Parallelize Closeness and Betweenness Centrality with APSP

Team 3: Abanoub Farag, Mitchell Humphries, Arze Lu, Rubayet Mujahid, Harshitha Patnaik

Objective

Implement two algorithms:

- Closeness Centrality
- Betweenness Centrality
- Parallelize

Calculate the following networks:

- Facebook Network
- Twitter Network

Compare runtimes between serial and parallelized algorithms.

Provided Data Overview

1. Amazon Product Co-purchasing Network

- Origin: Facebook.
- Nodes/Edges: 4039 / 88,234
- Directed: No
- Avg. Clustering Coef.: 0.6055

2. Twitch Gamers Network

- Origin: Twitter
- Nodes/Edges: 81,3.06 / 1,768,149
- Directed: No
- Avg. Clustering Coef.: 0.5653

Closeness Centrality (Floyd Warshall)

Objective: Get all shortest paths between all pairs of nodes

Time Complexity: O(n^3)

Pseudocode:

```
FLOYD-WARSHALL(W)
1. n ← rows[W]
2. D^(0) ← W
3. for k ← 1 to n
4.     do for i ← 1 to n
5.         do for j ← 1 to n
6.             d_ij^(k) ← min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
7. return D^(n)
```

Algorithm Comparison

- Ran different sample sizes with each algorithm
- Both serial and parallelization
-

Results – Closeness Centrality (Facebook)

- Applying Floyd-Warshall
- Runtime : 3 seconds for 120 iterations
  10 seconds for 240 iterations
  106 seconds for 780 iterations

Observations

- Runtimes:
  - Faster with Floyd-Warshall than Dijkstra
  - Parallelizing Dijkstra is less efficient for large graphs

Conclusions

- BFS is faster than Floyd-Warshall
- Dijkstra in theory is faster than Floyd-Warshall, but not in practice

Thank You!