Two versions of 2048

Aarchi Agrawal
20250017
aarchia@iitgn.ac.in
Indian Institute of Technology
Gandhinagar, Gujarat

ABSTRACT

2048 is a puzzle game based on sliding numbered tiles. In the game, tiles consist of numbers like 2,4,8,16,32,64 and so on, that is power 2 numbers. The main purpose of the game is to slide numbered tiles on a grid to combine them to create a tile having the value 2048. The game is very fascinating as well as popular. People spend a lot of time trying to make a 2048 tile on the game grid. Through this work, two variations of the game are shown. In the first version, the user can play the game and win it by getting a 2048 tile. Whereas in the second version the computer itself plays the game 2048 intelligently using minimax algorithm to take the best move possible.

KEYWORDS

Minimax algorithm, heuristic function, pruning, adversarial game

1 INTRODUCTION

The 2048 game is a fun game with tiles that are numbered and can slide in four directions. The purpose of the game is to slide numbered tiles of the grid to combine them in such a way that they create a tile with the number 2048. The player can slide the numerical tiles in four directions and the target is to win the game by getting a 2048 numbered tile on the board without causing the grid to overflow. There are many challenges in the game like the addition of random tile after every move, the constraint that the board has an empty tile, or else the game ends.

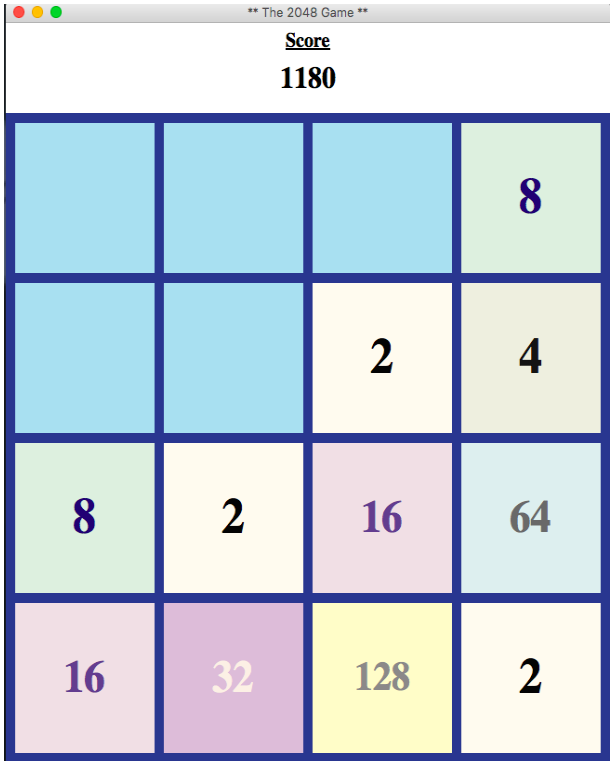

Figure 1: The 2048 game version 1

The above figure shows an instance of the first version of the 2048 game that is created. In this version, the player can the 2048 game using the up, down, left, right arrow keys with the aim of getting the 2048 tile. When the player gets a 2048 tile on the grid the player wins. In this case, when the user is not able to get a 2048 tile on the grid and all the tiles are filled with some numbers, such that no move is possible then the game ends and the player loses the game.

Figure 2: The 2048 game version 2

In the second version of the 2048 game computer plays the 2048 game. Here, the game of 2048 is played with the help of the Minimax algorithm. It is treated as an adversarial game where the player is a computer whose aim to maximize the value of the highest tile in the grid and the opponent here is that computer turn which randomly places tiles in the grid in such a way that it leads to minimization of the maximum score.
The game can go on after getting 2048 tile, creating tiles with larger numbers. The game ends when all the tiles are filled and no move is possible that is it is not possible to combine two tiles. The use of the minimax algorithm is helpful in this case as the game is played in such a way that one player tries to maximise the tile value whereas the other player adds a random tile to the grid and tries to minimise the chance of getting the goal state of having 2048 numbered tile. This version is played in the terminal and an instance where the computer wins the game is shown in figure 2.

2 RELATED WORK

The original version of 2048 was developed by Gabriele Cirulli[2] in a single weekend to test if he could program a game from scratch. It was inspired by an iOS game Threes. James Vincent of The Independent called 2048, "a clone of a clone". [7]AI system that would play 2048 on its own was one of the works that won the second prize of a coding contest at Matlab Central Exchange.[3] Many works in the development of 2048 include using alpha-beta search, a well-known game search method for two-player games, and expectimax search, often used in stochastic games.
The work done by Szubert and Jaśkowski is based on Temporal Difference (TD) learning along with n-tuple networks for 2048. Using this they successfully achieved the rate of reaching 2048-tiles as 97% but the TD learning method is found to maximize the average scores and is not driven to reach large tiles.[5] In the paper titled "Multi-Stage Temporal Difference Learning for 2048-like Games" by Kun-Hao Yeh et al.[1], they have proposed a multi-stage temporal difference learning, which is a type of hierarchical reinforcement learning method, to increase the performance for the rates of reaching large tiles, which is considered as good metrics to analyze the power of 2048.

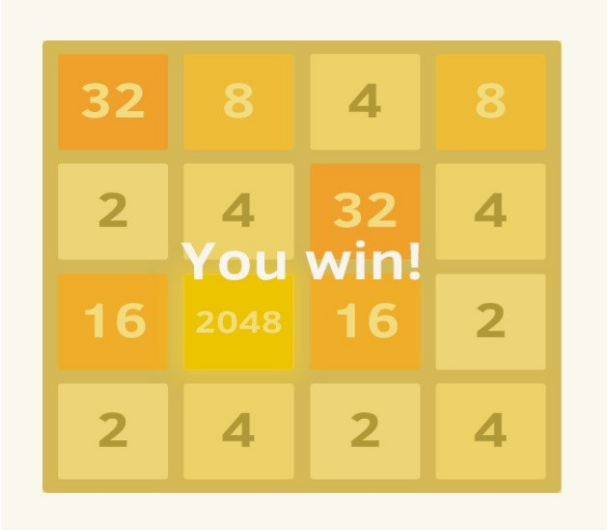

Figure 3: The original 2048 game

3 METHODOLOGY

In this section, the working of two variations of the game 2048 is discussed. In the first version where the player can play the game by using the arrow keys and in the second version computer plays the game 2048.

3.1 2048: First version

The game of 2048 is very interesting and even compulsive to most human players, the reason being the simple rules yet hard to win the game. So for the first version of the 2048 game, a user interface is created. Originally the game is a 4x4 grid with numbered tiles, but in this version, the player has the liberty to select the grid size for which the player wants to play.

The game starts by asking the player the choice of grid size for which the player wants to play. Once a player enters the grid size, the game begins with the desired board size along with 2 tiles randomly placed on the board. The player can move left, right, up, down to play the game. In every turn, a random new tile appears in any empty place available on the board, with a value of either 2 or 4. As per the input direction that the player gives, all tiles on the grid slide as far as possible in that direction, until either tile collides with another tile or the tiles collide with the edge of the grid. If two tiles with the same number collide, the tiles merge into a single tile with a value twice as that of the individual tiles. Scores are given after each merge with the value equal to the value of the tile formed by merging. By different observations, it is estimated that a score higher than 20000 in a 4*4 grid leads to getting a 2048 tile on the game board.

The main constraint of the game is to get a 2048 tile in such a way that the board is not stuck with the tiles having no possibility to either move or merge. If such a situation arises then the player has to abort the game. The game can continue even after a 2048 tile is achieved only if any possible moves are left that can be played.

Python's Tkinter library is used to creating the player's interface as illustrated in Figure 1. Each numbered tile has been colored differently to distinguish better between the numbers. The dynamic score of the player is updated with every move and can be seen at the top of the game board. This version is inspired from [6]. Different functions used for playing the 2048 game, moving and merging the tiles on the board is as follows :

- **stack() :** This is used to compress all the non-zeroes numbers to one side of the board removing the gap in between the tiles.
- **combine():** The function is used to add all horizontally adjacent non-zero numbers of the same value and merges to the left side.
- **reverse ():** This is used to reverse the order of each row of the matrix as and when required, according to the input given by the player.
- **transpose():** This is used to flip the game board over the diagonal.
- **addnewtile() :** In order to randomly add a new tile(2/4) after each move to an empty cell.
- **left() :** When the player presses the left arrow key this function is called and it is the combination of the stack, combine, stack function in this specified order.
- **right() :** When the player presses the right arrow key this function is called and it is the reverse of the left function.
- **up() :** When the player presses the up arrow key this function is called and it is the equivalent to the transpose of the left function.
- **down() :** When the player presses the down arrow key this function is called and it is the combination of transpose and then reverse of the left function.

Other functions for checking the horizontal and vertical possible moves, updating the GUI, dynamic score update are also used. And finally, we have functions that check the board and display the prompt "You win :)" if there is a 2048 tile on the game board or else the prompt "You lose :(" is displayed to the player.

3.2 2048: Second version

2048 game has been able to attract many programmers to develop artificial intelligence (AI) programs to play it. This is because of the fact that the game has challenging situation with simple protocols to follow in order to successfully complete the game.

The rules of the game remains the same with the objective of getting a 2048 tile on the board in order to win. The major change in the second version, when compared to the first version, is that the player in the second version is the computer itself and it is played on a terminal. Minimax algorithm which is a recursive algorithm is used for the decision making process and for maximizing/minimizing the move. This version is inspired from [4]. The 2048 game is treated as an adversarial game where the computer player attempts to maximize the value of the highest tile in the grid and the opponent computer player attempts to randomly places tiles in the grid to minimize the maximum score. Minimax algorithm would be suitable in this case. The game involves moving tiles with the target of increasing the score along with the constraint of not filling the board completely and a random tile is added after every move.

In the minimax algorithm there are two players, one player is called Min and other is called Max. Both the players play alternatively. The played named Max moves first and its objective is to maximize a heuristic score. While the aim the player named Min is to minimize the score. For every player, a minimax value is computed. This value is the best achievable payoff against their play. The move with the optimum minimax value is chosen by the every player. But the number of nodes that this algorithm explores is huge. So pruning is used. Here alpha-beta pruning is employed. Alpha–beta pruning is adversarial search algorithm that helps in decreasing the number of nodes that are evaluated by the minimax algorithm in the search tree. Here we have used depth limit of 4 for pruning.

- **Minimax Algorithm** The game starts with two tiles randomly placed on the board. Then MAX player that is computer only, makes the first move with the help of the minimax algorithm. While using the minimax algorithm, the MAX selects the move (UP, DOWN, RIGHT and LEFT) for finding the possible children nodes. After that the opponent player randomly generates a 2/4 tile and can place it in any of the empty cells. So it can be concluded that, for Max's move, there will be at most 4 children corresponding to each and every direction. And for MIN's move, the number of children will be 2*n where, n is the number of empty cells in the grid. The entire process goes on a loop until the game is over.
- **Heuristic Function** It has been found by various experiments and observations that the game will head positively if the highest valued tile is in the corner while other tiles are linearly decreasing in value as they moves away from the highest tile. Hence, the best four possibilities are : Maximum tile can be at the (1) Down -left (2) Top-left (3) Top-Right and (4) Down-Right corner. For computing the score, multiply the current configuration with a gradient matrix associated with each of the above defined possible cases.

Components of this version are:

- **Gamemain :** It starts the game on the terminal by loading both the computer players and then they compete with each other. The time for the move is set as 2 seconds.
- **Display :** It is used to control the visual properties of the game board on the terminal.
- **Grid :** Used to define the grid object and includes useful functions for the grid.
- **Helper :**Includes the eval function which is used to evaluate the heuristic score for a given configuration and other utility functions are defined.
- **Minimaxa :** The minimax algorithm along with alpha-beta pruning is defined.

- **Max_Player :** The Max player functioning is defined that gets the next move for the player using Minimax Algorithm.
- **Min_Player :**The Min player functionality, here The getMove() function is present that returns a tuple (x, y) indicating the place you can place a tile.

Using all the functions and minimax algorithm with pruning computer solves 2048 intelligently. If there were no time and space constraints then there was no need for pruning, and still, the performance would remain the same. But pruning is used to meet real-life time constraints and space constraints.

4 CONCLUSION

In this work, two versions of the game 2048 are created. The traditional 4x4 game can now be played of the board size of your choice. The first version is GUI-based for a human player to play the game using the arrow keys, get the scores and win by getting a 2048 tile. In the second version computer intelligently plays the 2048 game using minimax algorithm and alpha-beta pruning to get the best possible moves and successfully win the game with a 2048 tile on the board.

5 FUTURE SCOPE

Some of the things that can be added or improved are as follows:

- Use Deep-Reinforcement Learning to train a Neural Network To Play 2048.
- Finding more powerful heuristic function by learning more about the different patterns formed in the game which can be more effective in terms of providing better metrics related to time and space complexity.
- Better understanding of the game by doing mathematical analysis, using Mathematical Induction, Number Theory, Fuzzy Theory and Topology, this will help in the development of optimal strategies to ensure victory.

REFERENCES

[1] Kun-Hao Yeh; I-Chen Wu; Chu-Hsuan Hsueh; Chia-Chuan Chang; Chao-Chin Liang; Han Chiang. 2016. Multistage Temporal Difference Learning for 2048-Like Games, Vol. IEEE Transactions on Computational Intelligence and AI in Games. https://doi.org/10.1109/TCIAIG.2016.2593710

[2] Gabriele cirulli. 2014. Game 2048 [Online]. http://gabrielecirulli.github.io/2048/.

[3] Athi. 2048 Game Solver. 2014. https://www.mathworks.com/matlabcentral/fileexchange/46483-2048-game-solver.

[4] SrinidhiRaghavan. 2017. https://github.com/SrinidhiRaghavan/AI-2048-Puzzle.

[5] M. Szubert and W. Jaśkowski. 2014. Temporal difference learning of n-tuple networks for the game 2048, Vol. IEEE Conference on Computational Intelligence and Games (CIG). https://doi.org/10.1109/CIG.2014.6932907

[6] How to Build 2048 (Python and Tkinter tutorial). 2020. https://www.youtube.com/watch?v=b4XP2IcI-Bg.

[7] 2048 (video game). [n.d.]. https://en.wikipedia.org/wiki/2048_(video_game).