COMP20003: Assignment 2 Experimentation

In this project, we were assigned to make a solver for the 2048 game. A variant of Dijkstra’s algorithm was used to come to the solution. A graph was made, where each vertex represented a state of the game i.e. the configuration of each of its tiles. An edge represented a move to reach from the on state to another. The graph was to find the maximum score up to different depths ranging from 0 to 6. The propagation style was also varied, one for maximized score at each node, and other for the average score.

A Python script was used to run multiple cases. The propagation method was varied from maximum to average, the maximum node depth was also varied from 0 to 6 and finally, 50 cases were taken from each of the cases to minimize randomness. The average score below represents the average value taken of the 50 tests from the raw CSV data.

# Average Propagation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Depth | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Average Score | 1029.44 | 4071.84 | 7842.32 | 9877.04 | 8192.8 | 7612.16 | 6021.84 |
| Score Deviation | 545.3439342 | 3507.872984 | 4015.533032 | 5779.528337 | 3612.208415 | 3374.142149 | 2781.50031 |
| Average Time (s) | 0.00009802 | 0.49967832 | 0.614280238 | 0.50130044 | 0.281430779 | 0.263782258 | 0.41695322 |
| Time Deviation | 3.18071E-05 | 0.484258879 | 0.460777917 | 0.459714698 | 0.344243459 | 0.246992186 | 0.218399323 |

Now from the graph above, we can see that it is a crude form of a bell curve. The optimal depth seems to be 3. Comparing the two types of propagations together, we see that the average propagation would be the best to get the highest score.

We see a rise at the start of the graph as increasing the depth from 0 would definitely help the algorithms to predict the probable best moves to get a higher score. Now the more computations, the algorithm can make, the higher the score should be. This is because we are taking into account more possibilities. The standard deviation for the average scores is generally very high, due to the random nature of the game and since the current implementation does take any randomness nature of the game into account whatsoever.

Theoretically, we would always get a higher score, if we increase the depth, but that is not the case. This can be easily explained by the fact, that a number is added at a random position after each move. If our algorithm explores greater depths, the board would eventually become the size of a tile. The current implementation assumes, the board is left with one more tile space after each move and therefore has more flexibility, and therefore has an easier chance of getting a higher score. If we were to add a random component to our implementation, we would surely see a rise in the maximum scores.

The runtime of the algorithms generally increases as we increase the depth, but the few anomalies are caused by the fact that the higher the score gets, the more time the algorithm has to spent trying to get achieve a higher score.

# Maximized Propagation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Depth | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Average Score | 1117.28 | 2794.72 | 7807.6 | 7702.48 | 9254.96 | 8240.48 | 7616.08 |
| Score Deviation | 637.7646882 | 2500.218236 | 3397.43954 | 4043.027509 | 4445.31374 | 4856.842051 | 4361.942012 |
| Average Time | 0.00011088 | 0.61905326 | 0.65831366 | 0.691195501 | 0.65580602 | 0.557447642 | 0.666970161 |
| Time Deviation | 3.90884E-05 | 0.455974841 | 0.451035024 | 0.422955477 | 0.408229833 | 0.367537465 | 0.361792074 |

For our maximized propagation, we see an even more deviation from a bell curve. Here we observe high scores generally, for about depths of four. But one must also observe, the standard deviation of each of the scores which is almost half the score. Overall an average propagation seems to give higher scores due to the fact that maximization is a crude form of a greedy algorithm. Therefore, it does not take into account that although a path leads to higher scores quickly, that doesn’t mean it will to higher scores in the actual game. Now, since average propagation takes this into account, and picks the path that generally leads to higher scores, it usually ends up achieving higher scores itself. Another factor of this being the lower achieving propagation of the two is that averaging anything leads to a normalization effect, this works great here, since we add a random tile after every action.