Dynamic Graph Edge Clustering

Michael Ostroski
michael.a.ostroski.civ@mail.mil
Department of Defense

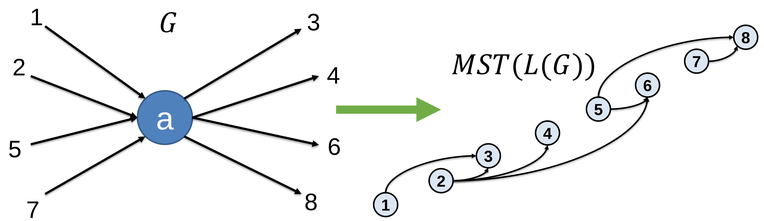
Motivation and Approach

We often know which nodes in a graph are important and need to answer the question of "when" we should be paying more attention to those nodes. To accomplish this we look for a way to group record data, effectively clustering the edges of a graph.

The Line Graph $L(G)$ of a graph G is another graph that represents the adjacencies between edges of G. Forming $L(G)$ for record data and clustering would result in a grouping of records we are looking for. Unfortunately, $L(G)$ can be several orders of magnitude larger than G, which is unacceptable for many data sets. We present a scalable approach to edge clustering that uses an approximate line graph, filtered by time, as a way to capture potential causal relationships between records. Analysis of the time-filtered line graph captures higher-level phenomenology that are not explicitly coded in the data, suggesting that it is an effective model for these problems.

Single Node Minimum Spanning Tree (MST)

For any given node $a \in G$, consider the set of incoming and outgoing edges. Here we will let the edge labels $\{1 \dots 8\}$ also be timestamps for when the edge occurred.



We only look for connections “forward” in time and the MST puts more emphasis on records that are “closer”. This can be computed quickly without generating the full line graph.

Any given edge weight will be the difference in time stamps of the associated records. More generally, the weights could depend on any number of edge or node features.

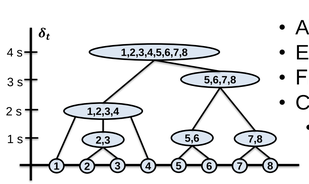
Distributed “M”ST Algorithm

Input: List of records (edges) of length E
- Create Adjacency list representation (segmented array)
- For each node compute a local MST

Output: list of at most $2E$ edges of $L(G)$ that at least contain a MST

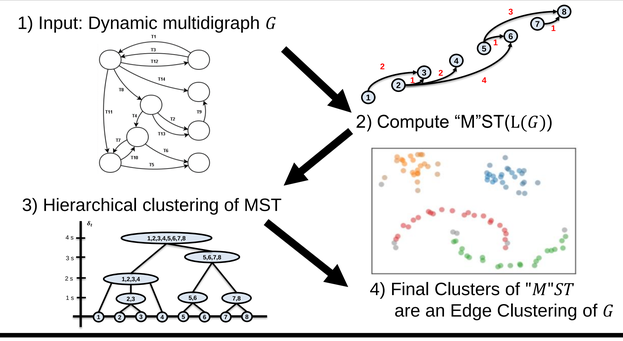
Hierarchical Clustering

From	To	δ_t
2	3	1 sec
5	6	1
7	8	1
1	3	2
2	4	2
5	8	3
2	6	4



- Algorithm based on HDBSCAN
- Each record starts in its own cluster
- First, combine clusters that are “closer”
- Continue until the full tree is built
  - Computed efficiently using a Union-Find Data Structure

Putting it All Together



Example Data Set

Enron Metadata

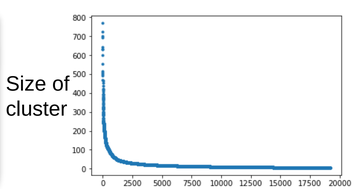
- ~4 years of email data
  - Anonymized
  - Topology only, no content
- Number of Records:
  - 1,341,535
- Number of Nodes:
  - 86,868
- Total Processing Time:
  - ~5 mins

MST Info

- # Nodes (Records): 1,341,535
- # Edges: 1,636,158
  - Note: this is larger than the true minimum spanning tree due to distributed and multithreaded optimizations

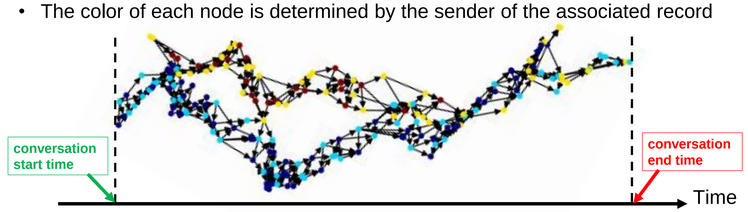
Clustering Info

- Min Cluster Size: 5 Records
- Max Cluster Size: 770 Records
- # Clusters: 19,169
- # Noise Records: 652,132



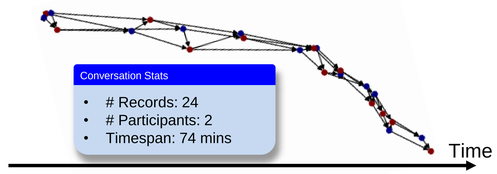
Conversation (Cluster) Visualization

- Each node is a record with an association timestamp.
- The X-axis is time
- Y-axis is chosen to make edges short using Kamada–Kawai algorithm
- The color of each node is determined by the sender of the associated record

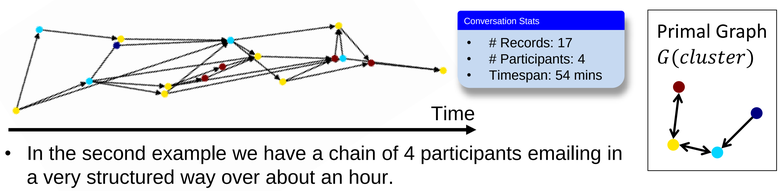


Example Conversations

- Conversations are built from “close” (in time) connections but can span much longer time frames
- The “Primal Graph” of a cluster is the graph that is induced by the records of the cluster

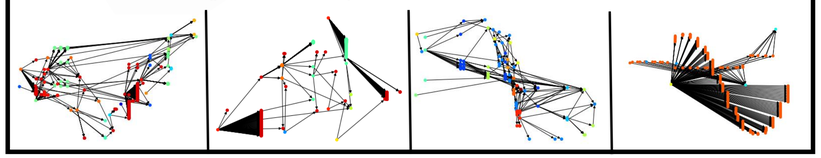


- In this first example conversation we see a back-and-forth of 24 emails between 2 participants over a little more than an hour.



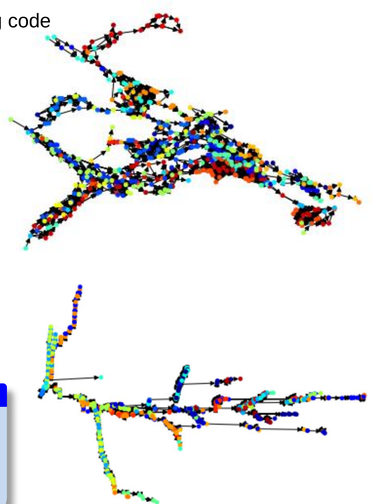
- In the second example we have a chain of 4 participants emailing in a very structured way over about an hour.

- There’s no shortage of more complicated structures that occur.



Next Steps and References

- Optimize MST generation and Clustering code
  - Arkouda + HavoqGT integration
  - Current top scaling notes:
    - ~50 billion records
    - 90 nodes of CRAY XC40
    - ~30 mins
- Cluster/Classify Conversations
  - Feature engineering on groups of records
- Test on more dynamic graph datasets
  - Netflow, Social Network graphs, ???
- Refine MST
  - Include properties
  - Base weights on historical data
- Content analysis on conversations



References
- Arkouda:
  - github.com/Bears-R-Us/arkouda
- HavoqGT:
  - github.com/LLNL/havoqgt
- HDBSCAN Documentation:
  - https://hdbscan.readthedocs.io/