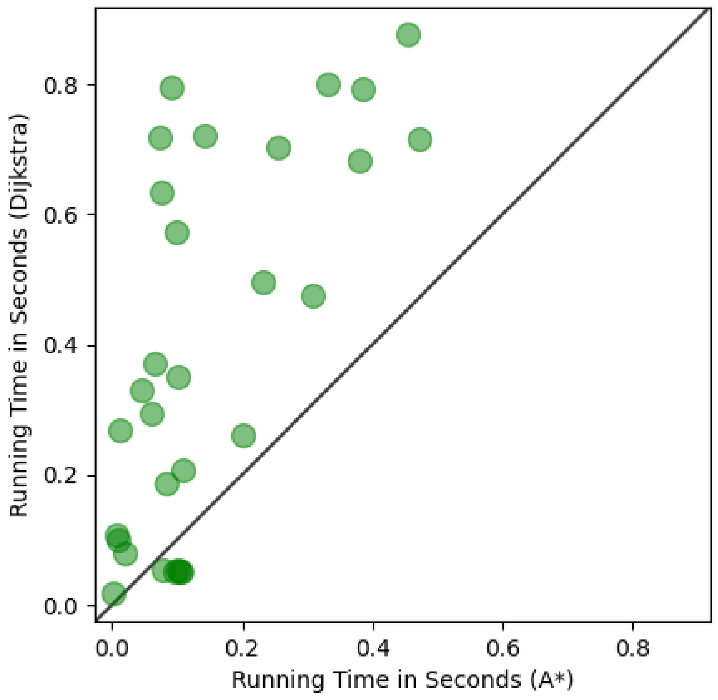
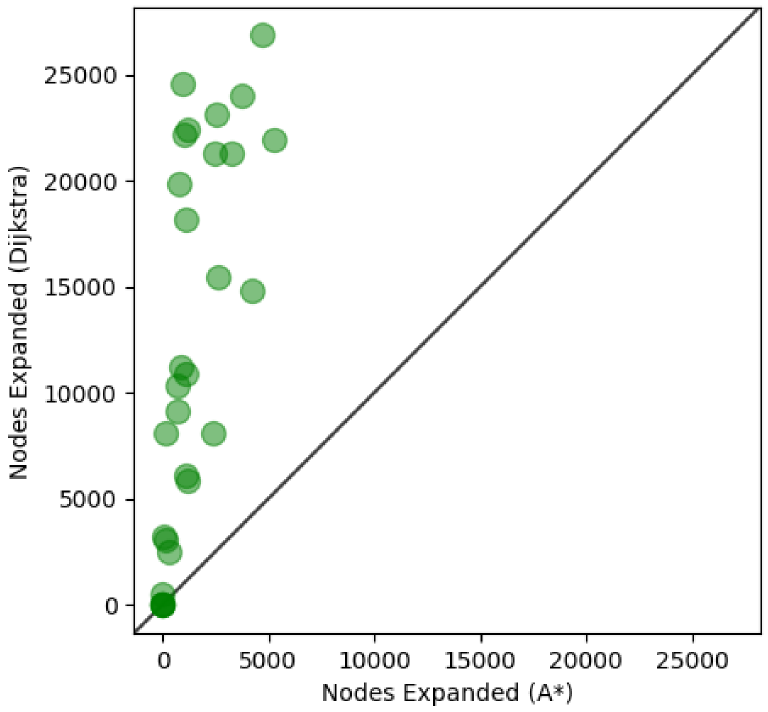
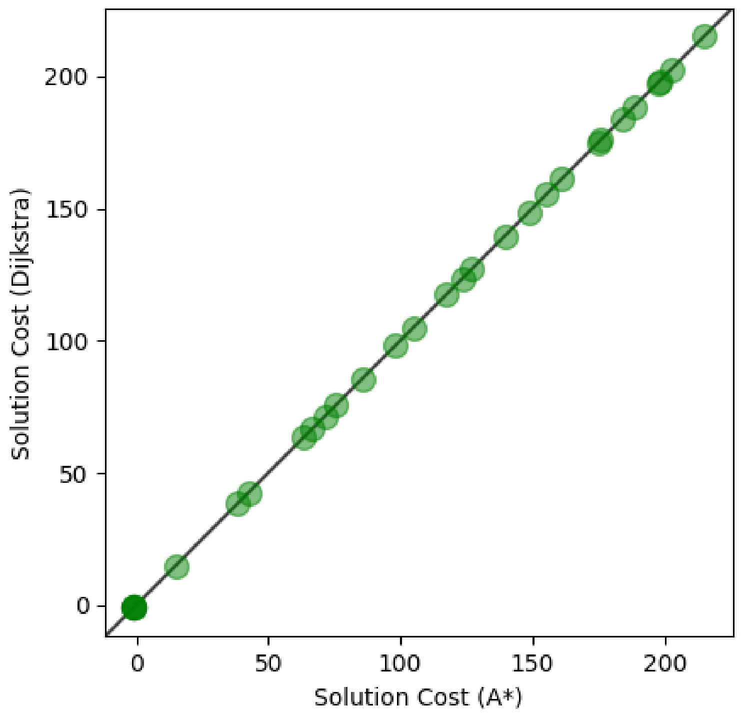
1)


Running Time in Seconds (Dijkstra)
Running Time in Seconds (A*)
0.0
0.2
0.4
0.6
0.8

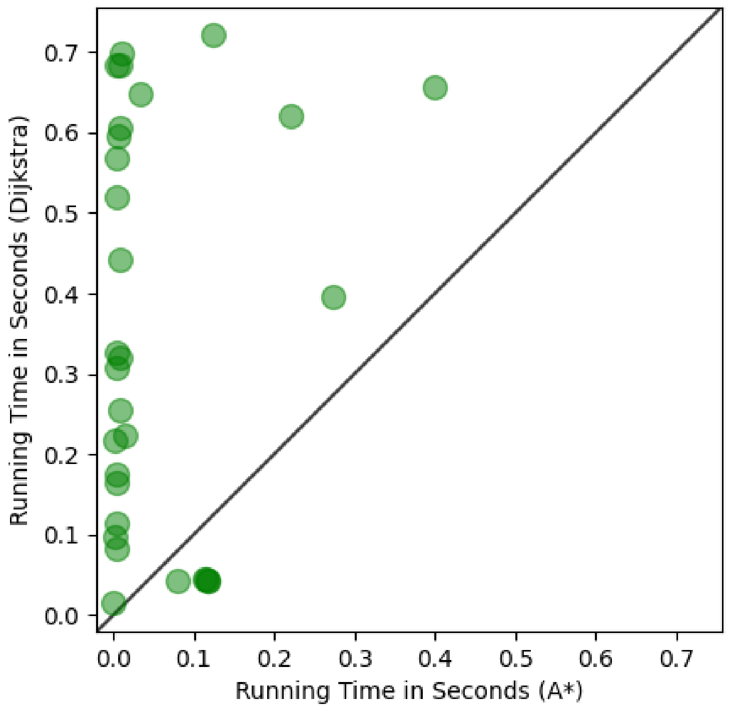

Nodes Expanded (Dijkstra)
Nodes Expanded (A*)
0
5000
10000
15000
20000
25000



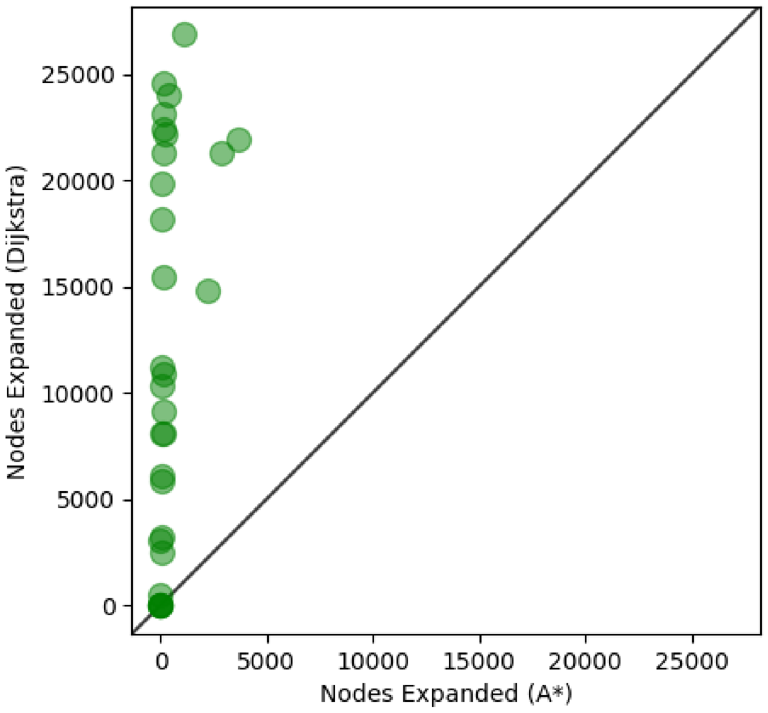
Three plots generated are 1) comparison between Running time of A* and Dijkstra. We can clearly see that Dijkstra is taking more time than A*. 2) comparison between Nodes Expanded of A* and Dijkstra. Now we can observe that Dijkstra is opening way more nodes than A*. 3) comparison between Solution cost of A* and Dijkstra. We get same answers from both algos. Running time and nodes expanded graph is kind of similar because if more nodes are expanded which mean more time is used to expand these nodes and. Reach the solution as we can clearly see in the graph that Dijkstra takes more time because it expands more nodes. Solution Cost is basically length of shortest path. Since both A* and Dijkstra are supposed to give shortest path, they will have almost equal solution and hence that's what the graph shows.

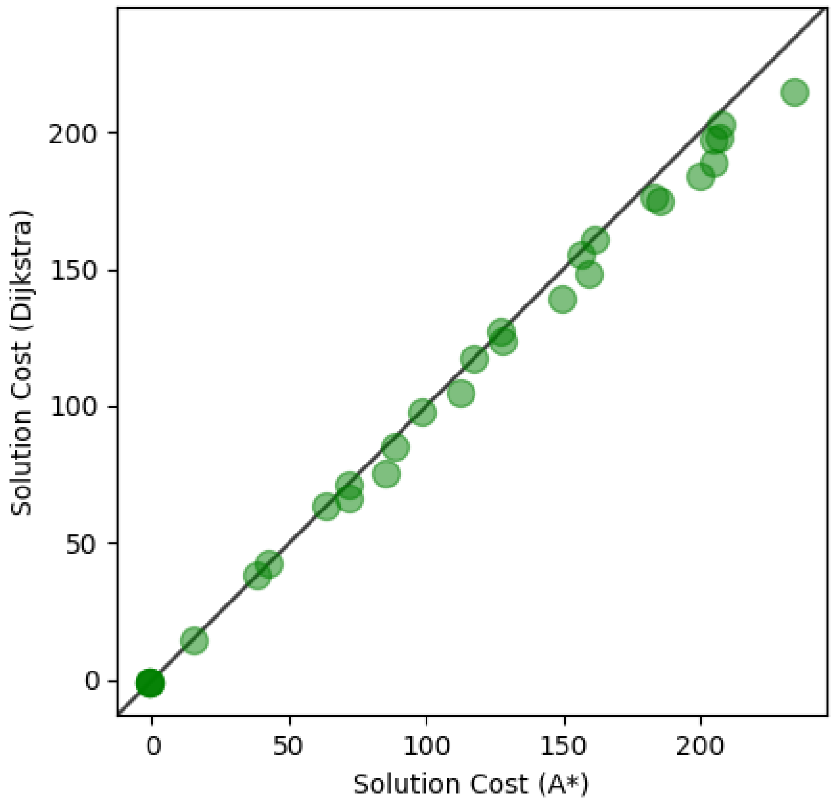
We can see few exceptions for A* where more nodes are expanded or run time is slightly higher compared to rest of test cases. It could be because when h(s) or heuristic is zero (in which case A* will act as Dijkstra) or smaller than number of nodes expansion will increase and so will runtime for those cases, but it is guaranteed to find the shortest path because it is lower than the cost of moving from n to the goal. But when h(s) is exactly equal it is then the most optimal and finds the best shortest path.

2)



Running Time in Seconds (Dijkstra)
Running Time in Seconds (A*)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Nodes Expanded (Dijkstra)
Nodes Expanded (A*)
0
5000
10000
15000
20000
25000



We got same three plots again for which Dijkstra is same because no role of h(s) or heuristic in Dijkstra in it. But the results for A* have been changed since we changed the H(s) and multiplied it by 2 we can see the run time for it got faster but compromised the optimality of the solution. Runtime can be seen in the runtime graph and if we compare it to nodes graph one can see that nodes expanded were also reduced and hence faster runtime. We can also observe the solution cost is no more linear it is because the A* is no more giving optimal solution. Now if we compare it to previous graphs, we can see less run time for A* in this graph but almost similar for Dijkstra in almost all test cases. Less nodes expanded for A* as compared to previous graph but almost similar results for Dijkstra in almost all test cases. The solutions are not as optimal for A* as compared to A* in previous graphs.
In this case we can that the h(s) is greater than the cost of moving from n to the goal. Hence A* is not guaranteed to find a shortest path but it can run faster as we saw in above graphs.
And in case h(s) is very large as compared to g(n) it will act as a Greedy BFS.