Presentation Outline: Parallel Algorithms for Centrality in Dynamic Graphs

Nathan Bowness
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
nbown088@uottawa.ca

November 18th, 2020

Slide Topics:

1. Introduction
   - Introduction to the project
2. Betweenness Centrality
   - Brief definition of betweenness centrality
3. Where/Why is Betweenness Centrality Used?
   - Practical applications, where it is used
4. Calculating Betweenness Centrality
   - Overview of fundamental Brandes Algorithm (will include animations)
     - Run Breadth-first search (BFS) and Reverse-BFS (R-BFS) from every node
5. Betweenness Centrality in Dynamic Graphs
   - Brief overview, of how an edge update affects betweenness centrality
   - Applications
6. Calculating Betweenness Centrality in Dynamic Graphs
   - Introduction to Shukla et. Al's cutting-edge algorithm and its key concepts
     - Avoid BFS by using biconnected components
     - Using previous data stored from BFS/R-BFS
     - Computing betweenness centrality for a batch of edges
7. Biconnected Components:
   - How identifying biconnected components allows for less BFS and further parallelization
     - Discuss that re-computing the betweenness centrality within just the affected biconnected component is enough
8. Batch Update of Edges
   - Discuss how a batch of updates can be applied rather than recomputing the betweenness centrality after each edge insertion/deletion

9. Parallelization of Betweenness Centrality Calculation
    - Explanation of how the algorithm is parallelized for performance improvements (with animations for the graph)
10. Parallel Performance
    - Discuss how the performance was with multiple processors
11. Comparison of Results with Literature
    - On the same datasets
    - On different datasets
12. References
    - Reference any images used
    - Reference the papers talked about

Note: Likely sections **4** & **9** will contain multiple slides that include animations to show the process of computing betweenness centrality and showing the final parallelized algorithm, respectively.