Task 1: Statistical Analysis of Shortest Path Algorithms

Please refer to 'Statistical Analysis of Shortest Path.ipynb' notebook for code and graphs supporting observations made in this report.

1.0 Evaluating Time Complexity

In Figure 1.1, we observe that the average execution time to find a path varies dramatically between the heuristic and Dijkstra's. For Dijkstra's, the execution time to compute shortest path grows as a proportion of the size of the board. This is expected as the Big-O complexity of the Dijkstra's using a binary heap is O(E x log(V)), where E = the number of edges and V = number of vertices [1]. In our case, since E ≈ 4*V (i.e., each cell has 4 connected cells), this can be further simplified to O(V x log(V)).

On closer examination of the average execution time of the heuristic algorithm, it appears to grow linearly with the size of the input. Based on the implementation used, this is consistent with the expected time complexity of O(n) since in the worst-case scenario, the path would traverse through at most half the cells on the board. Here, we note that the heuristic algorithm appears to be the more computationally efficient of the two.

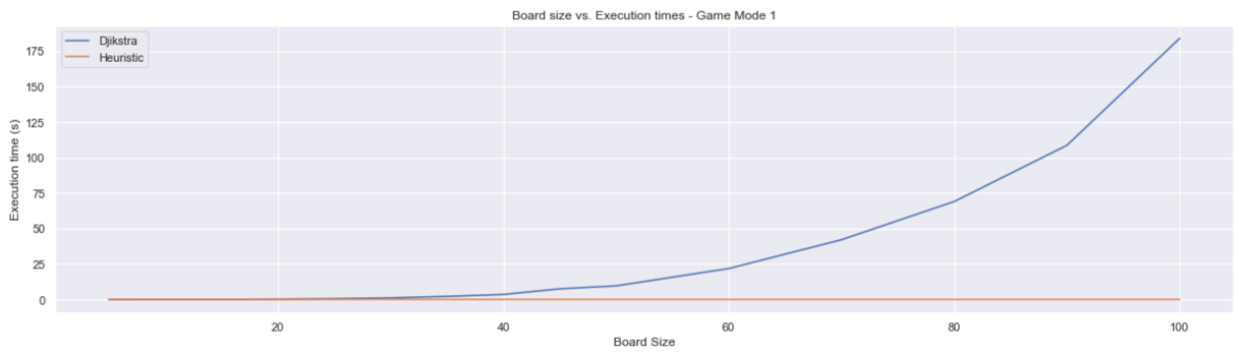

Figure 1.1 Average Execution Time (in seconds) vs. Board Size – Game Mode 1

2.0 Evaluating Game Modes

In this section, findings from evaluation of each game mode across both algorithms will be discussed. For reference, only square grids were evaluated and for each parameter, the values reported are the average across 3 iterations.

2.1 Game Mode 1

By inspecting Figure 2.1 we see a much clearer picture of the trade off in using the heuristic over Dijkstra's. The path cost plot (orange line) below shows on average, how proportionally larger the cost of the path was for the heuristic algorithm when compared to Dijkstra's. We observe that although no clear correlation with board size exists, the path computed by the heuristic was on average 23.7% longer with a standard deviation of 6.2%. The smallest difference in the path cost between the two algorithms was 13.8%, whereas the largest was up to 43.3% longer. The blue line in the same figure shows the average execution time of the heuristic algorithm as a proportion of Dijkstra's. In line with the observation from section 1, although no clear correlation exists with relation to board size, it can be noted the execution time for heuristic is comparably smaller.

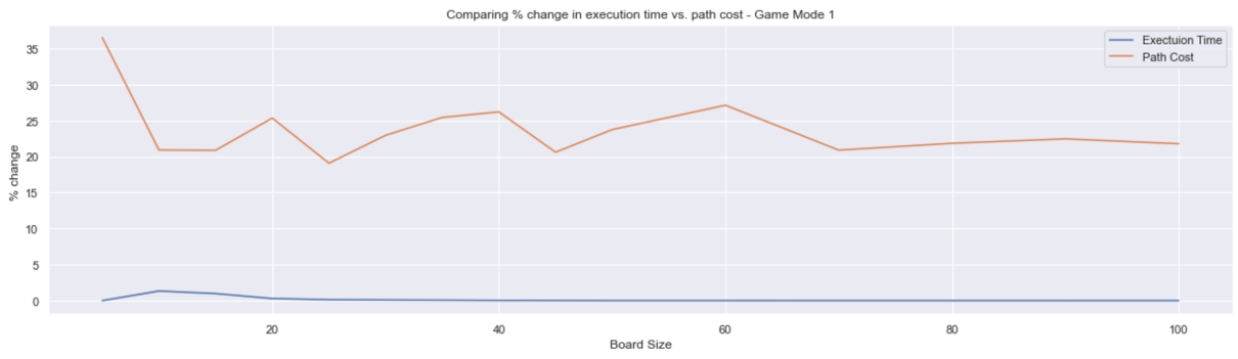

Figure 2.1 Comparing Percentage Change in Execution Time vs. Path Cost – Game Mode 1

2.2 Game Mode 2

Results obtained for Game mode 2 are comparable to that of Game Mode 1. The one interesting difference we note here is, the significantly worse performance of the heuristic function in terms of the cost of the path found. On average, for game mode 2 the path found by the heuristic was 70.9% longer with a standard deviation of 4.6%. Inspection of max and min values show that, at best the heuristic was only able to find a path that was at least 50% longer than the shortest path and at worst 79.3% (as opposed to 13.8% and 43.3.% respectively in game mode 1).

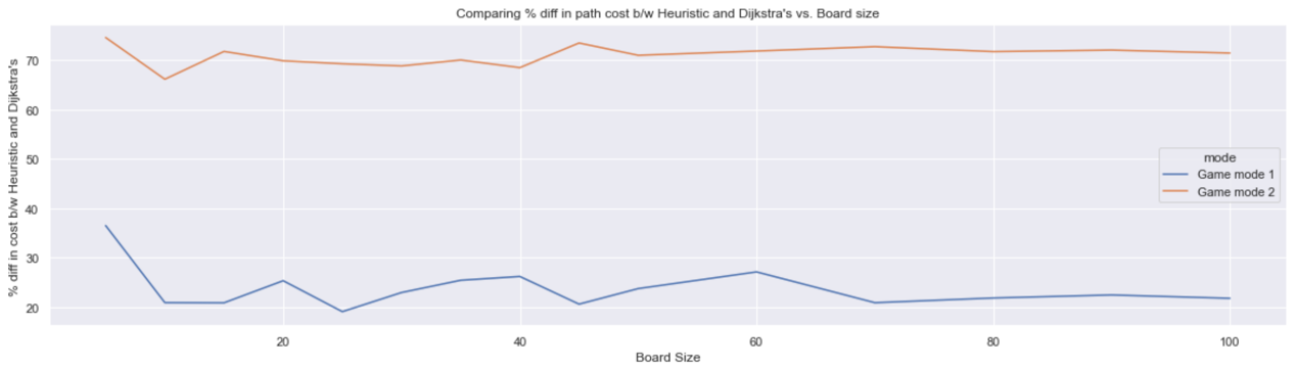

Figure 2.2 Comparing Percentage difference in path cost b/w Heuristic and Dijkstra's between game modes

3.0 Evaluating cost distributions

In this section, the observations made from using different distributions to generate costs on cost of the shortest path will be discussed. Four different distributions were used for study – normal, uniform, binomial and permutation. The values reported are the average across 3 iterations.

The metric being evaluated here is the percentage difference in cost of the path computed by the heuristic as a proportion of the cost of the shortest path (computed by Dijkstra’s). This helps us evaluate performance of the heuristic in relation to the cost distribution. In Figure 3.1 below, we observe how the metric changes across different cost distribution and different board configurations for Game Mode 1. Its interesting to note that as the size of the board increases, for binomial, normal and permutation distribution, the heuristic can find a path that is increasingly close to the optimal path. However, in contrasting this with normal distributions, the heuristic appears to perform relatively poorly independent of the board size.

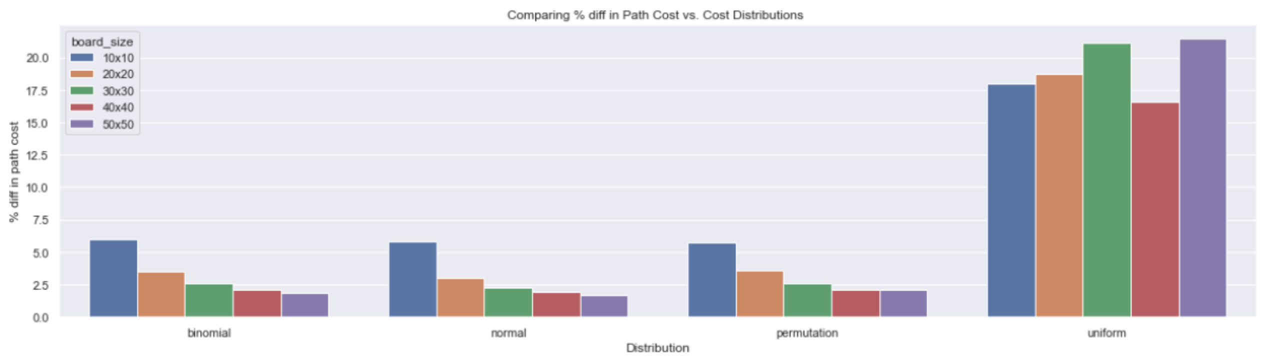

Figure 3.1 Percentage difference in Path Cost vs. Cost Distributions – Game Mode 1

In Figure 3.2, we observe the same metric, however for game mode 2. Here, we notice that the heuristic algorithm performs significantly worse regardless of distribution, with uniform distribution only performing marginally better by roughly 30%, however still at 70% longer than the shortest path.

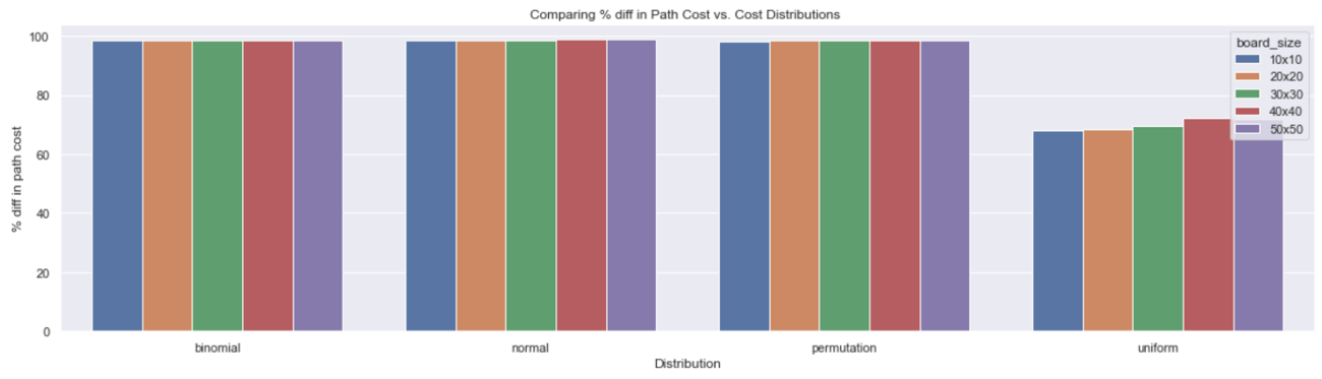

Figure 3.2 Percentage difference in Path Cost vs. Cost Distributions – Game Mode 2

4.0 Conclusion

The underlying cause for poorer performance in mode 2 could be due a multitude of reasons. One caveat of the heuristic is that it always picks a local minimum of the adjacent cells. It would likely perform better (i.e., find a path with lower cost) had it inspected the best move to make based on the adjacent cells' adjacent cells and plan based on cost of the next couple of steps before making a move. By evaluating the cost of the cluster of cells prior to traversing on it, this would have likely yielded a solution that is closer to the optimal solution as opposed to the currently employed greedy method of always picking the local minimum.

Through the analysis conducted, we can note that while Dijkstra's algorithm guarantees the shortest path is always found at the cost of higher time complexity, the heuristic while computationally more efficient performs variably (and consistently worse) in terms of the cost of the path found.

5.0 References

[1] Baeldung. (2021, October 13). *Understanding time complexity calculation for Dijkstra algorithm*. Baeldung on Computer Science. Retrieved December 31, 2021, from https://www.baeldung.com/cs/dijkstra-time-complexity

Code references:

[2] tacaswell. (2013, October 25). Drawing grid pattern in matplotlib. Retrieved November 17, 2021 , from https://stackoverflow.com/questions/19586828/drawing-grid-pattern-in-matplotlib.

[3] Kington J. (2014, January 9). Show the values in the gris using matplotlib. Retrieved November 17, 2021, from https://stackoverflow.com/questions/20998083/show-the-values-in-the-grid-using-matplotlib.

[4] Dijkstra's Shortest Path Algorithm. BogoToBogo. Retrieved November 17, 2021, from https://www.bogotobogo.com/python/python_Dijkstras_Shortest_Path_Algorithm.php

[5] Shortest Path with Djikstra's Algorithm. Bradfield School of Computer Science. Retrieved November 17, 2021, from https://bradfieldcs.com/algos/graphs/dijkstras-algorithm/

Percentage of code borrowed: 60%