IMPLEMENTATION OF UPPER CONFIDENCE BOUNDS FOR TREES (UCT) ALGORITHM FOR FINDING DECISION IN DAMDAMAN GAMES

Hilmi Abdul Aziz

Informatics Engineering – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : hilmi.a.aziz@gmail.com

ABSTRAK

The traditional game is the type of game that contains the values of the culture and in reality is a heritage that should be preserved its existence. There are games that are competitive and there are also the preferred just played to fill spare time as a form of recreation. One of the traditional game with the nature of the competition 1 versus 1 widely played is Damdaman or sometimes called Basbasan. With the development of technology, traditional games began to be abandoned due to lack of adaptation of these games. So this study will try to adapt the game Damdaman which also can give an example of making a good move on its players.

The use of artificial intelligence methods have been used for many kinds of board games similar to Damdaman. In addition to testing the computer's ability to solve problems, it can also be used as a means of entertainment. The search method used in this research is the *Upper Confidence Bounds For Trees* (UCT). This method can produce a good move by testing the accuracy and performance.

From these results it can be seen that the UCT algorithm can be applied to the game Damdaman, resulting winning percentage of 53%, this value is considered quite good. As well as 8011 milliseconds/turn search time performance and memory usage performance amounted to 11.338 bytes/turn.

Keyword : Board Game, Damdaman, *Upper Confidence Bounds For Trees*, *Artificial Intelligence,Tree Search.*

1. PRELIMINARY

The traditional game is a kind of game that contains the values of the culture in essence is a heritage that should be preserved its existence. There are games that are playing and there is also the preferred preferred pastime as a form of recreation. One of the many traditional games played is Damdaman or sometimes called Basbasan. Damdaman is one form of board games are competitive are played by two players, with the number of pieces of each player are 16 pieces. The final goal of the game is to spend pawns belonging to the opponent [1].

With the development of technology, traditional games began to be abandoned due to lack of adaptation of these games. As well as decrease a person who understands and is able to play, this study will try to adapt the game Damdaman which also can give an example of making a good move on his players.

Several previous studies have used Minimax and Negamax algorithm to find optimal decisions in the game Damdaman [2]. The algorithm used is ever Negascout method, which is a variant of the method Minimax, these methods have not shown satisfactory results [3]. Search results are also less satisfactory / rated low on Negamax algorithm with a winning rate of 40% [4]. These studies still produce victory values that have not been satisfactory, these studies suggest the use of heuristic algorithms and improve the depth of the search tree to be able to develop artificial intelligence in the game Damdaman.

In a kind of board games, one of the algorithms used and generate value algorithm is a good win Monte Carlo Tree Search (MCTS). This algorithm is used because of its advantages that can be operated anytime with meaning and discontinued at any time while still providing the appropriate result, require very little knowledge of the problems encountered, and has been successful on a problem that other methods have failed [5]. And the use of simulation of random search that also affect the speed of the algorithm to find the solution without having to evaluate every possibility that there [6].

This algorithm has been implemented and to give promising results in a variety of board games 1 to 1, including Go with a win rate of 75% against expert players, and 95% against ordinary players, as well as on the game Checkers [7] .Sedangkan UCT algorithm is an algorithm variants MCTS the most popular use. UCT algorithm combines Upper Confidence Bound into MCTS algorithm to optimize the process of selecting a solution in search MCTS [8].

Based on this problem, it is hoped that this research can show the results of the performance of the algorithm UCT on taking steps in Damdaman game.

2. RESEARCH

This section will explain about the game Damdaman description, the method used is UCT algorithm and its implementation on the Damdaman game

2.1 Damdaman Games

Damdaman games is one of traditional games, the game is played by two players, the game is played on a board or cardboard that has been drawn. In this game there are 16 pieces for each player.

To start a game of checkers daman, the steps are as follows:

a. Games played by 2 player
b. The player must find the stone / gravel to pawn but every player should not have the same piece
c. Total of player piece is 32 pieces
d. Board for the games drawn like Figure 1
e. Place every player's pieces like shown on Figure 1
f. determine a player's first turn.
g. Games end if a player's piece is all gone/eaten

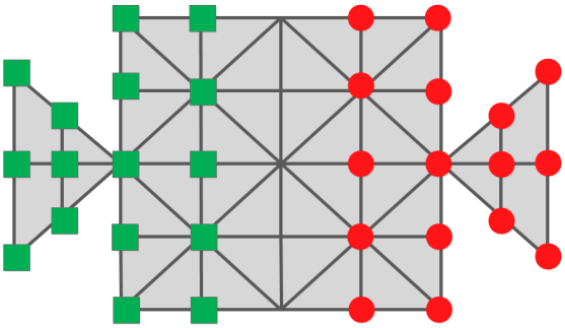


Figure 1. Games Board

Step Damdaman pawns in the game consists of two distinct steps, namely a step ordinary pawn and king pawn step / knight. Ordinary pawns can only make a step to the side, front and diagonal. While the king pawn can move freely. To be a pawn king must pass the tip of the triangle area of the opponent. Area.

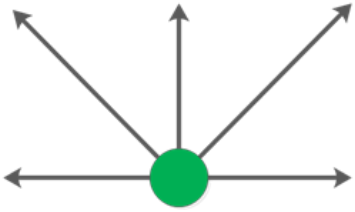


Figure 2 Valid move of a normal piece

As for eating pawns opponents, is that if there is an empty coordinate behind the opponent's pawn. As shown in Figure 3.

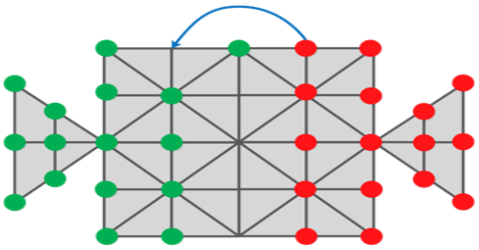


Figure 3 Eating move

Pawns past the end of a regional opponent, the pawn will be the status of magic. Sakti pawns can freely move more than one step forward, backward, sideways or move diagonally.

2.2 *Monte Carlo Tree Search*

Monte Carlo Tree Search algorithm is a heuristic search algorithm Best First Search commonly used as a method for selecting the optimal step. This method uses a search in the search tree form and also using simulated probability that a randomly selected by the Monte Carlo method [13]. MCTS already applied and gave promising results in a variety of board games 1 to 1, including Go, Othelo, Connect4 and Checkers.

The process of decision-making by MCTS algorithm that has four phases:

a. Selection, will make the tree search algorithm is based on a valid step in the game. Of the search tree node that is temporary, then will have a node which is the best step to the method Best-First Search.
b. Expanding the Tree, this step was taken when the circumstance / state unknown in search tree / if a simulation has not discovered the value of the next node level, then a new node will be created from the state. The new nodes are possible move pawns that can be used.
c. Simulation, simulating random game steps and in accordance with the rules of the game to find the optimal step. Each step will be assigned a weighting rules so that this process will tend to move the best.
d. Update, after arriving at the end of the simulation game, this process will renew the value and number of visits from each step that is passed to the root. Go back to step one if the state has not reached the end of the game due to a [14]

Algorithm will do the looping step of the selection process step by UCT algorithm to reaches the final state of the game. The following flow chart of the MCTS algorithm :

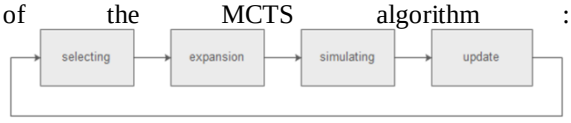


Figure 4 MCTS method's flow

2.3 *Algoritma Upper Confidence Bound to Tree* (UCT)

Algorithm is a variant of Monte Carlo Tree Search, the algorithm MCTS algorithm combined

with Upper Confidence Bound (UCB) on the method of selecting its search tree node. UCB is one of the algorithms used for problems that have many possible solutions, and each solution has taken a different reward. The algorithm then the aim is to get maximum reward from each step / electoral solution is taken [15].

In UCT algorithm, each node that has been known to be stored cost her, then the algorithm will create a random sample, but based on the rules of the game, if the sample size a little, then chances are the average in the sample will be less accurate. Therefore UCT to elect a node is optimistic, UCT will choose the measures which have the highest confidence value (Upper Confidence Bound) value is the highest triumph of the sample is known [8]. In this selection process will be used algorithms Upper Confidence Bound to Tree as the election rules. The formula is as follows:

$$\frac{Q(v')}{N(v')} + \sqrt{\frac{\ln\ (N\ (v))}{5\ \times N(v')}} = UCT \qquad (1)$$

With notations:
Q = win value
N = visit cont
v = parent node
v' = child node

Q (v ') obtained from the evaluation of a node, if the node is profitable based rules of the game then the node will be assigned a value of 1, the value will increase by 1 if the child of a node is also a step in profitable, so the higher this value, the more often this node prioritized drawing this search. The value of N (v ') show how many times the node is impassable to be evaluated by the UCT algorithm, while N (v) is the number of visits from the parent node. Both of these values will increase if a child node passed / evaluated.

UCT algorithm will search for the highest value as a solution UCT search. By searching for the highest value of UCT, from the child nodes, the importance of the node that will be the optimal move for the computer player.

2.4 Games Analysis

In general flow of the game on an application that will be created will look like the image below. Players will tersu switch turn up that shows the state obtained statu win one of the players.

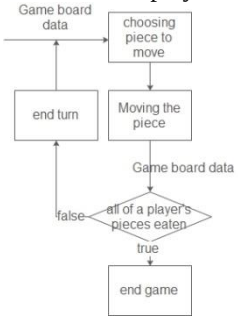


Figure 5 Game flow

In the human player decision making on the electoral process-driven pieces will be done naturally or spontaneously by relying on logic based on the rules of the game. For computer players, making such decisions must first use the search process. Just like the human player's decision-making process must also use the rules of the game as a reference.

So on this process will be implemented optimal search method step, in this case is to implement the algorithm UCT. In general the UCT algorithm implementation in this damdaman game is as shown in Figure 6.

.

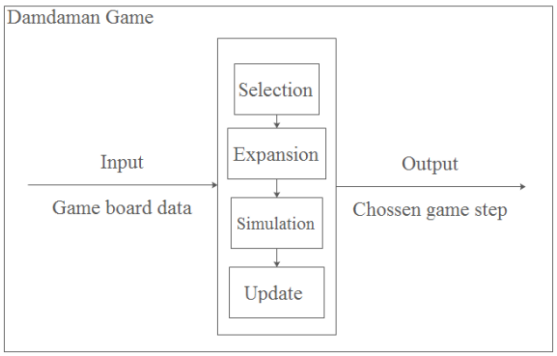


Figure 6 UCT search work flow on Damdaman

By taking data from the board game of pawns positions and measures that may be taken any pawn from the board game, then UCT algorithm will process such data by creating a search tree each leaf / node her is an array containing the state of the game board. And each node will be weighted to determine the node / step pawns which are favorable for the computer player. Node / pawns step that contains the selected measures will also be used as a reference output data to the next player steps.

Data input to be used in the search process is taken out of the game data, these data are:

a. The coordinates of each pawn, pawn coordinate input data will provide information about the coordinates pawns on the board and so the algorithm can determine the likelihood of each step of each piece.
b. Pawn's valid step, algorithm will need a valid step in the game in order to determine the simulation steps undertaken and measures to be taken.
c. The player's turn, variable game will give player's turn at UCT algorithm so that the algorithm can determine when it is appropriate to use a step results in the game.
d. Victory Status, Purpose of the game for the computer player is spent pawns human players so winning becomes true status. It will be a goal state in the search tree, so if it is achieved then the algorithm will determine what steps can be taken to achieve the goal state.

At UCT algorithm there are 4 electoral process that node, the expansion of the search tree, tree simulation search and update the value of each node. In the implementation process of updating the value will be merged in the other process and the expansion process and the simulation will run concurrently.

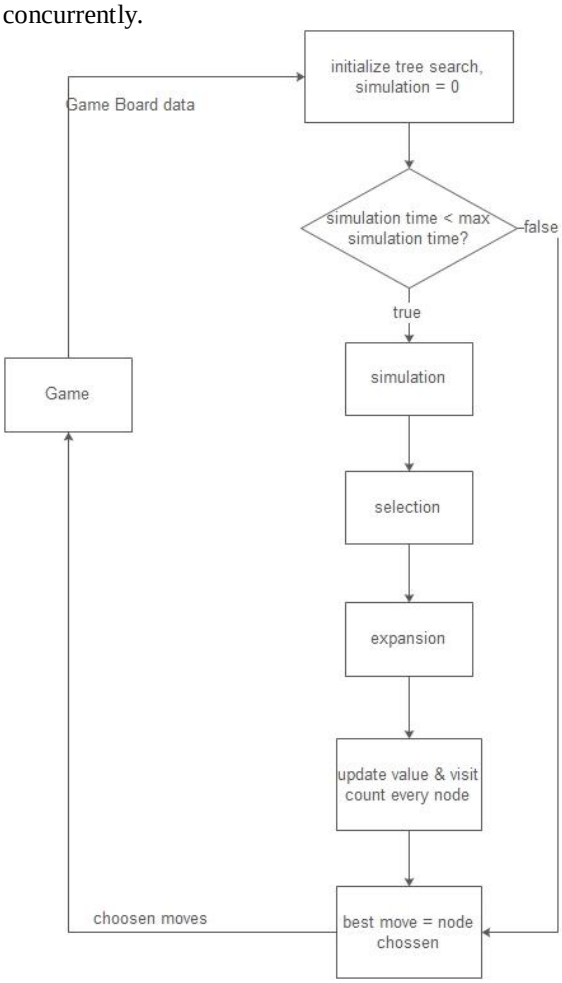


Figure 7 UCT work flow on Damdaman

At the beginning of the process will be done initialization root of the search tree. Root search tree will be taken from the input data coordinates pawn game board.

Next will be checked whether the simulation time has exceeded the maximum limit of the simulation time, because the new search begins, the value of simulation time is 0, it will be the expansion process and simulation.

In the process of expansion will be checked in the search tree if the node has children or not. At UCT algorithm, the value of the visit would be a reference to whether the algorithm will create a simulation or evaluate existing search tree. Visit value = 0 on the node, will make the algorithms run the simulation process, while for the visit above 0 it will be calculated by the formula UCT.

Because if the visit is more than 0, then the UCT value can be calculated, so that after this process, the values UCT obtained will be compared for decision making. While the value of a win will be added by 1 each time a node is simulated. However, if at the end of the search tree is a potential pathway defeat, then the value of the node to the defeat win will be updated to = 0.

Because there are only a root node in the search tree, the UCT algorithm will make the children of the root, based on the input data valid step.

Based on a step input data is valid, it will be created a new node as a child of the root is the selection steps that can be taken by the computer player in turn. Every possibility is the array to be formed into a sapling from the root by moving pawns in accordance with the relevant array. The following image is the result of the expansion process of the root.

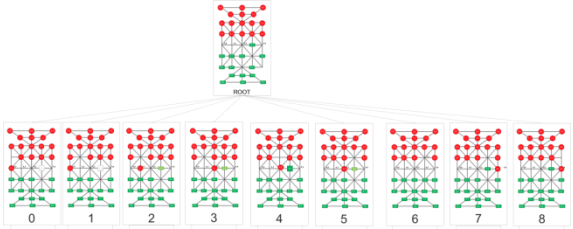


Figure 8 UCT tree search

The search tree has a value not visit / win on his node- because of impassable and evaluated, therefore, will be randomly selected nodes to be used in the simulation process.

Root node has 9 children, how selections for nodes that do not have the value of each node is to be given a UCT value with a random value, the following examples of random values prone 10,000 to 11,000 [20] given to the nodes. Node with the highest UCT score will be selected. If the selection process / benchmarking UCT value obtained at random or by the formula 1 turns out two values are the same, then it will have a value that is first discovered. It is based on a selection algorithm using the notation > (greater) in the comparison.

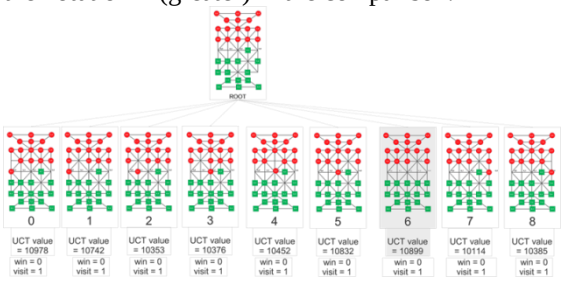


Figure 9 Tree search with random UCT value

In the simulation process, the algorithm will look for possibilities to be taken by human players as a reference for the next step. This process will be repeated with each iteration will do substitutions. Each node is undergoing a process of simulated values of his visit would be a plus 1 to the node and its parent node-. It is shown in Figure 3:15 where the parent node (node 6) has had a visit value 2 because it was done 1 time 1 time simulation and expansion

of the node. While every child node- it still has value and win visit = 1 = 0.

Win status will be obtained from the calculation of the number of human players pawns, if the amount = 0, then the opposite is true and the win if the sum> 0 then the win is false. At node 312 is a situation where human players run out of pawn. Then the nodes that leads to the node 312, will be stored in memory and will be updated value of its win value.

Table 1 UCT value of visited node

Tree search level	Chossen Node	UCT
1	6	5.824311
2	16	5.730502
3	23	5,635468
4	30	5,560089
5	45	5,441503
6	57	5,342448
7	68	5,241922
8	79	5,139849
9	91	5,036147
10	98	4,930726
11	109	4,823486
12	117	4,71432
13	119	4,603107
14	137	4,489712
15	149	4,373986
16	160	4,255761
17	169	4,134847
18	176	4,011027
19	194	3,884054
20	203	3,753641
21	212	3,619456
22	221	3,481104
23	236	3,338118
24	241	3,189929
25	253	3,035842
26	261	2,874982
27	272	2,706229
28	276	2,528101
29	284	2,338566
30	286	2,134703
31	300	1,912018
32	302	1,662906
33	312	1,37233

Its obtained node that could potentially win the computer players at every level of the search tree. at level 1 will be selected search tree node 6 as an optional step for computer players. In accordance with the node, the algorithm will generate step (3.3) is the beginning and the pawn position (5,3) is the position of the final piece. Comparison of the value of the search tree can be found in the appendix of the search tree.

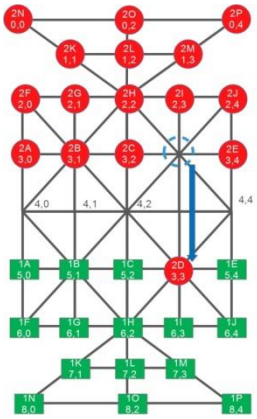


Figure 10 Move generated by UCT algorithm

2.5 Method Testing

This section will be testing the process carried out by the method / UCT algorithm on selecting Damdaman step on the game. In this test will be used several scenarios on the game board, which among others contained in the following table.

Table 2 Testing scenario

Components tested	Testing scenario	Type of test
Search result	Initial game	Selection accuracy
	State opponents pawns can be eaten	Selection accuracy
Winning precentage	Playing the game until there is a winner	Selection accuracy

At the beginning of the game, the conditions the board will look different depending on the choice of a movement that will be taken by the first player. In this example, the first player would move the pawns of coordinates (5,3) to coordinate without pawns in the coordinate (4,4). Furthermore, the search algorithm will take input data from the state of the game to produce a selection of measures appropriate to the circumstances and conditions of data input board games.

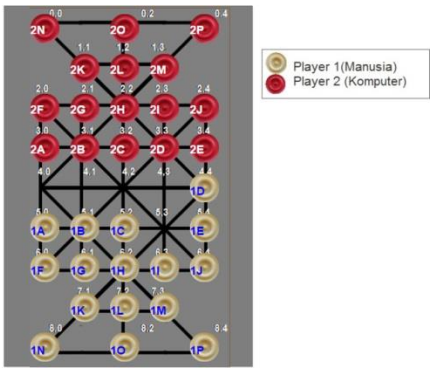


Figure 11 no eating game state

From the data obtained, will be generated search tree level 1 in accordance with the data earlier. The resulting search tree will look as follows.

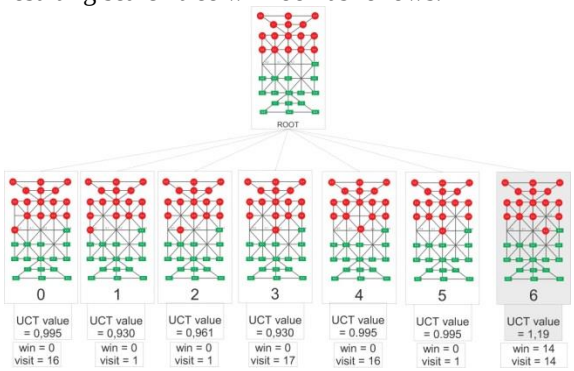


Figure 12 Tree search without eating chance

From the above picture will be taken in accordance with node 6 which has the highest UCT value from another node on the same level.

As for the state of the game board which met possibility of eating, then the algorithm will be directed to give priority to consuming pawn pawn opponent.

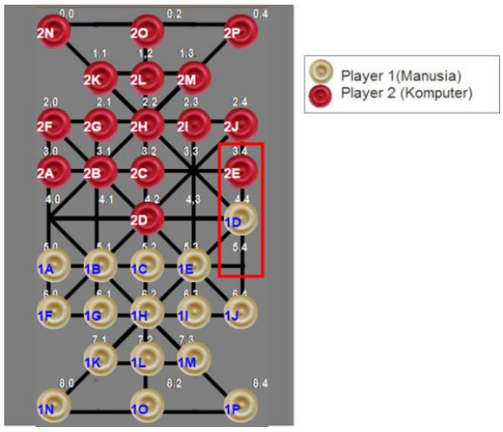


Figure 13 Game state with eating chance

From the picture above, the resulting search tree will prioritize pawns able to jump over / take pawn opponent. This is so the computer player has the possibility of better determine potential and good step according to the rules of the game Damdaman.

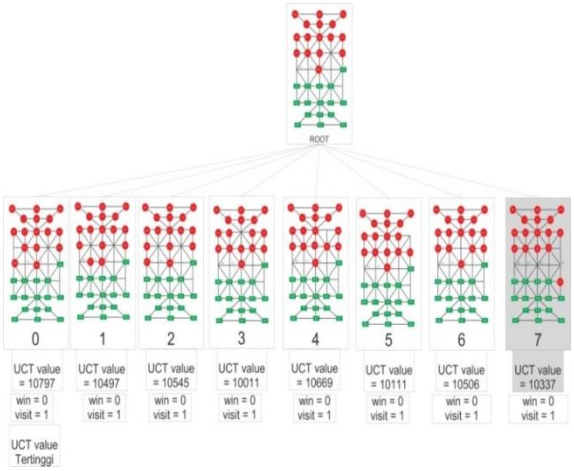


Figure 14 Tree search with eating chance

The above picture generating search tree with the final choice of selecting step moving pawn pawn the computer players to take an opposing player. In accordance with the rules of the game whereby if there is a possibility to eat, then the player should take the chance.

In the test scenario winning percentage computer players, will be the match against 5 human players in which each player will play against the computer players as much as six times. Each game will be recorded that player wins the game and the time spent on players in the game.

Table 3 Winning percentage test data

***Game* Number**	**Human Player Number**	**AI Time (second)**	**AI win status**
1	1	99,288	Lost
2	1	107,954	Lost
3	1	137,496	Lost
4	1	113,136	Win
5	1	111,176	Win
6	1	121,926	Lost
7	2	149,616	Win
8	2	140,032	Lost
9	2	125,342	Win
10	2	102,895	Lost
11	2	116,624	Lost
12	2	101,822	Win
13	3	78,331	Lost
14	3	165,452	Win
15	3	133,365	Win
16	3	117,992	Win
17	3	107,887	Lost
18	3	157,68	Lost
19	4	140,416	Win

Game Number	Human Player Number	AI Time (second)	AI win status
20	4	153,034	Lost
21	4	136,62	Win
22	4	96,668	Win
23	4	103,298	Win
24	4	88,891	Win
25	5	134,19	Lost
26	5	127,275	Win
27	5	114,24	Lost
28	5	84,756	Win
29	5	115,245	Win
30	5	136,686	Win

From the data obtained the number of computer player wins 16 times from 30 times a game or with a 53% winning percentage.

2.5 Performance Testing

This test will see how the performance of the method used, based on the memory used and the time taken by the search algorithm to obtain the selected step.

Based on the data it was concluded that the performance testing of memory used when the search is not affected by the number of pawns on the game board, but it depends on the time given to perform the search process. The more time available, the memory will be more widely used. Possibility of dining is available on an optional step will also affect the time and memory since the search will focus on a selection of these measures so that the search tree created will be smaller in size.

The values obtained is the value of search time performance with an average of 8011 milliseconds and memory performance with an average of 11 378 bytes (11 kilobytes).

3. CONCLUSION

This section will contain conclusions and suggestions. Conclusions will summarize the results obtained from this study, while advice will contain entries that the author can from the research to be used by future studies.

The conclusion of the research for the Implementation of Algorithms Upper Confidence Bounds For Trees (UCT) To Decision On Damdaman game is as follows.

1. Accuracy of selection steps taken are in accordance with the rules of the game with players winning percentage value computer generated by 53%.
2. UCT algorithm can be used to look for steps in the game Damdaman
3. Value of search time performance with an average of 8011 milliseconds and memory performance with an average of 11 378 bytes (11 kilobytes).

While the suggestion to increase the percentage of victories in the use of the algorithm UCT on Damdaman game is use of additional value as a value that is safe step that shows that if a move is likely to be hurt by the rules of the game Damdaman meal. This value can be used in variable win at UCT algorithm, such measures would harm always rated the win 0 so that the algorithms focus on profitable step.

BIBLIOGRAPHY

[1] S. Dharmamulya, *Permainan Tradisional Jawa*. Yogyakarta: Penerbit Kepel Press, 2005.

[2] A. Prasetyo, "Implementasi Kecerdasan Buatan Pada Permainan Damdaman Menggunakan Algoritma Minimax," Ilmu Komputer Universitas Pendidikan Indonesia, Bandung, Skripsi Juni 2013.

[3] A. Mustaram, "Game Development Of Traditional Damdaman," Teknik Informatika Universitas Komputer Indonesia, Bandung, Skripsi Juli 2013.

[4] Panji Febriyanto, "Penerapan Algoritma Negamax Untuk Menghasilkan Langkah Yang Optimal Pada Permainan Dam Daman," Teknik Informatika Universitas Komputer Indonesia, Bandung, Skripsi Juni 2016.

[5] C. Browne and E. Powley, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, Maret 2012.

[6] Oleg Arenz, "Monte Carlo Chess," *Darmstadt University of Technology Thesis*, April 2012.

[7] S. Gelly and D. Silver, "Monte-Carlo tree search and rapid action value estimation in computer Go," *Artificial Intelligence* , vol. 175, April 2011.

[8] M. Wiering and M. V. Otterlo, *Reinforcement Learning*, 12th ed. Berlin: Springer Berlin Heidelberg, 2012.

[9] Pressman, *Software Engineering : A Practitioner's Approach*, 7th ed.: McGrow Hill, 2010.

[10] D. H. and D. T. R. Siregar, *Permainan Tradisional Indonesia*. Jakarta: Direktorat Jenderal Kebudayaan , 1998.

[11] K. Knight and E. Rich, *Artificial Intelligence*.: McGraw-Hill, 1991.

[12] Suyanto, *Artificial Intelligence*. Bandung: Informatika, 2007.

[13] V.P. Singh, *System Modeling and Simulation*.:

New Age International, 2009.

[14] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo Tree Search: A New Framework for Game AI," *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, Oktober 2008.

[15] Yun-Ching L. and Y. Tsuruoka, "Adapting Improved Upper Confidence Bounds for Monte-Carlo Tree Search," *Advances in Computer Games: 14th International Conference*, vol. 14, pp. 53-64, Juli 2015.

[16] S. S. M. Rosa, *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Penerbit Informatika, 2014.

[17] P. Carey, *New Perspectives on HTML CSS*, 4th ed.: Cengage Learning, 2013.

[18] Marijn Haverbeke, *Eloquent Javascript*. San Francisco: William Pollock, 2011.

[19] Hariyanto B., *Rekayasa Sistem Berorientasi Objek*. Bandung: Informatika Bandung, 2004.

[20] Magnus Persson. (2013, Agustus) Sensei Library. [Online]. HYPERLINK "http://senseis.xmp.net/?UCT" http://senseis.xmp.net/?UCT

[21] M. Fowler, *UML Distiled*. Yogyakarta: Penerbit Andi, 2005.