1-Steiner Routing using Kahng and Robins Algorithm

RUI LI
MADHU SUDHAN LAKSHMIPATHY

April 2019

Goals

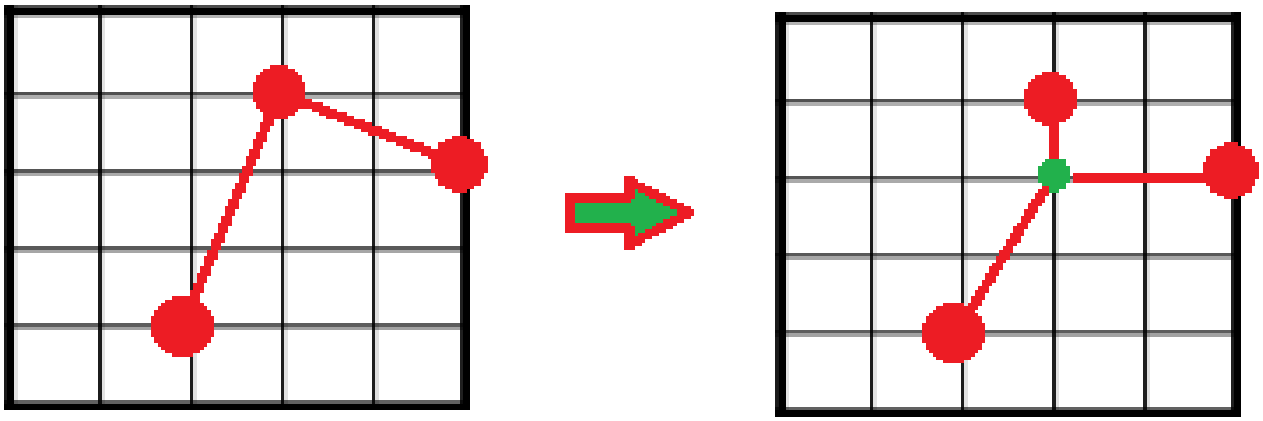
- Perform KR Routing with MST construction using PRIM's algorithm
- Implement Random Variant of KR Algorithm
- Speed up the code
- Verify the results using Matlab
- Create a video animation showing the wirelength improvement for points_30_50.pts testbench
- Extension: Add the feature to use Kruskal's algorithm in the code, instead of PRIM

PRIM’s Algorith for MST

- Get P be the set of points that are not connected initially.
- Move point ‘i’ from P to the graph set G
- Find the nearest point ‘j’ to ‘i’, draw and edge between ‘i’ and ‘j’.
- Next, add another point which is closest to any of the points in the tree by drawing an edge.
- Repeat the step till P becomes a Null Set.
- PRIM is a greedy algorithm, results in a minimum spanning tree

Steiner Point

- Points that are added to MST to reduce the wirelength



Without Steiner Point
Wirelength = 7

After 1-steiner point
Wirelenth = 6

Kahng and Robins Algorithm

- Hanan Grid:

  A Grid of points obtained by intersection of vertical and horizontal lines drawn on all nodes in the MST.

  Iterate through the points in the Hanan grid to find a Steiner point which gives best gain in the wirelength.

  Repeat the process with subsequent MSTs and keep adding Steiner points till there is no improvement in the Wirelenth

Random Variant

- Instead of choosing the best gain Steiner point in each iteration, just add any random steiner point.
- Final solution may have some redundant Steiner points with degree less than 3.
- The redundant points can be removed.
- Time spent on each step is less, but more Steiner points added.

Speed Up

- Initial Time Complexity of whole code: O (N^5)

- After using adjacency list, the time complexity of Prim function became: O(N log N)
- After the First iteration, the Hanan grid points that gave no gain were removed.

- This tremendously boosted the speed from $O(N^5)$ to $O(N^2 \log N)$

Implementation Chellenges

- Verification of output
  (MATLAB GUI was used)

- Choosing the correct data structures

- Speeding up the code

Results: Wirelength Improvement

Testbench	Wirelength					
	PRIM			KRUSKAL		
	Initial MST	Final (Basic)	Final (Random Variant)	Initial MST	Final (Basic)	Final (Random Variant)
Points_10_5	21	18	18	21	18	18
Points_10_10	24	20	20	24	20	20
Points_10_20	37	34	34	37	34	34
Points_30_50	183	163	163	183	163	163
Points_100_100	242	220	220	242	220	220

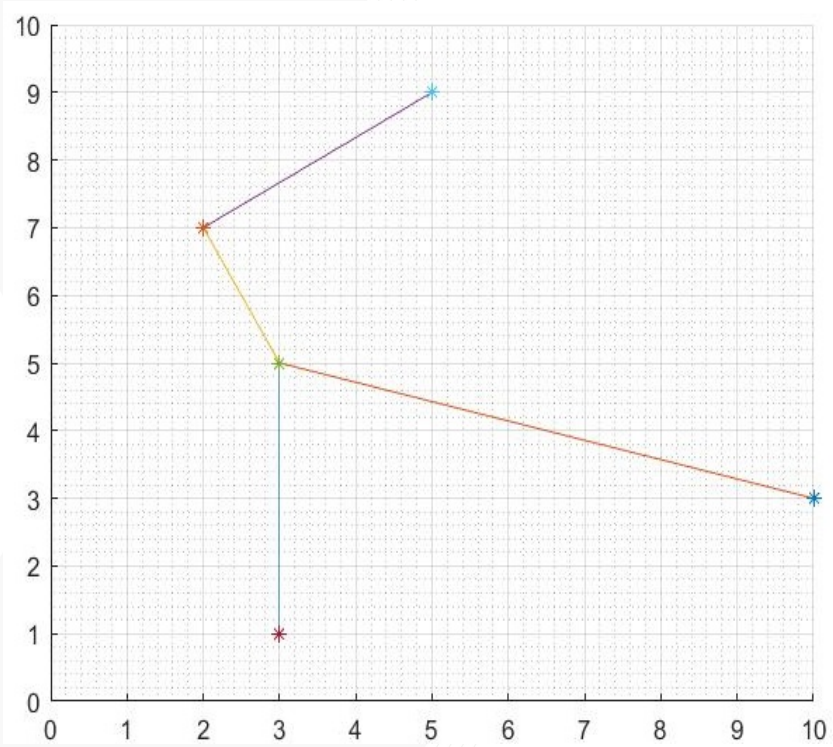
Results: Percentage Improvement in WL

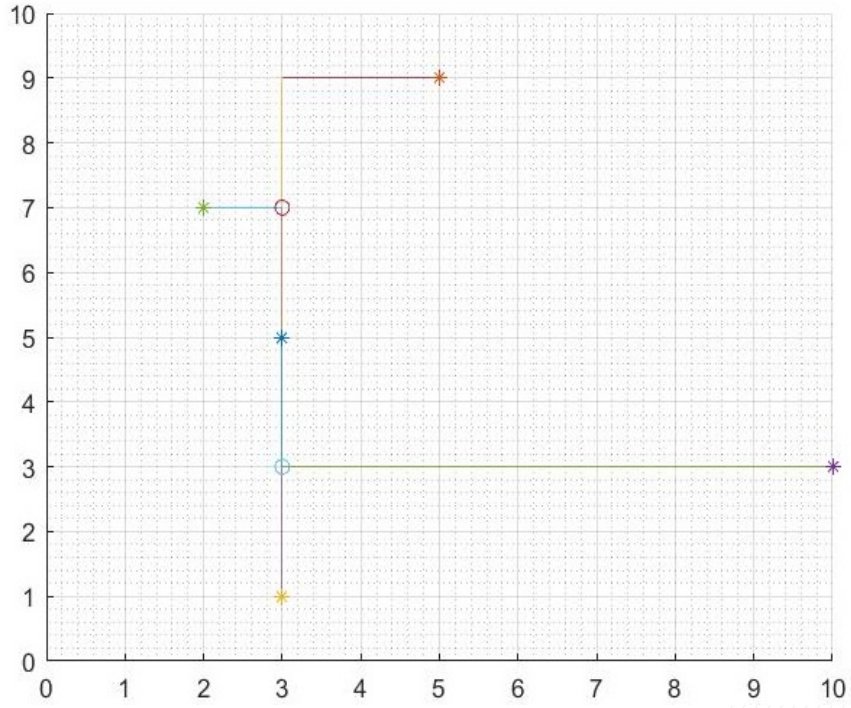
Testbench	% reduction in Wirelength from Initial MST to Final RMST
Points_10_5	14.29
Points_10_10	16.66
Points_10_20	8.1
Points_30_50	10.92
Points_30_100	9.1

Execution Time

Testbench	Execution time (ms)			
	PRIM		KRUSKAL	
	Basic	Random Variant	Final (Basic)	Final (Random Variant)
Points_10_5	0	0	0	0
Points_10_10	0	0	0.010	0
Points_10_20	0.030	0.030	0.040	0.040
Points_30_50	1.600	1.63	2.300	1.810
Points_100_100	14.720	13.880	19.030	18.070

10_5 (Prim, Basic)





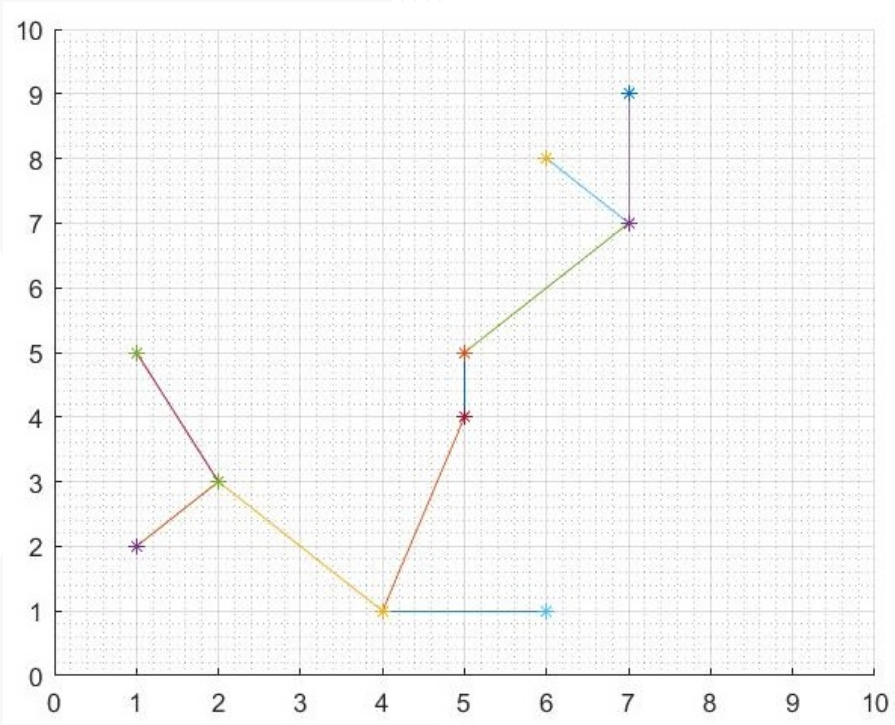
Before = 21

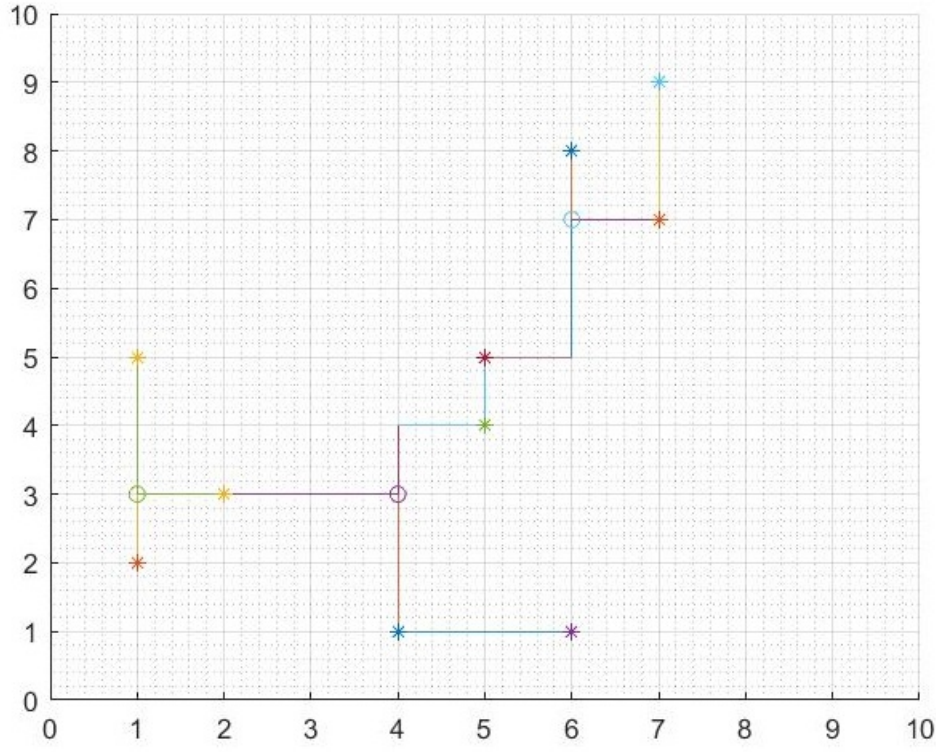
After = 18

Steiner Points = 2

Georgia Tech
CREATING THE NEXT

10_10 (Prim, Basic)





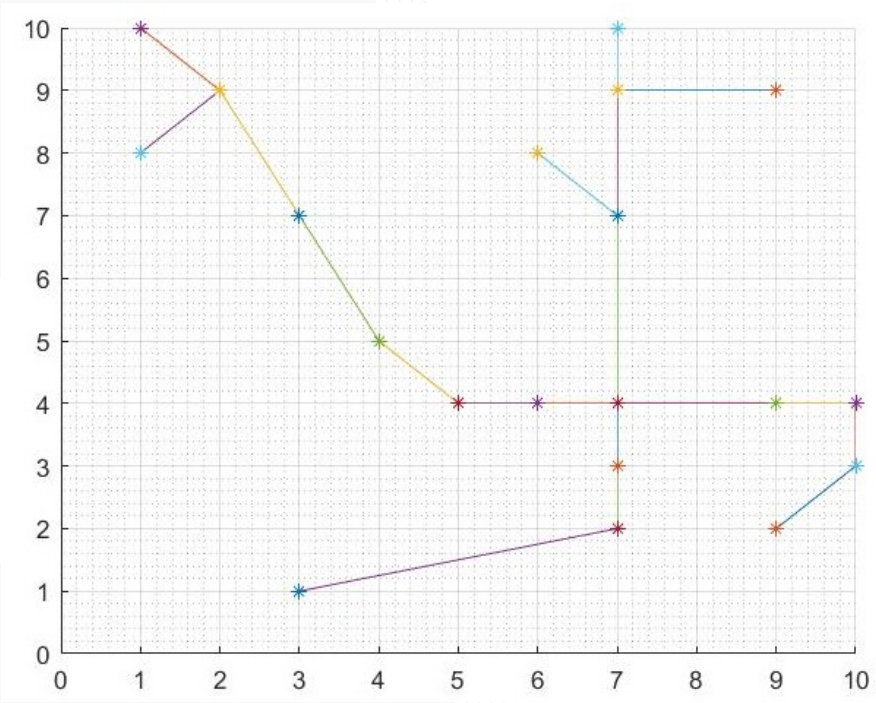
Before = 24

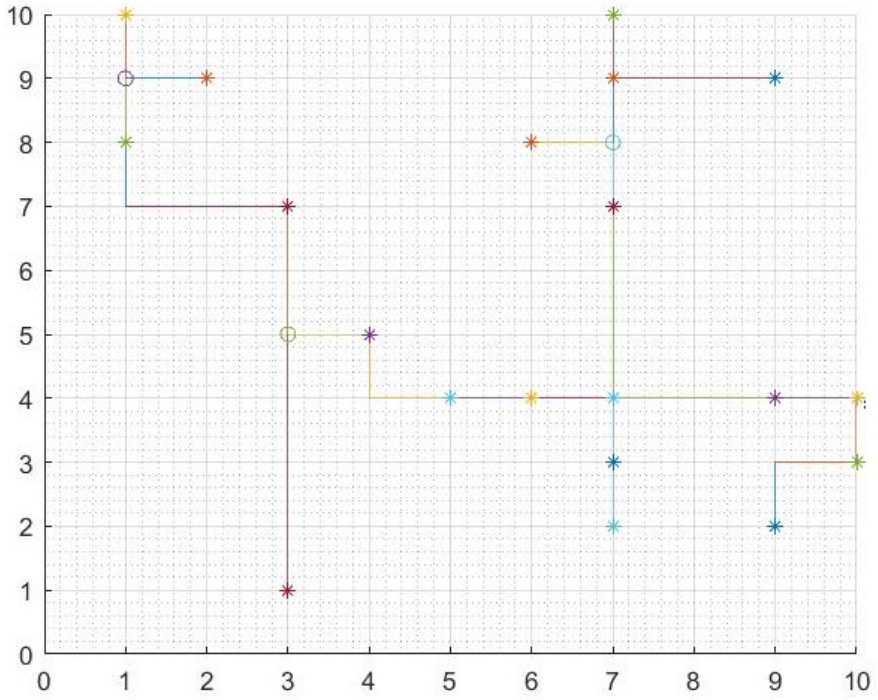
After = 20

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

10_20 (Prim, Basic)





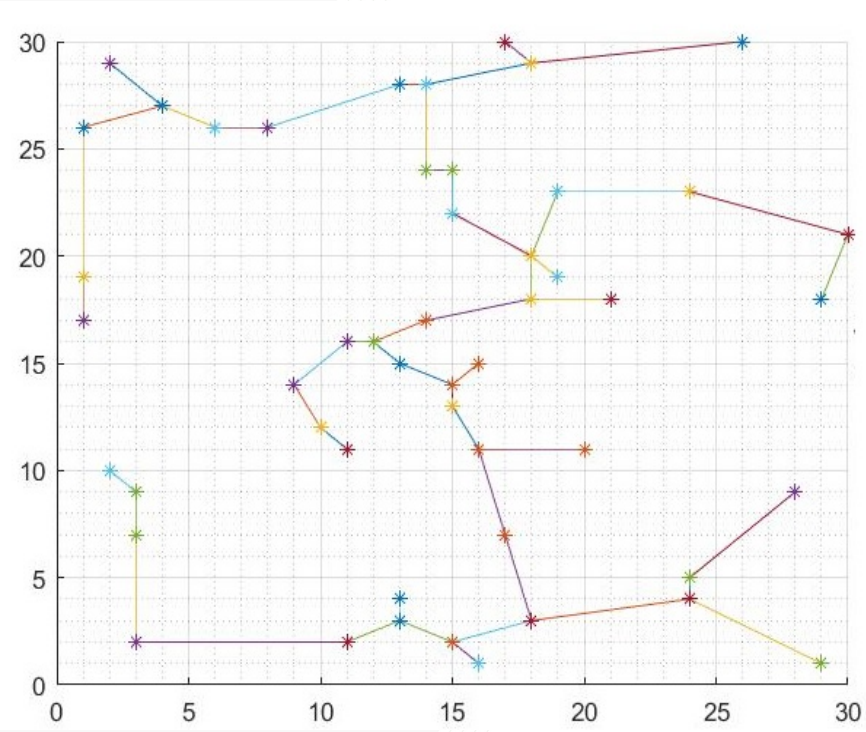
Before = 37

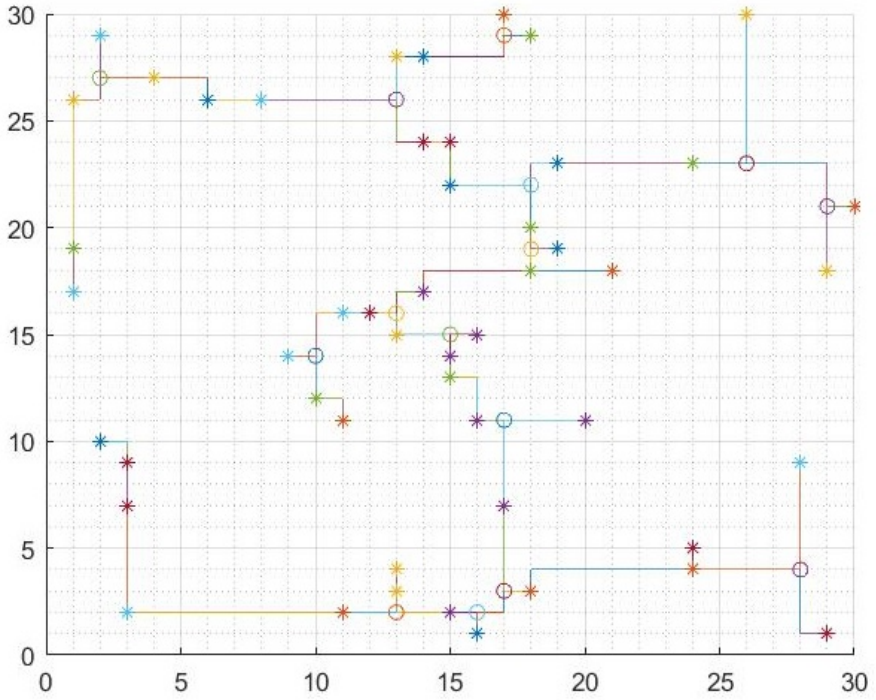
After = 34

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

30_50 (Prim, Basic)





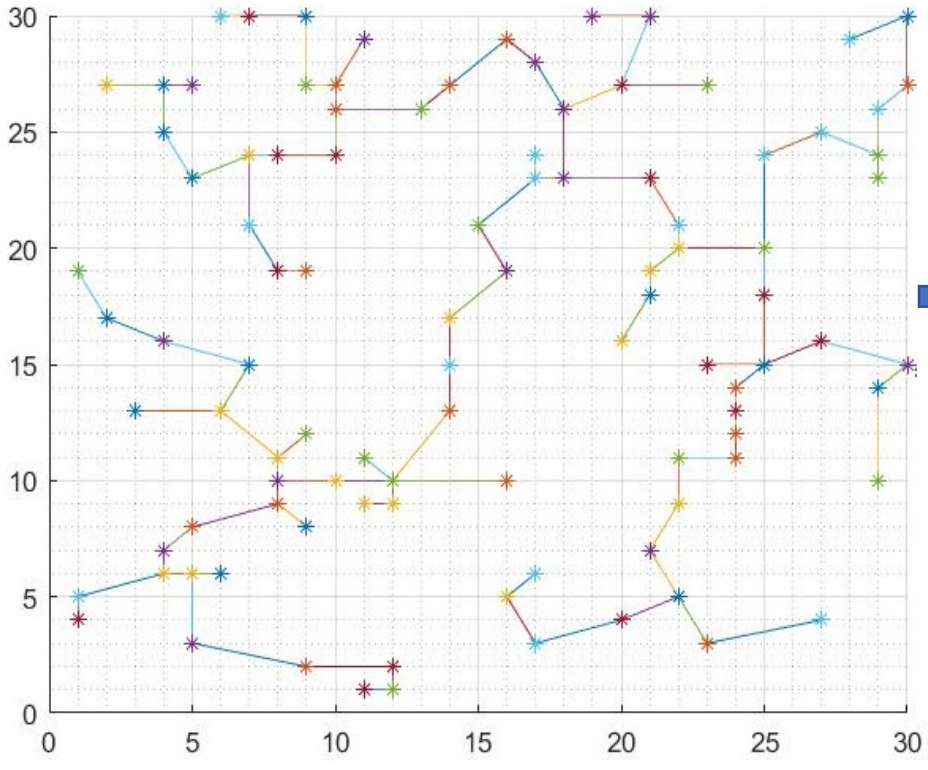
Before = 183

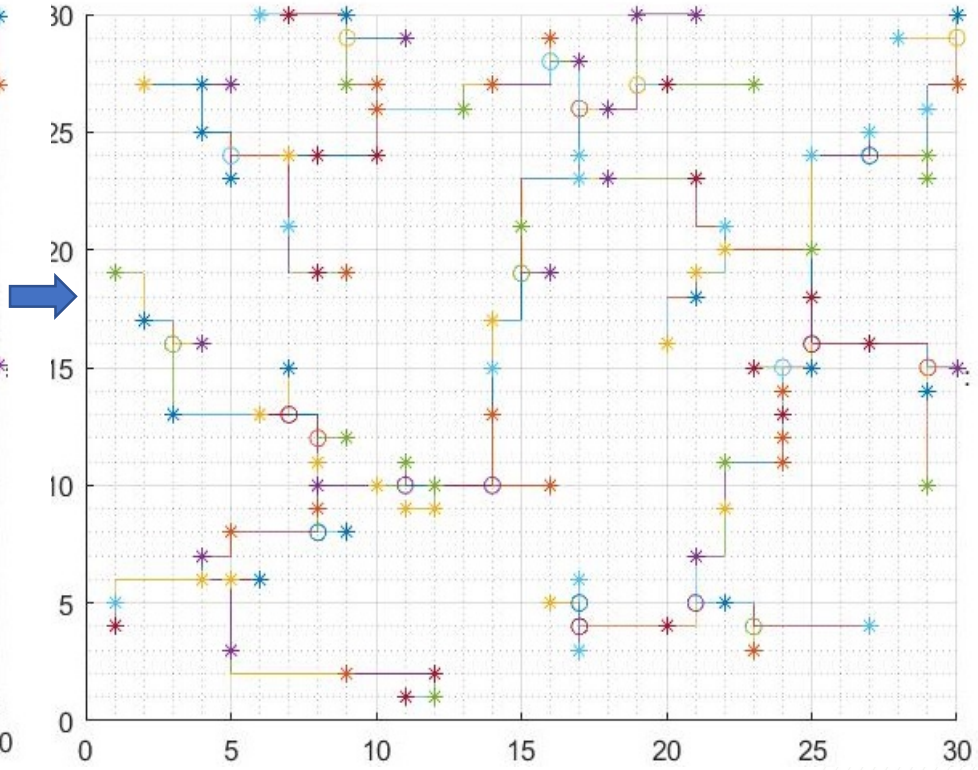
After = 163

Steiner Points = 15

Georgia Tech
CREATING THE NEXT

30_100 (Prim, Basic)





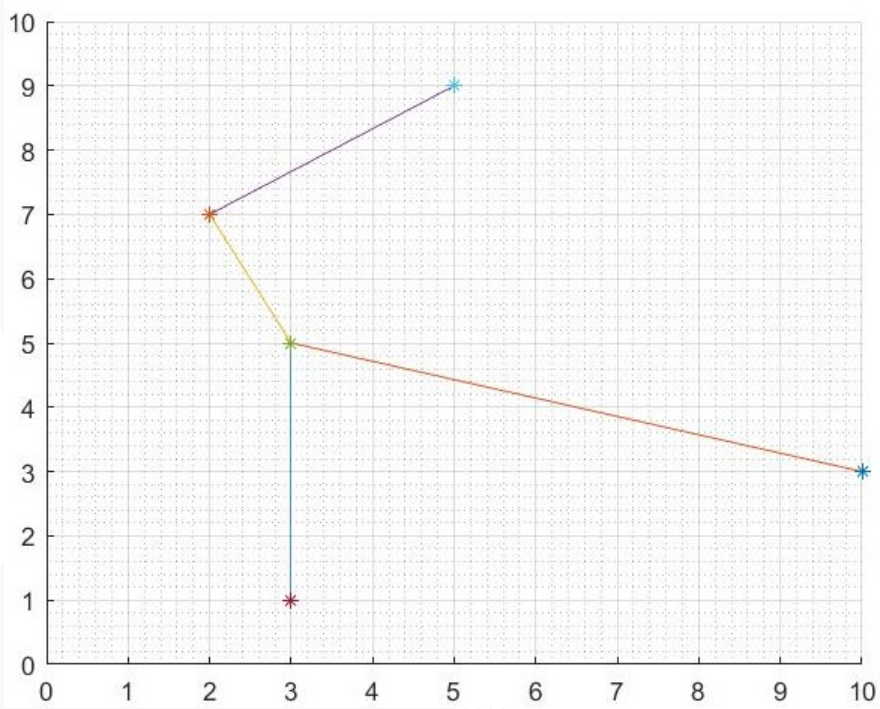
Before = 242

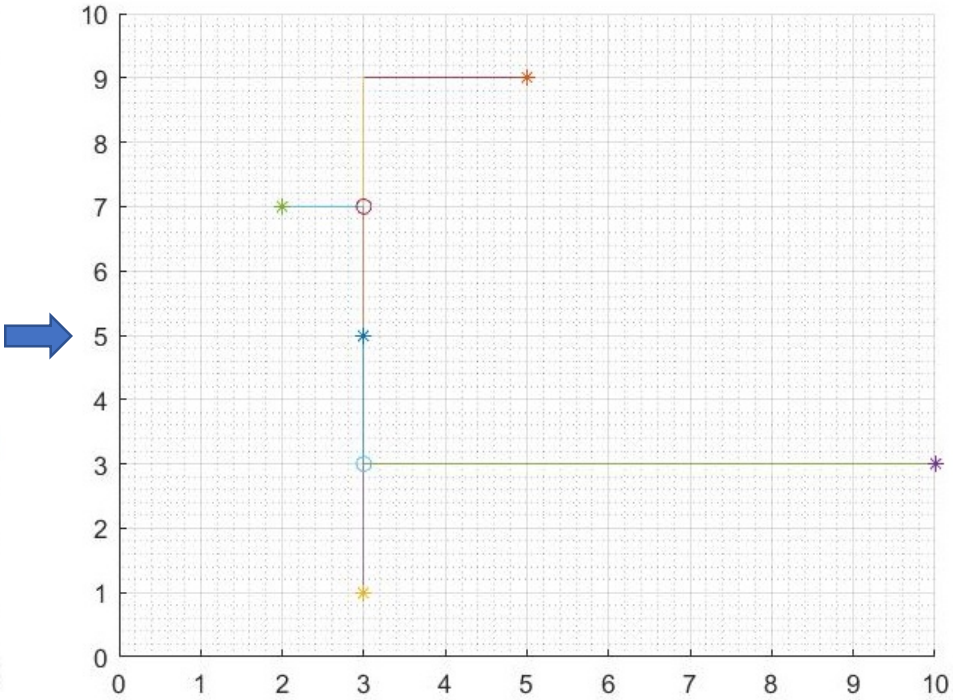
After = 220

Steiner Points = 21

Georgia Tech
CREATING THE NEXT

10_5 (Kruskal, Basic)





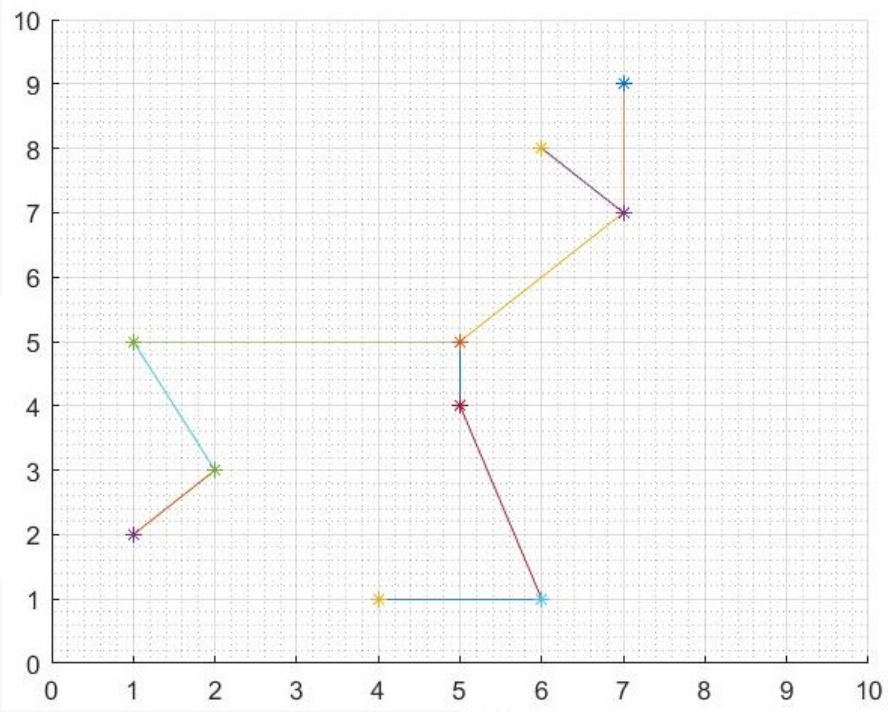
Before = 21

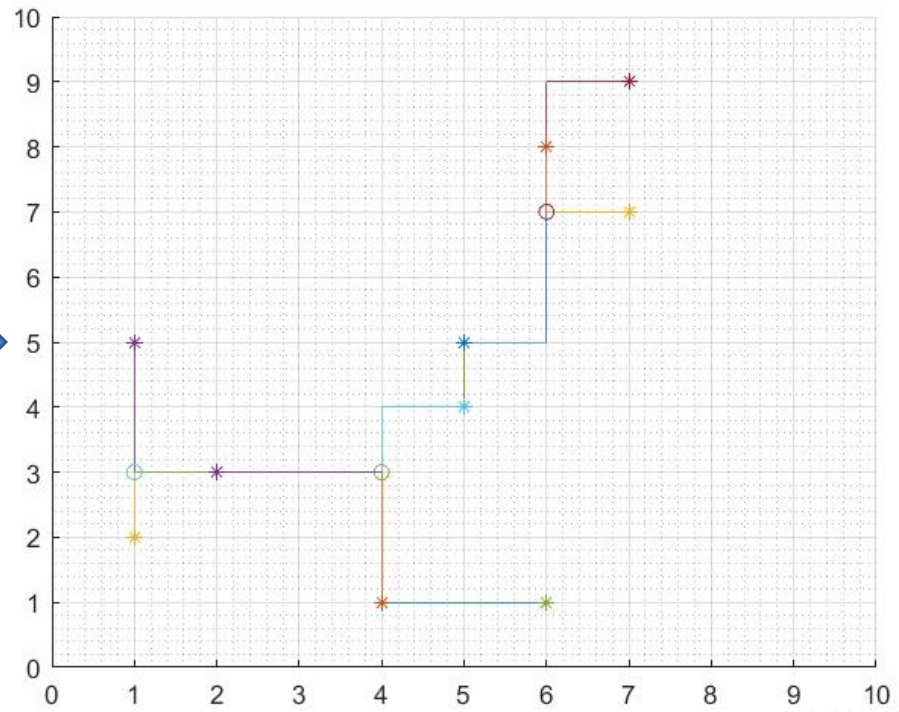
After = 18

Steiner Points = 2

Georgia Tech
CREATING THE NEXT

10_10 (Kruskal, Basic)





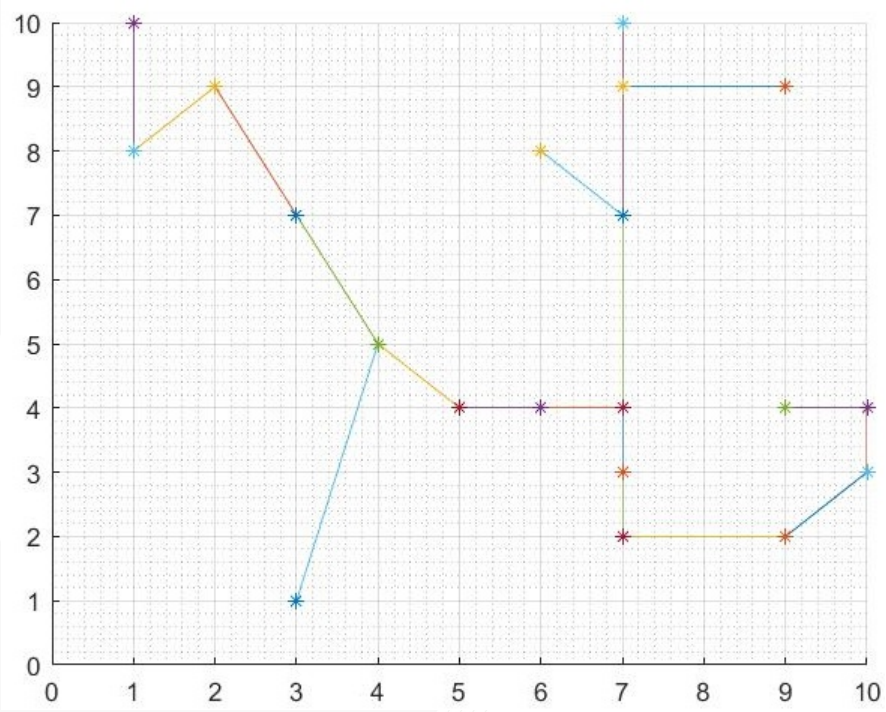
Before = 24

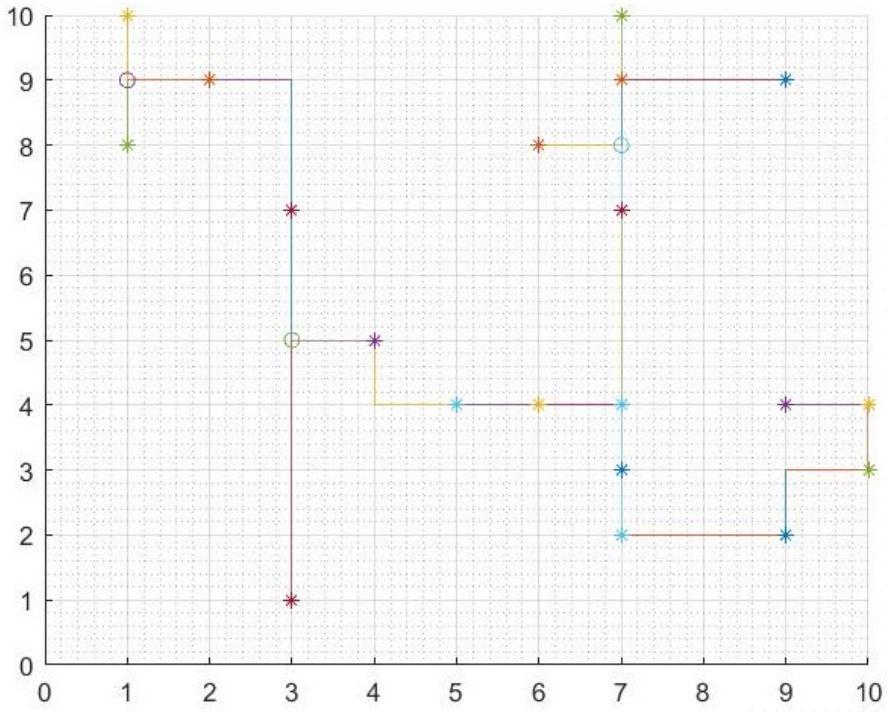
After = 20

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

10_20 (Kruskal, Basic)





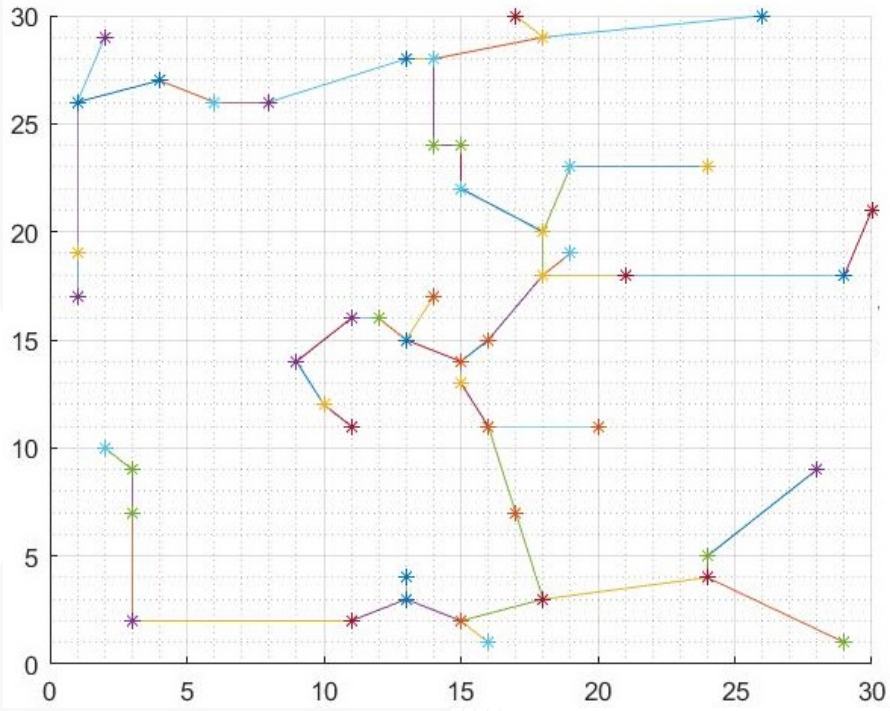
Before = 37

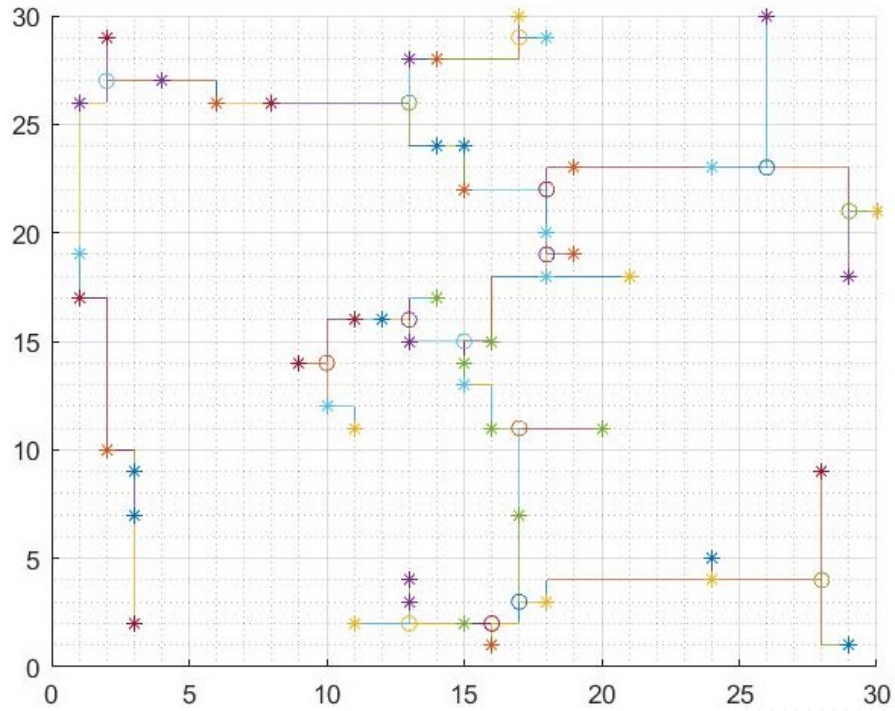
After = 34

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

30_50 (Kruskal, Basic)





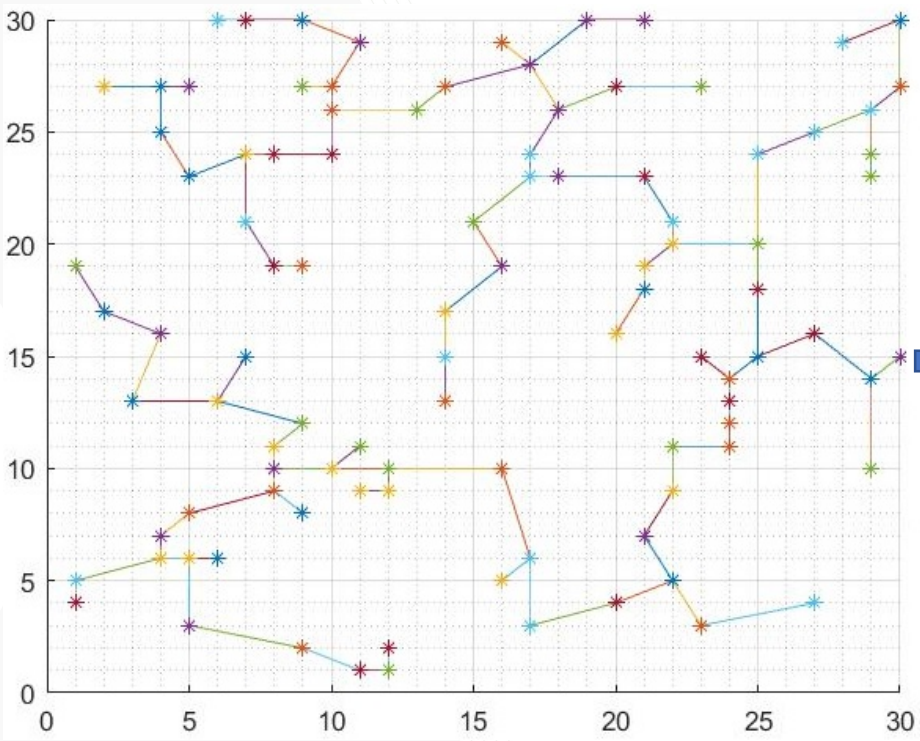
Before = 183

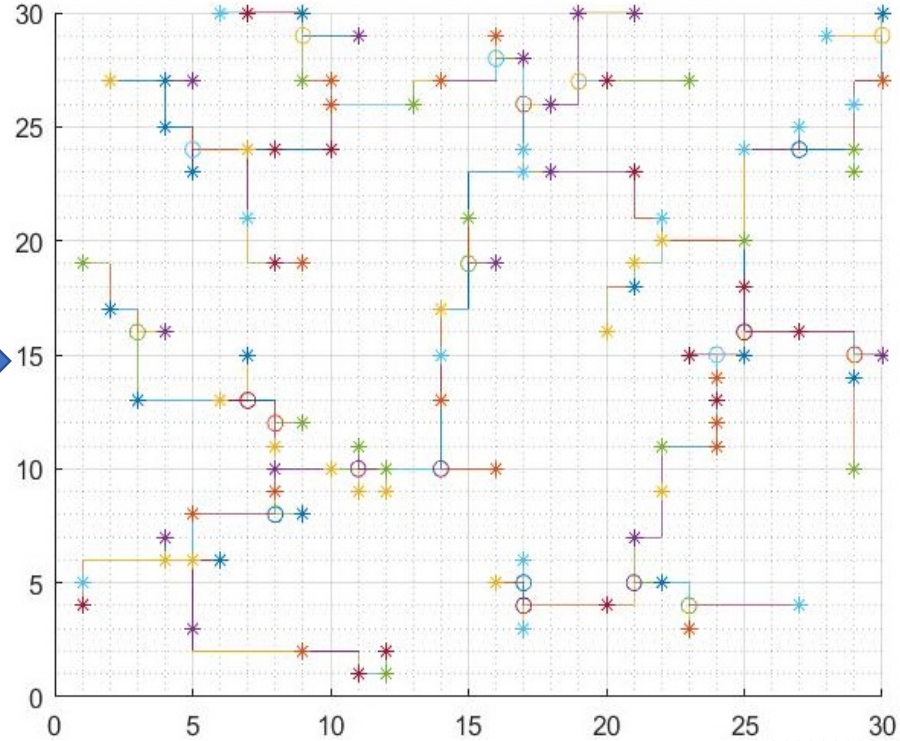
After = 163

Steiner Points = 15

Georgia Tech
CREATING THE NEXT

30_100 (Kruskal, Basic)





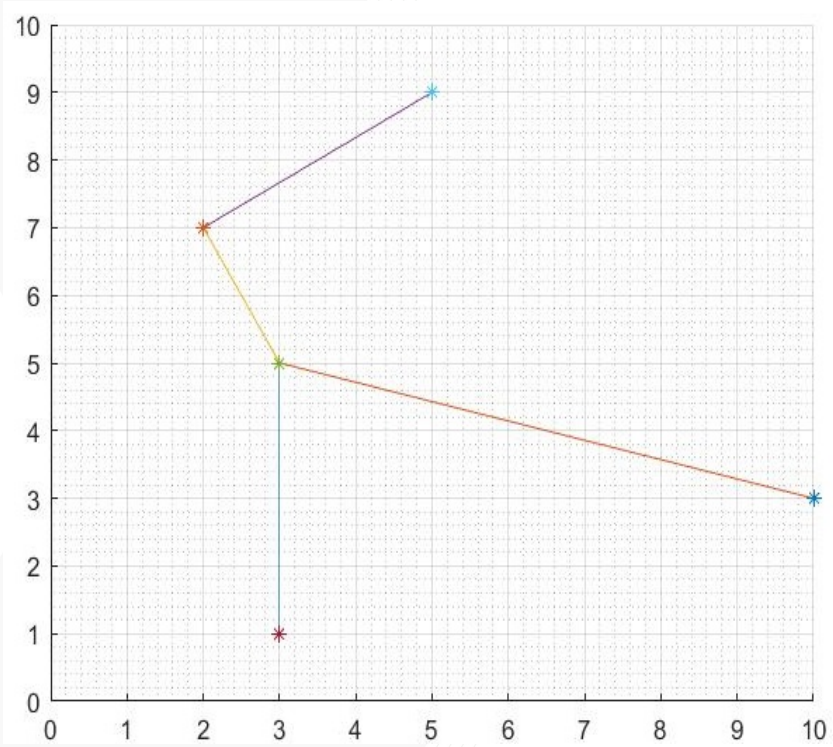
Before = 242

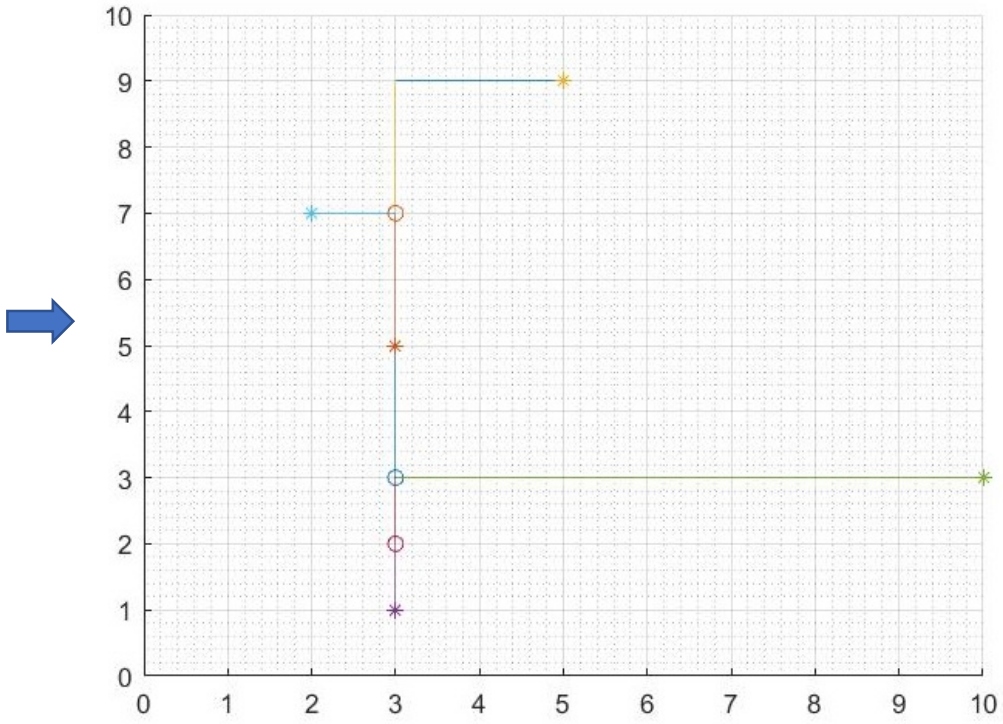
After = 220

Steiner Points = 21

Georgia Tech
CREATING THE NEXT

10_5 (Prim, Random)





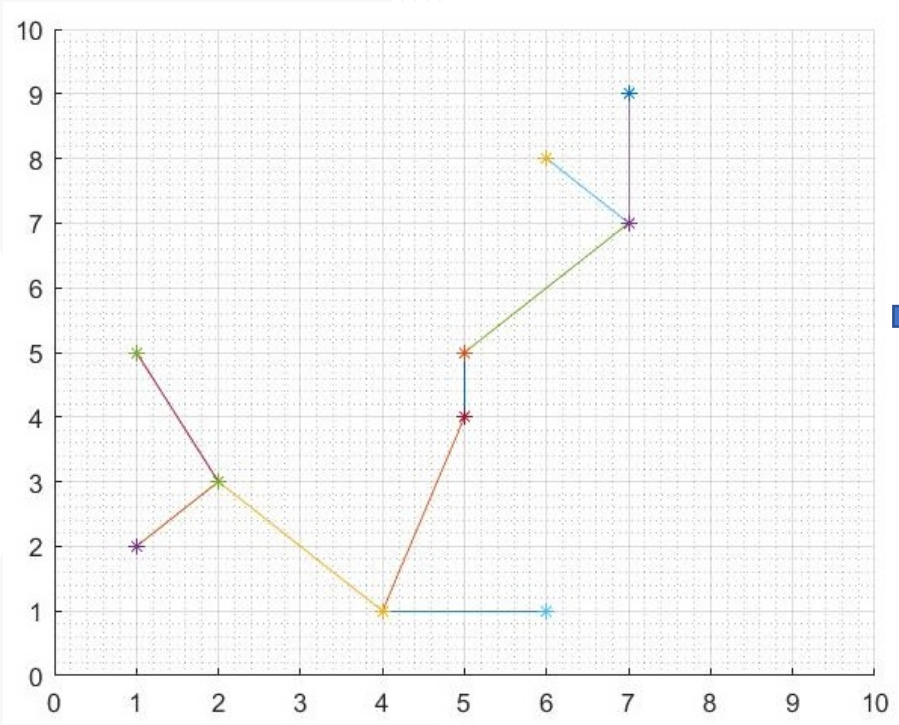
Before = 21

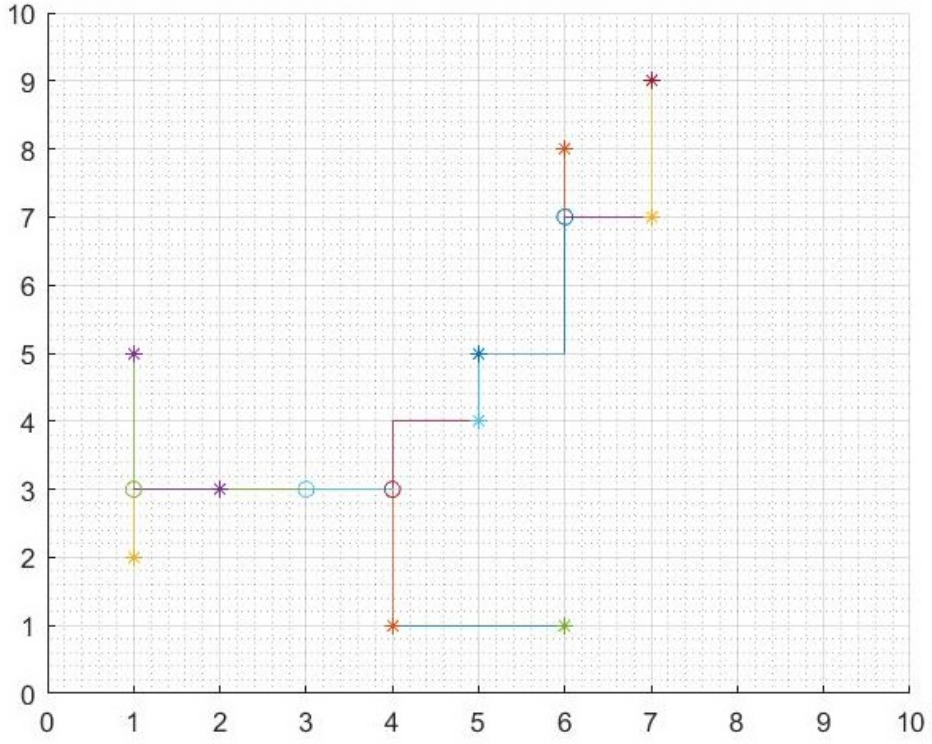
After = 18

Total Steiner Points = 3

Georgia Tech
CREATING THE NEXT

10_10 (Prim, Random)





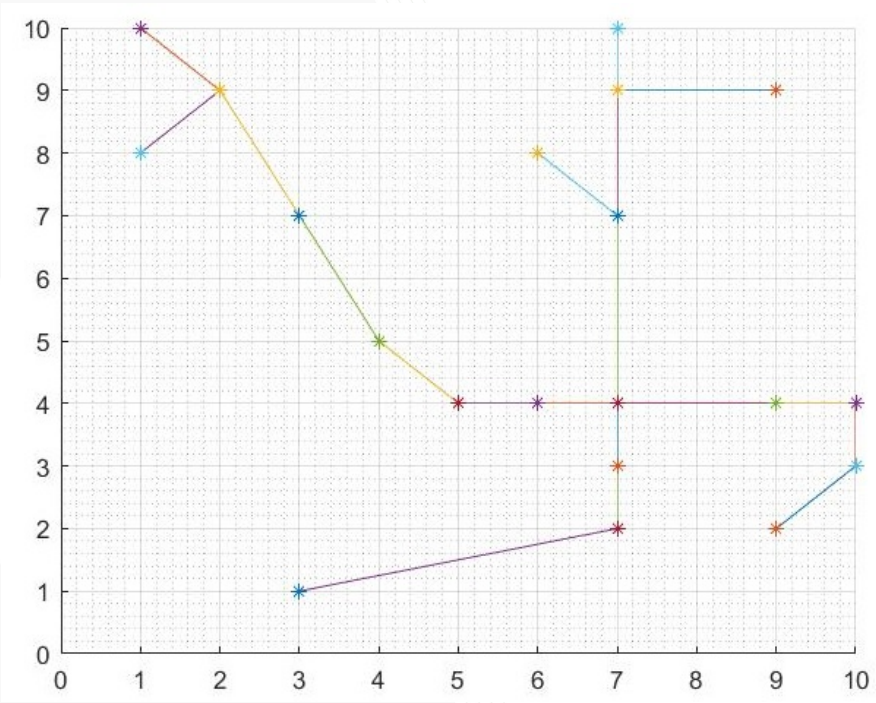
Before = 24

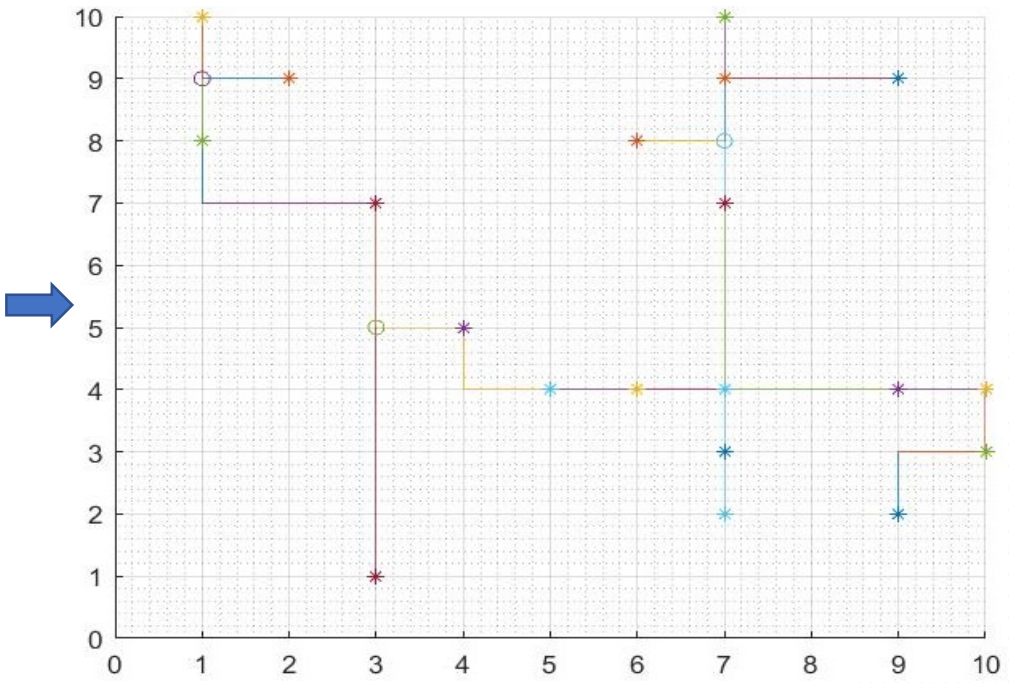
After = 20

Steiner Points = 4

Georgia Tech
CREATING THE NEXT

10_20 (Prim, Random)





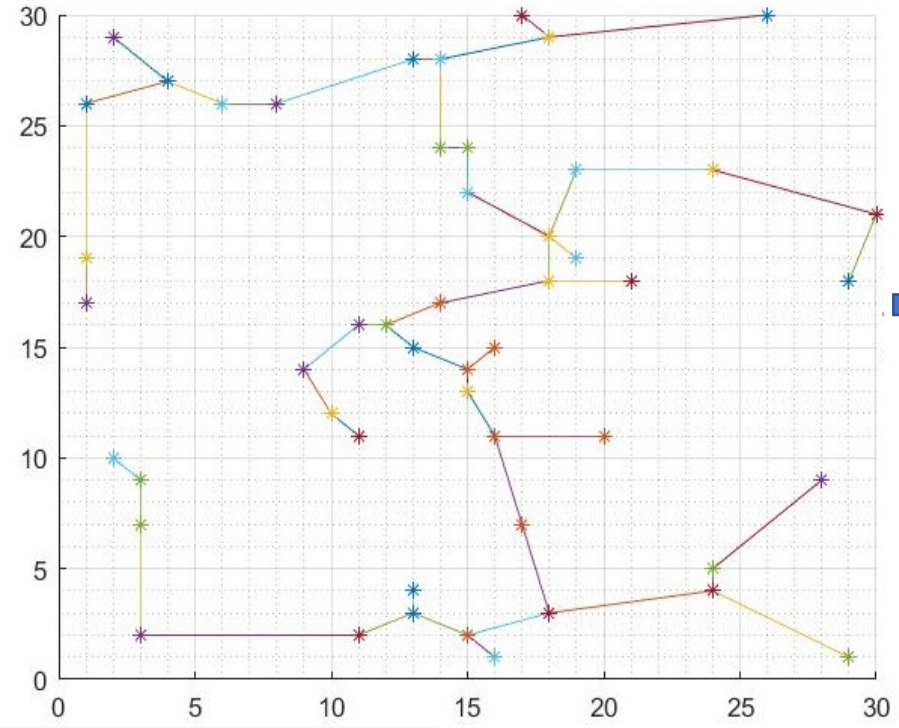
Before = 37

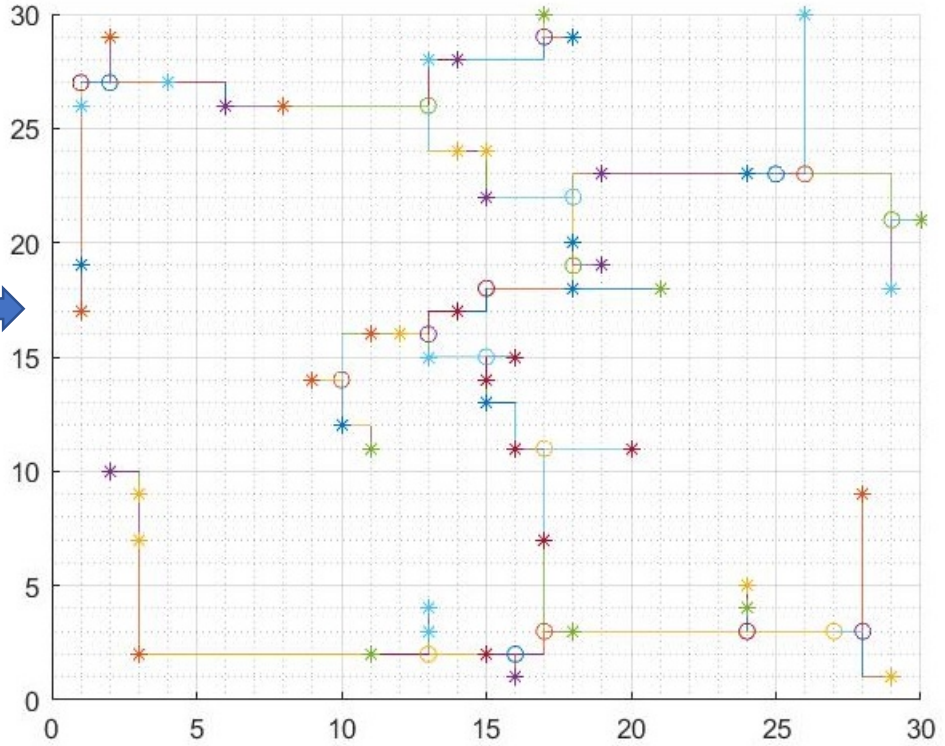
After = 34

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

30_50 (Prim, Random)





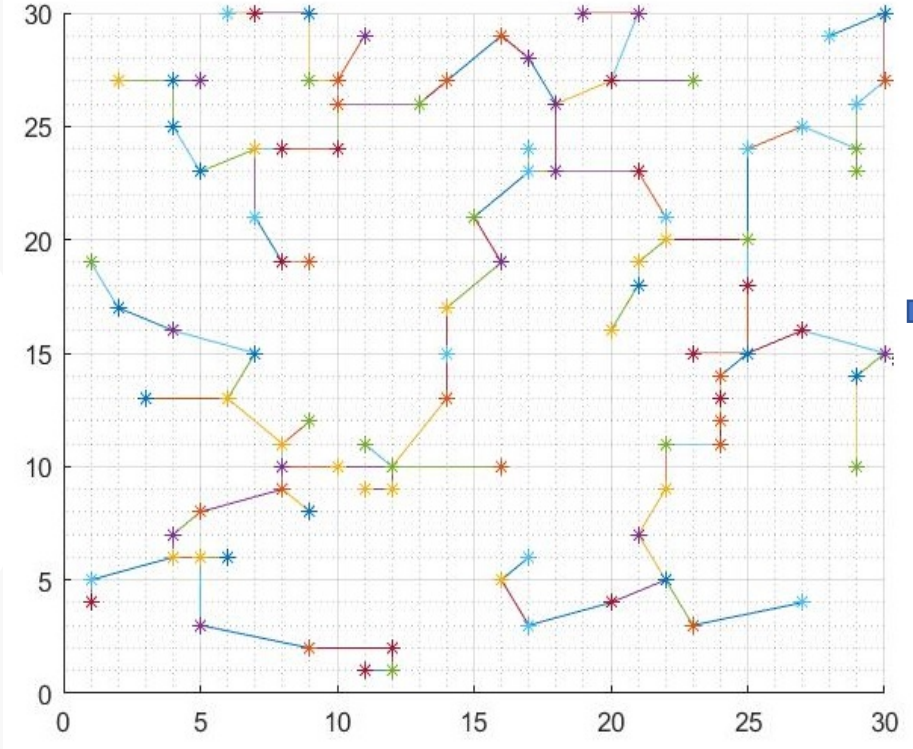
Before = 183

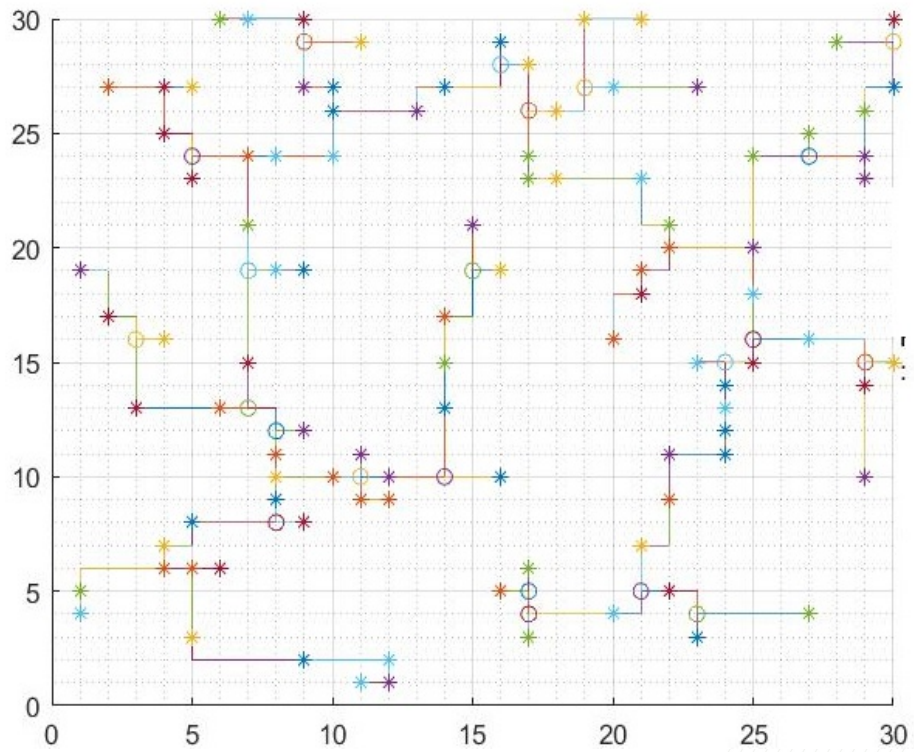
After = 163

Steiner Points = 20

Georgia Tech
CREATING THE NEXT

30_100 (Prim, Random)





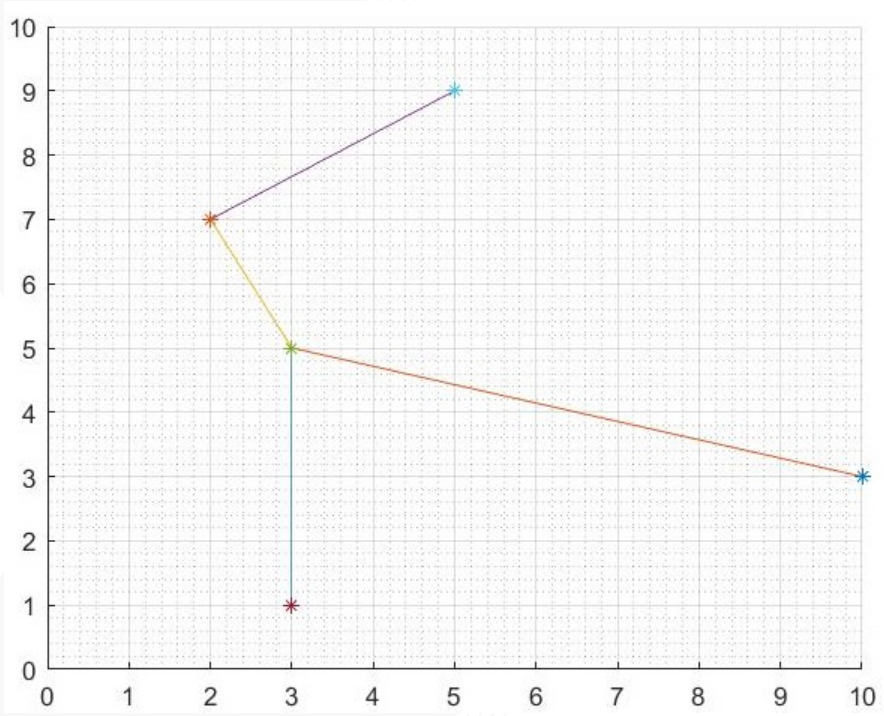
Before = 242

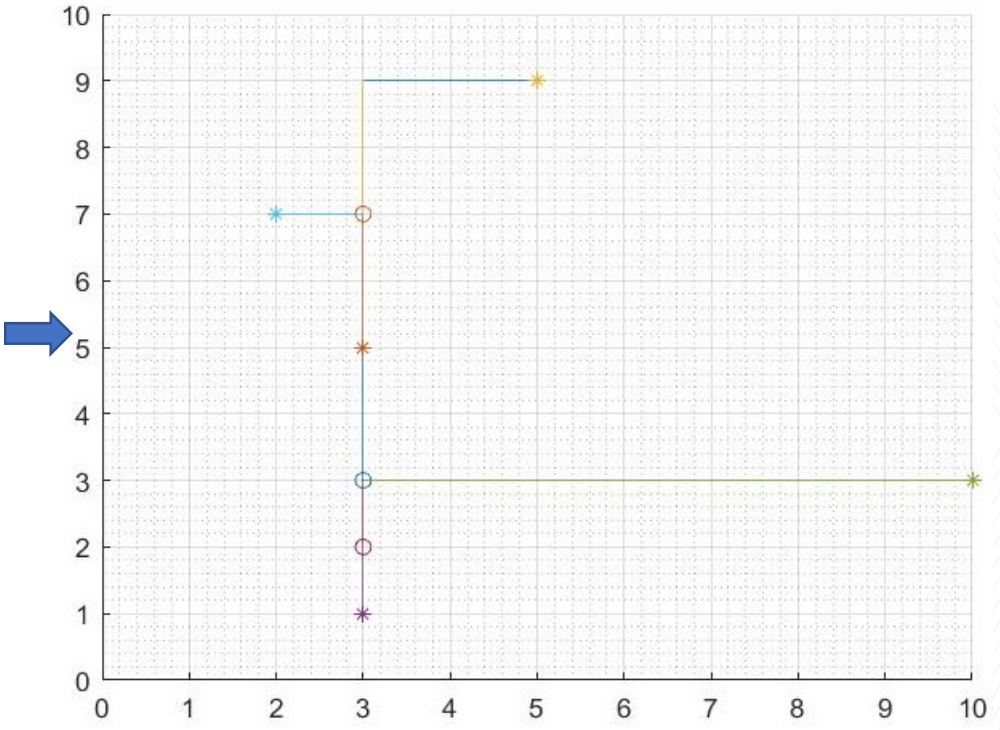
After = 220

Steiner Points = 22

Georgia Tech
CREATING THE NEXT

10_5 (Kruskal, Random)





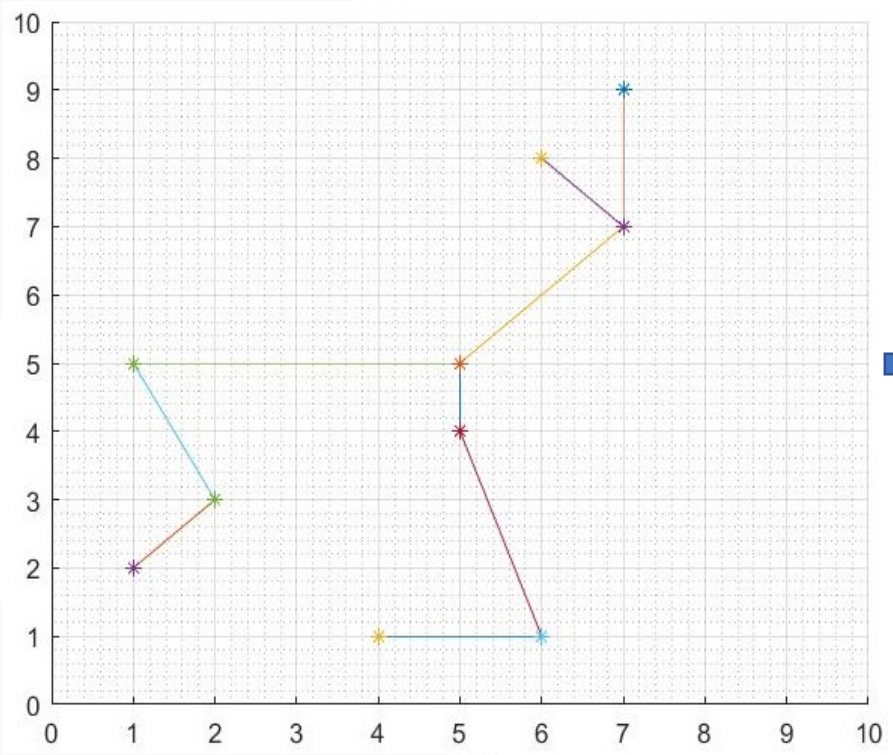
Before = 21

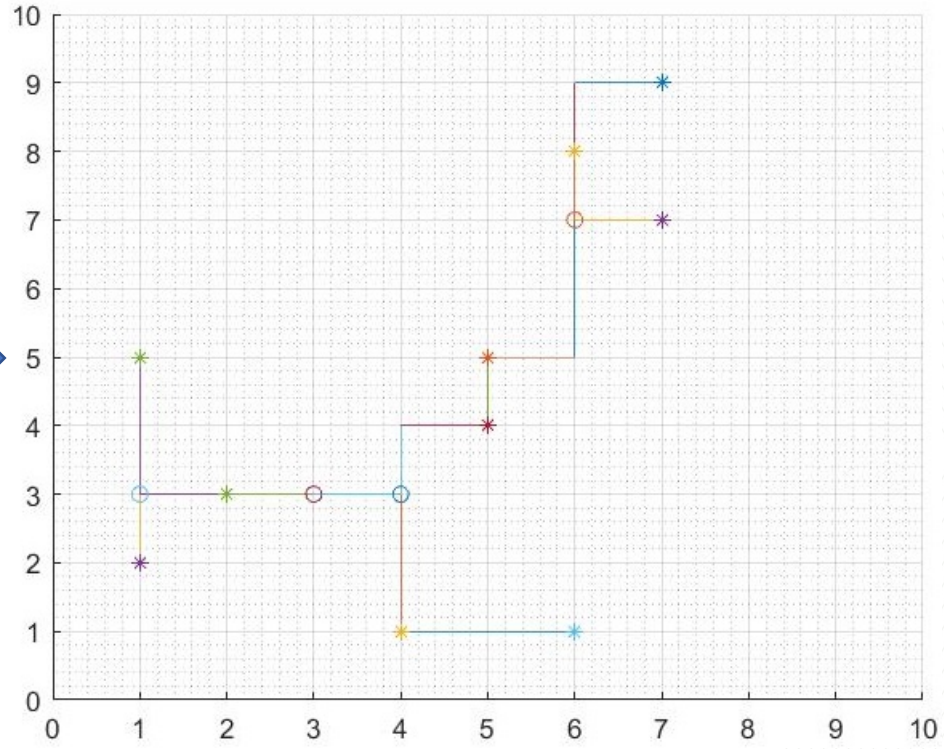
After = 18

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

10_10 (Kruskal, Random)





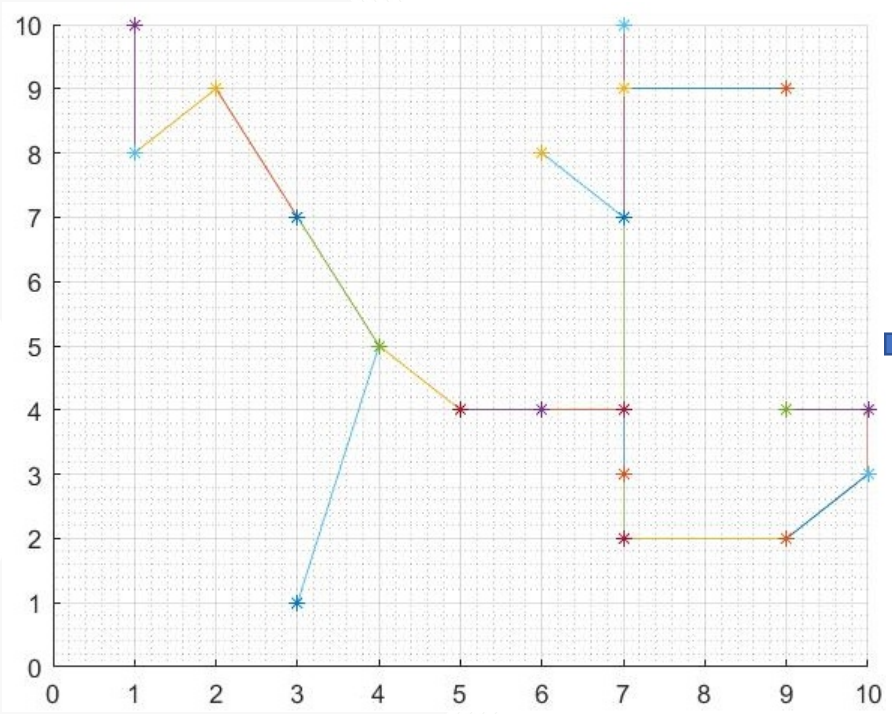
Before = 24

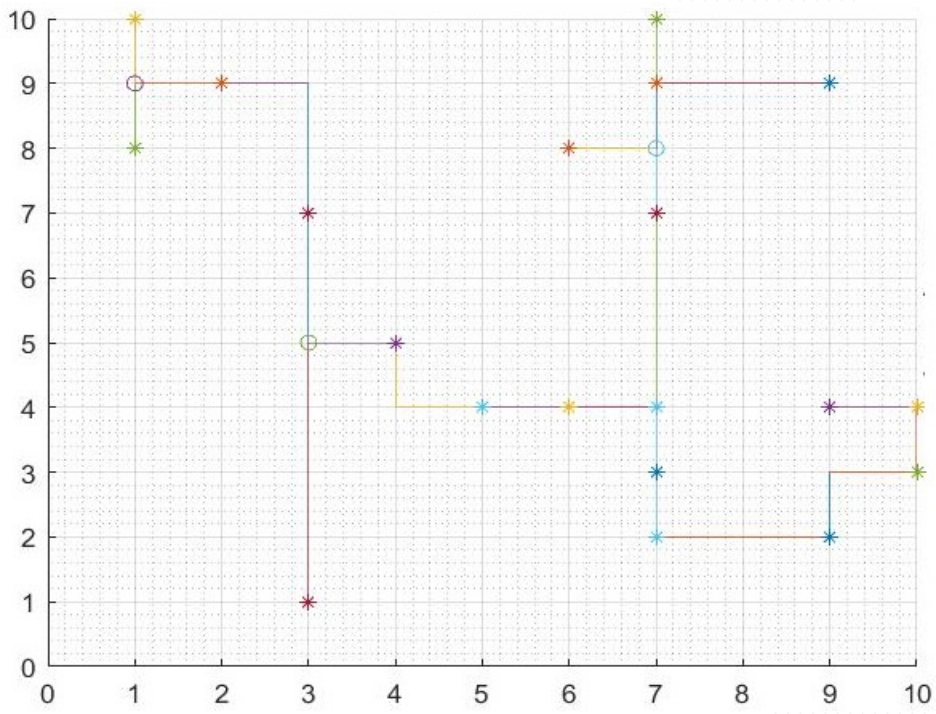
After = 20

Steiner Points = 4

Georgia Tech
CREATING THE NEXT

10_20 (Kruskal, Random)





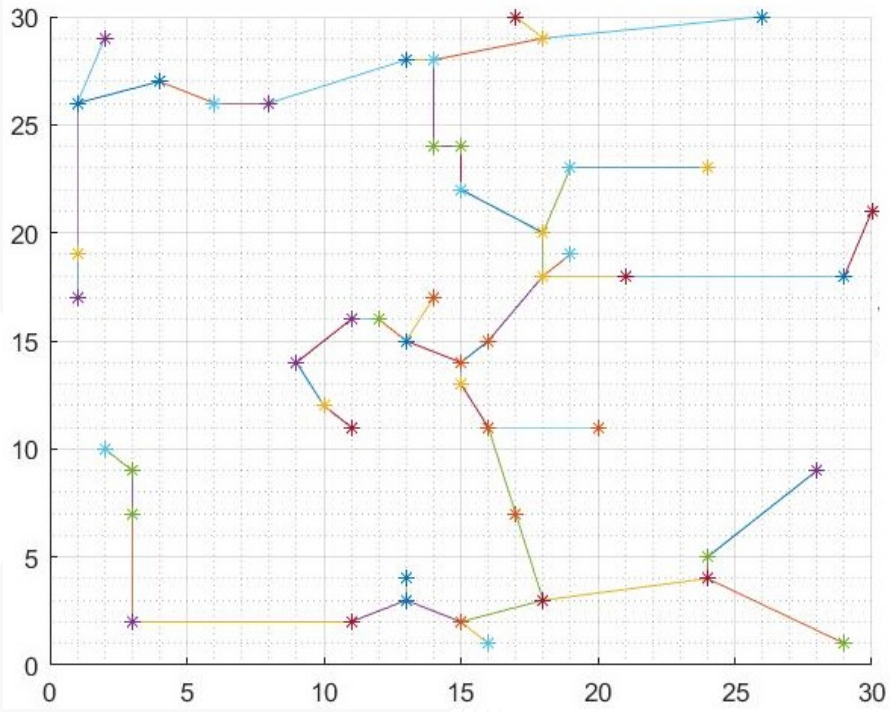
Before = 37

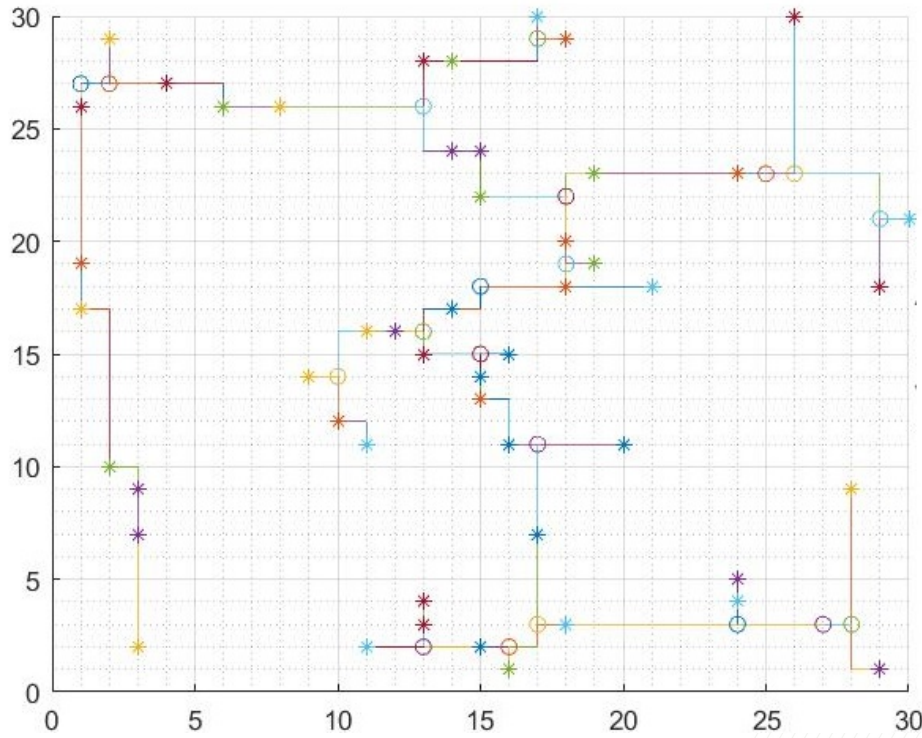
After = 34

Steiner Points = 3

Georgia Tech
CREATING THE NEXT

30_50 (Kruskal, Random)





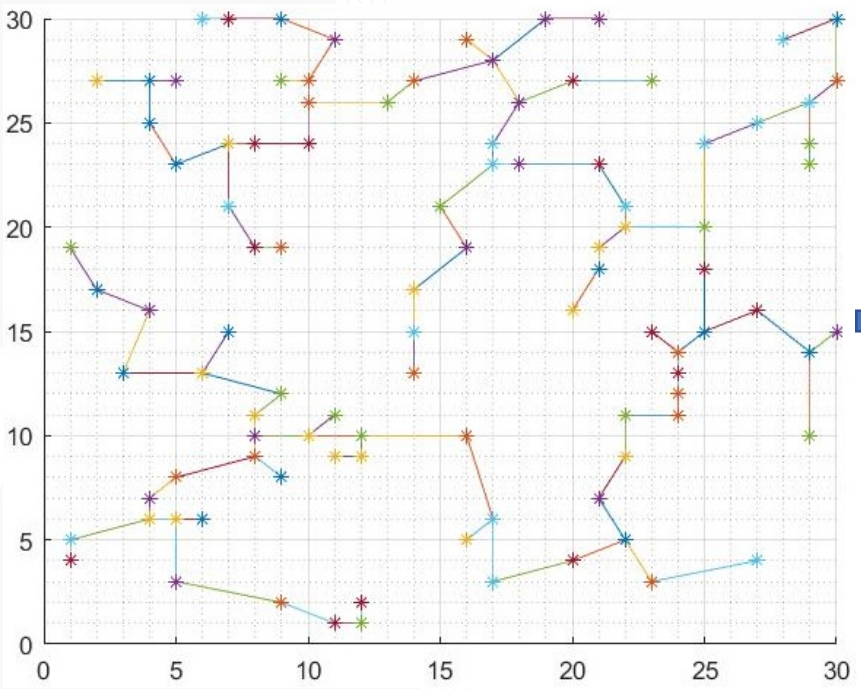
Before = 183

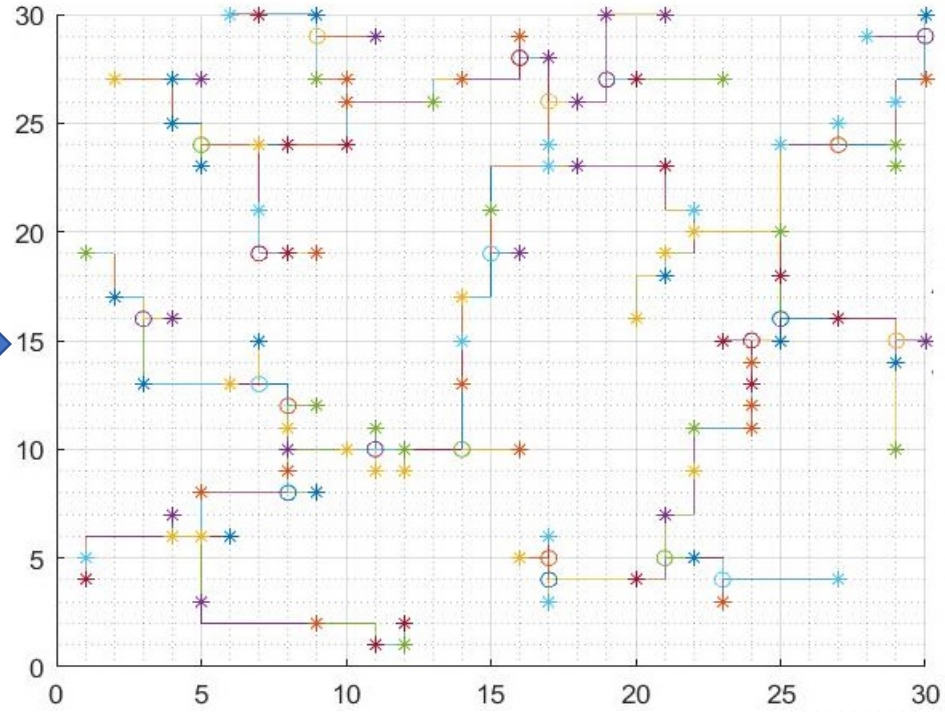
After = 163

Steiner Points = 20

Georgia Tech
CREATING THE NEXT

30_100 (Kruskal, Random)





Before = 242

After = 220

Steiner Points = 22

Georgia Tech
CREATING THE NEXT

THANK YOU

Georgia Tech
CREATING THE NEXT