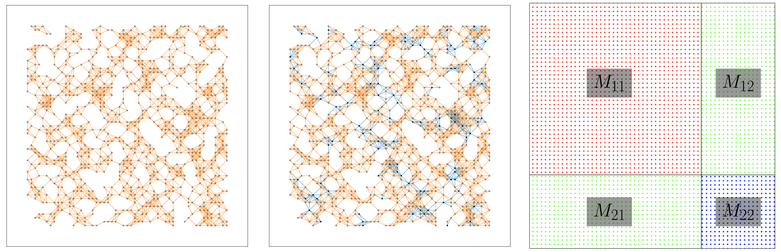
Incremental Graph Clustering in Parallel

Md Taufique Hussain (mth@indiana.edu) [1] Ariful Azad (azad@iu.edu) [1]

[1]Indiana University Bloomington

Incremental Graphs

Incremental Graphs: Graphs where new nodes and edges are being added incrementally



(a) A toy graph of 884 nodes, 6436 edges.

(b) 99 new nodes (black), 1540 new edges (blue)

(c) Adjacency matrix depiction (M_{all}) of the incremental graph. Red region represents old edges. Blue and green regions represent new edges.

Figure 1. Toy incremental graph

Clustering Incremental Graphs

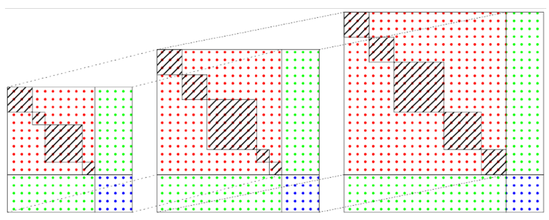


Figure 2. Pipeline of finding clusters in a graph over three incremental steps. Red regions represent old edges. Shaded regions represent clusters detected in previous incremental step.

Overall goal: Find clusters in the entire graph (M_{all}) assuming we know the clusters (shaded regions) in old graph (M_{11})

Simple solution: Create a combined graphs ignoring the previous clustering structure (Fig 1c), apply existing clustering algorithm. Set of clusters $C = \{c_1, c_2, ..., c_x\}$. High computational and memory cost (More than $O(nnz(old) + nnz(new))$).

Target solution: Remove $O(nnz(old)$ from the computational and memory complexity using previous clustering of M_{11}. Set of clusters $B = \{b_1, b_2, ..., b_y\}$.

Success measure: Agreement between B and C (F-score).

Application of Interest: Metagenomic Protein Family Detection

- **Step 1:** Samples are collected from the environment.
- **Step 2:** Micro-organisms are detected by genome sequencing. Each microorganism is a node.
- **Step 3:** Sequence alignment is done to measure similarity. Similarity is edge weight.
- **Step 4:** Detect protein families by clustering.

Motivation

Graphs are huge. Largest graph on which we could successfully detect clusters - *Metaclust50* having 282 million nodes and 37 billion edges with average degree 131 (https://metaclust.mmseqs.com/).

Graphs do not fit in a single compute node. Around 1 terabytes required just to store *Metaclust50* in memory (considering three 8 byte numbers for each non-zero). Single compute node memory of modern supercomputers is less than 100 gigabytes.

Computation requires even more memory. Requires large distributed computing resource. Clustering *Metaclust50* using *HipMCL* [1] required using 2048 compute nodes (131,072 cores) on NERSC Cori supercomputer for 3+ hours - which is one quarter of entire Cori.

Even larger graphs are possible. JGI IMG/M Database contains currently 77 billion sequences. Graph generated using entire database would contain 7 trillion edges (considering average degree 100).

Impossible to find clusters of such large graphs on any supercomputer. No supercomputer is capable of meeting the memory requirement of problems of such scale. Even there is any, no algorithm will scale due to communication being bottleneck.

Graphs are ever increasing. 65 billion sequences in August 2021, 70 billion sequences in October 2022, 77 billion sequences in August 2023.

Our two big aims -

- **Reduce memory requirement:** Find clusters of the new graph using smaller resources
- **Reduce resource wastage:** Find clusters of new graph faster than doing it again

Both without loosing much accuracy

Single Incremental Step: Our Solution

Substep-1: Detect clusters in M_{22} using regular clustering algorithm. For our target application. We use Markov Clustering(MCL).

Substep-2: Combine four pieces to create the incremental graph. Using full M_{11} and M_{22} is equivalent to full graph clustering. Simple solution is to remove inter cluster edges from M_{11} and M_{22}. But we can do better as we explain in the next section. To combine four pieces, we use parallel sparse matrix assign(*SpAssign*) and sparse matrix addition(*SpAdd*) operations. Load balancing these parallel operation is tricky due to the block structure.

Substep-3: Merge or split clusters. Simple solution is to apply regular clustering algorithm on the incremental graph. We choose MCL for our target application.

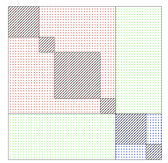


Figure 4. After substep-1. Clusters are detected in M_{22}(shaded areas in the blue region).

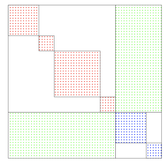


Figure 5. After substep-2. Incremental graph is prepared by sparsifying M_{11} and M_{22}.

Detour: Markov Clustering

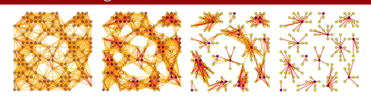
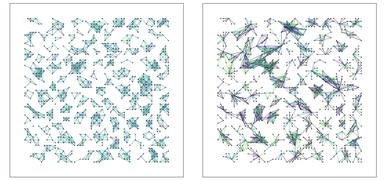


Figure 3. MCL iterations. Figure from van Dongen.

Iterates the following steps until convergence - simulate random walk (sparse matrix-matrix multiplication), inflate/deflate edge weights (elementwise power), discard low weight edges (sparsification). Matrix first become denser then keeps getting sparser when clustering structure starts to emerge.

Sparsification of M_{11} and M_{22}

In our solution, we keep MCL state of a state after first few MCL iterations as sparse summary. If average degree of the entire graph is d, it is being clustered in k incremental steps, at each incremental step our sparse summary keeps fraction r of the edges that it starts with, then average degree of the incremental graph at k^{th} incremental state can be expressed as a sum of the geometric series - $\frac{d}{k} + \frac{d}{k}(r + r^2 + ... + r^{k-1})$. When r is smaller than 1, the term is smaller than d. Because simulating random walk with SpGEMM is most expensive operation of MCL, and SpGEMM complexity depends on the average degree, our algorithm runs faster than full graph clustering at any individual incremental step.



(a) M_{11} after inter-cluster edge removal (4776 edges)

(b) M_{11} saved from 5^{th} MCL iteration (2721 edges)

Figure 6. Sparsification of M_{11} (6436 edges).

Experimental Evaluation

- **Dataset:** Metagenomic protein similarity network of 3 million nodes and 360 million edges
- **Platform:** 16 compute nodes (1024 cores) on NERSC Cori supercomputer
- **Incremental setting:** 50 incremental steps (equal number of new vertices in each step). Two variants - one is starting incremental clustering from beginning, another is running incremental clustering for last 20% incremental steps.

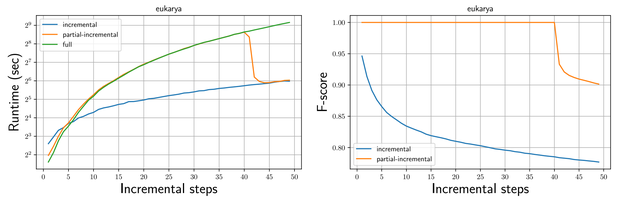


Figure 7. Both variants detected significantly good quality clusters while being at least 10× faster than the full graph clustering on the last incremental step. Difference would be more significant for larger graphs and higher parallelism.

Conclusion

- For graphs that are not so massive – trade off between accuracy and computational resources
- For massive graphs – the only scalable way
- Promising result for metagenomic protein interaction graphs
- Has potential to enable unprecedented scale of graph clustering
- Accelerate scientific discovery

References

[1] Ariful Azad, Aydin Buluç, Georgios A Pavlopoulos, Nikos C Kyrpides, and Christos A Ouzounis. HipMCL: a high-performance parallel implementation of the Markov clustering algorithm for large-scale networks. *Nucleic Acids Research*, 46(6):e33–e33, 01 2018.