PESQUIMAT 21(1): 1–10 (2018) ISSN:1560-912X/ ISSN-E:1609-8439
http://dx.doi.org/10.15381/pes.v21i1.15069 Facultad de Ciencias Matemáticas – UNMSM

Comparison of Expectimax and Monte Carlo algorithms in solving the online 2048 game

Efrain Noa Yarasca [1] ***y*** ***Khoi Nguyen*** [1]***.***

Abstract: In this work, two search algorithms Expectimax and Monte Carlo Tree Search (MCTS) were developed to solve the well-known "2048" puzzle online-game and compare their results. In both cases, ?ve heuristics were employed to obtain favorable tile positions within the game. These heuristics were combined to maximize the game-score in all possible board positions. As a result, the game-score, the maximum value of tile obtained, and the computing time employed in solving the game are shown. In addition, the e?ciency of each algorithm and its sub-cases are presented. This research concludes by arguing that Monte Carlo Tree Search was more e?cient in higher score than Expectimax algorithm, although in a longer time. Increments in level of depth-search in Expectimax and number of moves in MCTS do not necessarily resulted in obtaining higher score.
Keywords: 2048 game, Expectimax algorithm, Monte Carlo algorithm, heuristics.

Comparación de los algoritmos Expectimax y Monte Carlo en la solución del juego en línea 2048

Resumen: En el presente trabajo, dos algoritmos de búsqueda: Expectimax y Monte Carlo fueron desarrollados a ?n de resolver el conocido juego en linea "2048" y comparar sus resultados. En ambos casos, cinco heurísticas fueron empleadas para obtener posiciones favorables de las ?chas dentro del juego. Estas heurísticas fueron combinadas convenientemente para maximizar el puntaje del juego en todas las posibles posiciones. Como resultado el puntaje, el máximo valor de ?cha, y el tiempo de cómputo empleado en el juego son mostrados. Adem´as, la e?ciencia de cada algoritmo y sus subcasos son presentados. El presente trabajo concluye que el algoritmo de búsqueda Monte-Carlo fue más e?ciente en obtener un mayor puntaje que el algoritmo de Expectimax, aunque en un tiempo de cómputo mayor. Incrementos en el nivel de búsqueda en el algoritmo Expectimax y el número de movimientos en el algoritmo de Monte Carlo no necesariamente resultaron en un mayor puntaje del juego. **Palabras clave:** Juego 2048, Algoritmo Expectimax, Monte Carlo, heuristicas.

Recibido: 21/04/2018. *Aceptado*: 18/06/2018. *Publicado online*: 30/06/2018

[1]Oregon State University, School of CCE, Corvallis, OR 97331,USA. e-mail: noayarae@oregonstate.edu
[1]Oregon State University, School of EECS, Corvallis, OR 97331, USA., e-mail:nguyenkh@oregonstate.edu

1 Introducción

The "2048" puzzle is a single-player online game that has rapidly gained gamers attention (http://2048game.com/). It consists of 4x4 grid with numbered tiles that can be slipped by the player in four available directions: up, down, left, and right (basic rule). Rules for this game consider that two neighboring tiles with same number can be merged into a single tile with the sum of their values. Thus, some tiles increase their values at each turn. Initial state for "2048" game is given by two filled cells (tiles) with either 2 or 4 randomly placed into the 4x4 board, and the final state is given when the game is won or lost. The player wins when the value of "2048" is obtained on some tile on the board and loses when there are no legal moves to make [3].

Since the game was released in March 2014, it has gained thousands of followers. Online foros for discussing strategies of solution were opened. Artificial-intelligence researchers have rapidly started to develop some algorithms to obtain high score and maximum tile value in the game. Chowdhury G. and Dhamodaran V. [1] attempted to solve 2048 by using Q-Learning and Expectimax approaches. While memory limitations due to large state space were found by Q-Learning approach, 90% of won games were obtained by Expectimax approach. Maintaining the four cells of the left edge filled was used as a heuristic. Rodgers and Levine [4] discuss application of Monte-Carlo Tree-Search and Averaged Depth Limited Search in 2048 solving. Xiao [5] implemented an Expectimax algorithm considering bonuses for empty squares and placing large values near edges and corners as heuristics. Later, Xiao [6] released some improvements and recommendations to its algorithm by "StackOverflow" web-page.

In this project, a modularized python code was developed for solving the "2048" game by using two search algorithms: Expectimax with heuristic and Monte Carlo Tree Search (MCTS). Performance of these algorithms were evaluated in 100 experiments for each case. MCTS involved five cases with different maximum number of random moves: 100, 150, 200, 250 and 300. Expectimax algorithm involved five cases, one for each depth-search established on: 2, 3, 4, 5, and 6. Regardless of winning or losing the game, experiments were scored by three indicators: the highest tile-value obtained, the sum of merged values (game score), and computational time consumption (CTP). Towards the end of this report a discussion of results are presented in tables and plots. Finally, the report provides overall conclusions.

2 Description and Methods

Game Description:

Developed by Gabriele Cirulli, the online game "2048" rapidly became widespread among gamers since its releasing. Available online (http://2048game.com/), it is a single player game on a square board of 4-by-4 cells, which are partially filled with tiles numbered with powers of 2. Rules and game steps are described below.

(1) Given an initial state, which shows only two filled tile with either 2 or 4 (randomly placed into the board), the player may slide the board in four directions: up, down, left, and right (Figure 1).

(2) One game turn means one sliding of board tiles. (3) At each turn, tiles also slide as far as possible within the board. If two tiles with same number collide, they will merge into a unique tile obtaining the sum of both tile-numbers. For example, Figure 3a shows two intermedia sequential states. In the left side, rectangles show pair of tiles that will be merged on the next state (right side) if the player chooses to swipe tiles toward up.

(4) After each turn, new tile arrangement will be obtained with the value 2 or 4 with

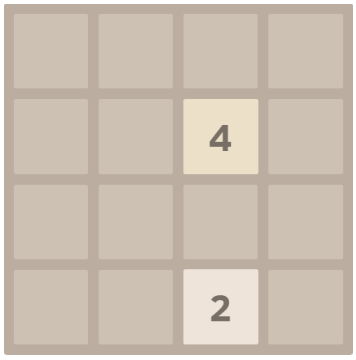


Figure 1: Initial state of the game with two tile values randomly placed

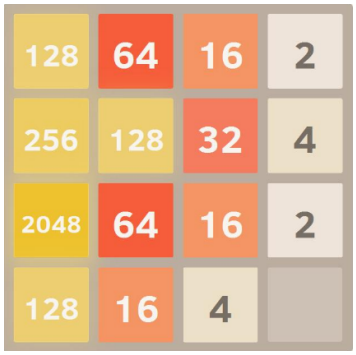
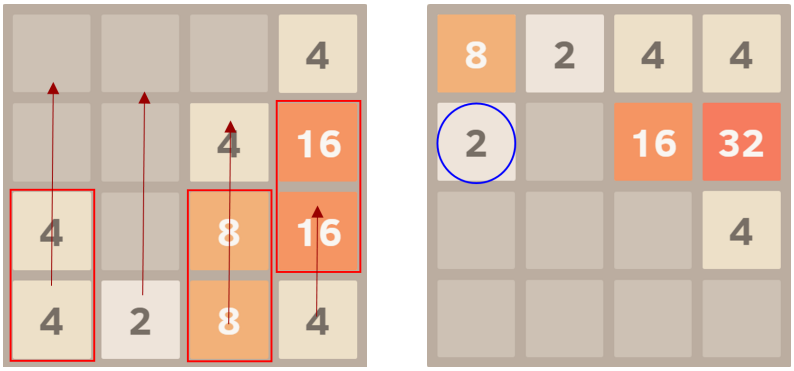


Figure 2: Winning state. No move is available to keep going



(a) a (b) b

Figure 3: Two intermedia sequential states. (a) Red rectangles show pair of tiles that will be merged on the next state if the player chooses to swipe tiles toward up. (b) Tiles slipped as far as possible to top border, merged tiles, and a new tile in blue circle

probabilities of 0.8 and 0.2 respectively, new tile values will be obtained if some merged, and a new tile will appear placed randomly in an empty cell. For example, the right side of Figure 3b shows all the tiles slipped as far as possible to the top border, merged tiles, and a new tile in circle

(5) Thus, values in merged tiles will increase frequently. (6) The player's goal is to obtain a tile with a value of 2048 (2^{11}), however the player may remain playing to attain higher score or higher tile value ($2^{12}, 2^{13}, \dots$) until no legal move is available (Figure 2, side shows also no more legal moves available). (7) No moves are available when there are no empty cells and no possible tiles to merge then game is over. (8) At each merge of tiles, their numbers are summed and then accumulated to provide a score called "game score". (9) The final state is given when the game is won or when no more legal moves are available

Heuristics:

Five heuristics were considered to obtain favorable tile positions. All heuristics were combined to maximize the game-score of all possible boar positions. (1) Maximum tile value in a corner. This heuristic grants bonus to those board positions with the highest tile value in any of the four corners. (2) Maximum empty cells. This heuristic penalizes the positions of the board with less empty cells, since fewer empty cells implies a cramped board with risk of running out of options in the coming turns. (3) Monotonicity along a snake-like path, rows and

columns (Figure 4). This heuristic seeks to ensure ascending and descending tile values along a snake-like path, rows, and columns. This heuristic also involves the first heuristic mentioned above that states the highest tile value should go on the corner. This heuristic tries to maintain an organized board with smaller tiles following and filling up to larger tiles. (4) Maximum tile value of a line along an edge. This heuristic in some way is related to the previous heuristic. It seeks also to obtain an organized ascending or descending board. (5) Neighbor tiles with equal values to be merged. Merging neighbor values help to obtain a clear board with more empty cells.

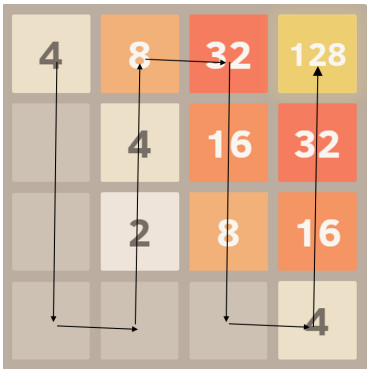


Figure 4: Scheme of snake-like path considered in heuristic of monotonicity

Expectimax algorithm:

This depth-limited search takes turn between expectation and maximization. In expectation, it evaluates all the possible tile-values and tile-location of the next generations, and optimizes based on weights according to the probability of each possibility (20% for 4 and 80% for 2). In maximization, it assesses all possible moves and selects the choice with best score. The tree search finished when it reached the predefined depth limit, or when it reached a highly unlike board state.

Monte Carlo Tree Search (MCTS):

In this case, given a state board the next move was determined by playing the game employing random moves until the end (no possible move can be obtained). This process was repeated several times storing the track and the end-score. Then, the starting move with highest average end-score was selected as the next move. Despite a random selection of possible games, it gave an excellent game since the best end-scored game of attempted games was selected for the next step. In this work, the number of random moves was established as a variable, thus five number of moves were taking into a account Lanctot et.al [2].

3 Experiment Setup

Five cases of depth-search were considered for Expectimax algorithm: 2, 3, 4, 5 and 6. For each depth-search, 100 experiments were conducted. Regarding Monte Carlo Tree Search, five cases of maximum number of random moves were considered: 100, 150, 200, 250 and 300 moves. Note that for MCTS the sample width (minimum of sample needed to evaluate a move) were kept in 20. For each case, also 100 experiments were carried out. In all cases, the initial states

were randomly considered as established by the rule of the game (Table 1). Experiments were executed through a python code which were developed and implemented for this project. These codes are available at `https://github.com/noayarae/games`.

Table 1: Experiments for each algorithm

Expectimax	Number of experiments	Monte Carlo Tree Search	Number of experiments
Depth-search = 2	100	Max-moves = 100	100
Depth-search = 3	100	Max-moves = 150	100
Depth-search = 4	100	Max-moves = 200	100
Depth-search = 5	100	Max-moves = 250	100
Depth-search = 6	100	Max-moves = 300	100

4 Results

Table 2 shows a summary of results corresponding to mean of scores (Mean score), max tile reached (Max tile), median of tiles reached (Median tile), mean of tiles reached (Mean tile), and mean of computational time consumption-CTC (Mean time consumption) for both algorithms. CTC time, measured in seconds, is the amount of time that the Central Processing Unit (CPU) employed in processing the python code instructions for solving "2048" game. Table 3 shows a summary of experiment results grouped according to the maximum value of tile attained. This table also represents efficiency since the number of experiments for each sub-case is 100.

Table 2: Mean score, Max-tile, Median of tiles, Mean of tiles, and Mean of computing time

Algorithm	Depth search - # of moves	Mean score	Max tile	Median tile	Mean tile	Mean time spent (sec)
Expectimax	2	2836	2048	512	472	5.0
	3	3812	2048	512	610	24.4
	4	2991	2048	256	474	32.8
	5	3318	2048	512	531	142.1
	6	4157	2048	512	642	301.0
	100	13410	4096	2048	1720	395
Monte Carlo Tree Search	150	15899	4096	2048	2015	663
	200	15877	4096	2048	2010	874
	250	16970	4096	2048	2148	1251
	300	15843	4096	2048	1987	1364

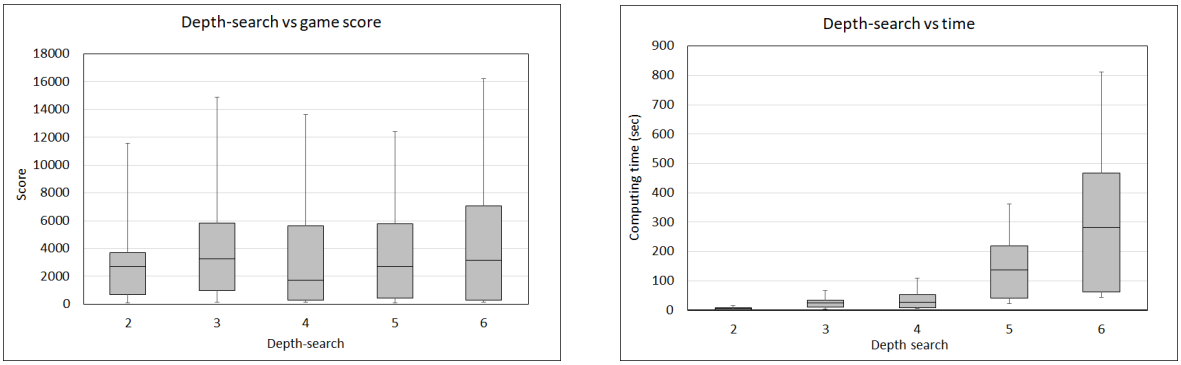
5 Discussion

Expectimax algorithm.

Figure 5(a) shows plots between depth-search and score reached. Singular results were obtained for depth-search level 3. Interestingly depth-search at level 3 provided better results than depth-searh at level 4 and 5, and almost same as depth-search level 6. Theses results suggest

Table 3: Results grouped by the maximum value of tile obtained

Tile	Expectimax Algorithm Depth search					Monte Carlo tree search Number of moves				
	2	3	4	5	6	100	150	200	250	300
16	1	0	1	0	0	0	0	0	0	0
32	9	4	19	13	17	0	0	0	0	0
64	13	15	15	15	21	0	0	0	0	0
128	6	8	8	8	1	0	0	0	0	0
256	13	8	9	6	1	0	1	1	1	0
512	36	27	19	26	21	6	5	2	3	2
1024	21	34	26	29	28	29	18	15	12	21
2048	1	4	3	3	11	62	64	74	70	68
4096	0	0	0	0	0	3	12	8	14	9



(a) (b)

Figure 5: (a) Depth-search vs score reached. (b) Depth-search vs computing time in seconds

to state that depth-searh 3 is enough to obtain acceptable results for the game. Regarding time, Figure 5(b) shows a monotonic relation between depth-search and computational time consumption. It appears there is a threshold at depth-search level 4, since after this point increases in depth-search resulted in exponential increases in computing time.

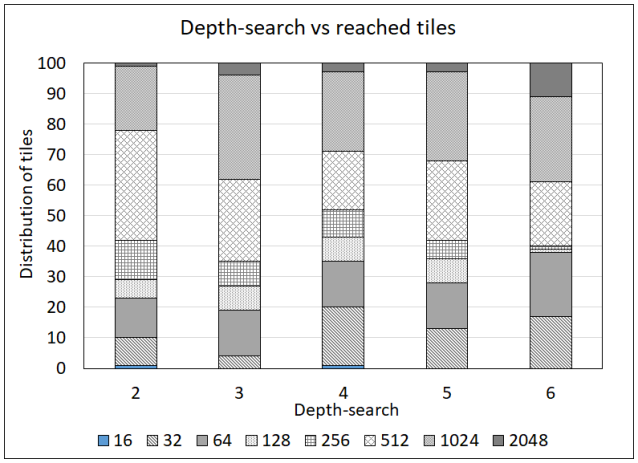
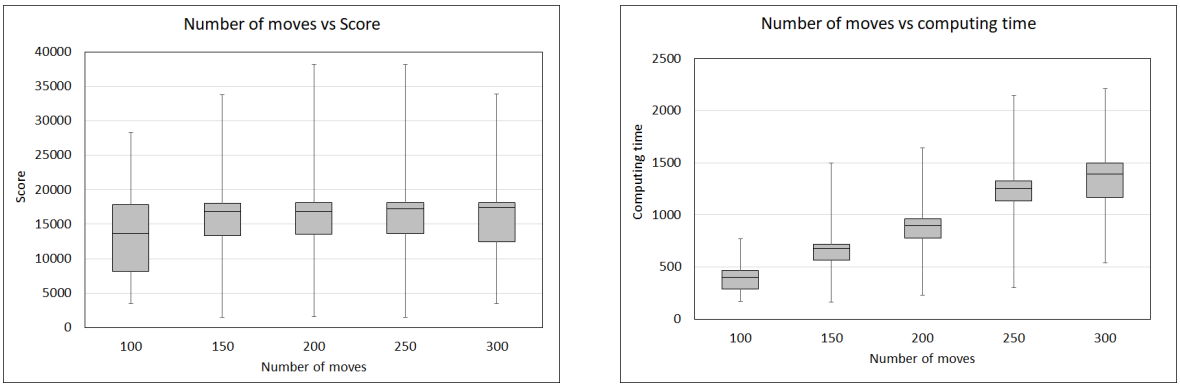


Figure 6: Distribution of tile-values reached on each depth-search case. Notice that depth-search at level 3 shows better performance that others.



(a) (b)

Figure 7: (a) Number of moves of Monte Carlo Tree Search vs game score reached. (b) Number of moves of MCTS vs computing time in seconds

Figure 6 shows the the distribution of tile-values reached on each depth-search case. Although, depth-search at level 6 resulted in a greater number of games won, depth-searching at level 3 is highly competent as well. For instance, depth-search at level 3 presented greater achievements of tiles equal to 1024 than other levels.

Monte Carlo Tree Search.

Relationship between number of moves and score follows a monotonicity behavior, although with a low gradient after 150 moves. While from 150-moves to 300-moves the mean-score shows small rises (almost constant), the computing time for those moves keeps ascending constantly. It can be inferred that in the cases of 150 and 300 moves, tiles values are getting up trying to reach goals even higher than 2048 (goal to win) but not all of them reached 4096 or more (Figure 7(a) and (b)). As many games did not reach tile-values equal or higher than 2048, the average of tile-values remained low. Regardless of whether or not tiles have reached 2048 or 4096, or have failed in their attempt to reach, the computing time keeps running. In short, increase in computing time should go associated to increases on number of moves and level of difficulty. Figure 7(a) and (b) that show the relationship between number of moves and score reached and computing time support this assertion.

Figure 8 shows histograms of maximum-tiles reached by using Monte Carlo Tree search. This algorithm won at least 65% of games. Interestingly with this algorithm, the case with greater moves (300-moves, 77 games won) was not one that won more games but the case with 250 moves (84 games won), followed very close by the case of 200 moves (82 games won). Hence, cases with 200-moves and 250 moves resulted better in performance for having won more games.

5.1 Trade-off analysis

Regarding expectimax algorithm, relationship between score of tries and time shows a strong association with Pearson coefficients close to 1. However, since the optimal scenario is to win the game employing minimum time these two variables are conflicting (Figure 9). Pareto front for each depth-search can be drawn as a first order function (line). In the same way, relationship between maximum-tile and time resulted associated but strong conflicting. On the other hand, relationship between score and maximum-tile resulted strong redundant and associated with Pearson coefficients close to 1 as well.

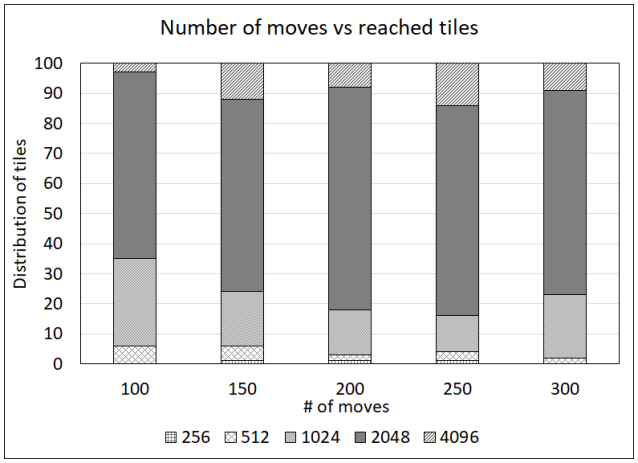


Figure 8: Distribution of tile-values reached on each number of moves of MCTS. Notice that depth-search at level 3 shows better performance that others.

Table 4: Pearson Correlation Coefficient (PCC) values and Trade-offs between the objective functions for depth-search 2, 3, 4, 5, and 6

Pair of variables	Pearson Coefficient					Trade-off
	h = 2	h = 3	h = 4	h = 5	h = 6	
Score - Time	0.99	0.98	0.99	0.98	0.96	Strong conflicting
Max. tile - Time	0.96	0.95	0.96	0.95	0.94	Strong conflicting
Score - Max. tile	0.98	0.97	0.98	0.97	0.98	Strong redundant

Relationship between score and max-tile shows a common sense association: as the value of the tile increases, the value of the score also increases. Trade-off between these variables are strongly redundant. One thing that is important to highlight is that the game is won with scores over 10,000, whatever be the depth-search level (The first game won was at 10708 of score). This does not mean that reaching a score of 10000 ensures winning the game, but it can be inferred that by reaching a score of 10000, the goal to win the game is close.

With respect to MCTS algorithm, results showed similarities to expectimax, strong association among score, max.tile and time with Pearson coefficients close to 1. Trade-off between Time and Max.Tile, and Time and Score resulted to be strong conflicting. Figure 10 shows also a global Pareto front in continuous line and Pareto front for 300-moves in dashed lines. If score is prioritized before the goal (win the game), it would be convenient to use a MCTS process with the least number of moves.

Table 5: Pearson Correlation Coefficient (PCC) values and Trade-offs between the objective functions for 100, 150, 200, 250, and 300 moves

Pair of variables	Pearson Coefficient MCTS					Trade-off
	100 m	150 m	200 m	250 m	300 m	
Score - Time	0.87	0.95	0.93	0.94	0.86	Strong conflicting
Max. tile - Time	0.79	0.94	0.89	0.90	0.79	Strong conflicting
Score - Max. tile	0.98	0.97	0.94	0.95	0.93	Strong redundant

Relationship between score and max-tile resulted strongly redundant. By this algorithm like expectimax, tile values equal to 2048 (goal to win the game) started after scores of 10,000 (first won game was at 11,610 of score), whatever be the number of moves. However, by MCTS

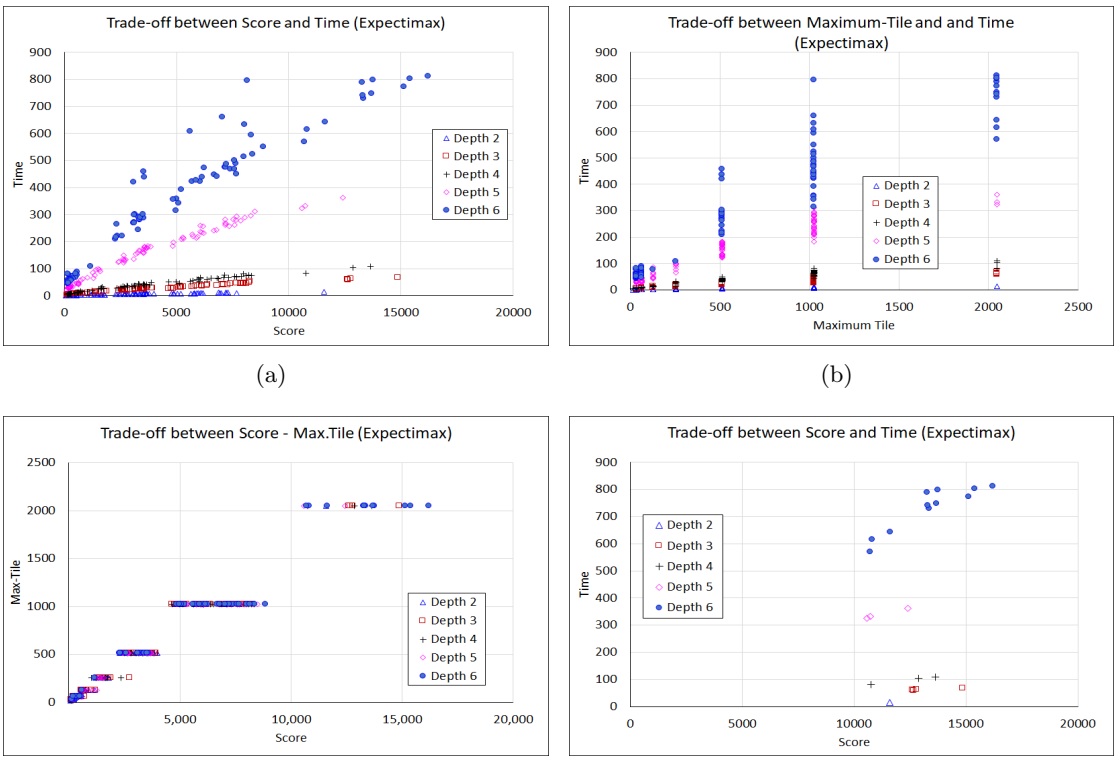


Figure 9: Trade-off of expectimax results: (a) trade-off between score and time; (b) trade-off between max-tile reached and time; (c) trade-off between score and max.tile reached. Scores for achieving 2048 tile start after 10,000; and, (d) trade-off between max.tile reached and time considering only winner experiments.

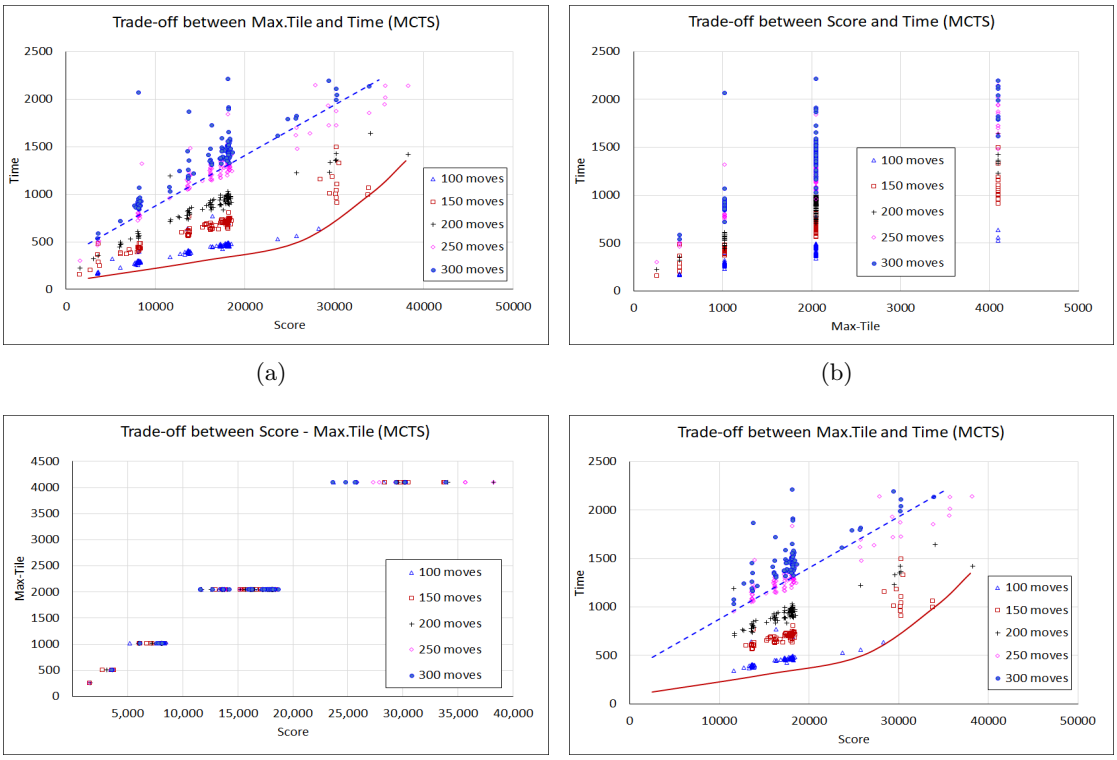


Figure 10: Trade-off of MCTS results: (a) trade-off between score and time; (b) trade-off between max-tile reached and time; (c) trade-off between score and max.tile reached. Scores for achieving 2048 tile start after 10,000; and, (d) trade-off between max.tile reached and time considering only winner experiments.

algorithm were able to obtain tiles with values of 4096 (the next level of tale value: 2048), starting at 26,600 of score, which were not obtained by expectimax.

Regardless of subcases (depth-search in expectimax or number of moves in MCTS), feasible space of games won by MCTS is larger than that obtained by expectimax as it is noticed in Figure 10(d) and Figure 9(d).

6 Conclusion

Monte Carlo Tree Search algorithm was able to solve more games than Expectimax although in a longer time. In Expectimax, a depth-search at level 6 achieved to obtain more games won than other depth-search levels with an efficiency of 11% followed by depth-search level 3 (4%) which resulted more competent than levels 4 (3%), 5 (%), and 2 (1%). Regarding MCTS, the highest performance was obtained by 250-moves with (84%), followeb by 200-moves with 82% and 300-moves, 150-moves, and 100-moves with 77%, 76% and 65% respectively. Hence, feasible space by MCTS algorithm regarding games won is highly larger than that obtained by Expectimax.

In both algorithms, trade-off between score-game and maximum-tile resulted strong redundant with pearson coefficiets close to 1. Trade-offs between score-game and time, and maximum-tile and time resulted strong conflinting with pearson coefficients close to 1 as well. In both cases, the target value (2048) is reached with scores greater than 10,000, i.e. the game is won at scores higher than 10,000. However, this does not mean that reaching a score of 10,000 ensures winning the game, but it can be inferred that by reaching a score of 10000, the goal to win the game is close. Regarding to computing time, Expectimax optimization offers a better alternative than MCTS algorithm although with a low efficiency (no more than 11%).

Finally, this article concludes arguing that Monte Carlo Tree Search was more efficient in higher score than Expectimax algorithm, although in a longer time. Increments in level of depth-search in Expectimax and number of moves in MCTS do not necessarily resulted in obtaining higher score.

References

[1] Chowdhury G., and Dhamodaran V. (2014). *2048 Using Expectimax*. University of Massachusetts Lowell Department of Computer Science.
`http://www.cs.uml.edu/ecg/uploads/AIfall14/vignesh_gayas_2048_project.pdf`

[2] Lanctot, M., Saffidine, A., Veness, J., Archibald, C., and Winands, M. H. M. (n.d.). *Monte Carlo * -Minimax Search*. Maastricht University, Netherlands.
`https://dke.maastrichtuniversity.nl/m.winands/documents/mc_star_minimax.pdf`

[3] Neller, T. W. (2015). *Pedagogical Possibilities for the 2048 Puzzle Game*. The Journal of Computing Sciences in Colleges 30.3 (January 2015), 38-46.

[4] Rodgers P. and Levine J. (2014). *An Investigation into 2048 AI Strategies*. Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK.
`https://www.cse.unr.edu/šushil/pubs/newestPapers/aux/paper_106.pdf`

[5] Xiao R. (2014). *2048 ia*. GitHub repository.
`https://github.com/nneonneo/2048-ai`

[6] Xiao R. (2017). *What is the optimal algorithm for the game 2048?*.
`https://stackoverflow.com/questions/22342854/ algorithm-for-the-game-2048`