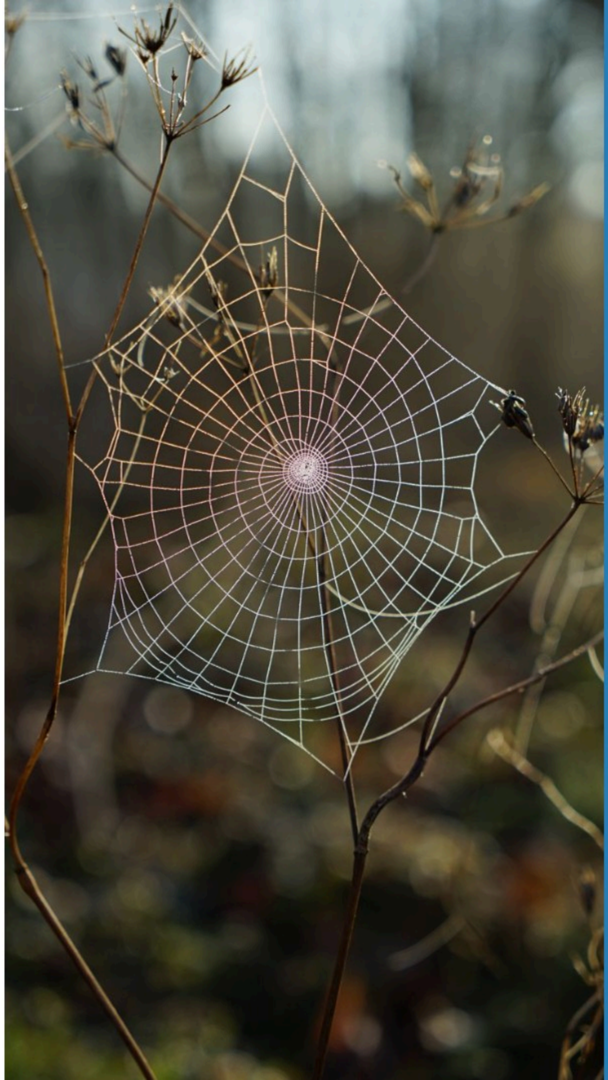


An approach to determine public facilities placement - a case study based on Canberra bus network

COMP4880 project presentation

team members:
Shidong Pan u6342277
Yinheng Chen u6341895

Catalogue

Introduction and Motivation

Algorithm and Method

Result and Evaluation

Conclusion and Future work

What are public facilities?

Ticket vending machines

https://www.otiglobal.com/ticket_vending_machines/

Electric vehicle charge stations

https://inhabitat.com/nations-first-nrg-evgo-all-ev-supercharger-station-installed-in-san-diego/

Sharing bike docks

https://www.alamy.com/stock-photo-bike-sharing-dock-rack-rome-italy-32280563.html

https://www.transport.act.gov.au/tickets-and-myway/get-myway

https://twitter.com/transport_cbr/status/1087869010509815808

How to distribute ticket vending machines in bus network?

We only have 5 ticket vending machines in Canberra. Not enough!

If the Canberra bus transportation company wants to set more auto charge machines to meet users's requirement. What's the placement plan to achieve most efficient distribution?

Also, we have to consider the robustness.

Nodes: machines randomly broke down;

machines in high-degree nodes are more likely not available (i.e. city center)

Edges: random edges failures (bus line broke down or road closed by accidents);

high-betweenness edge fails (i.e. traffic jam)

Dataset overview

	Canberra	Canberra_merged
Number of nodes	2520	1523
Number of edges	2908	1976
Average degree	2.30794	2.59488
Average clustering coeffcient of all nodes	0.0118474	0.0760809
Fraction of possible triangles	0.0210773	0.103448
Maximun degree	10	16
Minimum degree	1	1
Network diameter	51	44
Radius	26	22
Maximum closeness centrality	0.0848577	0.118149
Maximum betweenness centrality	0.184854	0.582854

- **Data pre-processing:**
  - merging nodes: decreasing the size of our network and make our approach more efficient.
  - opposite stops
  - close stops
- **Interesting properties:**

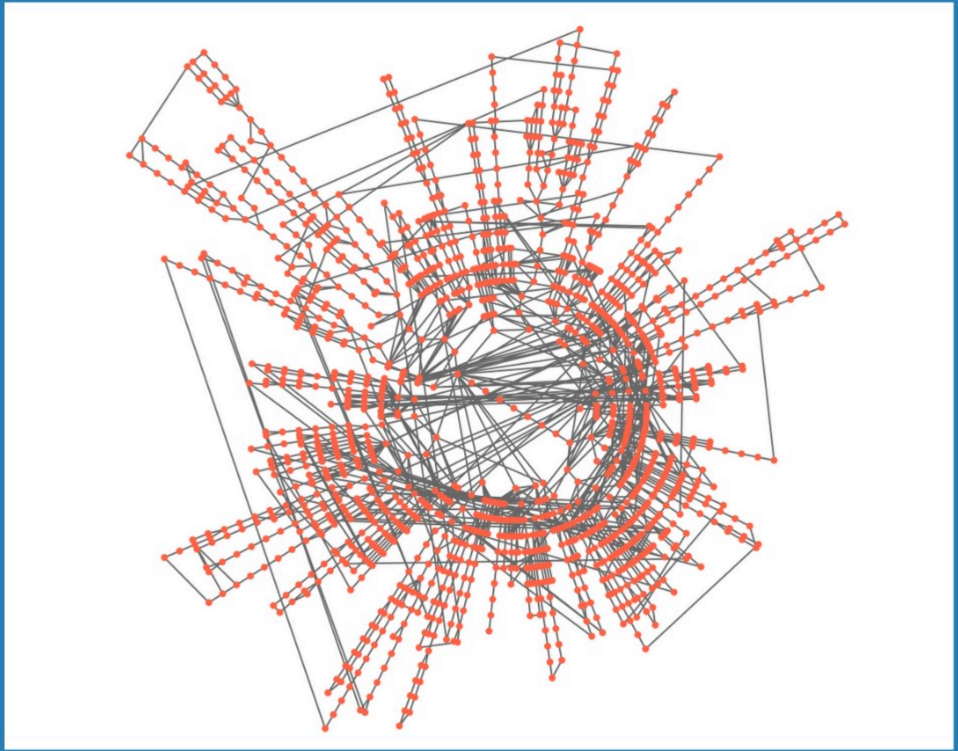
  average degree

  diameter

  betweenness

  ...

Dataset overview



- **Network topology:**

  red nodes represent merged bus stations

  edges represent trips

Distribution methods

Random

Randomly arrange a number of ticket vending machines in this network

High degree first

Firstly arrange ticket vending machines in the node with a high degree

Optimized solution

The best placement plan, but it's a NP-hard problem

Greedy algorithm

Every time arrange a ticket vending machine, minimizing the average shortest distance to the nearest machine

Our method: community + locally greedy

Based on communities detection method (greedy, Lovain etc.), and use greedy method within each community

Algorithms

How to measure the average shortest distance to the nearest machine?

1. The unit of distance is the stops.

2. We use breath first search to find the shortest distance between the node and the nearest machine. Then we compute the average shortest distance.

Greedy Algorithm

Assume we have n nodes(stops), m edges and we want to set b machines.

1. We assign a machine to a stop which can minimize the current average shortest distance most.

2. Repeat the step 1 until all machines are assigned.

Our Algorithm - Locally greedy by using community detection

1. We use a community detection method (Greedy/Louvian,etc.) to find the community partition of the network, assuming the best partition has q number of community. The average number of nodes of each community is n/q and the average number of edges are m/n .

2. Assume the number of machines is greater than the number of communities.

2.1. First, we assign q machines to each community.

2.2. In each community, we trade it as a sub-graph, we assign a machine to a stop which is in the community and it can minimize the current average shortest distance within the community most.

2.3. Once step 2.1 finished, then we choose a community has the least dense of machines (b_c/ size(community)), then we apply step 2.2.

2.4. Repeat step 2.3 until all machines are assigned.

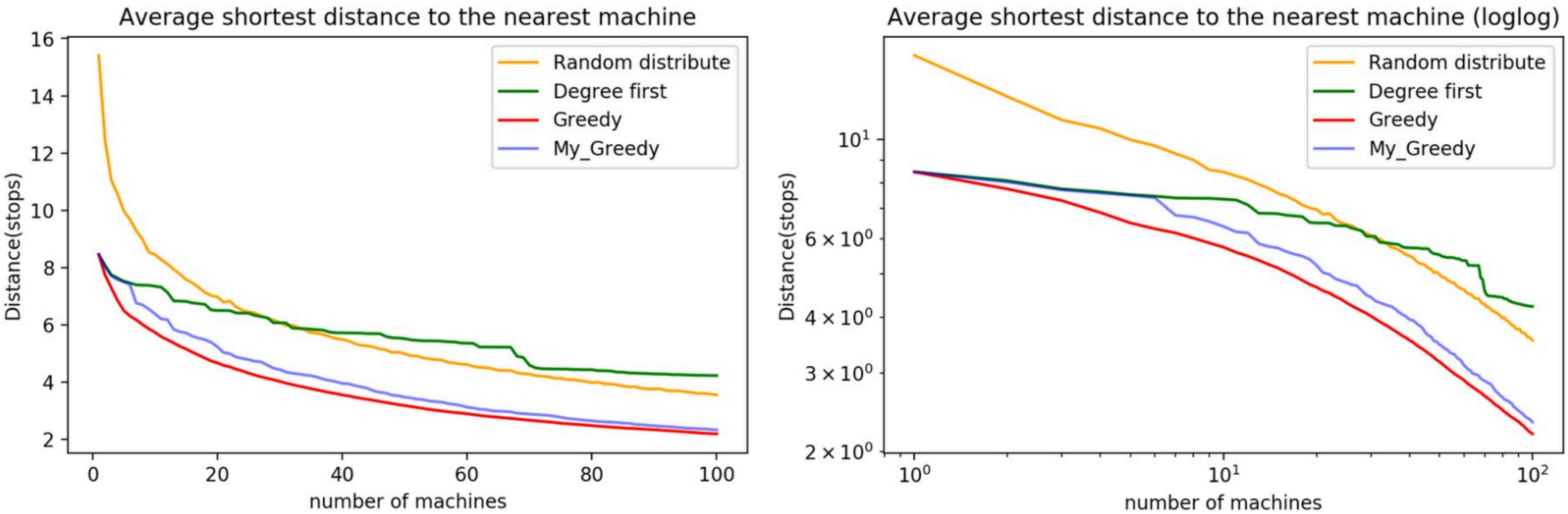
3. If the number of machines is less than the number of communities, we just assign machines to those communities have the largest size.

Compare of algorithms

n, nodes; m, edges; b, number of machines; q, number of communities

Distribution methods	Availability	Computational time complexity	Result
Optimised solution	No	N/A	unachievable
Random	Yes	O(1)	bad
High degree first	Yes	$O(nlgn)$	bad
Greedy algorithm	Yes	$O(bn^3 + bmn^2) \rightarrow$ $O(b^2n^2)$	good
Our method: community + locally greedy	Yes	$O(\frac{bn^3+bmn^2}{q^3}) \rightarrow O(\frac{b^2n^2}{q^3})$	good

Result



Average shortest distance to the nearest machine
Random distribute
Degree first
Greedy
My_Greedy
Distance(stops)
number of machines
16
14
12
10
8
6
4
2
0
20
40
60
80
100
Average shortest distance to the nearest machine (loglog)
10^1
6×10^0
4×10^0
3×10^0
2×10^0
10^0
10^2

Robustness analysis

machine random failure

- the machine randomly broke down
- under maintenance

machines in high degree failure (degree first)

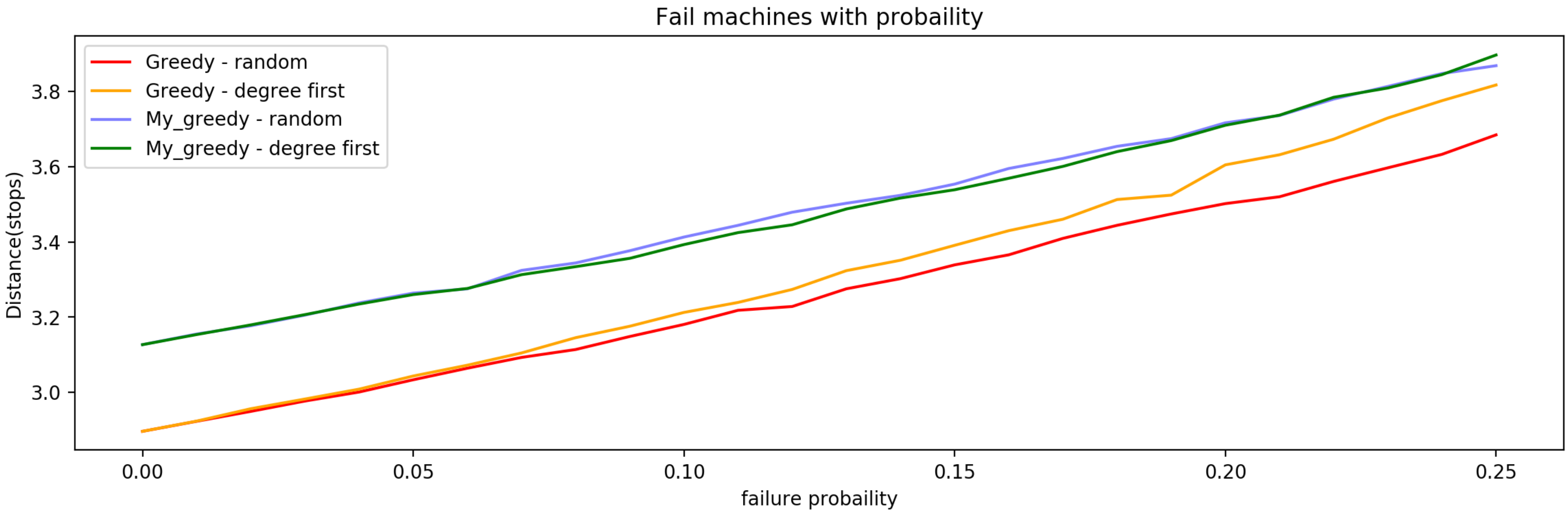
- a long queue
- tend to have problem due to high using frequency

edge random failure

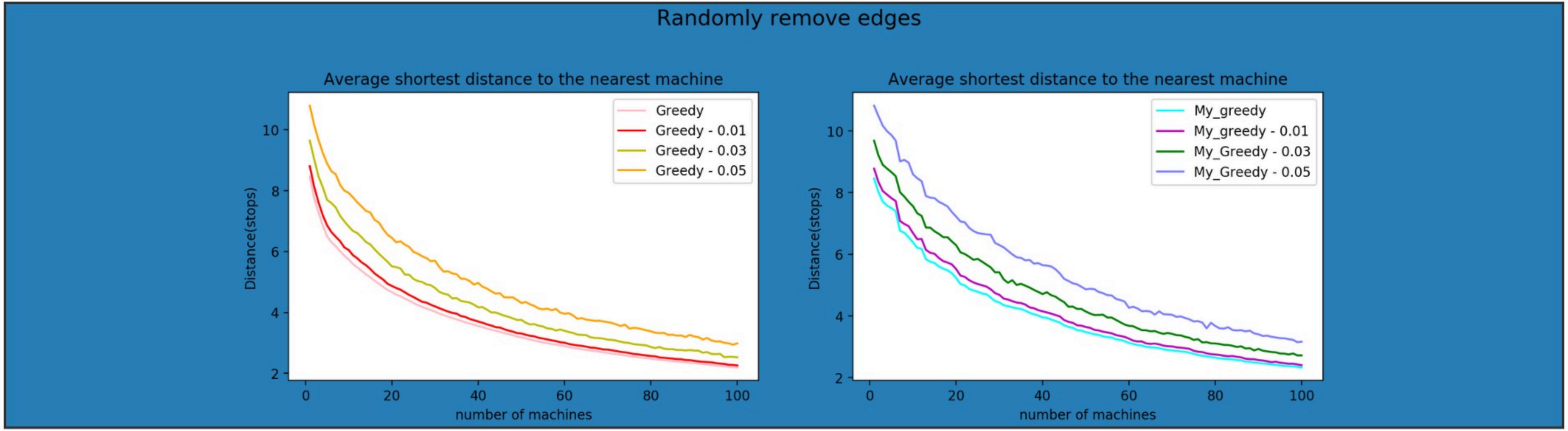
- bad weather
- road work

high betweenness edge failure

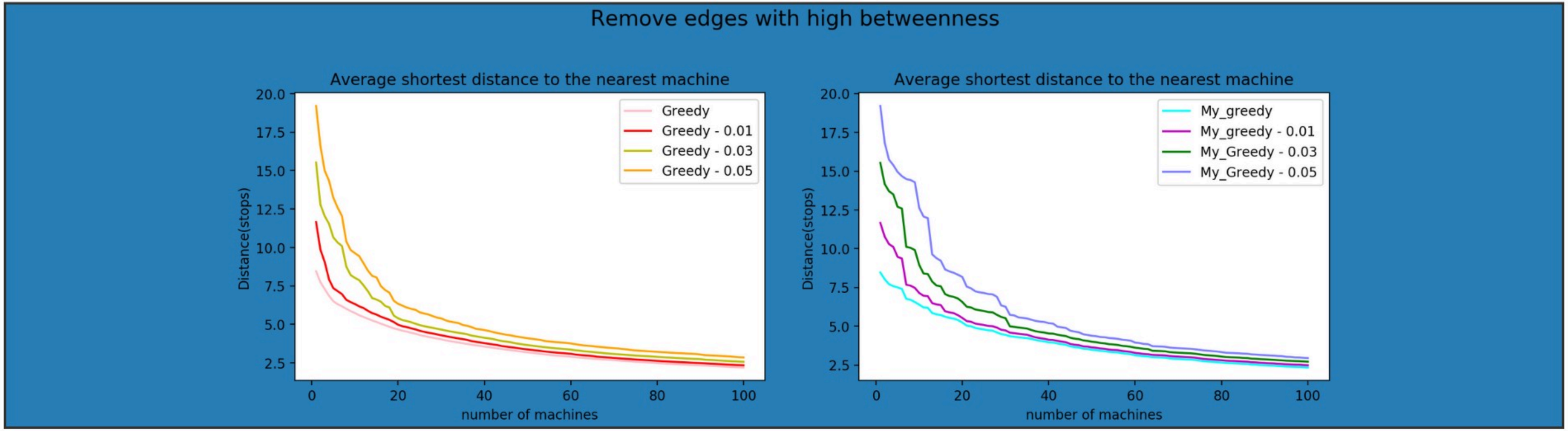
- traffic congestion at rush hours
- accident



Failure probability	0.00	0.05	0.10	0.15	0.20	0.25
Greedy-random	2.90	3.03	3.18	3.34	3.50	3.68
Greedy-DF	2.90	3.04	3.21	3.39	3.60	3.82
My_greedy-random	3.13	3.26	3.41	3.55	3.72	3.87
My_greedy-DF	3.13	3.26	3.39	3.54	3.71	3.90

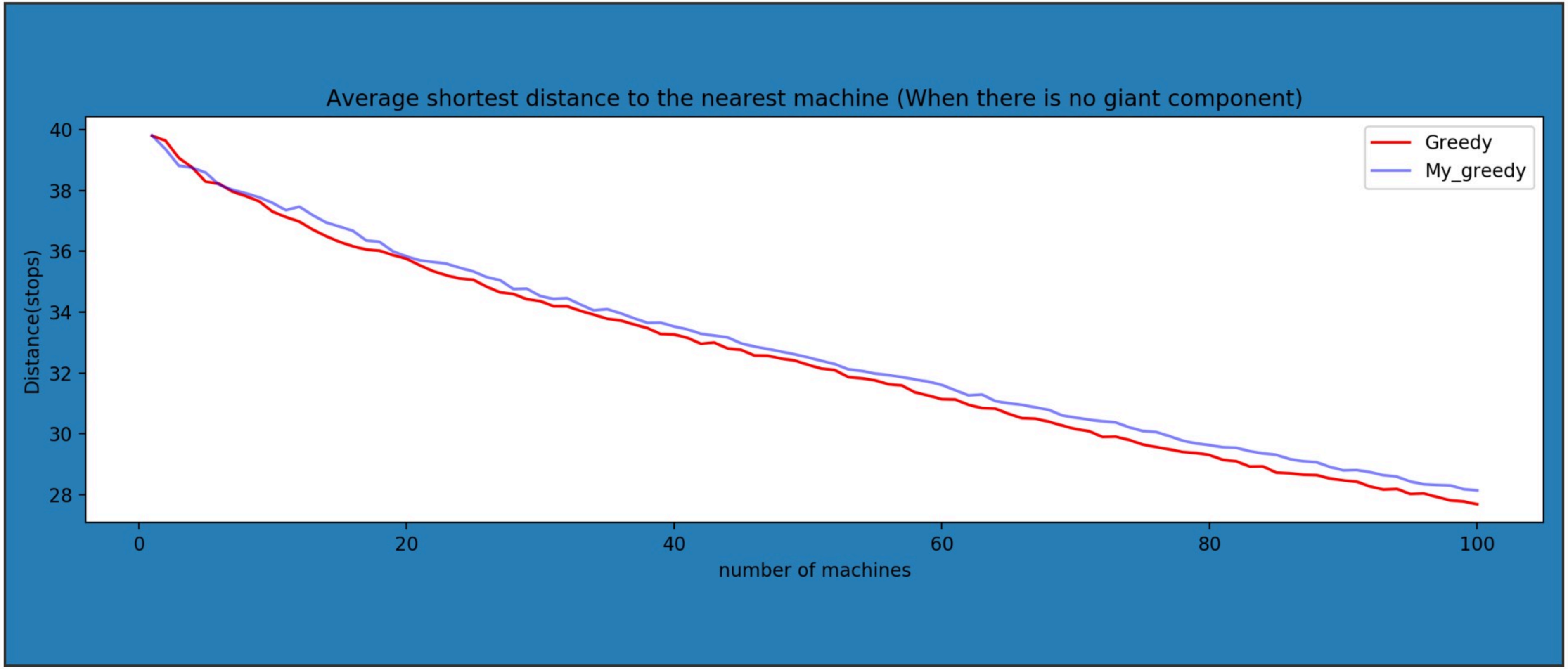


Randomly remove edges
Average shortest distance to the nearest machine
Greedy
Greedy - 0.01
Greedy - 0.03
Greedy - 0.05
Distance(stops)
number of machines
Average shortest distance to the nearest machine
My_greedy
My_greedy - 0.01
My_Greedy - 0.03
My_Greedy - 0.05
Distance(stops)
number of machines



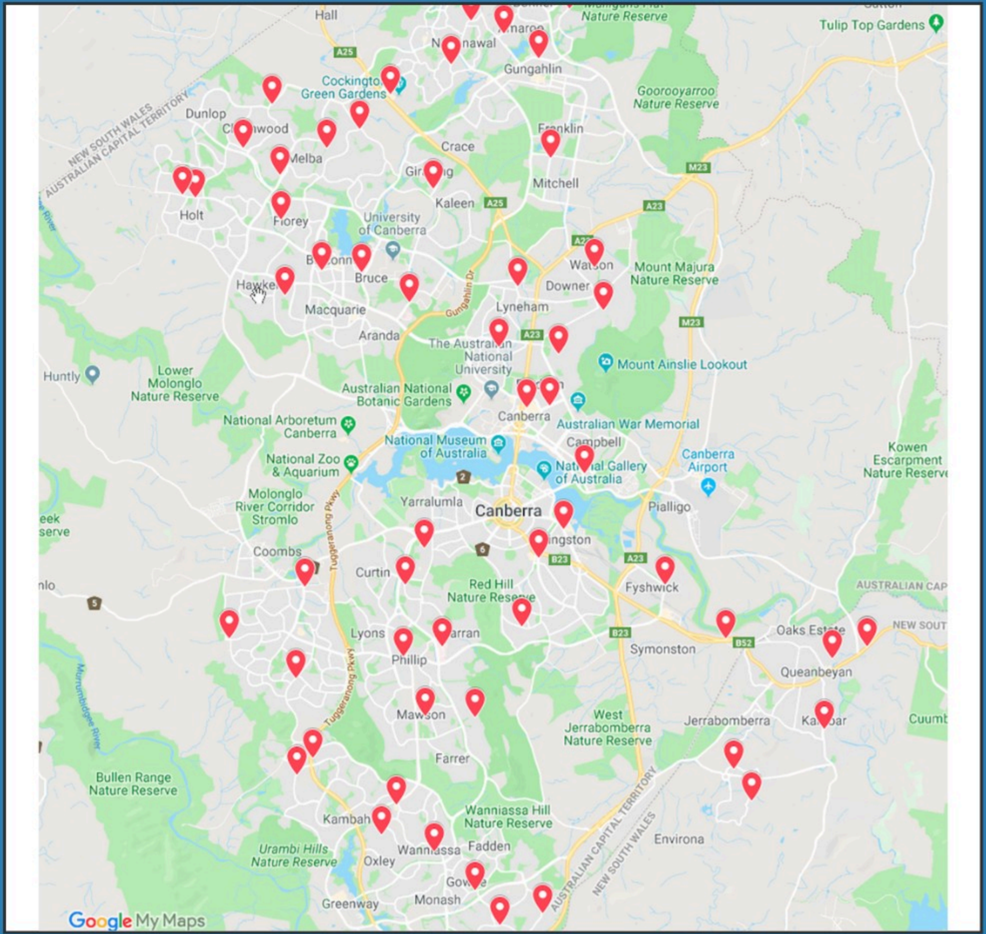
Remove edges with high betweenness
Average shortest distance to the nearest machine
Greedy
Greedy - 0.01
Greedy - 0.03
Greedy - 0.05
Distance(stops)
number of machines
Average shortest distance to the nearest machine
My_greedy
My_greedy - 0.01
My_Greedy - 0.03
My_Greedy - 0.05
Distance(stops)
number of machines

Remove edges to no giant component...



Conclusions

- Our results are basically same as the original greedy algorithm
- But our method is much more fast than original greedy algorithm
- Our distribution has a high robustness



Applications

- **Other city bus network..**

  ticket vending machines

  stops arrangement

- **City planning..**

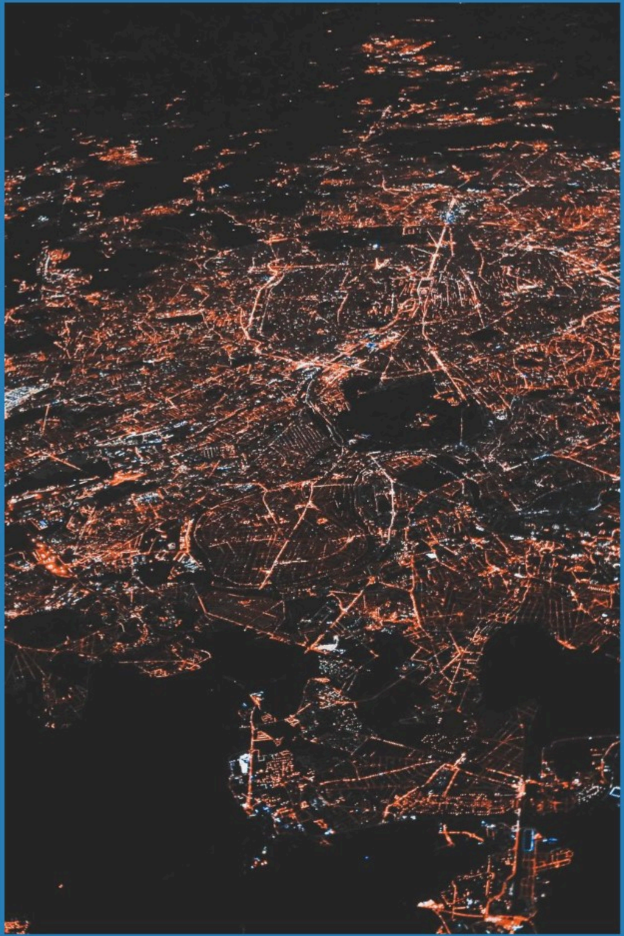
  self-help parcel lockers

  public toilets

- **New sharing-economy company..**

  sharing bike docks

  sharing power bank boxes



Limitation and future work:

1. The method of merging stops can be improved.

2. More random simulations are expected even though we already tested hundreds times.

3. Population density or people flow data can be added in future.

Thanks for listening!
Vote us~