Differences in Efficiencies of Pathfinding Algorithms on Mazes of Varying Size Complexities

Samyak Shrestha and Sariyah Sintayehu

Kenyon College
Gambier, OH

April 22, 2024

1. Introduction

Pathfinding algorithms are widely used in computer science, particularly in fields such as artificial intelligence, robotics, computer gaming, and global positioning systems (GPS). These algorithms address the challenge of navigating from a source to a destination while avoiding obstacles and minimizing various costs such as time, distance, risks, fuel, and price (H.urna, 2020). Given our regular reliance on pathfinding algorithms, we sought to determine which algorithm performs best in a race through a randomly generated maze. We hypothesize that more complex mazes will require more time for each algorithm to complete, with A* expected to perform consistently well across varying maze sizes due to its heuristic-based search strategy.

Our variables of interests are:

Algorithm: Categorical explanatory variable that refers to the type of pathfinding algorithm used. The three levels are:

- Depth-First Search (DFS): Traverses a graph by exploring as deeply as possible along each branch before backtracking, using a stack for memory.
- Dijkstra's: Calculates the shortest paths from a start node to all other nodes in a graph with non-negative edge weights, utilizing a priority queue to explore the most promising paths first.
- A* (A-star): Extends Dijkstra's by using heuristics to estimate costs from each node to the target.

Complexity: Categorical explanatory variable that refers to the size and obstacle complexity of a maze. The three levels of Complexity are 30 (simple), 50 (medium), and 70 (hard). For example, a maze of complexity 50 will have 50 rows and 50 columns. We opt into saying size complexity rather than just size since time complexity—the efficiency of an algorithm—changes based on the type of algorithm employed. Additionally, larger mazes have more dead ends and obstacles which makes them more difficult for the algorithm to reach the goal.

Time: Quantitative response variable measured in milliseconds which refers to the time taken by each algorithm to find the shortest path under our controlled experimental conditions.

2. Study Design and Procedure

In this experiment, we aim to compare the efficiency of three pathfinding algorithms— Dijkstra's, A*, and Depth-First Search—by solving mazes of varying size complexities. We will test these algorithms on size 30 (simple), size 50 (medium), and size 70 (hard) mazes. The purpose is to understand how the complexity of a given maze affects the time taken of these algorithms to find the solution.

For this study, we used a 2022 MacBook Air running macOS Sonoma 14.0 with the Apple M2 chip, 8 processor cores, and 8 gigabytes of random-access memory (RAM). We created a code script from pseudocode for generating mazes and implementing the three algorithms as functions (GITTA, n.d.; Sandeep, 2024; Barcelona, n.d.).

To ensure that each algorithm is tested on each size complexity in a randomized order, we used RANDOM.ORG to shuffle all possible orderings. To start the experiment, for each size complexity level we generated a set of mazes using our script and saved its seed. We then ran each pathfinding algorithm on each maze and recorded time taken to reach the end point from the start point. For a balanced factorial design, we recorded 10 data points for each algorithm and size complexity combination for a total of 90 observations.

3. Statistical Analysis and Results

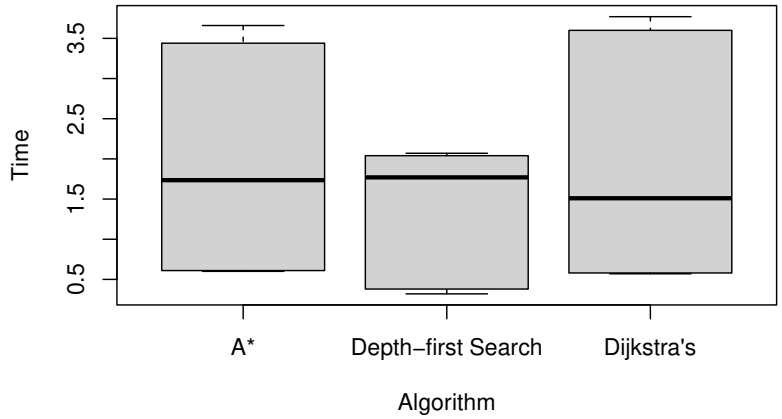

Figure 1. Box-plot of Solving Times Across Algorithms.

We plot the efficiencies of the three algorithms across all size complexities in Figure 1. It is difficult to discern if there is a difference in solving times across the algorithm groups. The vertical spread for A* and Dijkstra's are larger than for Depth-first Search, which indicates that these algorithms had a wider variation in efficiencies. The vertical spread for DFS is on the lower end of the Time scale, which indicates that this algorithm had less variation across the complexities compared to A* and Dijkstra's.

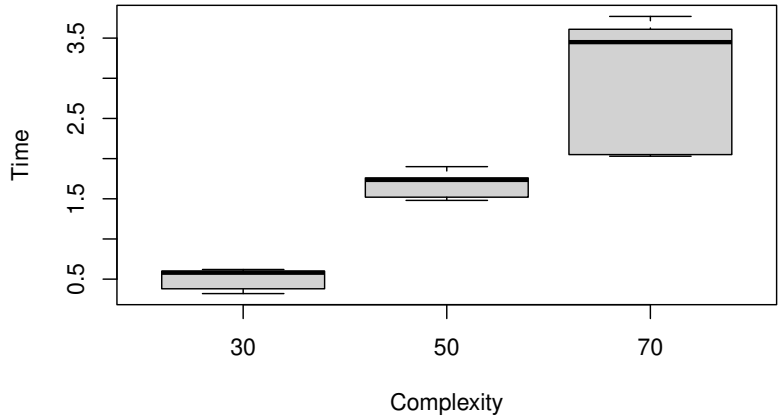

Figure 2. Box-plot of Solving Times Across Size Complexities.

We plot the efficiencies of the three size complexities across all algorithms in Figure 2. There do appear to be differences in the group size complexities since the vertical spreads do not overlap. The time taken to solve each type of maze increases from the simple, medium, and hard complexities. The vertical spread for the simple maze is approximately similar to the medium maze, but the hard maze has a much wider spread which suggests more variability in solving times for this size complexity.

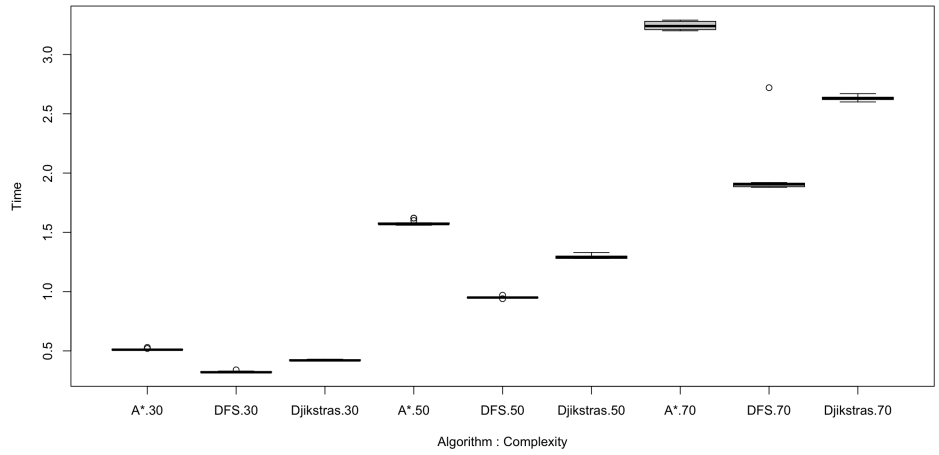

Figure 3. Box-plot of Solving Times Across all Algorithm and Complexity Pairings.

We observe in Figure 3 that there are relative clusters of three box-plots grouped by the complexity of the maze, correlating to the same trends we observed in Figure 2 with the simple mazes taking a lower time, medium mazes taking a medium time, and hard mazes taking more time to solve. We also observe some outliers. Additionally, DFS on the size 70 maze performed much faster than A* and Dijkstra's on the same size complexity, which is a different trend to the other sizes.

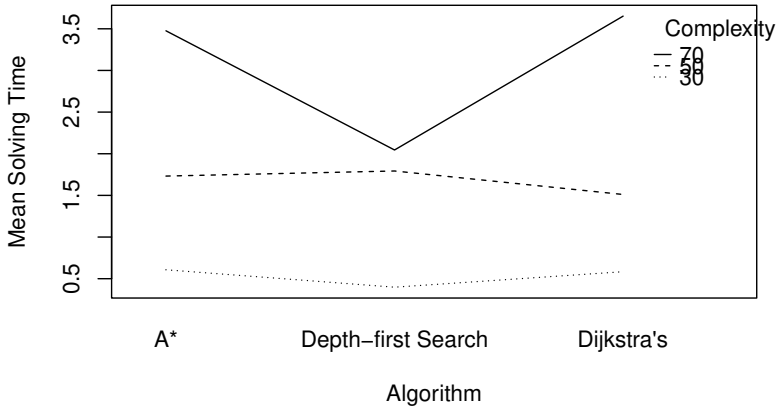

Figure 4. Interaction plot between Algorithm and Complexity.

Figure 4 depicts the mean solving time for each complexity ranging across the algorithms. Each line changes slope when switching from A* to Depth-first Search and from Depth-first Search to Dijkstra's. Additionally, there are inverse trends between the simple and medium and between the medium and hard complexity mazes. Based on these findings, there is likely an interaction between the algorithm used to solve the maze and the size complexity of it.

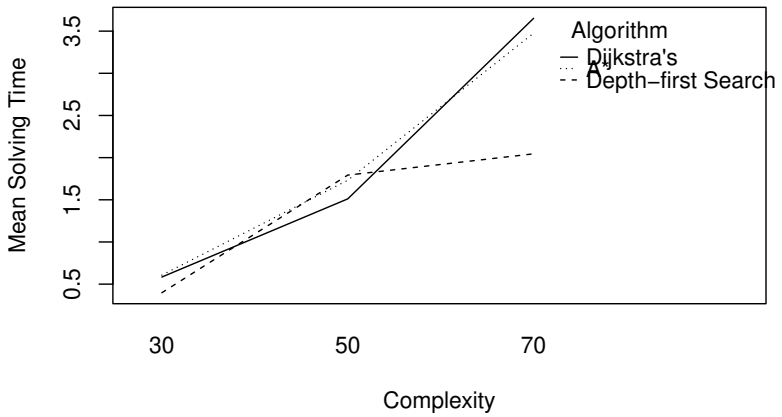

Figure 5. Interaction plot between Complexity and Algorithm.

Similarly, Figure 5 shows that there are crossed lines between the algorithms when going from one size complexity to another, which lends greater evidence that the time taken to solve a maze depends also on some interaction between these variables.

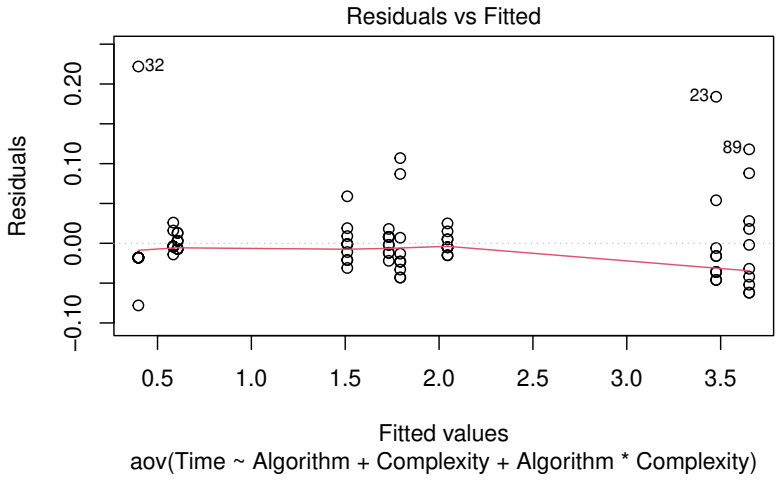

Figure 6. Residuals v. Fitted plot on 2-way Interactive Model.

Figure 6 reveals that the zero-mean condition is satisfied since we have residual points above and below the zero line. Similarly, we do not see alarming trends. There is an approximately balanced vertical spread throughout the plot, so constant variance is reasonable.

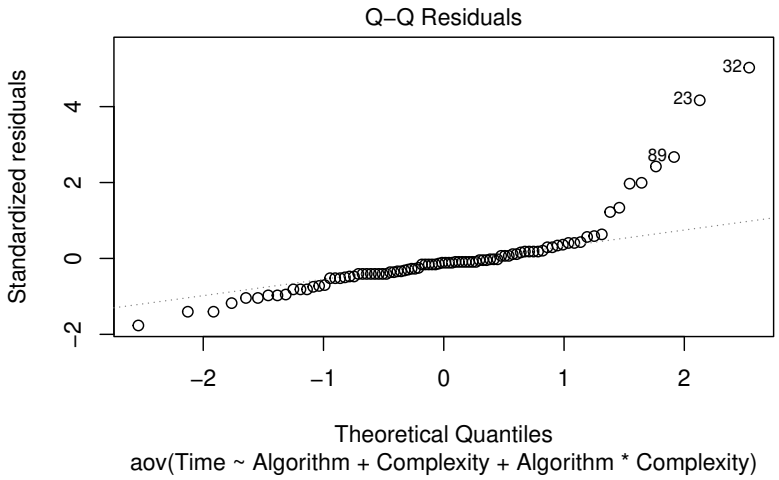

Figure 7. Q-Q Residuals plot on 2-way Interactive Model.

Normality appears concerning since the residuals stray from the normal line outside of the -1 to 1 theoretical quantile range. However, normality is less of a concern since we have a balanced factorial design.

A diagnostic plot indicated to test a square root transformation on the response: time.

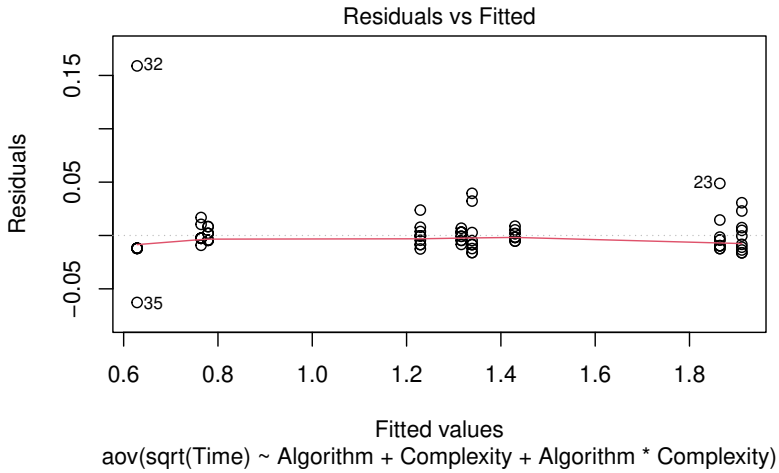

Figure 8. Residuals v. Fitted plot on Transformed 2-way Interactive Model.

From Figure 8 we see that constant variance got worse after applying the transformation. Residual points 32 and 35 make a megaphone-type trend and are far from the zero-mean line unlike the majority of the other residuals.

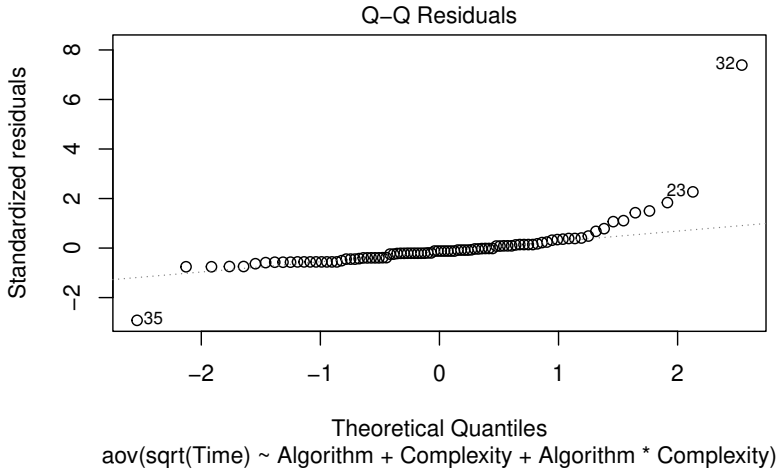

Figure 9. Q-Q Residuals plot on Transformed 2-way Interactive Model.

Comparing Figure 7 to Figure 9, normality improved slightly, but residual point 32 still has a large distance from the normal line. Since the square root transformation made the constant variance condition worse and did not improve the normality condition substantially, we proceed with a non-transformed 2-way interactive ANOVA model.

Our treatment effects within each combination of algorithm and complexity are constant because the population mean time is fixed for each level of algorithm and complexity. Independence between each algorithm is given since we only run one at a time, so one run through does not affect the next. The

random order of algorithm and size complexity combinations reduces human error for inadvertently creating a predictable order, which satisfies independence across observations of the same algorithm and size complexity. Moreover, each observation resets the maze to minimize dependence within an algorithm of the same size. We use the same random seed for each size complexity so that each algorithm solves the same maze.

The full 2-factor interactive ANOVA model is defined as

$$Y = \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon, \text{ where } \varepsilon \sim N(0, \sigma^2) \text{ for } i = 1, 2, 3 \text{ and } j = 1, 2, 3, \tag{1}$$

where the treatment groups are the algorithm used and the size complexity.

We now test if this full model helps explain the variation in time or if the overall mean is sufficient by running an overall F-test. Our hypotheses are

$$H_0 : \alpha_i = \beta_j = (\alpha\beta)_{ij} = 0 \text{ for all } i = 1, 2, 3 \text{ and } j = 1, 2, 3 \tag{2}$$

H_A : at least one $\alpha_i, \beta_j, (\alpha\beta)_{ij}$ is different for $i = 1, 2, 3$ and $j = 1, 2, 3$.

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	89	112.542				
2	81	0.176	8	112.37	6481.9	< 2.2e-16 ***

Table 1. Analysis of Variance Table for Overall F-test.

We see from Table 1 that with $F = 6481.9$, df = 81, and p-value $\approx 0 < 0.05 = \alpha$, we reject the null hypothesis. We have sufficient evidence to conclude that some factor in our 2-way interactive model is significant. We now look at the model output to see which factors are significant in describing solving times, which is shown in Table 2.

Response:	Time				
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Algorithm	2	5.312	2.656	1225.8	< 2.2e-16 ***
Complexity	2	96.126	48.063	22180.5	< 2.2e-16 ***
Algorithm:Complexity	4	10.928	2.732	1260.8	< 2.2e-16 ***
Residuals	81	0.176	0.002		

Table 2. Analysis of Variance Table for Model.

From Table 2 we observe that all factors in our model are significant. Since the interaction term is statistically significant and the most complex in the model, we proceed to look at group differences and ignore the non-interactive factors. Our hypotheses are detecting if there are group differences in the

interactions:

$$H_0 : (\alpha\beta)_{ij} = 0 \text{ for all } i = 1, 2, 3 \text{ and } j = 1, 2, 3 \tag{3}$$

$$H_A : \text{at least one } (\alpha\beta)_{ij} \neq 0 \text{ for some } i = 1, 2, 3 \text{ and } j = 1, 2, 3.$$

We see from Table 2 that with $F = 1260.8$, df $= (4, 81)$, and p-value $\approx 0 < 0.05 = \alpha$, we reject the null hypothesis. We have sufficient evidence to conclude that there are differences in interaction effects between the type of algorithm used and the size complexity it ran on.

We use Tukey's HSD for creating the pairwise comparisons since we have 9 groups and are concerned about inflating the possibility of making a Type I error.

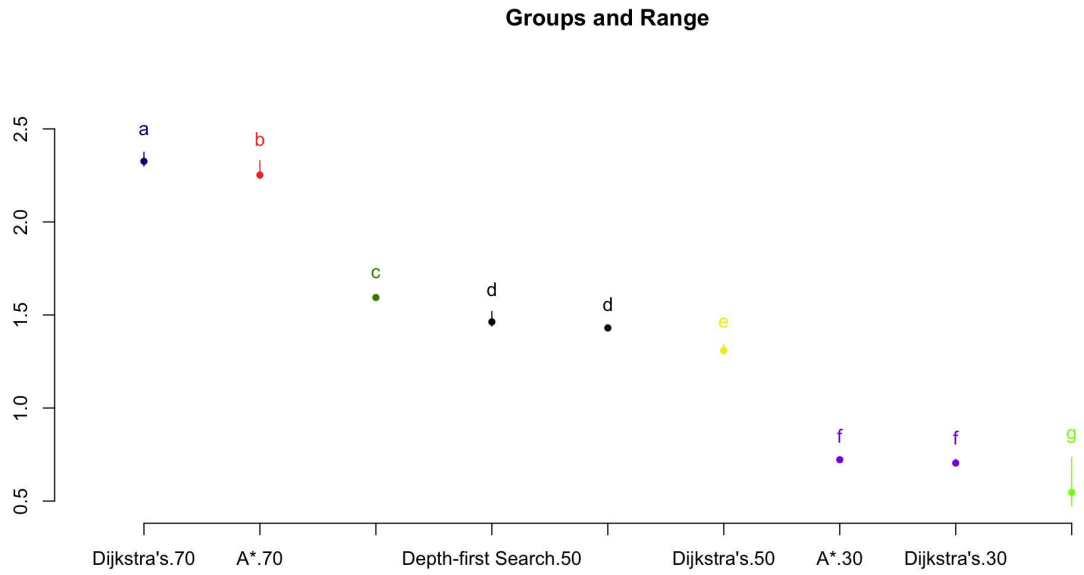

Figure 10. Pairwise Comparisons with Tukey's HSD.

From Figure 10, most groups are significantly different from each other apart from Depth-first Search and A* on the medium complexity maze as well as A* and Dijkstra's on the simple complexity maze. The group labeled ''g'' appears to have a larger range, which suggests more variability in the mean difference for Depth-first Search of complexity 30, but it does not share a letter with any other group. Across both simple and medium complexities, DFS performs comparably to A* and Dijkstra's algorithms in terms of average time taken. However, at a complexity level of 70, DFS significantly outperforms the other algorithms. This performance advantage is clearly illustrated in Figure 3, where the box

representing the hard complexity for Depth-first Search is noticeably lower than those for the other methods.

Our results confirmed one of our hypotheses: We observed that as the complexity of the mazes increased, so did the mean time required for each algorithm to complete the mazes. This trend of ascending mean completion times across mazes of greater size and complexity supports the idea that more complex mazes with harder obstacles and greater size pose challenges to the three pathfinding algorithms.

However, A* did not perform consistently well across all size complexities. Surprisingly, Depth-first Search performed the most consistently across all three size complexities. We assume that this is due to the use of recursive backtracking in our maze generation code, which inherently produces mazes that are less complex in terms of branching and obstacles. These mazes do not challenge the algorithm with complex decision points or multiple route choices that would typically showcase the strengths of A* or Dijkstra's.

Our results are most applicable to mazes generated using similar techniques of recursive backtracking since this process was consistent for each algorithm to run on.

4. Further Discussion

We encountered data collection inconsistencies from the manual recording process of data into a comma-separated-values (CSV) file. To address this issue, we developed a script to automate the data collection into a CSV format. Consequently, we conducted a second round of data collection using this automated process to ensure consistency and accuracy in our dataset.

For further work, we can conduct additional experiments using different maze generation algorithms (e.g., Kruskal's, Eller's, or Randomized Prim's) to see how the pathfinding algorithms perform across a broader range of maze types.

5. Conclusion

To reiterate, we studied the time taken for pathfinding algorithms (A*, Dijkstra's, and Depth-First Search) to complete randomly generated mazes of varying size complexities. We hypothesized that more complex mazes would take more time for each algorithm to finish, with A* expected to perform consistently well. Analyzing our final results, we found out that as complexity increases, so does the mean time taken to complete each maze. However, instead of A*, Depth-First Search was the fastest and most consistent across each size complexity.

References

Universitat Autònoma de Barcelona. (n.d.). A* Algorithm psuedocode. Retrieved April 15, 2024, from https://mat.uab.cat/˜alseda/MasterOpt/AStar-Algorithm.pdf.

GITTA. (n.d.). Dijkstra Algorithm: Short terms and Pseudocode. Retrieved April 15, 2024, from http://www.gitta.info/Accessibiliti/en/html/Dijkstra_learningObject1.html.

H.urna, Hybesis. (2020, January 20). Pathfinding algorithms: the four Pillars. https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b.

Sandeep, Jain. (2024, February 16). Depth First Search or DFS for a Graph. Retrieved April 15, 2024, from https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/#.