BIRZEIT UNIVERSITY

Computer Science Department
Dr. Radi Jarrar
ARTIFICIAL INTELLIGENCE COMP338
Section: 1 T, R.

Problem-solving by Search

the best path using A* and Greedy Best-First algorithms

This report was written for the Artificial intelligence coursework, specifically the first project. In which, the development process of the project is described.

Summary

Throughout the development of this project, the process was divided into 3 sections. The first section (Data Collection) collects data including Cities, roads, and air distance comma-separated Values (CSV) files. The second section (Methodology) is concerned with the selected algorithms, their implementation, and the justification of that selection. The third section (User Interface) presents the user interface specifications.

Sondos Ahmad Aabed
1190652
Project 1

Problem-solving by Search

Best path using A* & Greedy Best-First

Contents

Problem-solving by Search

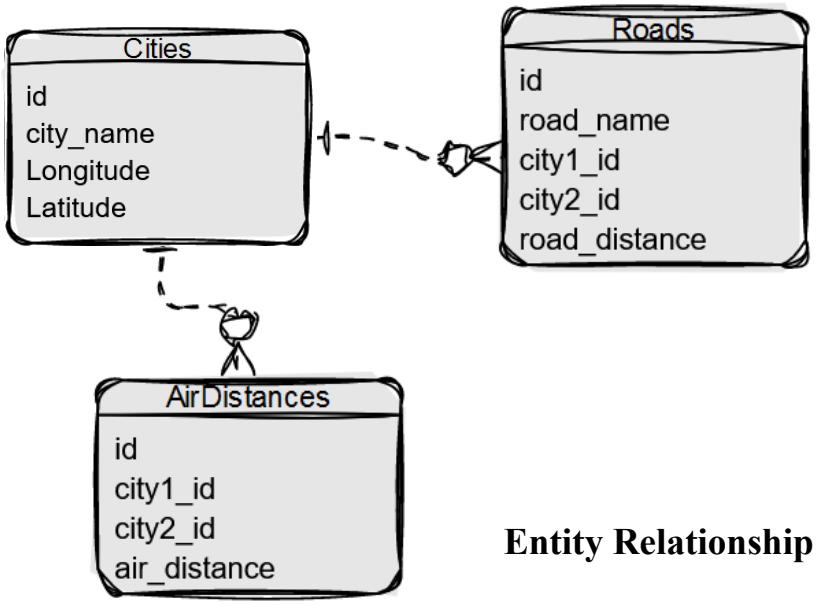
Best path using A* & Greedy Best-First

Introduction

In this project, search is used as an approach to solving a problem. This problem is finding the best path, meaning the shortest path between a source and a destination defined by the user. The solution will be found using algorithms that depend on three entities (cities, roads, and air distance), these entities construct a Map. The road distance beside the Air distance between the two cities is considered.

The following picture shows the Entity Relationship Diagram (ERD) of the Map where:

- Each city has an identifier (integer), a name (string), a Longitude (double), and a Latitude (double).
- Each road has an identifier (integer), a name (string), connects two city ids (integers), and road distance in KM (double).
- Each air distance has an identifier (integer), two city ids (integers), and air distance in KM (double).
- Each city may have many roads that connect it with different cities.
- Each city may have many Air distances from other cities.



Entity Relationship Diagram (ERD)

Data Collection

In this section data collection is presented, which was done using different web pages cited alongside google maps.

Cities

Starting with city names, the following table shows 15 Palestinian cities (Mongarby, 2019) alongside a number identifier, Longitude, and latitude (Google Maps, 2023). The following CSV file will be used as a sample input for the application:

id	city_nam	latitude	longitude
1	Jerusalem	31.77092859	35.21209453
2	Al-Bireh	31.9141773	35.22528028
3	Al-Khalil	31.53315381	35.10197177
4	Gaza	31.50220687	34.46830362
5	Nablus	32.22314778	35.26175553
6	Beir Al-Sabi	31.25293014	34.79586518
7	Yaffa	32.04968926	34.75914145
8	Bethlehem	31.7061114	35.20480278
9	Jenin	32.46445523	35.29434086
10	Haifa	32.79395954	34.98722461
11	Tulkarm	32.31904401	35.02431843
12	Akka	32.93350057	35.08539111
13	Jericho	31.86135947	35.46223949
14	Nasreh	32.69901236	35.30097544
15	Arad	31.26193383	35.21770829

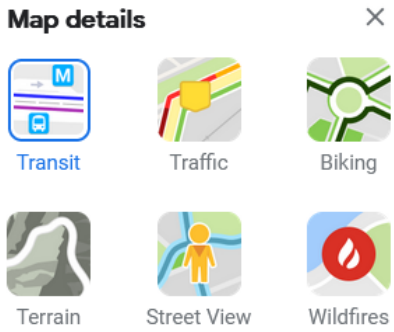
Cities table

Problem-solving by Search

Best path using A* & Greedy Best-First

Roads

For adjacent cities there exists at least one road that reaches a city from the other adjoining city. To collect the roads information, the transit mode of Google Maps was used as shown in the picture below:



(Google Maps, 2023)

The following CSV file contains road distances between different cities and it will be used as input for this project:

id	road_name	city1_ID	city2_ID	road_distance
1	Al-Quds road	2	1	22.6
2	Al-Khalil road	1	8	9.5
3	Jericho-Al-Quds road	13	1	44.4
4	Yaffa-Alquds road	7	1	64.6
5	Nablus road	2	5	73.9
6	Allon road	13	2	57.7
7	road 60	8	3	39.9
8	road 40	3	6	102
9	road 25	4	6	60
10	Amman road	5	9	44
11	Haifa road	5	11	40.8
12	road 60	6	3	60
13	road 5	11	7	59.5
14	road 60	9	14	31.2
15	road 22	10	12	25.5
16	Al-Sikkeh road	11	9	50.5
17	road 77	12	14	41
18	Ghandy road	5	13	111
19	road 79	14	12	44.8
20	road 31	15	6	47.2
21	road 3	7	4	88
22	road 22	14	10	48.5
23	Al-Bahr Al-Mayet road	13	15	119

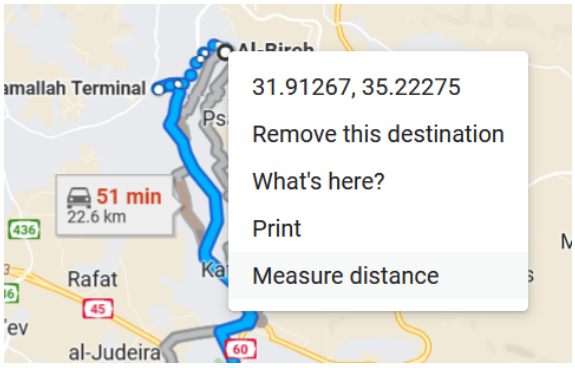
Roads Table

Problem-solving by Search

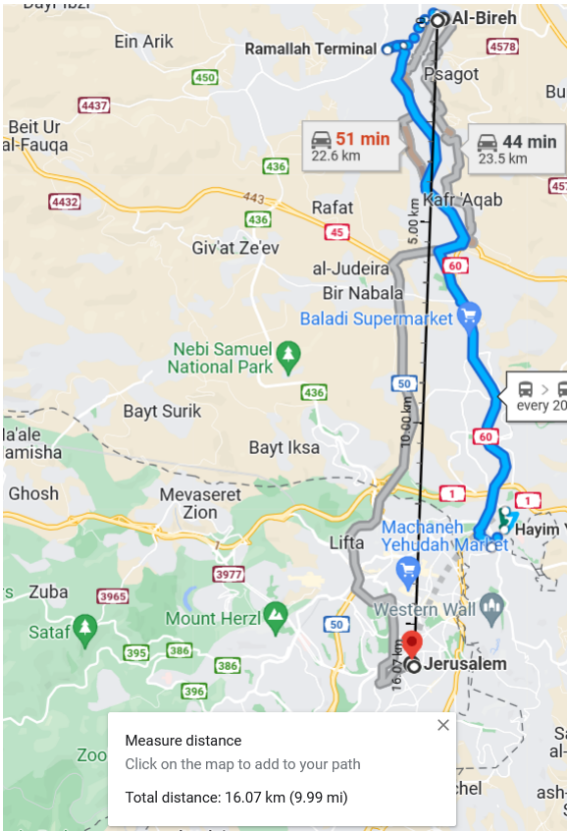
Best path using A* & Greedy Best-First

Air distances

For each road there exists an aerial distance in the best-case scenario it is the distance that is taken by plane disregarding the Palestinian context (occupation). To collect the air distance of the early collected roads information, the measure distance tool of Google Maps is used as shown in the pictures below:



Measure distance tool (Google Maps, 2023)



The air distance is shown (Google Maps, 2023)

This process results in the following table:

id	city1_ID	city2_ID	air_distance
1	2	1	16.07
2	1	8	7.11
3	13	1	25.44
4	7	1	52.59
5	2	5	34.67
6	2	13	23.28
7	8	3	17.16
8	3	6	46.09
9	4	6	40.88
10	5	9	26.98
11	5	11	28.83
12	6	3	43.39
13	7	11	39.14
14	9	14	26.19
15	10	12	18.11
16	11	9	29.79
17	12	14	33.29
18	13	5	44.76
19	14	12	33.3
20	15	6	40.32
21	7	4	67.04
22	14	10	31.41
23	13	15	69.96

Air distances Table

Methodology

As stated early in the introduction, the methodology used in this project approaches the problem using search as a solution. The A* algorithm was required to be implemented alongside another algorithm. In this section, both algorithms are overviewed and the justification for the other algorithm is selected.

A* algorithm

A* works by making a lowest-cost path tree from the start node to the target node. What makes A* different and better for many searches is that for each node, A* uses a function f(n) that gives an estimate of the total cost of a path using that node. Therefore, A* is a heuristic function, which differs from an algorithm in that a heuristic is more of an estimate and is not necessarily provably correct. (Brilliant, 2023)

A* expands paths that are already less expensive by using this function:

$$\mathbf{f(n) = g(n) + h(n)}$$

f(n) = total estimated cost of the path through node n.
g(n) = cost so far to reach node n.
h(n) = estimated cost from n to goal.

This is the heuristic part of the cost function, so it is like a guess.

A* properties: complete, optimal if the heuristic function is admissible, its time complexity is exponential O(bd), its space complexity with the worst case of O(bd).

In this case, the g (n) is obtained using the collected data in the Air distances file.

Problem-solving by Search

Best path using A* & Greedy Best-First

Greedy Best-First Search

Greedy best-first search is an informed search algorithm where the evaluation function is strictly equal to the heuristic function, disregarding the edge weights in a weighted graph. To get from a start node to a target node, the lowest value resulting from some heuristic function, h(x), is considered as the successive node to traverse to. The goal is to choose the quickest and shortest path to the target node.

The evaluation function, f(x), for the greedy best-first search algorithm, is the following:

$$f(x) = h(x)$$

Here, the evaluation function is equal to the heuristic function. Since this search disregards edge weights, finding the lowest-cost path is not guaranteed.

Suppose a bot is trying to move from point A to point B. In a greedy best-first search, the bot will choose to move to the position that brings it closest to the goal, disregarding if another position ultimately yields a shorter distance. In the case that there is an obstruction, it will evaluate the previous nodes with the shortest distance to the goal, and continuously choose the node that is closest to the goal. (Young, 2022)

The problem with the greedy search is that it can keep expanding paths that are already very expensive.

Problem-solving by Search

Best path using A* & Greedy Best-First

Justification

The choice was made to implement the Greedy Best-First search algorithm for the following reasons:

1- **Informed search starter (or heuristic-based approach)**: Since the heuristic air distance was already collected in the early process of this project, this was taken into consideration choosing the other algorithm. Because an uninformed algorithm would not use the collected data, the decision was to choose from among informed search algorithms.

2- **Computational cost (or efficiency)**: after considering informed search algorithms only, the option is left between Uniform cost search and greedy search. In that case, the trade-off is between reaching the optimal solution with costing computational time and reaching a greedy solution with less computational time. Since the optimal solution will already be reached using the A* search algorithm, then the focus was on the computational time. Therefore the greedy search algorithm was selected.

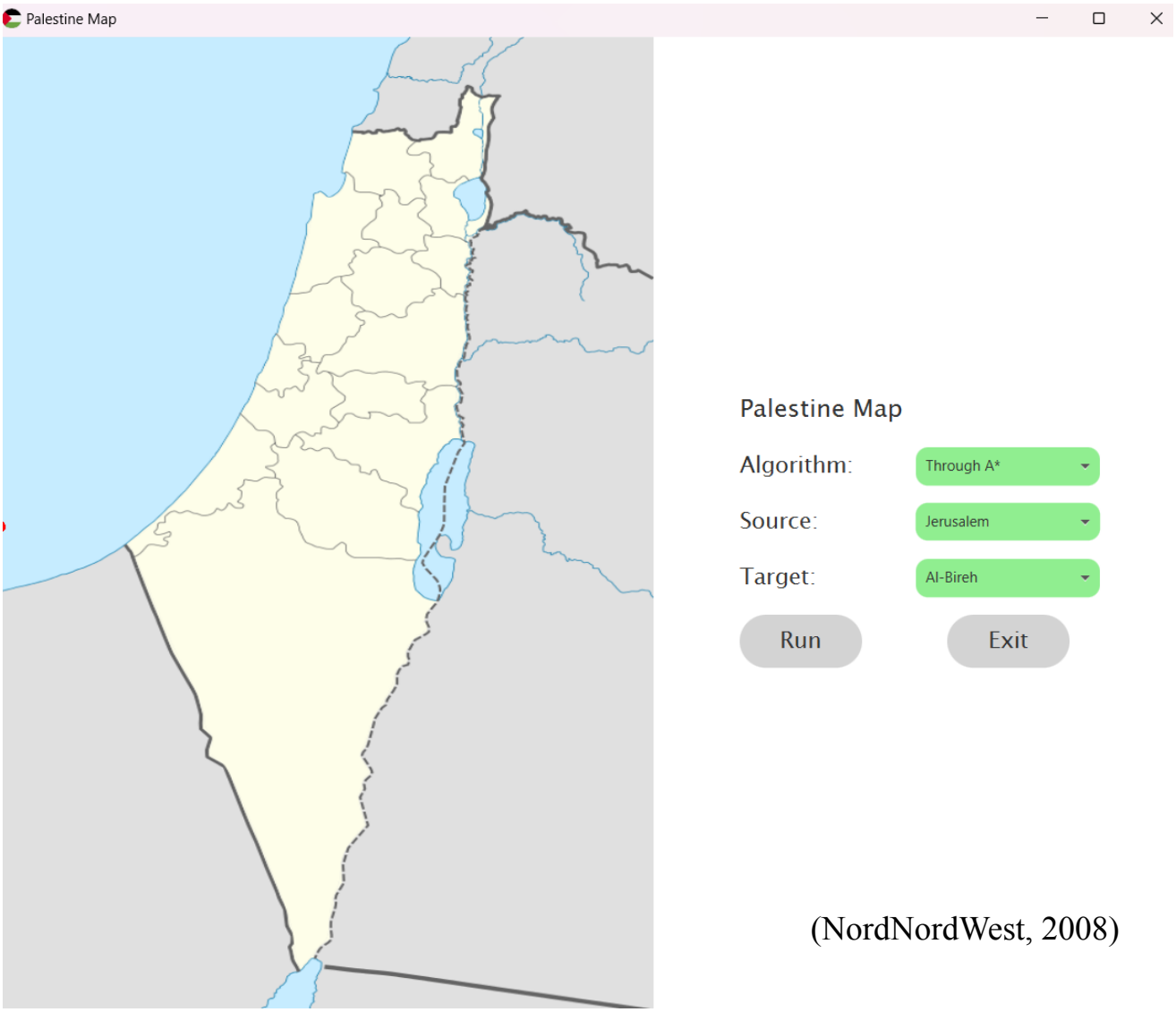
Problem-solving by Search

Best path using A* & Greedy Best-First

User Interface

In this section, the work done regarding the user interface specifications is presented. An input that the user inputs and the actual outputs are shown beside the time complexity of the running program.

The below image of Palestine was used as the map of the program. Where each of the input cities will be added to it alongside the streets, in addition to the Palestinian flag as a Logo:



(NordNordWest, 2008)

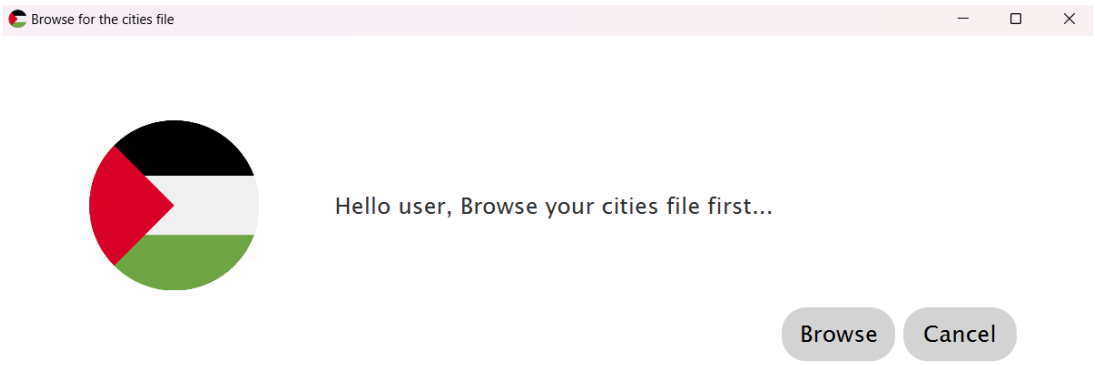
Problem-solving by Search

Best path using A* & Greedy Best-First

Inputs

There are two types of inputs in this project:

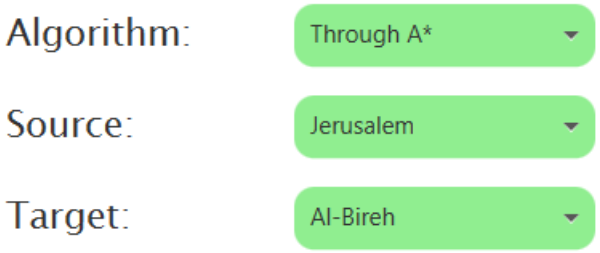
1- **Map inputs:** the inputs regarding the map information. Including cities, roads, and air distances CSV files. Which will be asked to be inputted one by one the moment the user starts the program.



2- **Path inputs**: the inputs regarding the best path (cities). Which will be asked to be inputted when the Map is ready to run, Including

   a. **Source city:** the user has the option to select a starting point.

   b. **Destination city:** the user has the option to select the ending point.



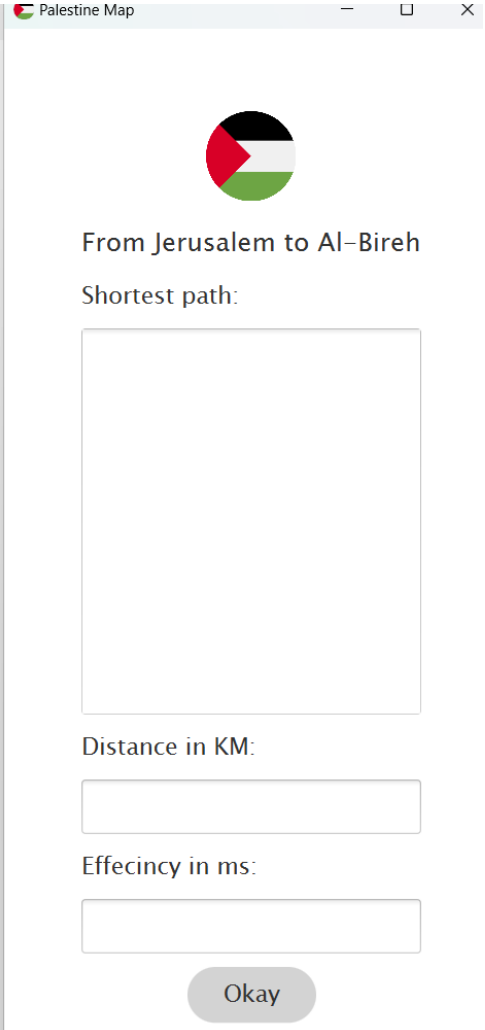
Problem-solving by Search

Best path using A* & Greedy Best-First

Outputs

The outputs when running the program with the previously explained inputs includes the following:

1- **Best path**: the full best path should be shown on the map.
2- **Node distance**: The distance between each city should be present.
3- **Total cost**: the total cost (distance) should be shown.
4- **Efficiency**: the time complexity of the program, and the running time in milliseconds.



Improvement: Palestinian Roads Context

One of the improvements that could have been made to the project is considering the context of the Palestinian roads while working on data collection. The state of the streets and the road distances are continuously and actively changing. It is said to be unpredictable by the people due to the outer forces of occupation (road obstacles), corruption (closing a road for no urgent reason), and the working nature of the municipalities (not necessarily planned before time).

For example, Gaza is one of the main Palestinian cities, but due to the Gaza siege, it was found to be not reachable in the stage of data collection. Despite that, two roads have been virtually created from Jaffa to reach Gaza and Bier Al-Sabi from Gaza.

Conclusion

Overall, this project provides a problem-solving approach using search to find the best path taken. It asks the user for inputs of the map, shows them on the user interface, and lets the user interact with the map. The data was collected as a sample for testing the project. The user is able to decide which destination they want from a specific starting point they chose. The project implements A* to reach the optimal solution, alongside the Greedy best-first search.

References

Brilliant. (2023, 6). *A* search.* Retrieved from Brilliant: https://brilliant.org/wiki/a-star-search/

Google. (2023, 6). *Google Maps Palestine Country.* Retrieved from Google Maps: https://www.google.com/maps/place/Palestine/@31.7719527,34.140914,8.58z/data=!4m3!3m2!1s0x151cf2d28866bdd9:0xee17a001d166f686!16zL20vMDFrMHA0!5m1!1e2?entry=ttu

Mongarby. (2019, 7). *Largest cities in Palestine.* Retrieved from Mongarby: https://population.mongabay.com/population/palestine/

NordNordWest. (2008, 12). *Mandatory Palestine Location.* Retrieved from Wikipedia: https://ar.m.wikipedia.org/wiki/%D9%85%D9%84%D9%81:Mandatory_Palestine_location_map.svg

young, C. (2022, 4). *Greedy best-first search.* Retrieved from Codecademy: https://www.codecademy.com/resources/docs/ai/search-algorithms/greedy-best-first-search