CMSC 818B: Decision-Making for Robotics (Fall 2019)
Homework 1

Amrish Baskaran

116301046

Problem 1

Minimum Spanning Tree (MST) based 2-approximation algorithm

MST Method-

Preparing the edge information-

1. Get all the vertices
2. Construct a list containing edge information- [u,v,w] where u is vertex 1, v is vertex 2 and w is the weight. Here the weight is the Euclidean distance between the two points. Duplicates are avoided while doing so, that is [1,2,w] is same as [2,1,w].{In the class **graph**}
3. Sort this list using the weight of the edges.

Start collecting edges for MST-

1. Select edges in order from this sorted list of all edges.
2. Check if edge when added to the required edge list would create a loop or not.
3. If no loop, then add the edge to the required edge list. Else discard this edge.
4. Do this till V-1 edges are collected. Where V is the number of vertices. {In the class **graph**}

Creating the tree-

1. Choose an arbitrary starting vertex
2. Transverse all the vertices by depth first search (DFS) and record the sequence of vertices. Both visited and unvisited. Eg: A-B-C-D-C-E-C-B-A. {In the class **DFS**}
3. Use the shortcut strategy to generate a tour. Eg:
   A-B-C-D-(C)-E-(C-B)-A
   A-B-C-D-E-A
4. Create a final list of vertices in order

Heuristic-

1. Run a loop to run through the pairs of vertices in order(edge after edge). {In the class **heuristic**}
2. Check for intersection of edges. A-B-C-D-E-A, AB and DE intersect.
   Reference- https://bryceboe.com/2006/10/23/line-segment-intersection-algorithm/
   very simple and logical way to check for intersection with very few lines of code
3. If there is an intersection flip the whole tour between these vertices.
   A-B-C-D-E-A
   A-D-C-B-E-A.
4. Run this in a while loop until there are no more intersections.

Criteria for loop check-

Parent and rank method- when an edge is added to the tree its parent is found. It is then compared with he previous parent. If they are the same then the loop is complete and the edge is neglected. If not, it will added to the tree and the information for parent and rank for both are updated.

Reference-

1. https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/
2. https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
3. https://en.wikipedia.org/wiki/Disjoint-set_data_structure

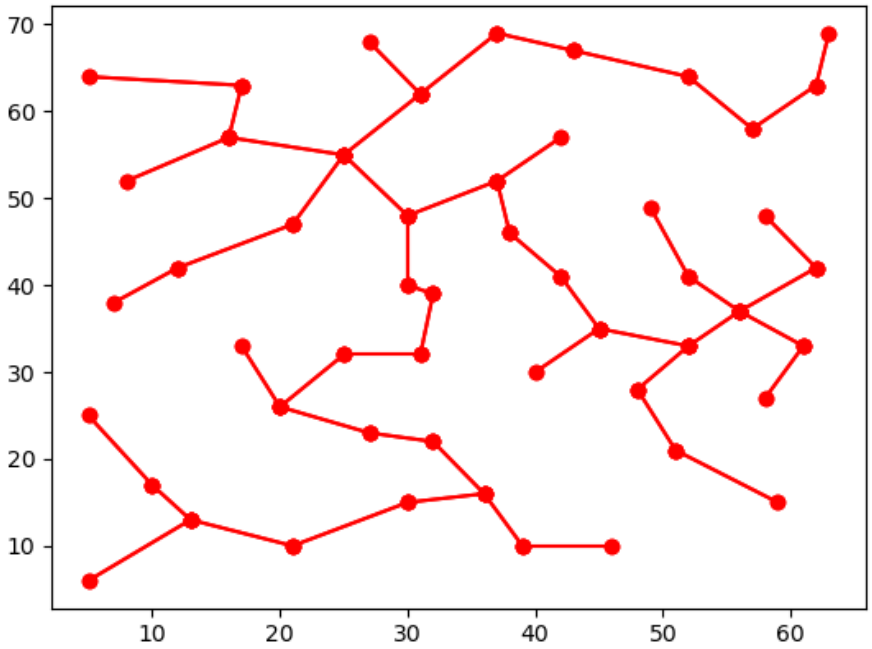
Running on the files provided-

File Name	Length of Final Tour	Optimal Tour Length	Length of MST	Length **Without** Heuristic	Time Taken to Solve
eil51	485	426	376	604	0.224
eil76	603	538	472	690	0.588
eil101	733	629	562	844	1.158

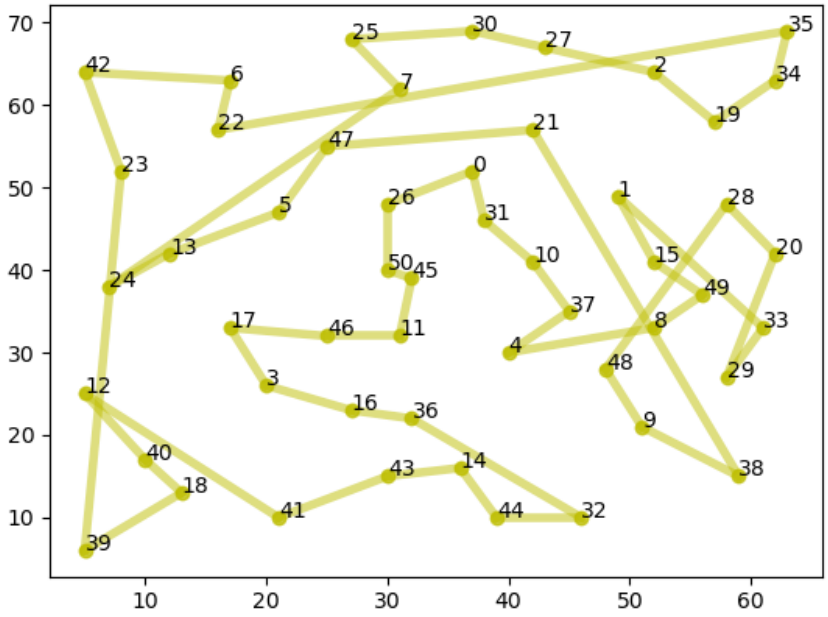
The tours are then plotted-

For eil51-

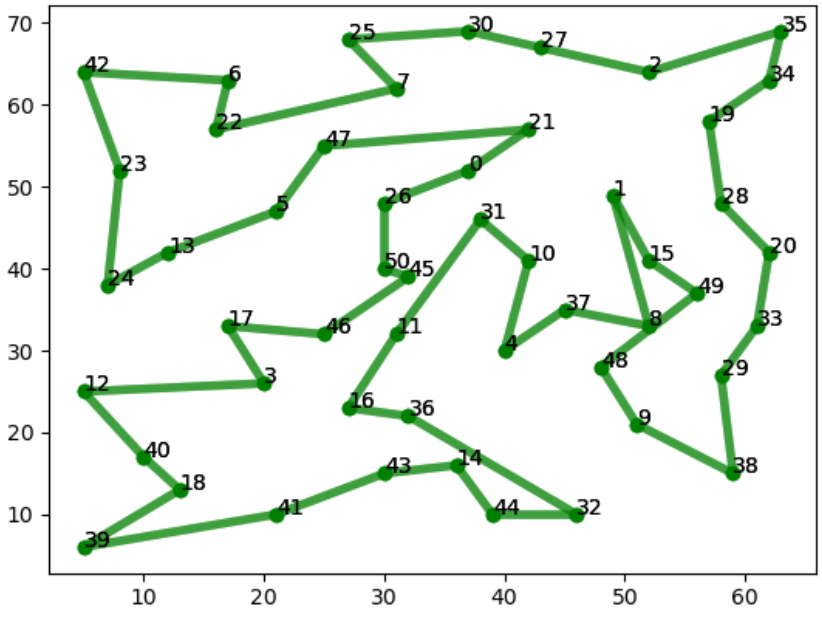
MST-



Tree after shortcut is applied-



Final tour after using heuristic-



Running with 100 Random points(Points are in the bounding box[(0,0),(100,100)]**)-**

Length of Final Tour	Length of MST	Length **Without** Heuristic	Time Taken to Solve
1007	705	1099	0.251
959	707	1078	0.315
989	709	1094	0.419
976	721	1044	0.22
978	676	1065	0.268
855	629	932	0.258
940	661	988	0.488
879	665	1013	0.32
924	669	1038	0.193
930	679	1044	0.253

Running with 200 Random points(Points are in the bounding box[(0,0),(200,200)]**)-**

Length of Final Tour	Length of MST	Length **Without** Heuristic	Time Taken to Solve
2754	1995	3140	1.616
2650	1921	2968	1.88
2556	1917	2835	1.682
2605	1930	2781	1.518
2646	1916	2994	2.054
2799	1917	2996	1.964
2857	2011	3172	1.867
2685	1940	2949	1.115
2680	1900	2882	0.94
2679	1935	2904	1.369

Running with 200 Random points (Points are in the bounding box[(0,0),(300,300)]**)-**

Length of Final Tour	Length of MST	Length **Without** Heuristic	Time Taken to Solve
4724	3495	5286	4.899
4629	3497	5096	3.967
4871	3572	5300	5.705
4972	3530	5505	5.93
4745	3472	5306	5.155
4895	3473	5335	5.725
4620	3466	5158	3.502
4656	3491	5222	5.79
4674	3437	5463	6.528
4748	3513	5487	6.238

Screencast of Running the program can be found in this link-

https://drive.google.com/open?id=1lyJdUFHYMI3rnixmSszDzqKGpJwI1dxX

Problem 2: k- robot metric TSP

Conditions- k robots, same starting vertex (s), k > 1, Return to starting vertex, N- vertices

Simple Clustering approach-

N- vertices are distributed in 2d space. Now these vertices can be clustered into k groups. One way to cluster them is by using k- means clustering [O(n^{2k+1})]. All the robots start from the same starting point. Hence after clustering each robot can be assigned a cluster.

Now any algorithm can be used for each algorithm to find the best Tour. This does not guarantee the length of maximum tour be minimized.

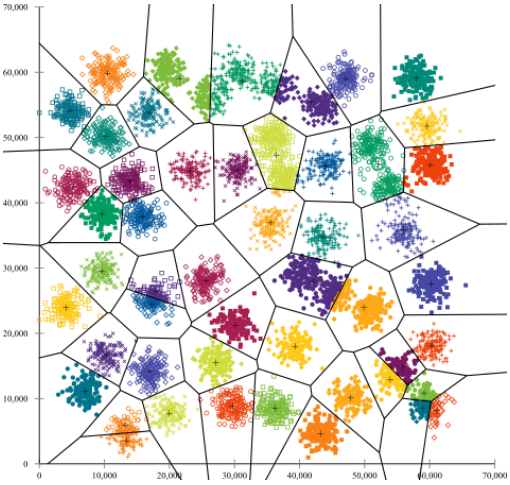
For the Maximum tour to be minimized each of the tour length of k – robots must be almost equal. Which may not be the case. Some clusters can have more points than the other or may more span between them.

Pros-

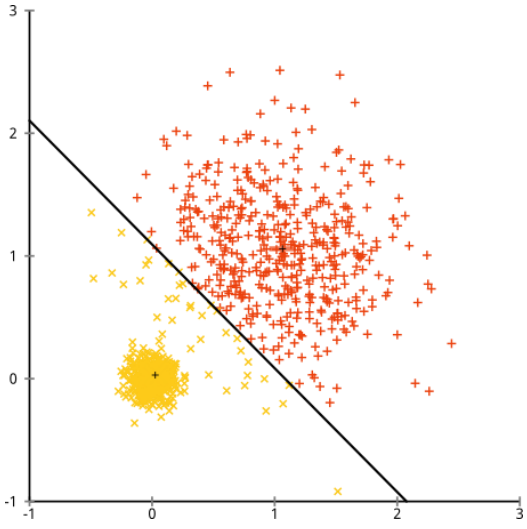
1. If the distribution of points is in clusters, then k- means clustering will converge faster. And then MST 2- approximation can be used to get length ≤ L(Optimal) solution. This may be a feasible solution even if it is not optimal.
2. Will give good results for uniform distributions as well as they will be clustered with equal number of points in each distribution.

Cons-

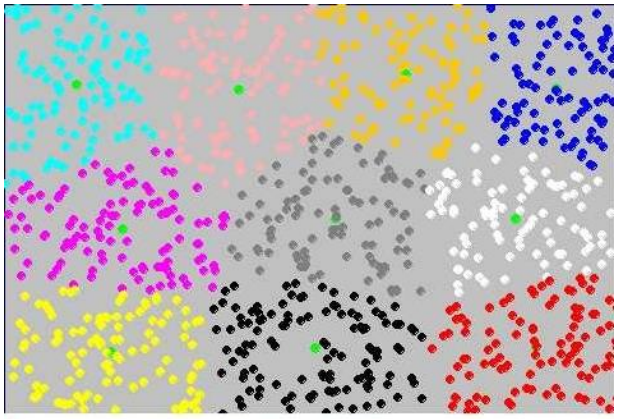
1. One cluster may have more vertices than the other or may be larger than the other. Resulting in a robot finishing its tour earlier than the others.
2. K- means is based on Euclidean distance, hence may not have a good MST and may not result in similar tour length while having equal number of points.



(a)



(b)



(c)

(a) Multiple clusters grouped using k- means

(b) Cons(1) – variation in cluster size and members resulting in varied tour length

(c) Almost uniform distribution clustered into groups having almost same size and number of members.

Reference-

1. https://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means
2. https://www.researchgate.net/figure/Uniform-distribution-output-K-Means_fig4_47554407

Clustering Using MST-

K- means uses Euclidean distance for clustering which as mentioned above may not give good results. Instead MST can be used as a criteria to update the clusters during the unsupervised learning. Including the condition of having almost equal MSTs across the k- robots will result in almost equal tour lengths.

Greedy over k- robots-

1. Construct a list containing edge information- [u,v,w] where u is vertex 1, v is vertex 2 and w is the weight. Here the weight is the Euclidean distance between the two points. Duplicates are avoided while doing so, that is [1,2,w] is same as [2,1,w].
2. Sort this list using the weight of the edges.
3. Choose top k edges that do not create a loop for each of the robots and add them to a list of lists containing the vertices.
4. Calculate and store the length of these edges in a list(forward list).
5. Use another list(total length list) similar to the forward list that will have the total length for each robot- forward and return which will be calculated every time a new point is added to the forward list.
6. The index of the robot with the least value in the total length list is chosen.
7. From the extreme vertex find the minimum edge connected to it that does not form a loop and add it to the robot vertex list.
8. Recalculate the forward and total length lists.
9. Repeat the steps 6-7 until you have iterated through all the vertices.

Vertices that belong to each robot are found. MST on these points for each robot may not be required as the steps above result in minimum edge lengths.

A heuristic to remove the intersections may further reduce the tour lengths. This heuristic can further be applied after addition every q number of points resulting in uniform tour length among robots.

[Incrementally adding vertices to a robot's tour if its total tour length is the least among the others. The added vertex is chosen as the shortest from the extreme vertex of that robot.]

Pros-

1. Faster solution as we are iterating through the points only once.
2. Reduced tour length of maximum tour as all the tours are almost equal.

Cons-

1. During the addition of the last few vertices the selected edges may be large resulting in a higher tour length.
2. In clustered distributions lot of intersection may be produced which when removed may result varied tour length among the robots.