Debojjal Bagchi
Bachelor of Science (Research)
debojjalb@iisc.ac.in

Abridged Project Write Up
Method of Successive Averages

Undergraduate (UG)
Date: 18.06.21

1 Details of the code

The code is divided into three sections:
1. Preliminaries
2. One to all shortest path algorithms
3. The MSA Algorithm

2 Travel times for MSA Algorithm

$$time_{UE} = time_{FreeFlow} * (1 + B(flow/capacity)^{power})$$

$$time_{SO} = time_{UE} + flow * time^{'}_{UE}$$

$$time_{SO} = time_{UE} + time_{FreeFlow} * B * power * ((flow/capacity)^{power})$$

3 Initialisation in MSA & other details

1. The x is MSA algorithm is initialised as follows: We find the times at 0 flow and then apply the shortest path algorithm to get initial flow.
2. The $time_{SO}$ is commented in the code. The code runs on $time_{UE}$. To see SO Results please uncomment it in $time\_fn(flow)$ function
3. Similarly the code uses Label Correcting algorithm label Setting algorithm is commented. Uncomment it in $shortest\_path(data)$ function to use it.

4 Summary of Results

Network	Nodes: Links: Zones:	TSTT (UE)	TSTT (SO)	% difference	Running Time (UE) MSA + LC	Running Time (UE) MSA + LS
Sioux Falls	Nodes: 24 Links: 76 Zones: 24	7497384.342333	7490919.22	0.086	8.17s	7.98s
Eastern-Massachusetts	Nodes: 74 Links: 258 Zones: 74	28225.81	27663.23	2.033	1.86s	2.31s
Anaheim	Nodes: 416 Links: 914 Zones: 38	1420110.10	1405632.089	1.03	1.76s	2.37s
Chicago-Sketch	Nodes: 933 Links: 2950 Zones: 387	18395240.49	18002791.86	2.17	387s	933s

5 Minimising time in large network like Chicago

We see in all relatively shorter networks. MSA was able to give results in less than 10s. But as the number of zones increase the polynomial time comes to play and our run time increases. I suggest the taking following **approximations** to reduce run time.
Approximation: In the demand matrix set $d[i][j]$ values to be 0 when $d[i][j] < 10^{-3}max(d[i][j])$
Why this works?: Demand between an OD Pair affects the final flows. But note that if demand at an OD pair is very low less than 0.001 times the maximum flow it will barely affect the flow values. We assume the demands are well distributed in a real life network and not skewed for this approximation to work the assumption is pretty true!
Does Doing this have any significant effect on Run Time?: Yes! Here are the results: (Also the equilibrium flow values are pretty close. As can be seen from the code)

		Without any approximation		Running time with Approximation of Order 10^-3*	
Network	Algorithm	TSTT (UE)	Running Time (UE)	TSTT (UE)	Running Time (UE)
Chicago-Sketch	MSA + LC	18395240.49	387s	14223775.623	154s

6 Concluding Remarks

1. It is difficult to predict in which situation MSA+LC is better and when MSA+LS is better. As LS terminates in n iterations but LC takes more iterations with less time per itteration. Still in most real case scenarios LC performed better.
2. MSA is slow! As said, ”Continental drift would beat it anyday!”. As Number of Zones & Nodes increase, MSA keeps taking more time as it runs on polynomial time. Also, convergence is slow based on inverse of iteration number as convex combination. Frank Wolfe Method would be much faster.
3. While finding Shortest Paths, topological ordering could help if the network graph is acyclic & can be topologically ordered.

END OF REPORT