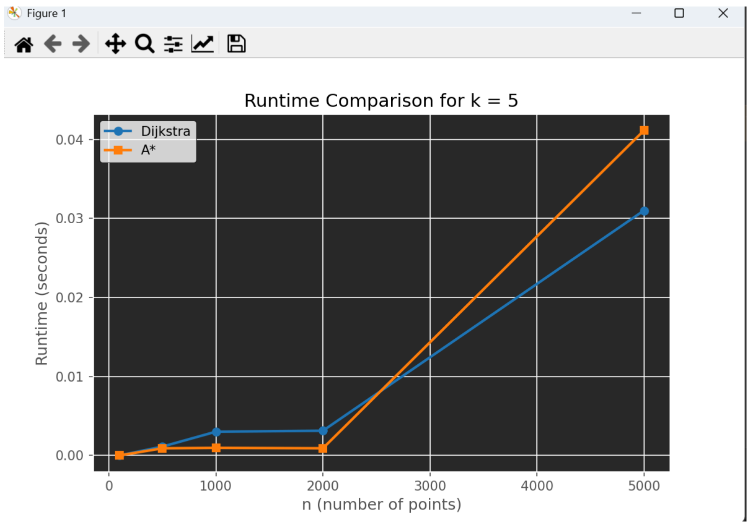
Adam Pinkos

CS4040

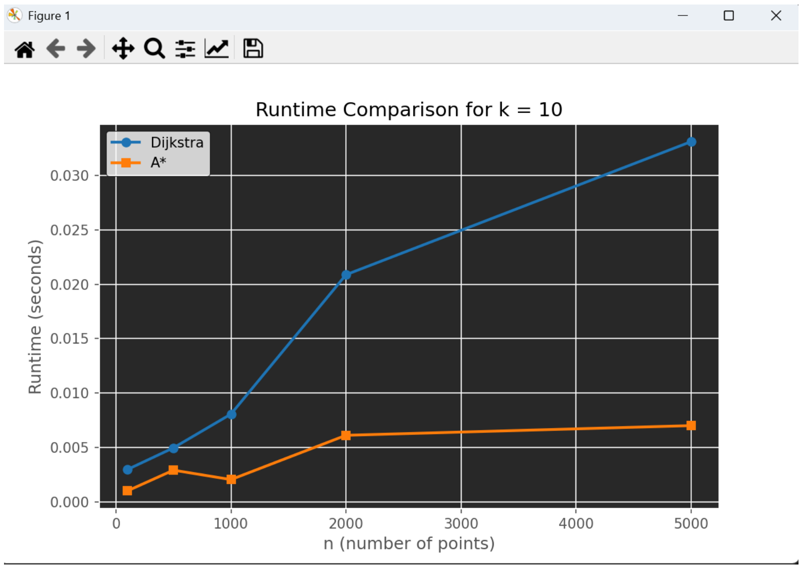
December 4th, 2025

Dijkstra’s and Heuristic Shortest Path Algorithms

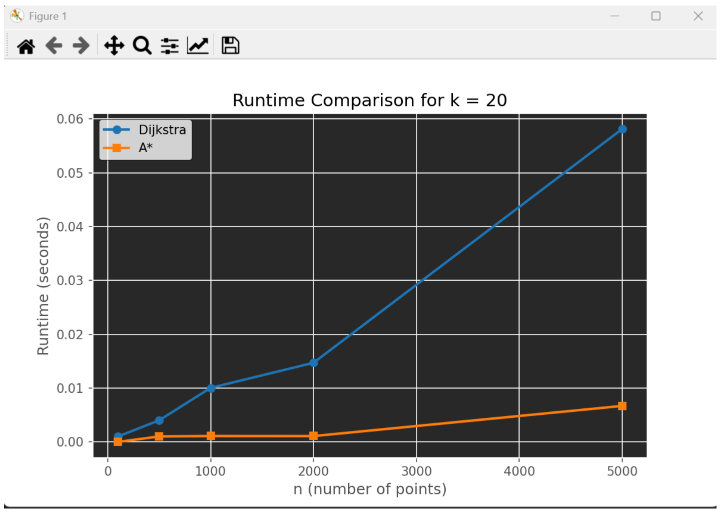
For this project, to evaluate how efficiently different shortest-path algorithms scale on geometric graphs, I compared the runtime of Dijkstra’s algorithm and the Astar search algorithm across several graph sizes and neighborhood densities. Using randomly generated point sets in the square [0,100]^2. I constructed k nearest neighbor graphs and measured how long each algorithm needed to compute a path from the lexicographically smallest point to the largest. By varying both the number of points n and the parameter k, these experiments reveal how the structure of the graph directly impacts performance.



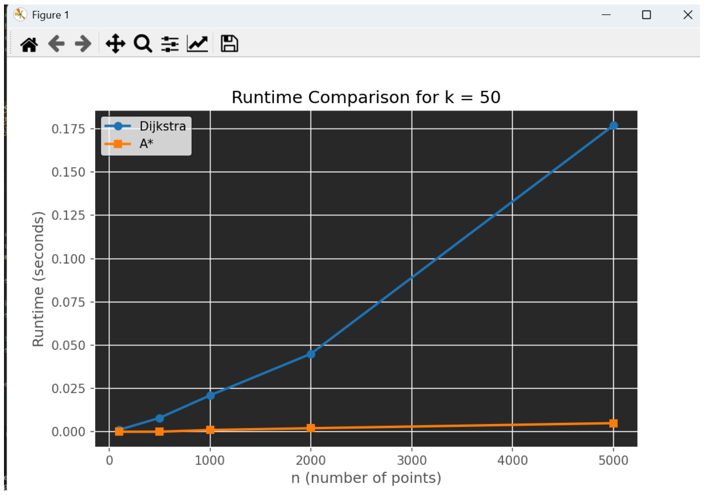
For **k = 5**, both Dijkstra’s algorithm and Astar begin with extremely small runtimes at low values of nnn, but their performance diverges once the graph becomes larger. Up to about n = 2000 the two algorithms run almost identically, with Astar slightly faster. However, as the number of points increases to n=5000, Astar becomes noticeably slower than Dijkstra, which is the opposite of what we see for larger values of k.



For **k = 10**, Dijkstra’s runtime increases steadily as n grows, becoming significantly slower for larger graphs. However, Astar stays much lower overall, rising only slightly with n. The gap between the two algorithms widens as the graph gets bigger, showing that Astar benefits more from its heuristic.



For **k = 20**, the performance gap between the two algorithms is a lot. Dijkstra’s runtime rises sharply as n increases, while Astar remains extremely fast, growing only slightly even at the largest graph size. With more neighbors per node, the graph is dense enough for Astar's heuristic to guide the search efficiently.



For **k = 50**, the difference between the two algorithms is a lot more Dijkstra's runtime rises steeply as n grows, becoming more than an order of magnitude slower by the time the graph reaches 5000 points. In contrast, Astar stays extremely fast across all sizes, increasing only slightly even as the graph becomes very dense.

Across all experiments, the results show a consistent pattern and that is that Astar outperforms Dijkstra's algorithm most of the time, and the advantage is shown as the graph becomes denser (larger k) or larger (greater n). When k is small, the graph is scattered, and Astar gains little from its heuristic, occasionally even running slightly slower due to overhead. But as k increases and each node has more neighbors, the heuristic becomes far more effective at guiding the search toward the target. This allows Astar to explore dramatically fewer nodes, keeping its runtime almost flat even as the graph size grows.