Implementation of A\* and TSP Solver to Determine the Shortest Route

Name: Yogesh Venkatesa Perumal

Roll: 106118110

Contents

[2 Heading 1 1](#_Toc161042498)

[2.1 Problem statement 1](#_Toc161042499)

[3 About A\* Algorithm 2](#_Toc161042500)

[3.1 About A\* algorithm and TSP 2](#_Toc161042501)

[3.1.1 Traveling Salesperson Problem (TSP): 2](#_Toc161042502)

[3.1.2 About A\* Algorithm: 2](#_Toc161042503)

[4 Project Setup 3](#_Toc161042504)

[4.1 Introduction 3](#_Toc161042505)

[4.2 Initialising grid 3](#_Toc161042506)

[4.3 Computation of Distance Matrix Elements 3](#_Toc161042507)

[4.4 TSP Solver 4](#_Toc161042508)

[5 Conclusion 5](#_Toc161042509)

[5.1 Determination of the Shortest Route 5](#_Toc161042510)

[5.2 Convergence Graph 5](#_Toc161042511)

# Problem Statement

Given a grid of a side length (say 20), and randomised waypoints(nodes) and obstacles within the grid, determine the shortest distance that needs to be travelled so that one can start from any node, travel through all nodes exactly once and reach back to the starting node.

# About A\* Algorithm and TSP

## Introduction

The Traveling Salesperson Problem (TSP) and the A\* algorithm are two approaches used in computer science to solve problems related to finding the shortest path.

In the context of this project, the distances between every pair of waypoints are computed via A\* algorithm. I.e. the elements of distance matrix are computed using A\* algorithm. This matrix is then used as the input to the tsp solver to determine the order in which we must travel through the nodes.

### Traveling Salesperson Problem (TSP):

The Traveling Salesperson Problem is a classic algorithmic problem in the field of computer science and operations research. Its intent is to find the shortest possible route that visits each city exactly once and returns to the origin city. The TSP is NP-hard, however, various approaches, including exact algorithms, heuristic algorithms, are used to tackle this problem.

Heuristic algorithms don’t guarantee optimal path but are yet used in practical sense as they provide faster solutions than exact algorithms. Some examples are Nearest Neighbour, Greedy, and Christofides' algorithms.

### About A\* Algorithm:

The A\* algorithm is a best-first path finding algorithm. It is used in graph traversal and pathfinding in computer science. Unlike TSP, A\* is used to find the shortest path between two nodes in a graph. It is widely utilized in applications such as network routing and AI in games.

#### How A\* Algorithm works:

A\* working can be considered as an extension to Dijkstra’s algorithm. The difference is, unlike Dijkstra’s algorithm, here the cost function of a node ‘f-score’ is given by g(n) + h(n). The intent behind calculation of cost functions is so that we could narrow down the search space of nodes, thereby speeding up the search process.

**Cost Function (g(n)):** This function represents the exact cost of the path from the start node to some node n. It indicates the distance already travelled to reach this node.

**Heuristic Function (h(n)):** This is an estimated cost from node n to the goal. Unlike g(n), h(n) provides an estimate of the shortest path from n to the goal. This heuristic function is the idea that differentiates A\* from Dijkstra’s algorithm, as it allows A\* to prioritize nodes that are believed to be closer to the goal.

It is necessary to choose a heuristic function that does not overestimate the actual cost to get to the nearest goal (also called admissible). The quality of the chosen heuristic function determines the no of steps needed to converge to the optimal solution.

**F-score (f(n)):** The F-score of a node is the sum of its g(n) and h(n) scores. f(n) represents the total estimated cost of the cheapest solution that goes through node n. A\* maintains a priority queue (often implemented as a min-heap) based on these f(n) values. The node with the lowest f(n) value is selected for expansion, as this node is considered most likely to lead to an efficient path to the goal.

# Project Setup

## Introduction

The notebook can be divided into 2 sections, the 1st is a walkthrough of all parts individually, and the 2nd section comprises of running multiple random simulations to understand the convergence of A\* algorithm.

## Initialising grid

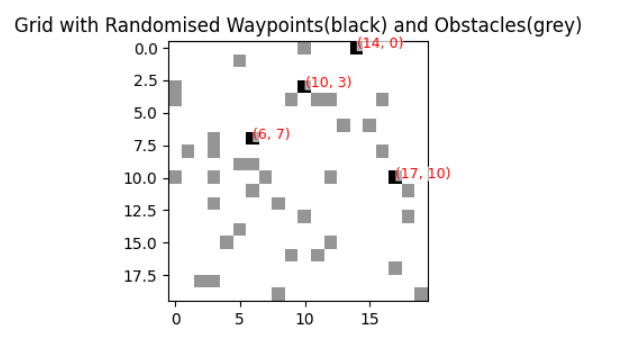
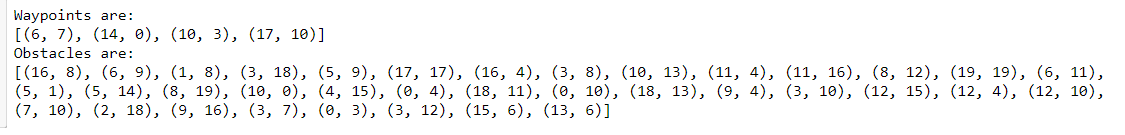


Figure 4‑1An Example Grid with waypoints and obstacles



## Computation of Distance Matrix Elements

We have 4 waypoints (6,7),(14,0),(10,3)(17,10).

A number grid with numbers

Description automatically generated

Each of these points have been determined by the A\* algorithm. The pathways are given below

A graph of a number of steps

Description automatically generatedA graph of a graph with a number of squares

Description automatically generated with medium confidenceA graph of a graph with a line of dots and a line of dots

Description automatically generated with medium confidence

A graph of a graph with numbers and a line of dots

Description automatically generated with medium confidenceA graph with numbers and a number of dots

Description automatically generated with medium confidenceA graph of a number of steps

Description automatically generated with medium confidence

A graph of a graph with numbers and a line

Description automatically generated with medium confidenceA graph with numbers and a number of dots

Description automatically generated with medium confidenceA graph with numbers and dots

Description automatically generated with medium confidence

A graph of a graph with numbers and a line of dots

Description automatically generated with medium confidenceA graph of a number of squares

Description automatically generated with medium confidence A graph with a number of squares

Description automatically generated with medium confidence

## TSP Solver

After determination of distance matrix, it is passed onto the implementation of a solution to the Traveling Salesman Problem (TSP) using Google's OR-Tools, a suite of optimization tools designed for solving combinatorial optimization problems.

If a solution is found, the path and total distance is calculated. It iterates through the route, starting from the first node, and continues until the end of the route is reached. The distance for each leg of the journey is accumulated to calculate the total route distance.

# Conclusion

## Determination of the Shortest Route

A visual plot of the route determined is shown below.

A graph of a graph of a number of squares

Description automatically generated with medium confidence

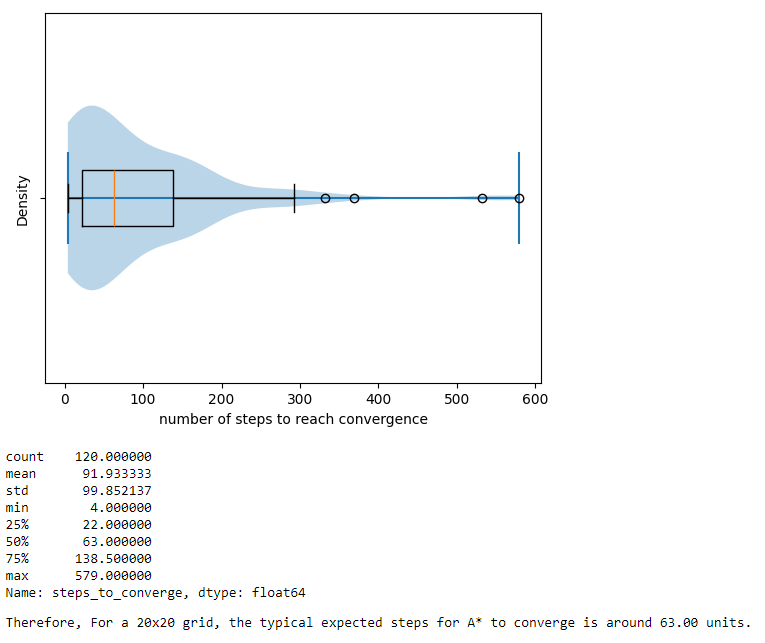
## Plotting Convergence Graph

On running 10 random simulations, we have obtained the following datapoints.

A screenshot of a white sheet with numbers

Description automatically generated

The distribution of the ‘steps\_to\_converge’ datapoints have been graphically shown below.



The distribution of the steps reveals that the number of steps to reach convergence does not follow any normal distribution. Most optimal paths tend to be computed quickly (~63-91 steps), but some distance take relatively much more time (~600 steps) making them outliers.

Thus, for a grid of a side length of 20x20, it can be estimated that number of steps that A\* algorithm would need to take to converge to the optimum is around 63 (median) to 91 (mean).