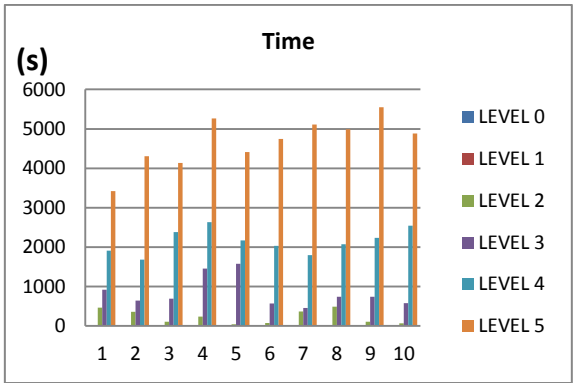


Figure 10 Testing Results Based on Time

3. FINAL

3.1 Conclusion

Based on the results of the study, analysis of system design and implementation as well as tests performed on the algorithms MTD (f) to finish the game in 2048, it could be concluded that the algorithm MTD (f) can be applied to the game in 2048, but to get the tile in 2048 in order to win the game in 2048 requires level searches search depth of at least three and takes about 13 minutes 59 seconds. The level of accuracy of the algorithm MTD (f) on a quest with a depth of level zero, one and two is 0%, whereas on level three is 20%, a level four was 30% and the level of five is 60%.

3.2 Suggestion

In the algorithm implementation MTDf) on the game in 2048 is still far from perfect and there are still many shortcomings. Therefore it is necessary to do a deeper analysis. The advice can be given by the researchers in this study is to apply other algorithms that can be applied to the game in 2048 in order to more quickly and accurately to complete the game in 2048.

REFERENCES

[1] K. Lu, "Game Theory of 2048," *Math 89s: Game Theory and Democracy*, 2014.

[2] Stepen, "Perbandingan Algoritma Greedy & Bactracking Dalam Penyelesaian Permainan 2048," *Makalah IF2211 Strategi Algoritma* , 2014.

[3] V. Vryniotis, "Using Artificial Intelligence to solve the 2048 Game (JAVA code) | Datumbox," 14 November 2014. [Online]. Available: http://blog.datumbox.com/using-artificial-intelligence-to-solve-the-2048-game-java-code/. [Accessed 20 September 2015].

[4] A. Plaat, "MTD (f) A Minimax Algorithm faster than NegaScout," Erasmus University, Rotterdam, 1997.

[5] M. Shalihin, I. Arwani and E. M. Adam, "Penerapan Algoritma Minimax Dengan Optimasi Mtd(F) Pada Permainan Dam-Daman," *Informatika*, 2014.

[6] R. S. Pressman, Software Engineering: A Practitioner's Approach, Seventh Edition, New York: McGraw-Hill, 2010.

[7] J. V. Neumann and O. Morgenstern, Theory of Games and Economic Behavior, Third Edition, New Jersey: Princeton, 1953.

[8] S. Russell and P. Norvig, Artificial Intelligence A Modern Approach, Third Edition, New Jersey: Pearson, 2010.

[9] A. Ilman, "Penerapan Algoritma Minimax Dengan Optimasi Mtd(F) Pada Permainan Catur," *Makalah IF2251 Strategi Algoritmik*, 2008.

[10] J. Lewis and W. Loftus, Java Software Solution Foundations of Programming Design, Sixth Edition, New Jersey: Pearson, 2008.

[11] D. A. Uston, Theory and Problems of Software Engineering, New York: McGraw-Hill, 2002.

[12] Microsoft, "Introduction to the C# Language and the .NET Framework," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx. [Accessed 18 12 2015].

[13] Microsoft, "Visual Studio," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/vstudio/aa718325.aspx. [Accessed 15 12 2015].