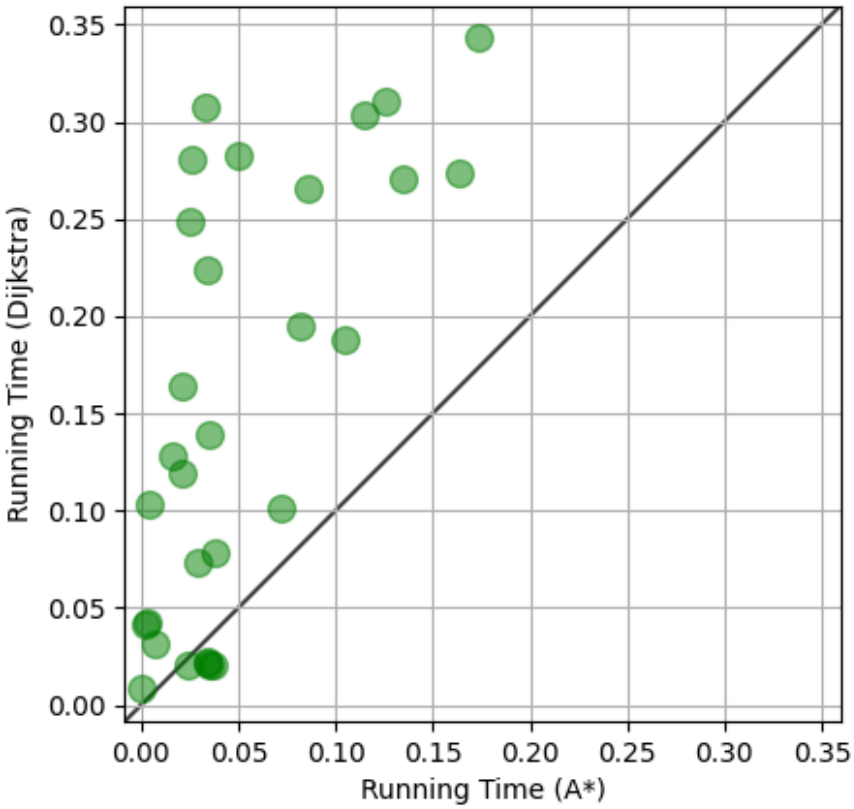
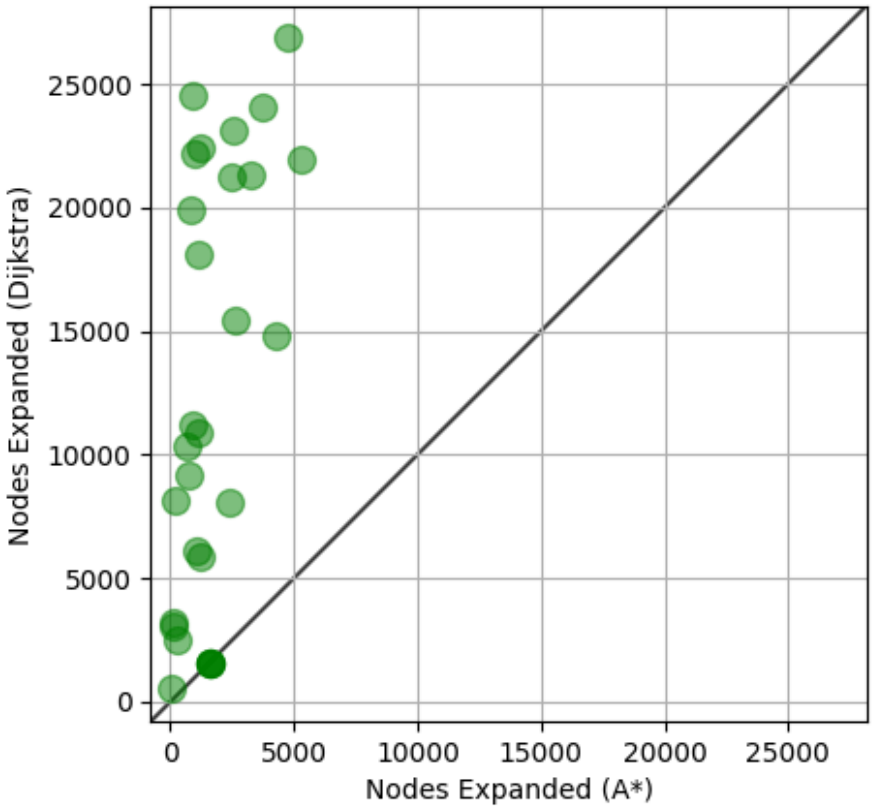
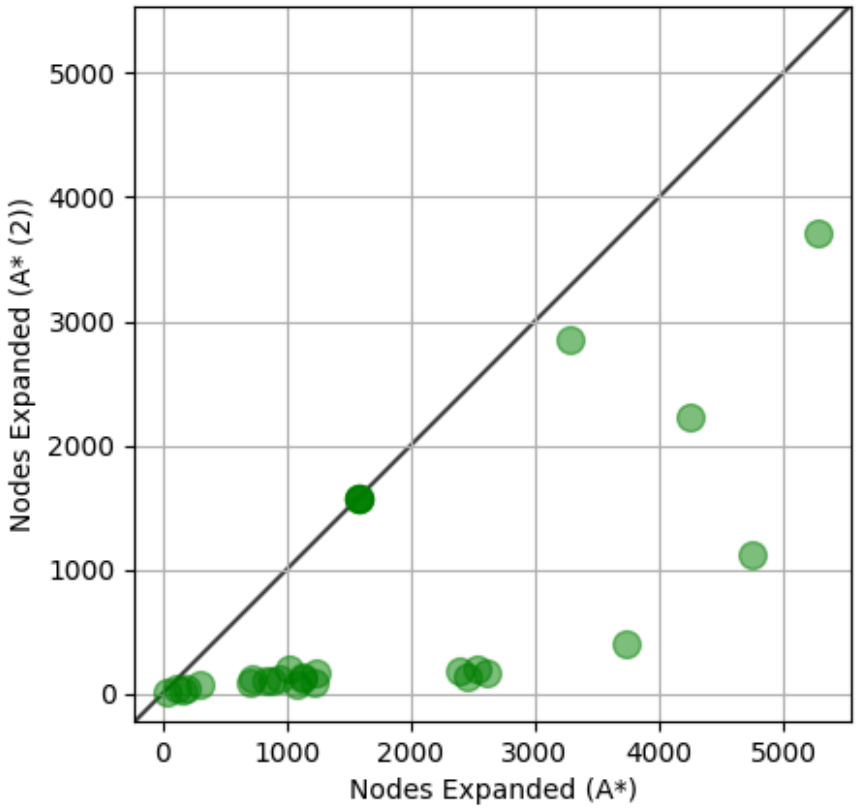
1. a) The overall distribution of points in the plots suggest that the number of nodes expanded and thus the running time of Dijkstra's algorithm is greater than those of A*. This is the result of A* expanding nodes that are closer to the goal first, though each expansion is slower, rather than expanding all nodes relative only to the start. The result is that the goal state is expanded sooner and with less nodes expanded prior to the goal node, resulting in a less expansions and time before termination.

   b) Overall, the running time is shifted slightly closer to the line indicating even running time, especially for problems that require more expansions, compared to the graph of expansions. For any given expansion, A* has to perform extra computation compared to Dijkstra to be able to compute the heuristic value and thus the cost. As such, when there are more expanded nodes, the difference in time of each expansion becomes more noticeable. Thus the difference in expansions will be larger than the difference in time, since for the same number of expansions A* will take longer, but since A* has significantly less expansions overall it still takes less time than Dijkstra's.


Running Time (Dijkstra)
Running Time (A*)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35



2. When you multiply the heuristic by 2, A* will then further prioritize expanding nodes that are closer to the goal. This results in less nodes being expanded. This is due to the fact that there are less tiebreakers between states, and also that the algorithm acts more greedy, picking states closer to the goal, thus expanding the goal sooner. Consequently, the weighter heuristic also has a lower run time. However, the consequence of this is that the heuristic given the situation becomes inadmissible, thus it still may return the optimal cost or path, but it is not guaranteed to return the optimal cost or path. This is evident in a 1x3 grid with start and goal on either end that the heuristic becomes inadmissible. The estimate of the middle state is 3 but the true cost is only 1 and the true cost of the start state is 4 when in reality it is 2. Thus in this case there is a trade off between speed and optimal path and cost. In testing, this was clear since the weighted heuristic of 2 provides a different non-optimal path and cost to the admissible heuristic, so it failed the test cases despite the same problems.



3. In the case of no longer updating the cost and parent for a given node when it finds a better path, the algorithms no longer are guaranteed to return the optimal path. The proof of this goes by nature of the situation. A path to a given node is suboptimal if there exists a path from the start to that node such that the path has a lower cost. The situation implies that if there exists a node n such that the cost of its current path is > the cost of a different path, thus its current path is non-optimal. Thus if there exists a path to n with a lower cost then its current path is suboptimal, thus we can conclude that A* and Dijkstra's no longer outputs optimal cost and paths for all problems. However, considering our specific problem, A* will no longer output the optimal path and cost. This is due to the fact that A* does not consider walls thus, it may expand a node that is closer to the goal, but not within the optimal path, before it expands a node along the optimal path, thus it cannot update the cost and parent to reflect the optimal path, thus returning sub optimal path and cost. This is evident through modifying the code resulting in modified A* overestimated the cost, and resulted in a suboptimal path. However, when running modified Dijkstra's algorithm there often is no change in the cost of the optimal path, the path is dependent on implementation, if random tie breaking, either way the modified will return the first

path it sees to the finish. This is due to the fact that vertical and horizontal movements cost 1 while diagonal costs 1.5, thus the combination of a vertical and horizontal is greater than a diagonal move. In combination with the fact that we will always expand diagonally before we are able to expand either vertically or horizontally then expand horizontally or vertically, when we expand diagonally the path will have its optimal value already, thus no other path may be better, but only as good. Thus given this situation, Dijkstra's cost is unchanged and potentially it’s path depending on implementation.