**B351 Final Project**

**2048 Solver**



**Group members**

Shiyue Pei

Yanlin Liu

Alexander Ong

11/29/2016

**Project Description**

1. **Understanding 2048.**

2048 is a simple yet enjoyable game to play with. 2048 is built on a 4x4 grid board where each tile in the grid can hold a block with integer values of exponential of 2. 2048 starts with two blocks randomly placed on the 4x4 board with either two 2’s block or one 2’s block and one 4’s block. For each turn in the game, the player are able to move in either 4 different direction which are Up, Down, Left or Right. Player can merge two tiles with the same value by moving towards that direction which will convert both of the tiles into a single new tile whose value is the sum of the two tiles. After every turn, a new tile with either value 2 or 4 spawns randomly in the empty tile of the board. The goal of 2048 is to keep merging tiles until the value of 2048 is reached. 2048 ends when the tile 2048 is achieved or when there are no more available moves where all the empty tiles are occupied and there are no more tiles to merge.

1. **Implementation**

Our implementation of 2048 is based on an open-source javascript version created by “Gabriele Cirulli”. Based on the open-source code, we’ve added several new functions in the grid.js such as Clustering heuristic, Monotonicity heuristic, findMax (find largest tile), largestTileInEdge (is the largest tile in the edge of the board), isGamePaused (check if game is over) and hasWon (check if game is won). Besides that, we’ve included our own AI.js file which contains an evaluation function (evaluates the heuristic scores), minimax function (minimax search with alpha-beta pruning) and nextMove function (for the AI to move).

1. **AI Strategy**

The strategy that we used for the AI to play and win the game is minimax algorithm with alpha-beta pruning. The reason that we chose this algorithm is because the player (max) who will try to maximize his/her score and achieving 2048 but the computer (min) can block the progress and stops the player from completing the game by inserting 2 or 4 randomly into the board. After playing 2048 for a few rounds, we noticed that we have a higher chance of winning by placing the highest tile in one column or row for the whole game and try to have as many empty tiles as possible to decrease the possibilities to lose the game. We came up with a few simple heuristics based on our analysis of the game. We tried to limit the AI to move only in 3 different directions (up, right or down) which ensures that the largest tile to be placed on the far right of the board and try to have as many empty tiles as possible (shown in pic1 and pic2). Furthermore, we’ve combined our heuristics with two other open-source heuristics that we found online to improve the performance of our AI. But sadly with the improved performance we still couldn’t reach the ideal target score which is 2048. Two of the improvement for heuristics that we imported are Clustering heuristics and Monotonicity heuristics. Clustering heuristics (shown in pic3) measures how clustered the tiles are in our board by measuring the distance between two of the same tiles. When tiles with the same value are close, it’s easier to merge them together and decreasing our chance to lose the game. The monotonicity heuristic (shown in pic4) tries to ensure that the values of the tiles are either increasing or decreasing along both the left/right and up/down directions. This is done to keep higher valued tiles in a corner which makes the board well organized.





**pic4**

**pic3**

**pic2**

**pic1**

**Here’s an example of a perfectly monotonic grid.**

**Here’s an example of a clustered grid.**

1. **Statistics**

The result we got from our solver are not as significant but on an average run this is what we could get as shown below. You can run our solver at <https://shipei.github.io/>.

1. **Deliverables**
   1. 2048 game by GabrieleCirulli.

You can play 2048 at here!

<https://gabrielecirulli.github.io/2048/>

* 1. 2048 AI solver by ShiYue Pei, YanLiu Lin and Alexander Ong.

You can test run our program at here!

<https://shipei.github.io/>

* 1. 2048 AI solver codes available at github.

<https://github.com/shipei/shipei.github.io>

1. **Conclusion**

In conclusion, we noted that achieving a high value tile is the most difficult part of the game and we’ve limited our branching depth to 5 because the branching factor will be too large to handle thus decreasing the speed and performance of the algorithm. But with a deeper depth, we can achieve better and more accurate result. Besides, one of the most important part in this project is to improve our heuristics in order for us to reach 2048 by trying to stick the largest value tile in a corner instead of placing it in row/column. Last but not least, after finishing this project we gained insightful information about heuristics and how we can actually create a bot to solve our problems!

1. **References**
   1. Original 2048 game created by “Gabrielle Cirulli” <https://gabrielecirulli.github.io/2048/>
   2. Open-source code from the original 2048

<https://github.com/gabrielecirulli/2048>

* 1. Open-source article guidance

<http://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048>