A Comparative Analysis of Parallel Louvain Algorithms for Community Detection

Naw Safrin Sattar, Shaikh Arifuzzaman
Department of Computer Science, University of New Orleans, New Orleans, LA 70148 USA

Overview
- Community detection in networks is very important to understand complex networks and extract information in graph mining.
- Louvain algorithm is one of the efficient algorithms for community detection.
- Emerging size of social networks, increased amount of data over time require parallelization of algorithms.
- Parallel algorithms are necessary to deal with networks [1] of billions of vertices and edges.
- We provide a comparative analysis of Parallel Louvain Algorithms.
- We present a hybrid parallel algorithm using both OpenMP and MPI.

Louvain Algorithm
- Detects community based on modularity optimization [2].
- Better than other community detection algorithms in terms of:
  - Computational time and space
  - Quality of the detected communities

Modularity Calculation
\[ Q = \frac{1}{2m} \sum_{i,j} A_{ij} \left( \frac{k_i k_j}{m} - \frac{2 A_{ij}}{m} \right) \delta(c_i, c_j) \]

Here,
- \( Q \) = Modularity
- \( A_{ij} \) = Link weight between nodes i and j
- \( m \) = Total link weight in the network
- \( k_i \) = Sum of the link weights attached to node i
- \( c_i \) = Community to which node i is assigned
- \( \delta(c_i, c_j) \) = Kronecker delta. Value is 1 when nodes i and j are assigned to the same community. Otherwise, the value is 0.

- Two Phases
  - Modularity Optimization - looks for “small” communities by local optimization of modularity.
  - Community Aggregation - aggregating nodes of the same community a new network is built with the communities as nodes.

Louvain Parallelization Challenges
- Shared Memory Challenges
  - do not scale to a large number of cores and large networks
- Distributed Memory Challenges
  - Communication overhead
  - Efficient load-balancing scheme
- Hybrid (Shared + Distributed) Challenges

Utilize the advantages and minimize the disadvantages of both shared and distributed memory.

Algorithm 1: One Parallel Louvain using MPI

```
1. Define Input Graph G(V, E), partitions P into p processors
2. For each vertex v in G, assign it to a community
3. While increasing in modularity do
   a. Gather Neighbors linked to v across processors
   b. Compute Community gutters
   c. Update Community assignments
   d. Compute new modularity
4. If final, return modularity
```

- Performance Analysis
- 4-fold speedup for several real-world networks.
- Limitations
  - speedup is limited by physical cores available to system

Shared-Memory (OpenMP) Based Louvain Parallelization
- Performance Analysis
- 2 phases
  - Graph Partitioning
  - Community Detection

Distributed-Memory (MPI) Based Louvain Parallelization
- Performance Analysis
  - existing only MPI algorithm [4] has shown scalability to only 16 processors, that we have increased to larger number of processors
- Limitations
  - Communication overhead bottlenecks
  - Graph internal properties
  - Time spent in communication among processors.

Community No. (%) Table
- Deviation of number of Communities for different parallel approaches
- Table 1. Percentage of Community Size for DBLP Network

Table 2. Deviation of number of Communities for different parallel approaches

Future Works
- Finding an efficient load balancing technique for graph partitioning to minimize communication overhead.
- Eliminating the effect of small communities hindering the detection of meaningful sized communities.
- Investigating the effect of node ordering (e.g., degree based ordering, koreas and clustering coefficients).

Acknowledgements
This work has been partially supported by Louisiana Board of Regents RCS Grant LEGSF(2017-20)-RDA 25 and University of New Orleans ORSP Award CON000000002410.

References