Experimentation – Assignment 2 | COMP20003

Algorithm

The AI finds the shortest path based upon path 'weights'. With the path with the highest weight being preferred.

Depth uncertainty.

As this AI looks at moves ahead of the initial move, a level of uncertainty appears, with predicted moves likely to become impossible due to the random addition of tiles. To counter this, the weights of higher depths have been penalised with the following algorithm.

Move Weight = (Move Score + Bonuses – Penalties)/Move Depth

This helps favours moves with lesser depth, as they are more likely to be able to be attempted.

Empty Tiles

An empty tile is a tile in the 2048 with no value. These are valued in the game, so a bonus weight is added. It's calculated through the following algorithm.

Empty Bonus = Number of Empty Tiles on Board x 2

Through this bonus, obtaining more empty tiles is favoured rather than adding some 2's and 4's together. This helps in later games, where it's often vital for empty tiles to be on the board.

Big Values on Corner Tiles

A common technique performed by 2048 players is keeping the largest tiles in the corner of the board. A bonus weight is therefore given to a board that has the largest tile in one of the 4 corners of the board. This really improves the initial stages of the 2048 game and lessens the likelihood of the AI having a low scoring game.

Highest Value Corner Bonus = (LargestTileValue)

So the algorithm will allocate to 1024 bonus weight if the 1024 is in the corner tile.

Nathan Bullen - 761249

MAX

Max Depth	Score	Max Tile	Nodes Generate	Nodes Expanded	Time (seconds)	Expanded per Second
0 *	906±410	86±37	0	0	15±5E-3	0
1	3,514±1,580	307±150	1,697±1,120	1,259±420	167±54E-3	6,974±1,774
2	7,300±2,897	589±243	9,183±2697	6841±2018	78±22E-3	87,686±4,210
3	7905±2706	614±216	30,445± 7949	20,813± 5,483	103±27E-3	201,332± 3627
4	11,237±3,957	870±247	110,940 ± 33,300	382,820 ± 5636	189±56E-3	382,821 ± 5,636
5	9,750±3284	768±270	257,394± 69,277	160,697 ± 43,540	275 ± 71E-3	527,924 ± 165,546
6	12778 ± 3865	921±216	810,771 ± 219,086	488,207 ± 133,433	661 ± 181E-3	738,510 ± 20,628

AVERAGE

Max Depth	Score	Max Tile	Nodes Generate	Nodes Explored	Time (seconds)	Expanded per Second
0 *	906±410	86±37	0	0	15±5E-3	0
1	2,705±1,174	230±118	1,161±339	1043±308	35±10E-3	29,214 ± 1,923
2	7,202± 3,990	538 ± 281	9,128± 3,825	6787 ± 2845	81 ± 34E-3	83,594 ± 4,781
3	9,987 ± 5,334	768 ± 499	36,895 ± 14,066	25,431 ± 9,765	123±48E-3	206,772 ± 6,083
4	9,501 ± 3,960	768 ± 270	101,479 ± 34,269	66,168 ± 22421	164 ± 55 E-3	402,851 ± 6,540 E-3
5	8,818 ± 3,307	717 ± 264	259,111 ± 74,074	162,464 ± 46,594	257 ± 76 E-3	631,388 ± 8,774 E-3
6	10,185 ± 3,704	768 ± 270	778,830 ± 232,182	475,411 ± 142,829	591 ± 176 E-3	732,379 ± 229,740

* At Depth – 0, both propagations revert to random movement allocation.

In terms of scores, both the average and the max propagation were quite similar at lesser depths. This could be due to the heuristics implemented via weight allocation.

However, as the depth increases, it is clear that Max is the more beneficial propagation, with its average max score increasing greater than the average propagation.

Broadly, the average propagation seemed to run slightly quicker than the max propagation. This would be due to the max propagation having to traverse up the nodes to the parent node.

Changed After Testing

Certain heuristics were added after the algorithm was tested.

Random Tile Addition

While the number of empty tiles is less than 5, a random tile is added. This helps with boards that have minimal empty tiles available, especially if there is only one tile available, it helps the randomness in the future.

Ordered Tiles (Large)

The ordered tile heuristic adds a bonus where if the tile next to the largest number is the size of the largest number / 2.

These heuristics helped improve my algorithm from an average of 12,778 (max 6) to 19,048 (max 6). However, the time to finish the game extended from 0.661 seconds to 1.27 seconds.