COMP 6721
Introduction to Artificial Intelligence

Montreal Crime Analytics

Project-1 Report
Submitted on 14th October, 2019

Submitted by:

Student Id	Name	Email
40082906	Harsh Deep Kour	harshdeep.kour93@yahoo.com
40082312	Iknoor Singh Arora	iknoor438@gmail.com

Table of Contents

1. Introduction

The Project focuses on analysis of crime in Montreal. The analysis is based on the data set given, which is fetched and modified from the City of Montreal's open data portal. The dataset consists of the category, time line as well as the location of crime given in the form of spatial points. The prime task lies in the creation of grids in this area to compute the crime rate statistics in each grid and come up with an optimal path from source to destination which avoids major crime affected areas.

1.1 Technical Details

IDE :- Jupyter notebooks

Language :-Python(version 3.7)

Libraries used:-

- **Geopandas**:-helps in working with geospatial data in python easier[1], we have used geopandas for reading,writing shape files , for analyzing data by creating dataframes and grids as well as plotting points on the xy plane.
- **Numpy:-** It is a fundamental package for scientific computations, we have used numpy to calculate different threshold values(mean, medium and standard deviation) from considering the no. of points a specific grid.
- **Matplotlib**:-It is a plotting library in python , we have used it for plotting final path on the xy plane.
- **Time**:-We have used this library to calculate the total elapsed time in the running of the algorithm for finding the optimal path from source to the destination.

2. Data Analysis

The dataset provided consists of 5 files which are:-

crime_dt.shp:-The main file (.shp) which contains the geometry data.

	CATEGORIE	QUART	timestamp	geometry
0	Vols qualifiés	soir	2017-08-02 00:00:00	POINT (-73.55952 45.51703)
1	Introduction	jour	2017-08-03 00:00:00	POINT (-73.55510 45.52185)
2	Méfait	jour	2017-08-23 00:00:00	POINT (-73.58359 45.49012)
3	Vol dans / sur véhicule à moteur	jour	2018-09-05 00:00:00	POINT (-73.58794 45.52297)
4	Vol dans / sur véhicule à moteur	jour	2017-08-25 00:00:00	POINT (-73.55280 45.52430)

(Extracting first 5 records of the spatial data from the shape file)

crime_dt.shx:- It contains a positional index of the feature geometry to allow seeking forwards and backwards quickly.

crime_dt.dbf :-It consists of columnar attributes for each shape.

crime_dt.prj:-consists of projection description using crs(Coordinate Reference Systems)

crime_dt.cpg:-helps to identify character encoding used.[2]

After reading the file using the geopandas read_file function we get a geopandas dataframe which consists of 19010 rows and 4 columns. The last column in the frame is the geometry column which consists of the geo spatial points identifying the crime areas on the Montreal map.

The closed area in the Montreal map provided in the project description is divided into smaller blocks representing a grid layout. The blocks are represented by the geopandas polygon data structure. These blocks have a specified width and height which is taken as an input from the user.

This structured data now in the form of blocks(polygons) consisting of 4 points each is written into a grid.shp file. The leftmost point on each block is stored in a python dictionary identifying individual blocks in the xy plane. These small blocks on the map are used for further crime analysis.

```
                                              geometry
0     POLYGON ((-73.58983 45.49007, -73.58913 45.490...
1     POLYGON ((-73.58983 45.49077, -73.58913 45.490...
2     POLYGON ((-73.58983 45.49147, -73.58913 45.491...
3     POLYGON ((-73.58983 45.49217, -73.58913 45.492...
4     POLYGON ((-73.58983 45.49287, -73.58913 45.492...
5     POLYGON ((-73.58983 45.49357, -73.58913 45.493...
```

We have used a sjoin function **(geopandas.sjoin(left_df, right_df, how='inner', op='intersects', lsuffix='left', rsuffix='right')** on the crime_dt.shp file and the grid.shp file so as to generate a geopandas data frame consisting crime point spatial coordinates and the location of the point i.e in which block it lies in according to its latitude and longitude.

If a point lies on the grid of the block, the point is considered under both the blocks using the intercept parameter of sjoin in order to avoid any data loss.

```
                        geometry  index_right   FID
0     POINT (-73.55952 45.51703)         2489  2489
1     POINT (-73.55510 45.52185)         2838  2838
2     POINT (-73.58359 45.49012)          456   456
3     POINT (-73.58794 45.52297)          161   161
4     POINT (-73.55280 45.52430)         3012  3012
5     POINT (-73.55334 45.50909)         2991  2991
```

The information gathered is further used to come up the threshold used for the grid. The threshold used here is median, which is used to identify blocks with higher crime and lower crime risk. We have also used a colour scheme to colour blocks with high risk as yellow and low risk as blue. The details explored are stored in python data structures on which we run heuristic algorithms.

2.1 HEURISTIC ALGORITHMS

The entire grid of blocks is represented as a weighted graph using dictionary data structure in Python and edges being assigned weights as per its self colour and the colour of the neighbouring blocks. This weighted graph is then used as an input to identify the optimal and the shortest path in the grid. In order to compute the optimal path, we have used two different heuristic algorithms, which are:

- **Greedy Best-First Search Algorithm**

  Both BFS and DFS blindly explore paths without considering any cost function. The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore. Best First Search falls under the category of Heuristic Search or Informed Search. [3]

  Best first search algorithm is often referred greedy algorithm this is because they quickly attack the most desirable path as soon as its heuristic weight becomes the most desirable.[4]

  Pseudo code followed :-

  1. Initialise a queue_list
  2. Put Start_key into a list.
  3. Initialise a closed_list
  4. While(list is not empty)
  5. u=Remove first node from the queue
  6. if(u is equal to goal) ,then “path found”
  7. Foreach neighbour v of u
  8. If v not in closed_list
     Insert v in the queue_list.
     Put v in closed_list
  9. Sort the queue_list according to the ascending order of the sum of distance from the goal(h(n)) and the cost of moving from u to v.

The heuristic function used is :-

h(n) - the estimated cost to reach the goal from the current node.

- **A* Algorithm**

It is really a smart algorithm which separates it from the other conventional algorithms.

A* (pronounced as A-star) evaluates every node by combining g(n) and h(n), where:

g(n) - the cost to reach the node from the source

h(n) - the estimated cost to get from the node to the goal

f(n) - estimated cost of the cheapest solution, and

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

To find an optimal and cheapest path, algorithm gives priority to the lowest value of f(n). A* is chosen above all algorithms as it provides completeness and optimality over other algorithms.[5]

Pseudo code followed :-

1. Initialise open_list with start_key and close_list as empty.
2. Initialise parent_dict, travel_dict_g and travel_dict_f as empty dictionaries.
3. While open_list is not empty
4. Select next = node with minimum f(n) from the open_list
5. If next == destination_key
6. Print path found and return path from parent_dict
7. Remove next from open_list and append to the closed_list
8. Generate all the neighbours of the next.
9. If Neighbour(next) is not in closed_list
10. Calculate g(n) and f(n) of Neighbour(next)
11. If Neighbour(next) is not in open_list
12. Update parent(next) in parent_dict, g(n) of Neighbour(next) in travel_dict_g, f(n)
13. in travel_dict_f
14. else if g(n) of Neighbour(next) < travel_dict_g[neighbour(next)]
15. Update parent(next) in parent_dict, g(n) of Neighbour(next) in travel_dict_g, f(n)
16. in travel_dict_f

2.2 HEURISTIC FUNCTIONS

Performance of the algorithm depends on how well the cost or evaluation function is designed. In order to rank many alternative paths from the source and target, heuristic functions have been applied. We have used two different heuristic functions on two different grounds in order to be note the difference. Two heuristic function employed are:

- **Distance Heuristic Function**

Each block of the grid is recognised solely by the bottom left co-ordinate of the block. In order to get the shortest distance of every block from the target co-ordinate. The absolute value of distance between the x-coordinates and y-coordinates of bottom-left point of every block the target point is calculated. It will give us two values namely:

x_diff - absolute distance between the x-coordinates

y_diff - absolute distance between the y-coordinates

The maximum of x_diff and y_diff is then used as the final estimated cost of travel from that block to the target point.

heuristic_distance = max(x_diff , y_diff)

estimated_cost = heuristic_distance

• Cost and Distance Heuristic Function

Since for the project description we also have a travel cost of the distance between the blocks, we have brought the cost of the diagonal distance of the blocks into the consideration with the above used distance heuristic function. The length of the diagonal of a block is calculated. heuristic_distance from the above formula is divided by the diagonal distance and multiplied by the travel cost of each diagonal in order to give the estimated_cost of travel.

a - side of block of a rid

d - diagonal distance of each block

$$\mathbf{d = square\_root\ (2) * a}$$

$$\mathbf{estimated\_cost = heuristic\_distance * (1.5) / d}$$

Both the heuristic functions applied aims to get the estimated cost lower or equal to the actual cost of reaching the goal state since there can be no distance shorter than the absolute distance between the co-ordinates of two points. Thus, we can say that the admissibility is achieved by the below heuristic functions.

2.3 Features of Analysis and States Considered

Features are :

- Different Threshold Parameters (Mean, Median, Standard Deviation)
- Runtime of the algorithms
- Cost of the path found(optimality)
- Space Complexity of the algorithms

States Considered : We are finding a path from source co-ordinates(top-left block) to the target co-ordinates(bottom right block of the grid).

- Initial State (grid with yellow unsafe blocks and blue safe blocks)
- Goal State (grid with a path from the source to the target avoiding unsafe blocks)
- Transitioning States (consideration of blocks w.r.t color(blue, yellow) along the path)

3. RESULTS

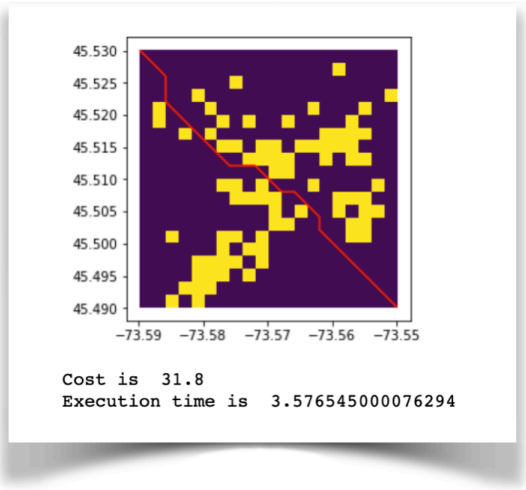
We have compared both the heuristic algorithms(A* and BFS) with both the heuristic functions stated above and then analysed the difference in path's optimality, path's cost and the run time of the algorithms. Following are the certain scenarios with different grid size and threshold values.

(1) - A* algorithm with distance heuristic
(2) - A* algorithm with cost and distance heuristic
(3) - BFS with distance heuristic
(4) - BFS with cost and distance heuristic

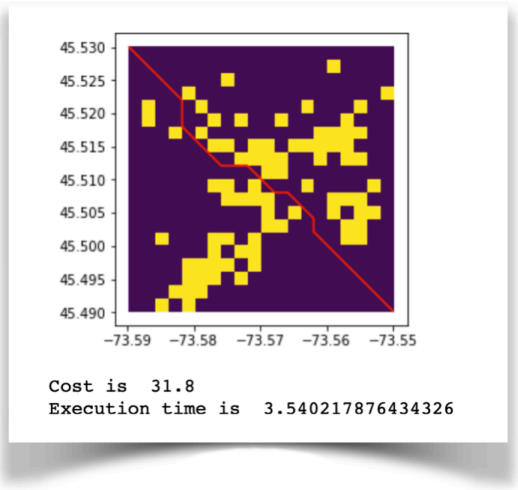
3.1 TEST SCENARIOS

- **Scenario -1**

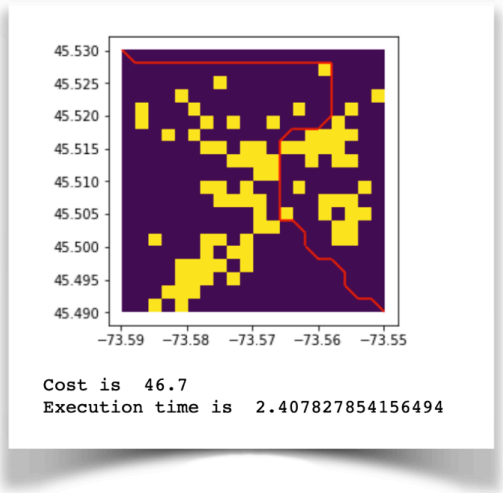
Threshold- 80, Grid size - 0.002



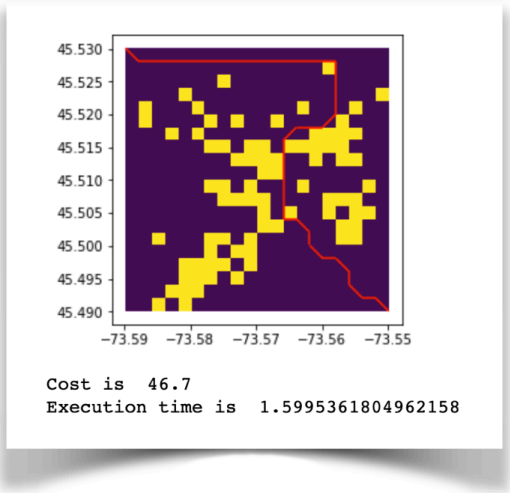
(1)



(2)



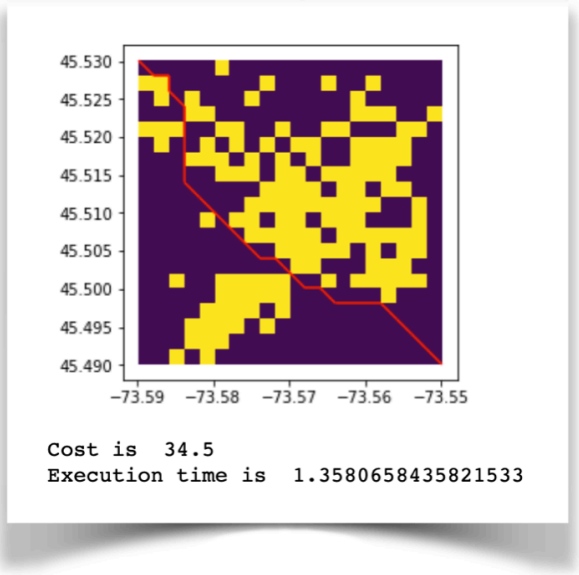
(3)



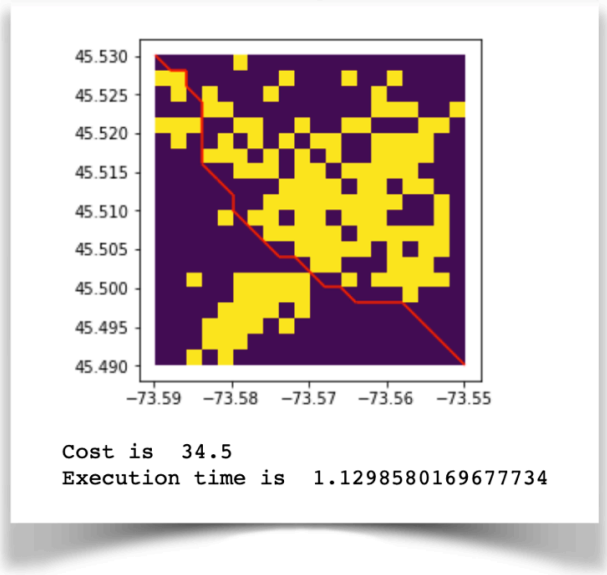
(4)

- **Scenario -2**

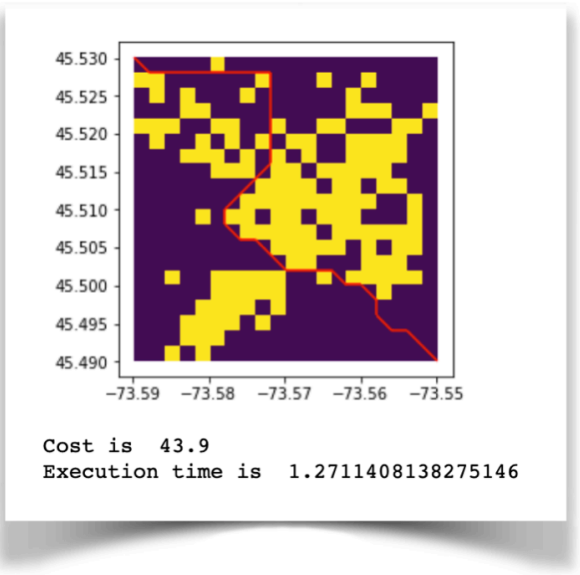
Threshold- 65, Grid size - 0.002



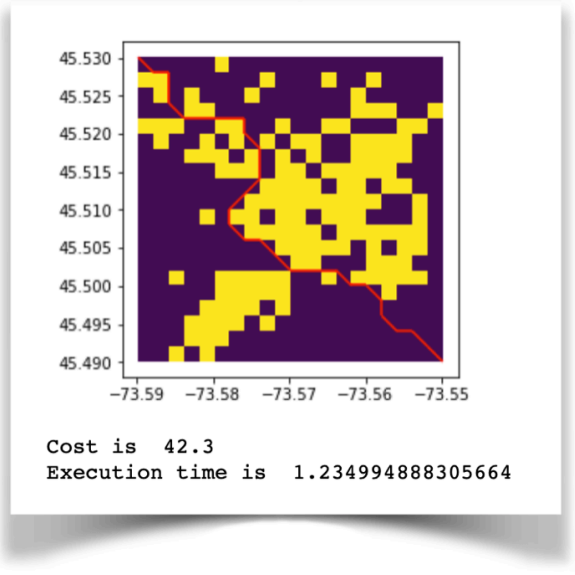
(1)



(2)



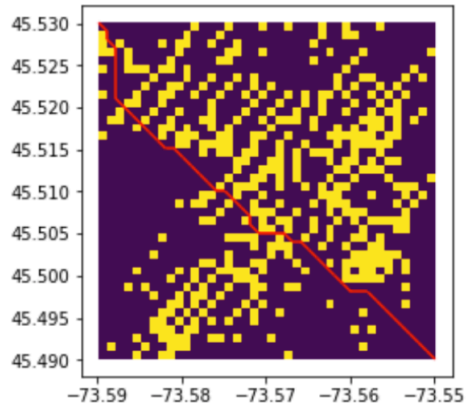
(3)



(4)

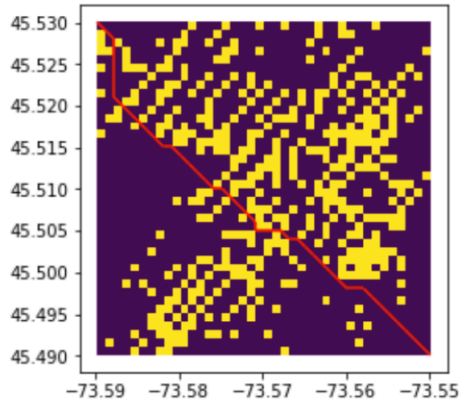
- **Scenario -3**

Threshold- 75, Grid size - 0.001



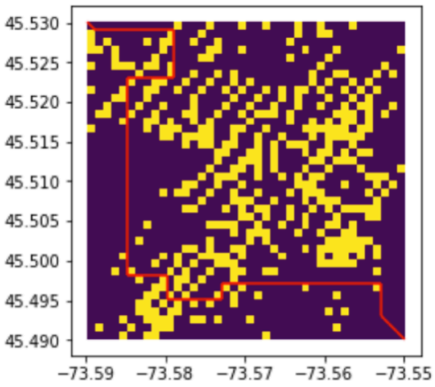

```
Cost is  66.1
Execution time is  1.333017110824585
```

(1)



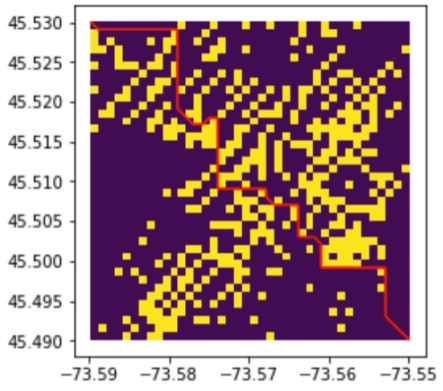

```
Cost is  66.1
Execution time is  1.1232988834381104
```

(2)




```
Cost is  105.09999999999992
Execution time is  1.2840828895568848
```

(3)

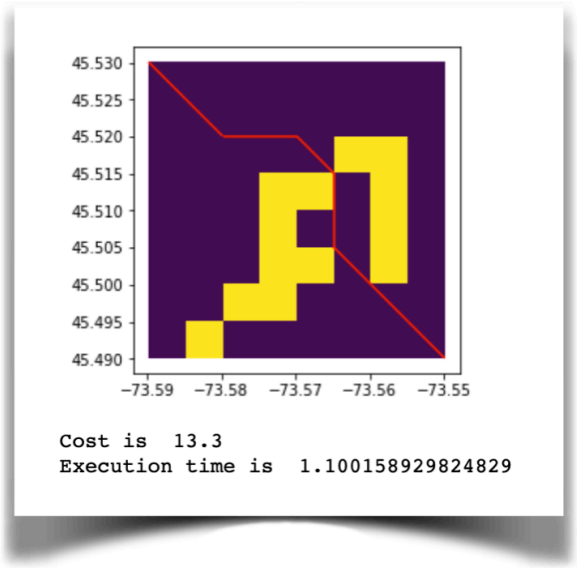



```
Cost is  89.19999999999993
Execution time is  1.2658588886260986
```

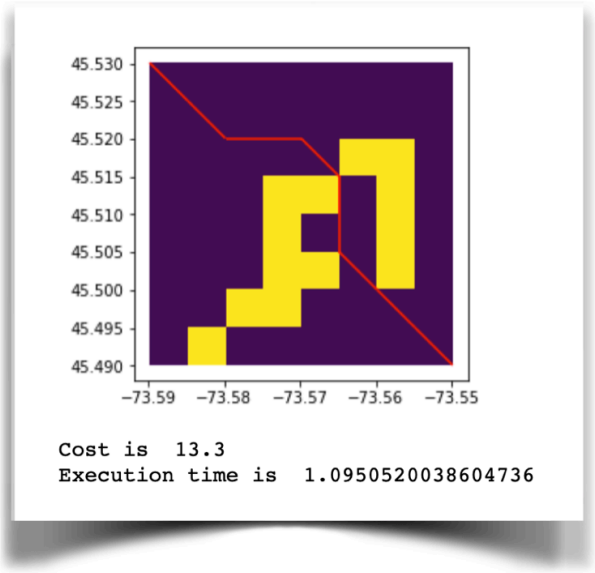
(4)

- **Scenario -4**

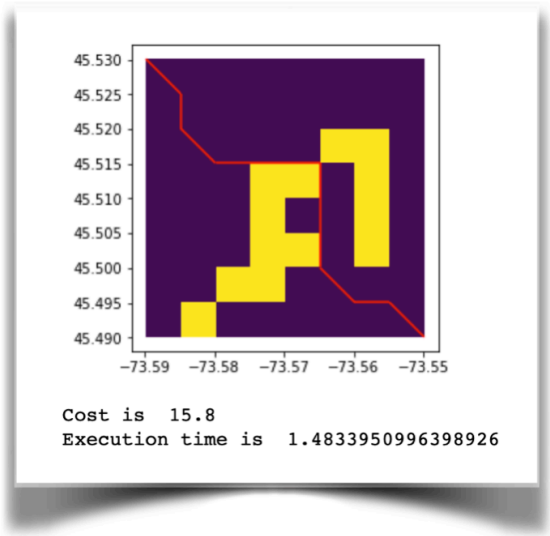
Threshold- 80, Grid size - 0.005



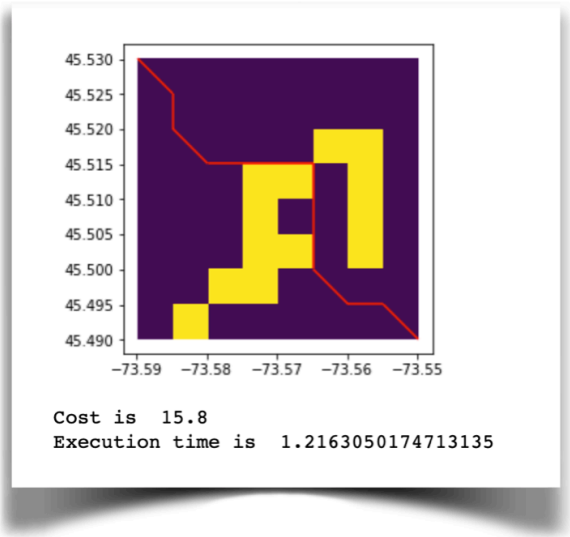
(1)



(2)



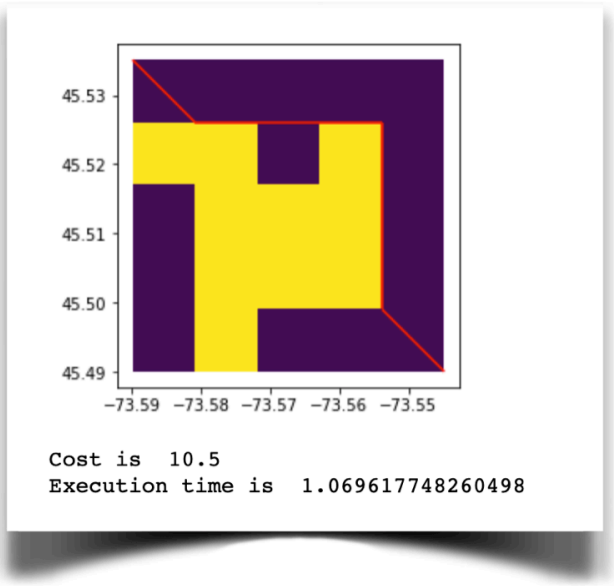
(3)



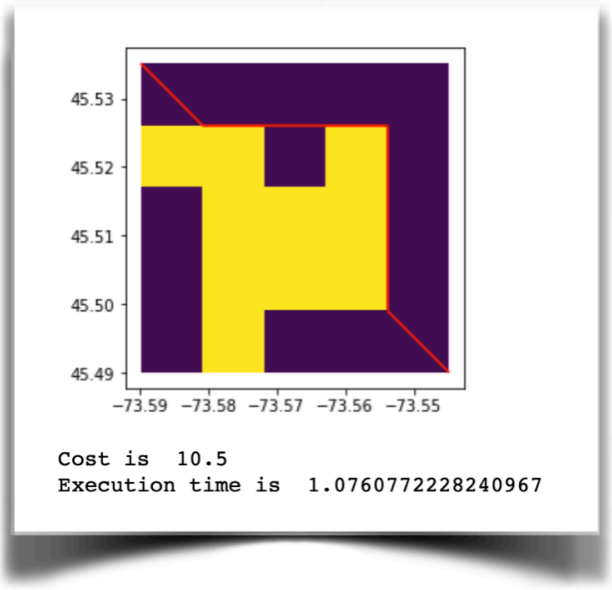
(4)

- **Scenario -5**

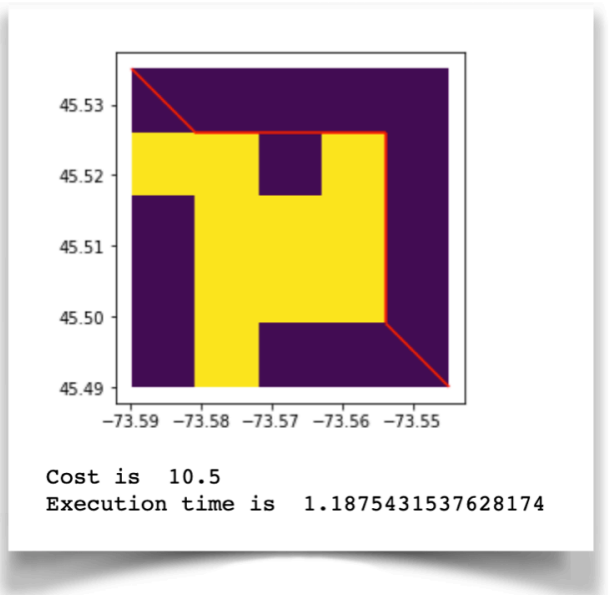
Threshold- 60, Grid size - 0.009



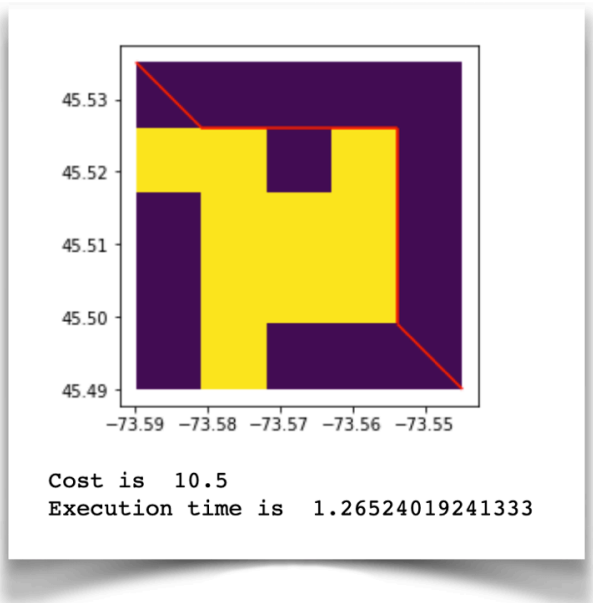
(1)



(2)



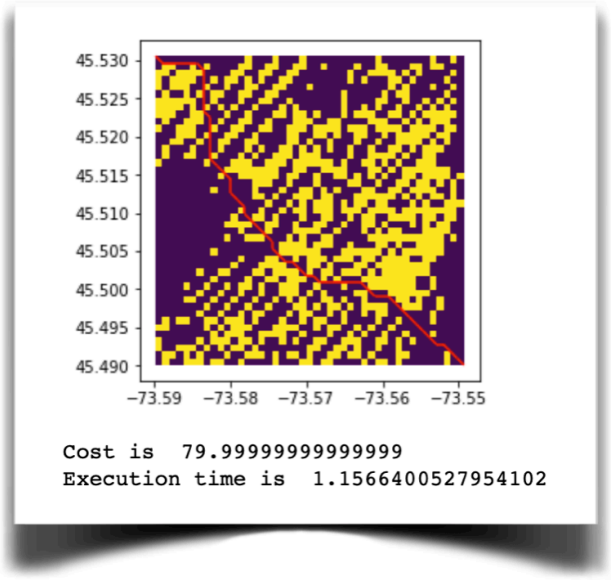
(3)



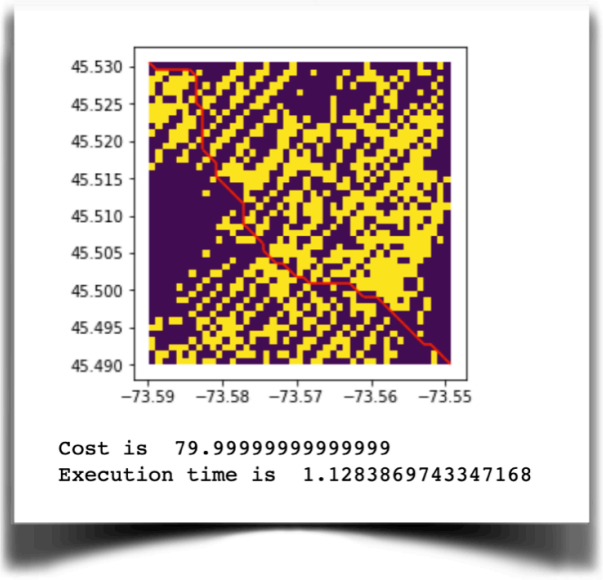
(4)

- **Scenario -6**

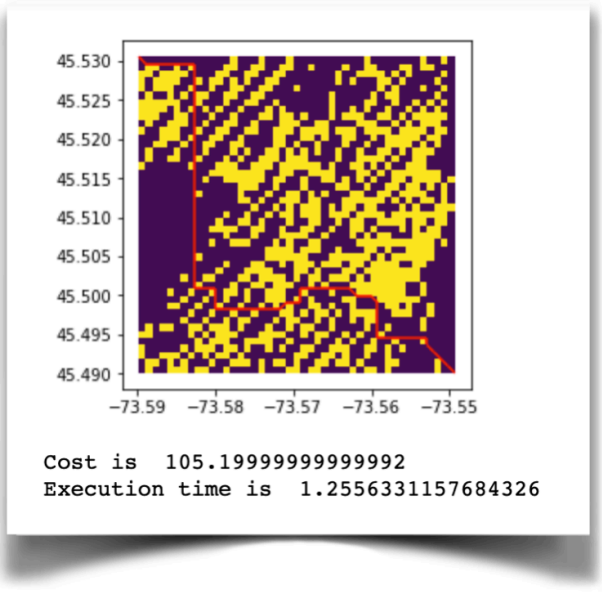
Threshold- 60, Grid size - 0.0009



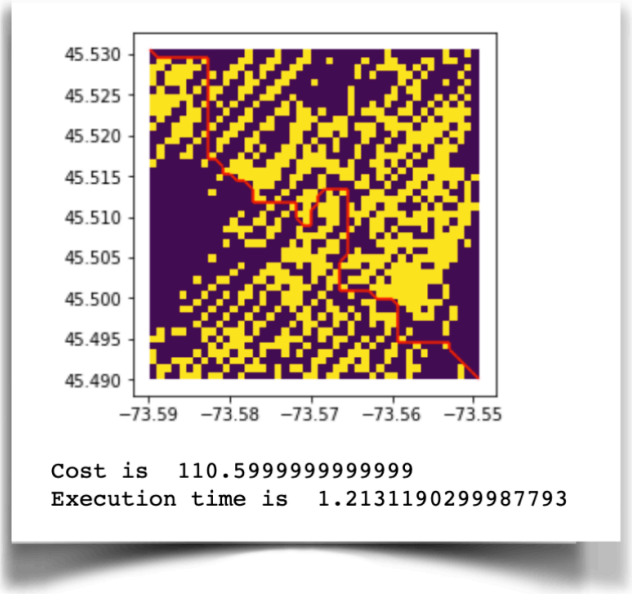
(1)



(2)



(3)



(4)

3.2 INFERENCE AND RESULT ANALYSIS

From the above executed scenarios we got following observations:

Observations for different Algorithms:

- A* gives the optimal path for all the scenarios, where as BFS will give optimal path only in 16% of the above cases.
- Both A* and BFS algorithms are complete algorithms and will always give the path, if the same exists.
- In few scenarios, BFS execution is faster than A*, but the path is more expensive as compared to A* since the difference of path cost is almost double.
- When the block size is more ,that is the blocks are more dense and are less in number, the path given by A* and BFS seems to be the same and the optimal(Scenario 5).
- As the block size decreases, the difference between the costs given by both the algorithms increases since the no. of edges to be considered to find the path increase with the increase in the number of blocks.

Observations for different Heuristic functions:

- Both the heuristic functions works same for A* algorithm, the path is always optimal and same.
- In some scenarios, the execution time for Cost and distance heuristic function is less compared to Distance heuristic function.
- A* gives the same optimal path for both the heuristic functions, this gives a conclusion that A* works in the same fashion if the heuristic is admissible.
- BFS gives different paths for both the heuristic functions when the block size is too small. The path given by the cost and distance heuristic function is cheaper than the path given by the distance heuristic function.

4. DIFFICULTIES & SOLUTIONS

- **Creating blocks with the given points in the shape file.**
  Used geopandas polygon data type to create blocks from the given spatial points in the crime data file.
- **Choosing the proper block size inside the grid to have more accurate results.**
  To ensure and display the accuracy of the data, smaller size less or equal to 0.002* 0.002 is used.
- **Coming up with a proper way to plot points inside the grid blocks.**
  Used sjoin function to come up with a geopanda data frame which consisted of points in a given block.

- **Finding an easier and accurate way to traverse each block of space.**
  After a lot of closed analysis coming up with considering left most point of each block to be used for finding the path and providing unique identity to each block.

- **Choosing a data structure used for storing the cost associated with traversing each block and their colour combinations which could be fed into the heuristic algorithm as input.**

  We have used dictionary data structure in python to store the colour combinations of each block and cost of exploring the block.

- **Designing the heuristic function, which takes less time for evaluation and takes us closer to goal state.**

  We have employed Distance heuristic function considering the distance between the co-ordinates and Cost-Distance heuristic function considering the cost of traversal as well.

- **Selecting the best algorithm which could help us in finding the optimal path from source to destination in minimum time.**

  We have used the most accurate informed Search algorithm for finding path i.e the A* algorithm as well as the fastest algorithm for finding path i.e the Best First Search algorithm.

5. RESPONSIBILITIES & CONTRIBUTIONS

Harsh Deep Kour	Iknoor Singh Arora
• Data Analysis. • Understanding Libraries. • Heuristic Function Designing. • Distribution of points among polygons. • A* Algorithm implementation. • Cost Calculation Implementation. • Execute Test scenarios and Result Analysis. • Report Writing	• Setting up development environment. • Data Analysis. • Understanding Libraries. • Points and Polygon plotting. • Best First Search Algorithm implementation. • Timer Function Implementation. • Heuristic Performance Analysis • Report Writing

Table 1 : Contributions Table

6. REFERENCES

1. http://geopandas.org/
2. https://en.wikipedia.org/wiki/Shapefile
3. https://www.geeksforgeeks.org/best-first-search-informed-search/
4. https://www.brainkart.com/article/Best-First-Search--Concept,-Algorithm,-Implementation,-Advantages,-Disadvantages_8881/
5. https://www.growingwiththeweb.com/2012/06/a-pathfinding-algorithm.html