

SOLVING 2048 PUZZLE GAME USING ARTIFICIAL INTELLIGENCE

MEMBERS:

T. SAIKIRAN KUMAR (IIT2015001)

DANISH IQBAL (RIT2015080)

SUPERVISOR:

DR. JAGPREET SINGH



INTRODUCTION

- 2048 is a popular single-player video game released by an Italian programmer Gabriele Cirulli.
- 2048 became a viral hit. The game has been described by the Wall Street Journal as "almost like Candy Crush for math geeks".
- There is also a free app version of the application in Android and iOS.
- The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048; however, we can keep playing the game, creating tiles with larger numbers (such as a 32,768 tile).



2048

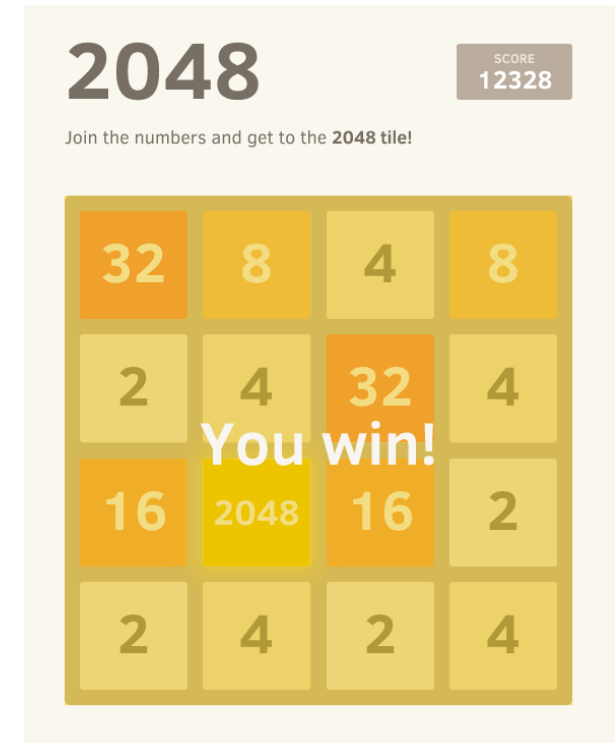
GAMEPLAY

- 2048 is played on a grey 4x4 grid, with numbered tiles that slide smoothly when a player moves them using the four arrow keys.
- Every turn, a new tile will randomly appear in an empty spot on the board with a value of either 2 or 4.
- Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided.
- The resulting tile cannot merge with another tile again in the same move.



WINNING THE GAME

- The game is won when a tile with a value of 2048 appears on the board, hence the name of the game.
- After reaching the 2048 tile, players can continue to play (beyond the 2048 tile) to reach higher scores.
- When the player has no legal moves (there are no empty spaces and no adjacent tiles with the same value), the game ends.



COMPLETION CONDITION

The algorithm halts or completes when such a configuration or state is reached where:

- **Win** : The algorithm halts when we generate a 2048 tile or the whole block.
- **Loss** : The algorithm halts when there is no empty cell and no more valid move is left on the board.

MOTIVATION

- Game playing was an area of research in Artificial Intelligence from its inception.
- Artificial intelligence (AI) in gaming isn't a recent innovation. As early as 1949, mathematician and cryptographer Claude Shannon pondered a one-player chess game, in which humans would compete against a computer.
- Creating effective artificial intelligent for a game is one of the most interesting challenges as an undergraduate student.
- Computers now a day armed with artificial intelligence are beating humans in many games like Chess, Pong, Go, Pac-man, DOTA, Backgammon, etc.



OBJECTIVES

- In this project we aim to create the web based 2048 game using JavaScript.
- Also using Artificial Intelligence i.e., using certain algorithms to solve the 2048 game, consistently trying to reach 2048.
- The AI should also play the game automatically and maximize the game score within reasonable number of moves.
- We also aim to reach the goal of reaching 2048 by computing the number of moves it takes, running each algorithm and also compute and compare each other to find the optimal algorithm.

LITERATURE SURVEY

We went through different papers and journals to learn about related research and theory in the field.



PAPER 1

TITLE OF PAPER:

AI Plays 2048

JOURNAL/CONFERENCE : Stanford University, 2016

PUBLISHER : Stanford University

MAIN TECHNIQUE OF METHODOLOGY:

Minimax ,Expectimax Search, Pruning

LANGUAGE AND TOOLS USED:

The entire system was developed in JavaScript.

PAPER 2

TITLE OF PAPER:

An Investigation into 2048 AI Strategies

JOURNAL/CONFERENCE : IEEE,2014

PUBLISHER : IEEE

MAIN TECHNIQUE OF METHODOLOGY:

Monte-Carlo Tree-Search, Averaged depth Limited Search

LANGUAGE AND TOOLS USED:

The entire system was developed in C++ and HTML.

PAPER 3

TITLE OF PAPER:

Evolving neural network as a decision support system-controller for a game of "2048" case study

JOURNAL/CONFERENCE: IEEE,2016

PUBLISHER: IEEE

MAIN TECHNIQUE OF METHODOLOGY:

Neural Network, Genetic algorithm, Neuro-evolution

ASPECTS OF PROBLEM SOLVED:

The self-learning neuro-genetic system, that combines the advantages of artificial neural networks and evolutionary algorithms in order to efficiently look for feasible solution of the given problem.

LANGUAGE AND TOOLS USED: C++

PROCEDURE

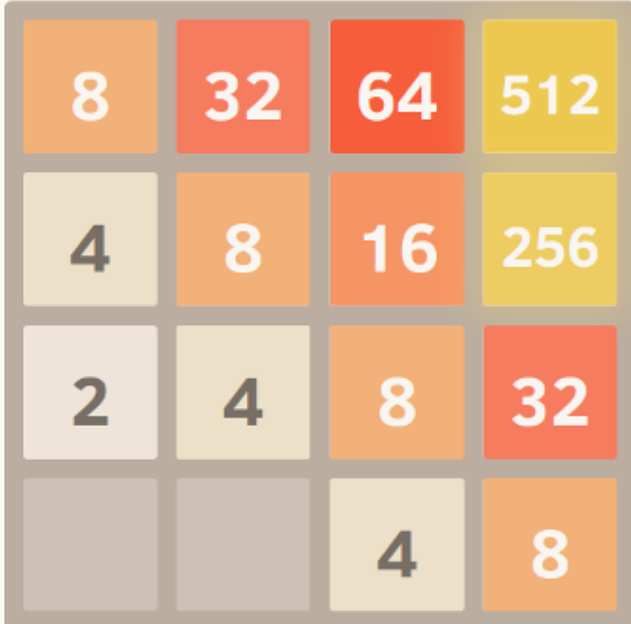
- The main algorithm that we will use is the **minimax algorithm** which is a recursive algorithm for choosing the next move in an n-player game, usually a two-player game.
- We will be using a heuristic function, also called simply a **heuristic**, which is a function that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow
- We then use **alpha-beta pruning** to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It provides a significant improvement over the naïve minimax.

PROPOSED METHODOLOGY

- 2048 is a discrete state space, perfect information, turn-based game like chess and checkers.
- We use the same methods that have been proven to work on these games, namely minimax search along with alpha-beta pruning.
- The most important part is to choose the suitable heuristics to get the best results i.e. to reach the 2048 tile more consistently and to ensure the maximum winning probability:
 - Monotonicity
 - Smoothness
 - Free Tiles

1. MONOTONOCITY

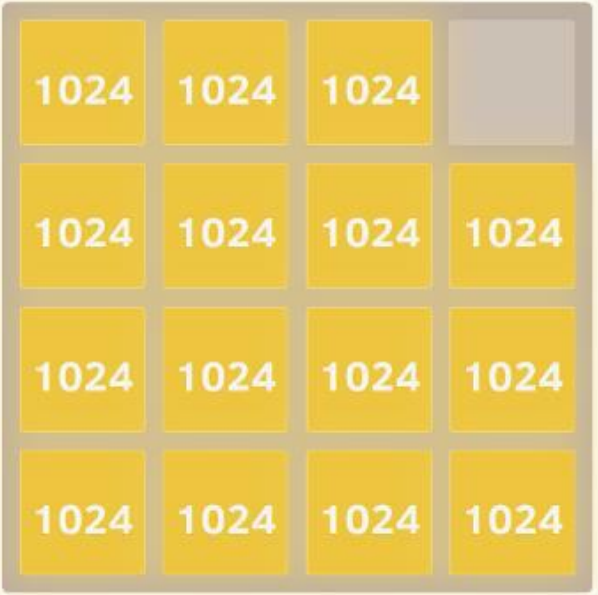
- The most prevalent approach to playing 2048 is maintaining monotonicity i.e. the heuristic tries to ensure that the values of the tiles are all increasing or decreasing along both left/right and up/down directions. Thus the higher valued tiles should be clustered in a corner.
- Monotonicity ensures that smaller valued tiles don't get orphaned and will keep the board very organized, with smaller tile cascading in and filling up into the larger tiles. Something like this:



8	32	64	512
4	8	16	256
2	4	8	32
		4	8

2. SMOOTHNESS

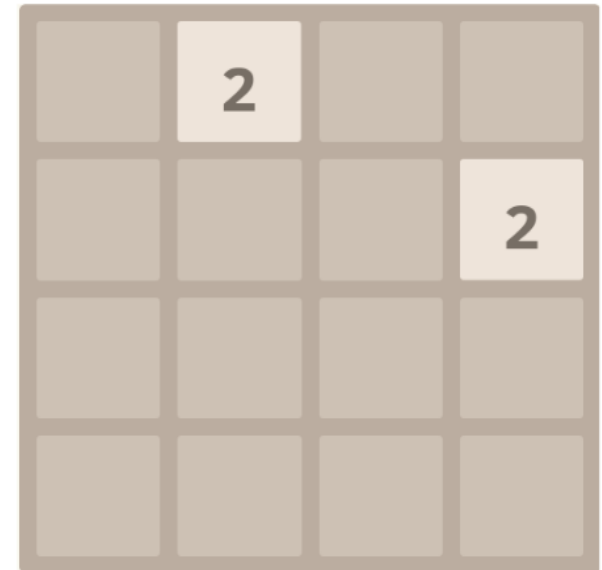
- In order to merge adjacent tiles and move forward in the game the adjacent tiles have to be of the same value.
- Therefore, the smoothness heuristic comes into play. It measures the difference between the neighbouring tiles and tries to minimize this count. Here's an example of a perfectly smooth grid:



1024	1024	1024	
1024	1024	1024	1024
1024	1024	1024	1024
1024	1024	1024	1024

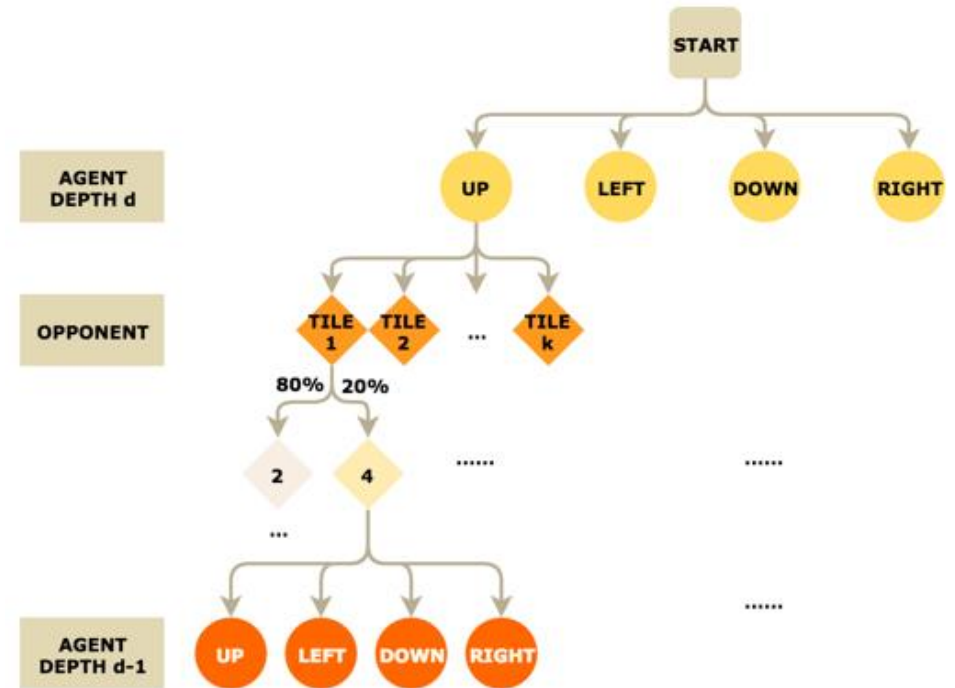
3. FREE TILES

- There is a penalty for having too few free tiles, since options can quickly run out when the game board gets too cramped. The greater number of free tiles we have the more options we would have to play for.



MINIMAX ALGORITHM

- Minimax is a decision-making algorithm, typically used in a turn-based, two player games. The goal of the algorithm is to find the optimal next move.
- In the algorithm, one player is called the maximizer, and the other player is a minimizer. If we assign an evaluation score to the game board, one player tries to choose a game state with the maximum score, while the other chooses a state with the minimum score.
- In other words, the maximizer works to get the highest score, while the minimizer tries get the lowest score by trying to counter moves.



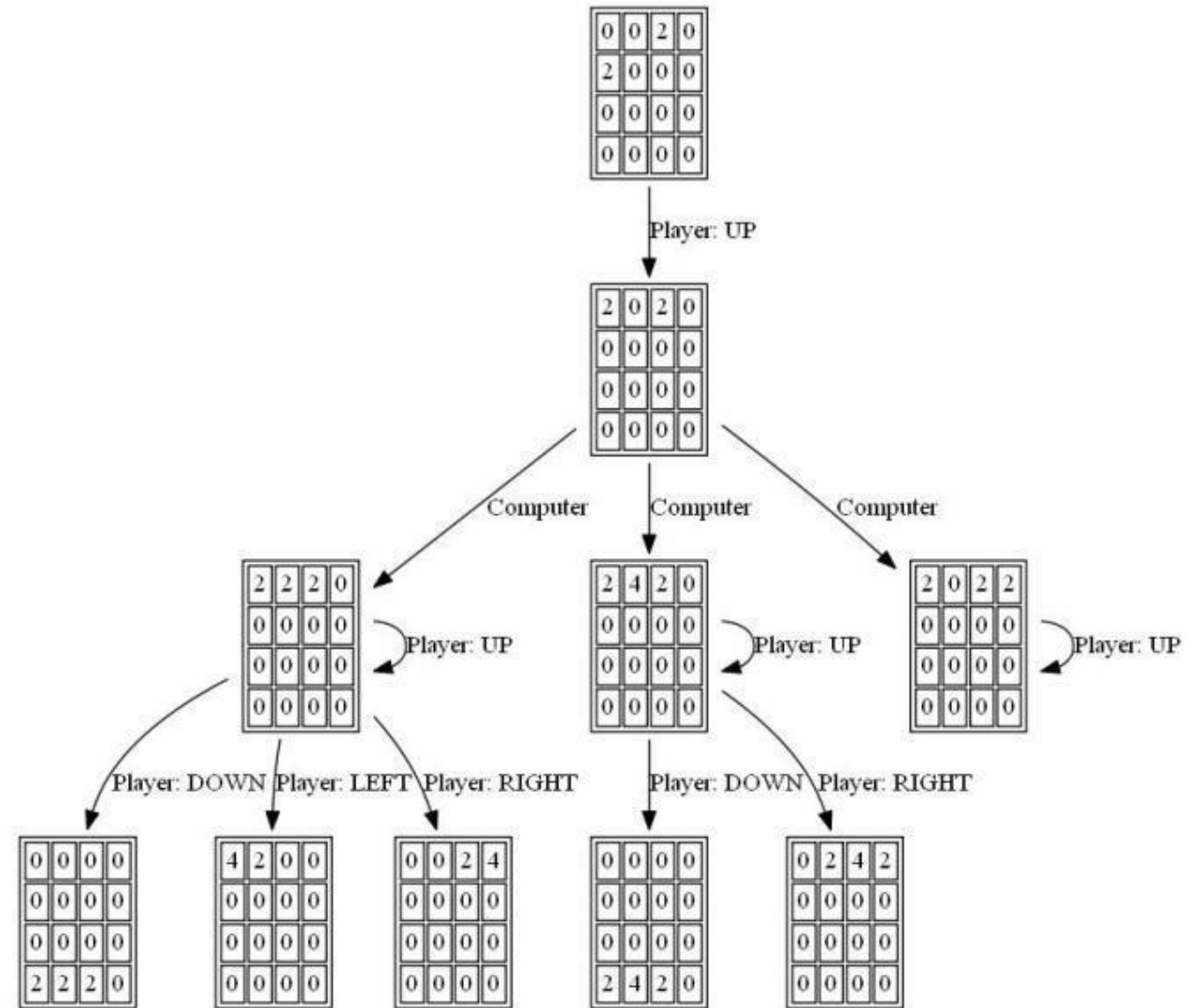
STEPS IN MINIMAX ALGORITHM

- Construct the complete game tree.
- Evaluate scores for leaves using the evaluation function.
- Back-up scores from leaves to root, considering the player type:
 - For max player, select the child with the maximum score
 - For min player, select the child with the minimum score
- At the root node, choose the node with max value and perform the corresponding move.
- We can supply a heuristic evaluation function which gives values to non-final game states without considering all possible following complete sequences. We can then limit the minimax algorithm to look only at a certain number of moves ahead. This number is called the "look-ahead", measured in "plies".

MINIMAX IN 2048

- In the 2048-puzzle game, the computer AI is technically not “adversarial”. In particular, all it does is spawn random tiles of 2 and 4 each turn, with a designated probability of either a 2 or a 4.
- It certainly does not specifically spawn tiles at the most inopportune locations to foil the player’s progress.
- However, a “Player AI” can be created to play as if the computer is completely adversarial. In particular, the minimax algorithm will be employed by assuming that the computer player is adversarial.

MINIMAX TREE

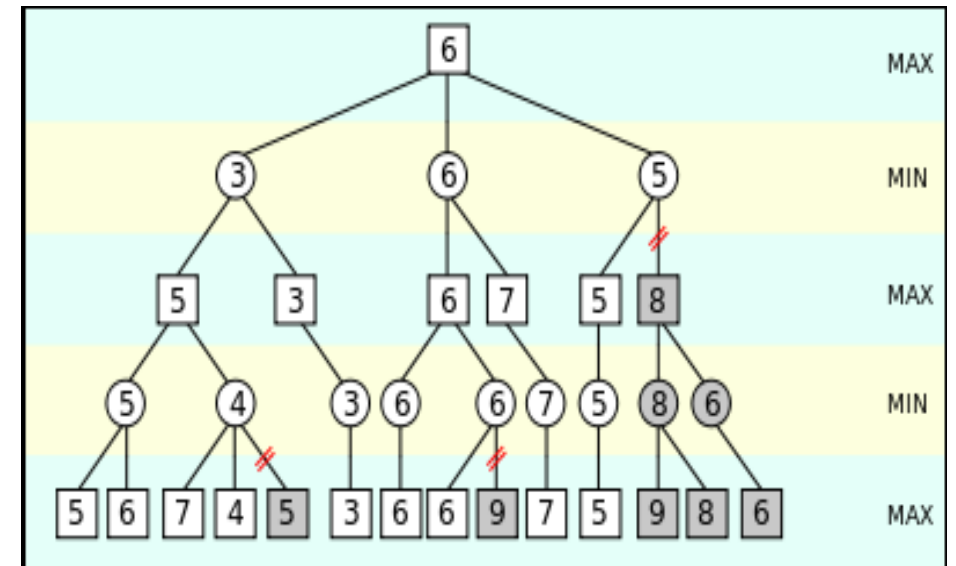


ALPHA-BETA PRUNING

➤ **Alpha – Beta Pruning** : It is used for finding an optimal solution while avoiding searching sub trees of moves which won't be selected

- Beta is the minimum upper bound of possible solutions
- Alpha is the maximum lower bound of possible solutions

➤ The benefit of alpha–beta pruning lies in the fact that branches of the search tree can be eliminated. This way, the search time can be limited to the 'more promising' sub tree, and a deeper search can be performed in the same time.



HARDWARE & SOFTWARE REQUIREMENTS

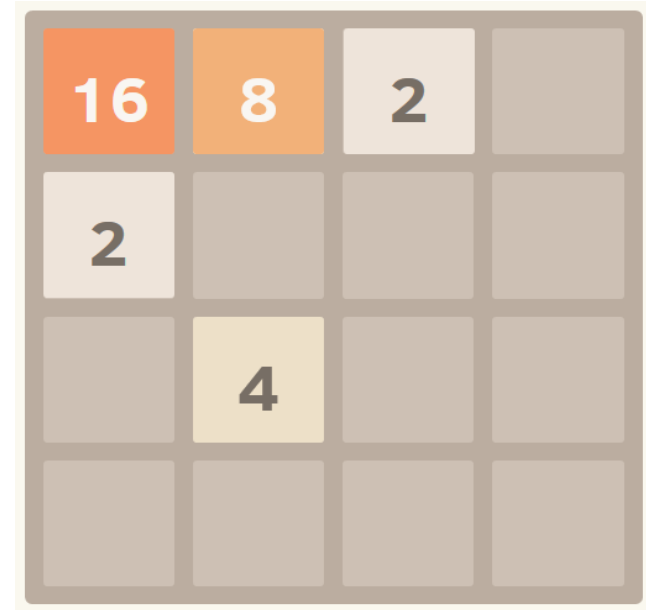
Hardware	Operating System	Processors	RAM	Graphics
System	Windows 8.1	Intel Core i5	8 GB	2GB (NVidia GeForce)

Software & tools	Requirement
JavaScript	Programming Language used for creating the 2048 browser based GUI and to implement the Artificial Intelligence Algorithms.
HTML & CSS	For creating an attractive user interface

Browser used: Google Chrome, Firefox

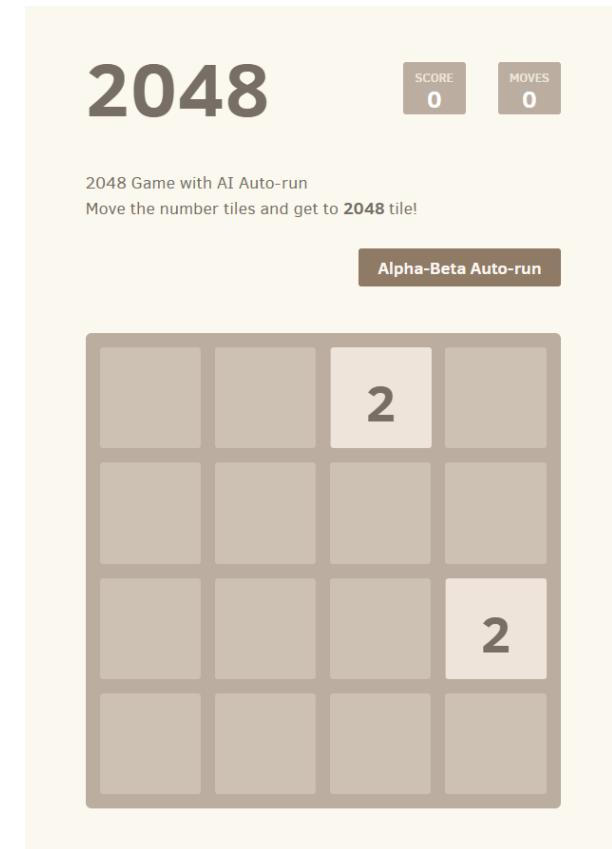
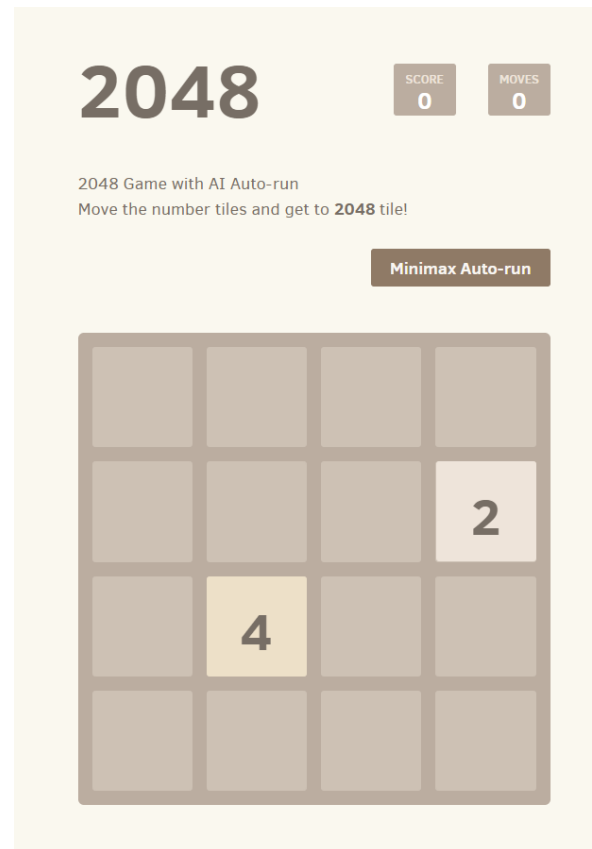
WORK DONE TILL MID-SEMESTER

- We have created a Javascript based graphical user interface for the game, which can be run in a browser like Chrome, Firefox etc.
- The player can play the game by moving the arrow keys as of now.
- Can reset the game by clicking 'New Game'.



RESULT

- We have implemented both the algorithms separately for AI auto-run.
- Also we have implemented a tab for calculating the maximum score and a tab for the number of moves it takes to reach 2048.
- A final look at the Game GUI, as shown:

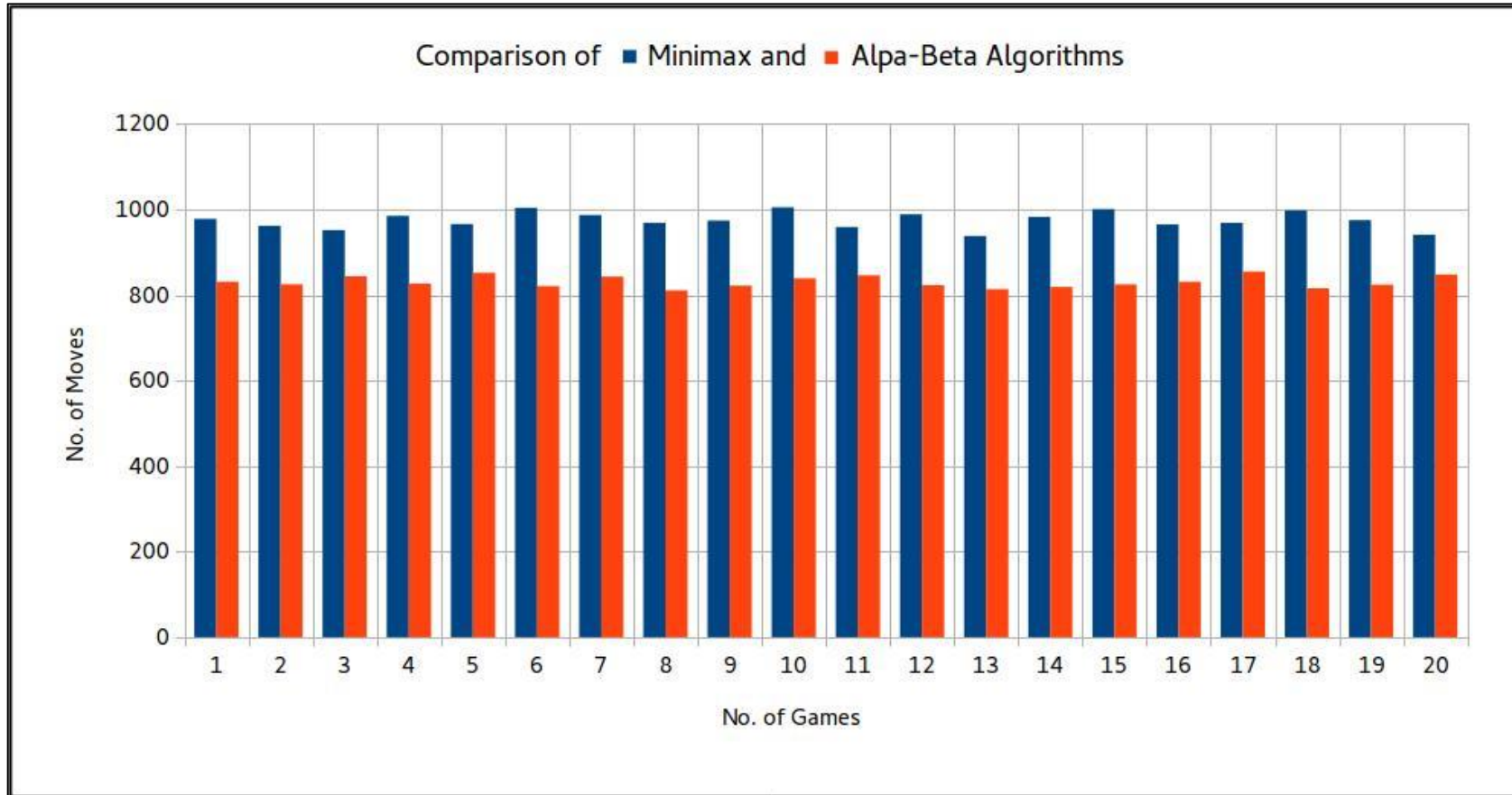


RESULT

- We executed the game several times i.e. AI auto-ran the game for both the algorithms separately and gathered some interesting results. Here are some of the results:
- Minimax: Ranges between 930-1010 moves.
- Alpha-Beta: Ranges between 810-865 moves.
- Scores range between 18000-24000 for both the algorithms.

Algorithm / No. Of Games	Minimax	Reached 2048	Alpha-Beta	Reached 2048
1	978	Yes	831	Yes
2	962	Yes	825	Yes
3	952	Yes	844	Yes
4	985	Yes	827	Yes
5	966	Yes	852	Yes
6	1004	Yes	821	Yes
7	987	Yes	843	Yes
8	969	Yes	811 (Best)	Yes
9	974	Yes	822	Yes
10	1005 (Worst)	Yes	839	Yes
11	959	Yes	846	Yes
12	989	- No -	823	Yes
13	938 (Best)	Yes	814	Yes
14	983	Yes	819	Yes
15	1001	Yes	825	Yes
16	965	Yes	831	Yes
17	969	Yes	875 (Worst)	Yes
18	998	- No -	816	Yes
19	975	Yes	824	Yes
20	941	Yes	864	Yes
Average	965	-	845	-

RESULT



FUTURE SCOPE

- **Improve the Speed:** Improving the speed of the algorithm will allow us to use larger depth and thus get better accuracy.
- **Improve the Number of Moves:** By improving both the algorithms and by making it to be more efficient, will certainly improve the number of moves.
- **Tune Heuristics:** One can experiment with the way that the scores are calculated, the weights and the board characteristics that are taken into account.

CONCLUSION

- Alpha-Beta pruning algorithm proved to have the highest success rate, than Minimax algorithm.
- Alpha-Beta pruning had quick runtime sometimes, taking lesser number of moves compared to Minimax algorithm.
- The Minimax algorithm however was unable to reach the 2048 tile at times which makes it quite ineffective algorithm sometimes, whereas Alpha-Beta pruning algorithm was able to reach 204 tile at all times.
- Alpha-Beta pruning algorithm proved to be much more consistent and efficient throughout than the Minimax algorithm, thus becoming the optimum algorithm.

REFERENCES

- [1] Yun Nie, Wenqi Hou and Yicheng An, “AI Plays 2048”, Stanford University ,California , Stanford University ,California, 2016.
- [2] Gayas Chowdhury and Vignesh Dhamodaran, “2048 Using Expectimax”, University of Massachusetts Lowell, 2015.
- [3] Philip Rodgers and John Levine, “An Investigation into 2048 AI Strategies”, IEEE, IEEE, 2014.
- [4] Arkadiusz Kwasigroch and Michal Grochowski, “Evolving neural network as a decision support system-controller for a game of "2048" case study” , IEEE, IEEE, 2016.
- [5] Kiminori Matsuzaki, “Systematic Selection of N-Tuple Networks with Consideration of Interinfluence for Game 2048” , IEEE, IEEE, 2016.

[6] Marcin Szubert and Wojciech, "Temporal Difference Learning of N-Tuple Networks for the Game 2048", IEEE, IEEE, 2014.

[7] Antoine Dedieu and Jonathan, "Deep Reinforcement Learning for 2048", NIPS, MIT, 2017.

[8] Kun-Hao and I-Chen Wu, "Multistage Temporal Difference Learning for 2048 Like Games", IEEE, IEEE, 2017.

THANK YOU!
