Vertex Clustering of Augmented Graph Streams

Ryan McConville Weiru Liu Paul Miller

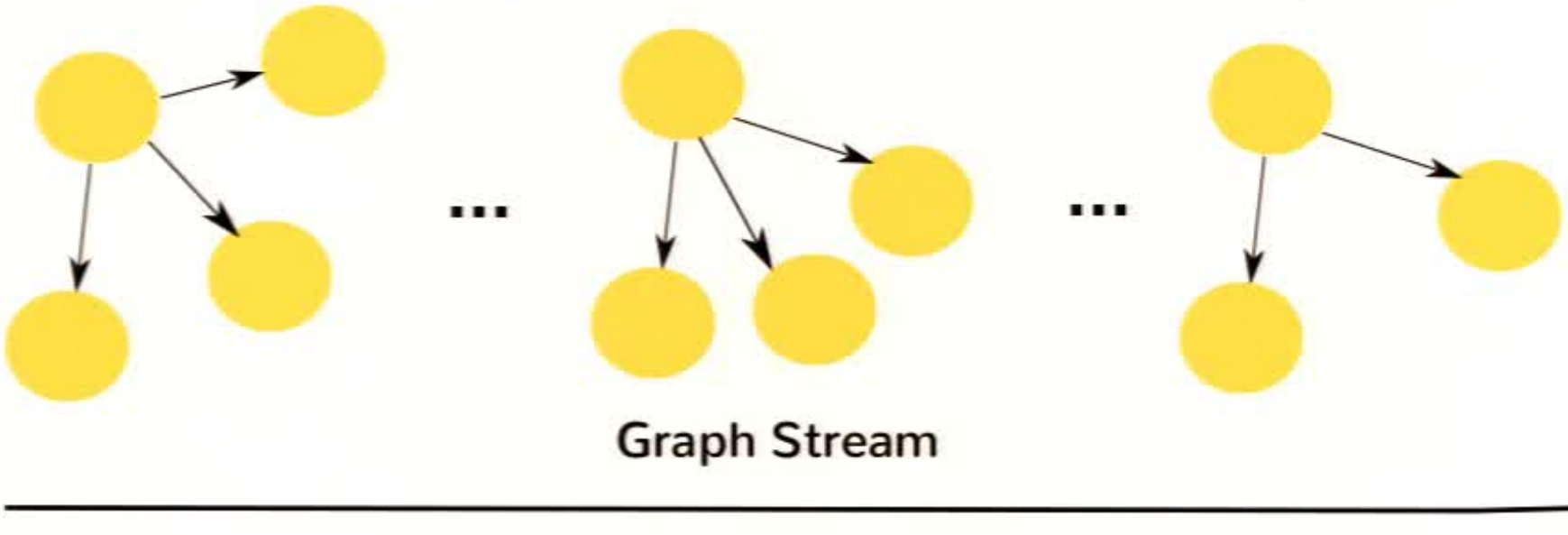
Queen's University Belfast

rmcconville07@qub.ac.uk

April 30, 2015

Graph Streams

- Graph data is everywhere and much of it is being constantly updated.
- Batch processing after update from the stream can be wasteful at best or infeasible at worst.
- Alternative?
  - Process each update as it arrives - update clustering incrementally



Definition

A graph stream $G = \{G_1, G_t, \ldots, G_n\}$ is a stream of subgraphs; For clarity, we call each G_t from the stream a transaction graph.

Side Information

- Side information to the graph is also useful.
- Each new streaming graph involves multiple vertices, which may have conflicting side information
  - On a social network an interest may be shared but location may be different.
- Therefore we associate each piece of side information with the relevant vertex rather than the graph as a whole.

Definition

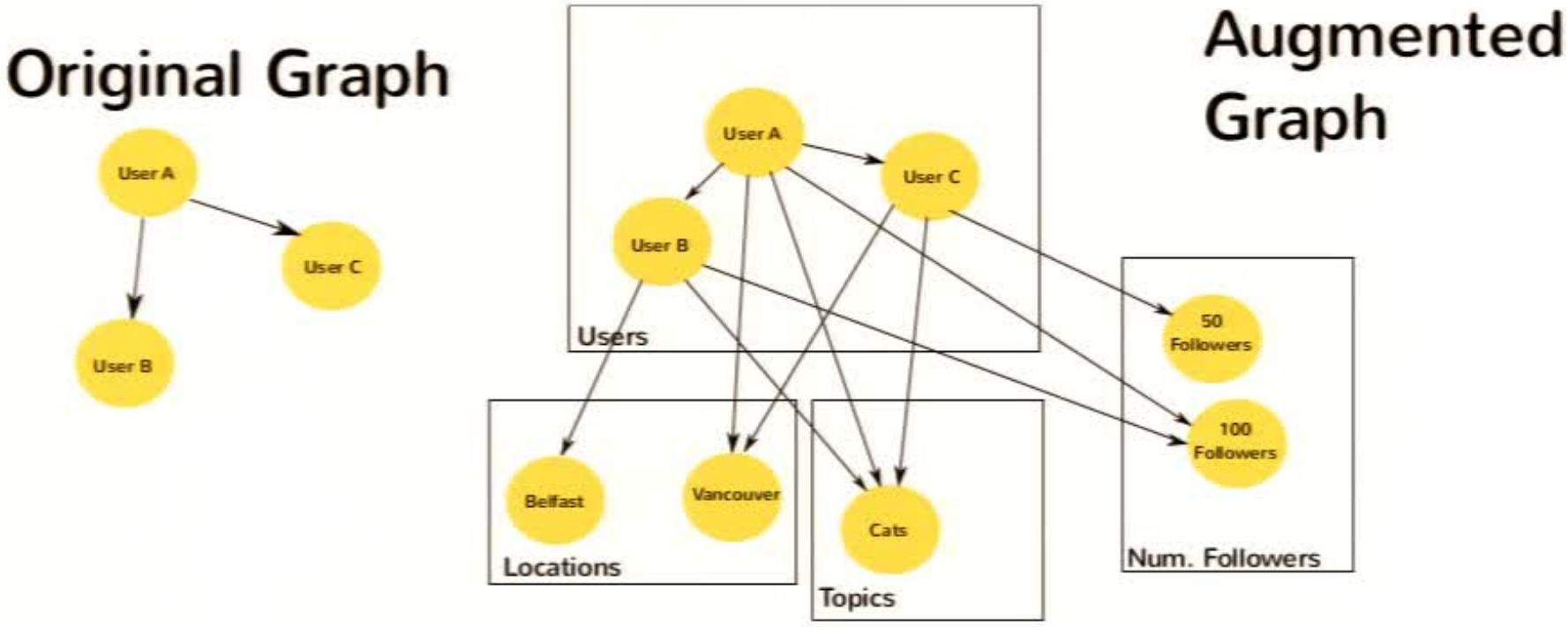
Let A be a set of discrete attribute labels $A = \{a_1, \ldots, a_n\}$ associated with a graph stream. Let $a \in A$ be an attribute label with the domain $D_a = \{d_1, \ldots, d_m\}$. The set of attribute-value pairs $\lambda = \{\langle a, d \rangle : a \in A, d \in D_a\}$.

Augmentation

- We augment the original structural graph with side information included as vertices.
- This provides a unified framework allowing a single similarity measure for both original structural and augmented side information.

Definition

An augmented subgraph G_s of the graph G_t is denoted $G_s = (v_t, V_s, E_s, \lambda_s)$ where v_t is the structure vertex of G_s, V_s is the set of other structure vertices in G_t, E_s is the set of edges, and λ_s the set of attribute-value pairs belonging to v_t.

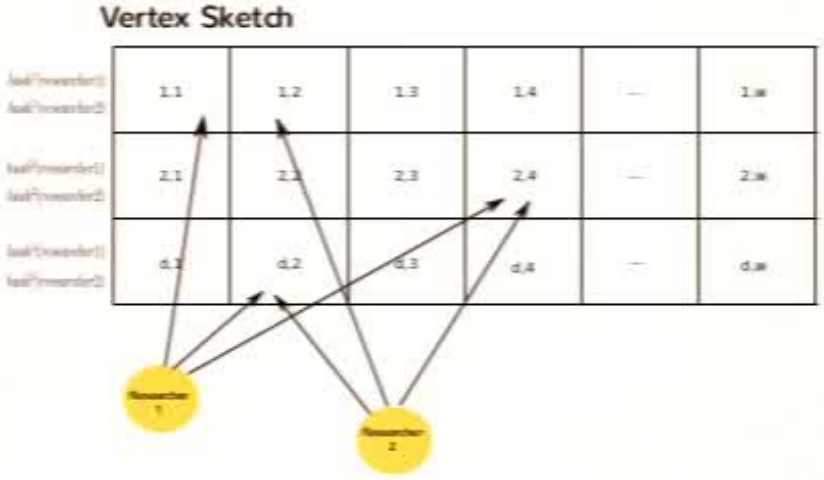


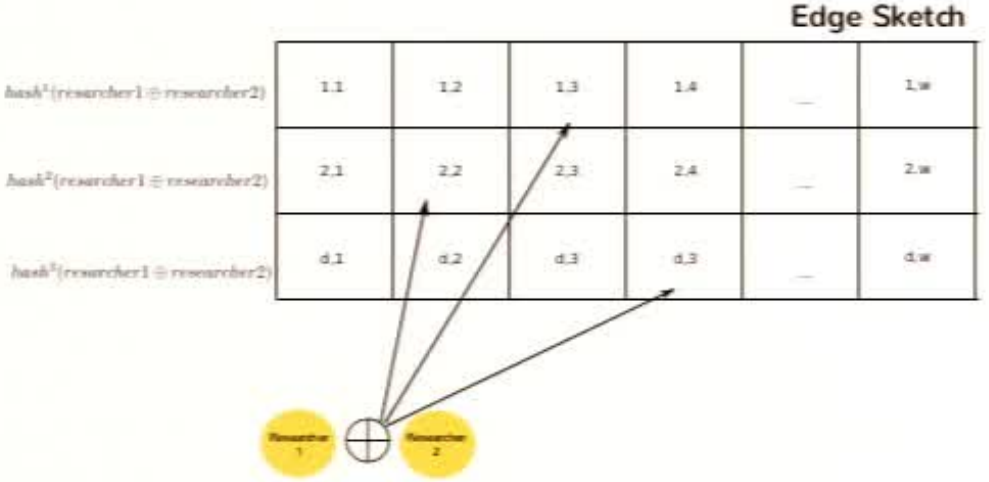
Each transaction graph G_t will be partitioned into $|V_t|$ augmented subgraphs. For each $v_t \in V_t$ we create one augmented subgraph that contains the set of attribute value pairs for vertex v_t in addition to original transaction graph G_t.

Count-Min

- Count-Min is a data structure that provides a number of useful features to deal with large quantities of data.
  - Constant sized with known error bounds.
  - Can scale size and accuracy up or down based on available hardware.

- We can't just cluster edges or graphs as we then lose individual vertex information!
  - So store both vertex and edge information within Count-Min.





Objective

- Modeled our data by augmenting original graphs with side information.
- Assign similar augmented subgraphs from the stream to the same clusters.
  - How many clusters will exist?
  - How do we manage these clusters over time?

Defining likeness between a subgraph and cluster

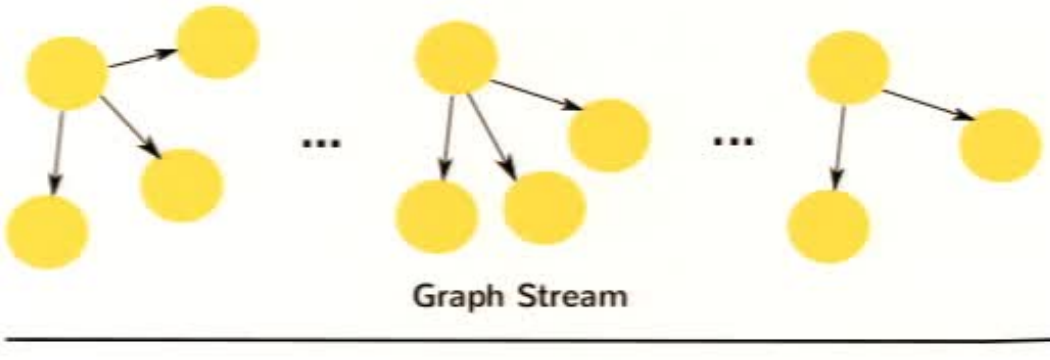
$$MS_V(v, c_l) = F_V(v, c_l)\, w(v) \tag{1}$$

$$MS_E(\langle v_t, v_j \rangle, c_l) = F_E(v_t \oplus v_j, c_l)\, w(v_j) \tag{2}$$

$SoB(c_l)$ is a vector containing the Sample of Best subgraphs in the cluster. The likeness expectation of the current subgraph G_s to an existing cluster c_l is calculated via the likeness.

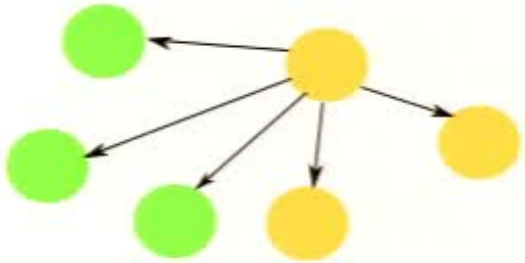
$$likeness(G_s, c_l) = \frac{\left(\sum_{\forall v \in \{v_t\} \cup \lambda_s} (MS_V(v, c_l)) + \sum_{\forall \langle v_t, v_j \rangle \in Es} (MS_E(\langle v_t, v_j \rangle, c_l)) \right)}{mean(SoB(c_l))} \tag{3}$$

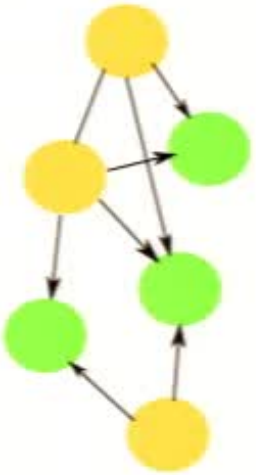
$max(likeness(G_s, c_l))$ is the most similar cluster to the current subgraph.



1

For each graph from the stream augment with side information (in green)

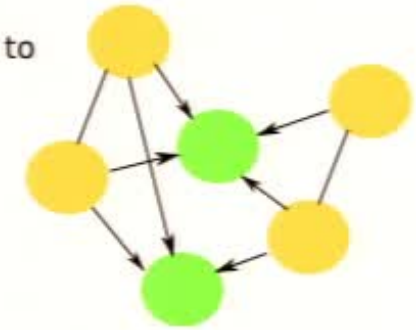




Cluster 1

Calculate likeness of augmented subgraph to each cluster

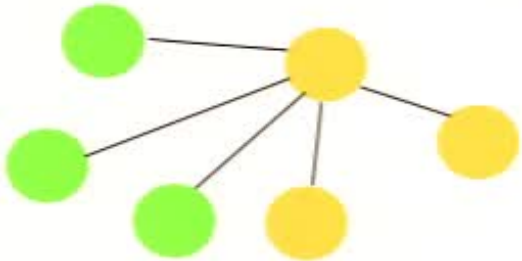
2



Cluster 2

3a

If outlier to most 'similar' cluster then create new cluster with subgraph.



Cluster 3

3b

Insert graph into most 'similar' cluster if not outlier.

Number of Clusters and Merging

- How to **predict** the number of clusters that will be in a stream?
- Create new clusters based when certain conditions are met.
  - No similar cluster found.
  - Subgraph is an outlier to the most similar cluster.
- Periodically merge clusters that are similar.
  - We choose average cosine similarity over heaps of of most frequent d vertices in each cluster.
  - Need to choose similarity threshold.

Datasets used in Experimentation

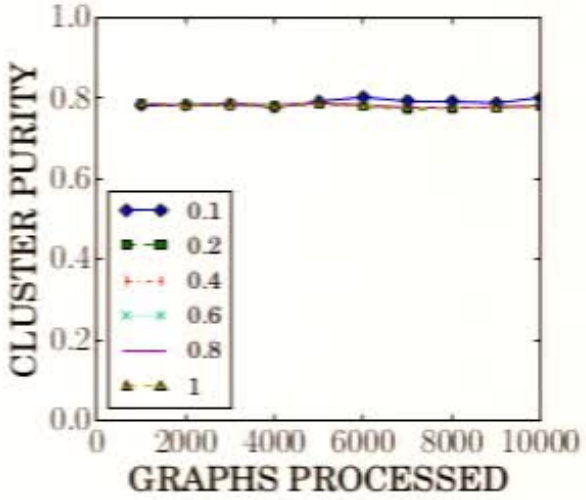
- Cora dataset consists of a publications in the Computer Science domain.
  - An edge between each pair of authors of the publication as the original structural edges
  - References and tokenised words from the publication title augmented as vertices into the graph.
- The Million Song Dataset consists of data for one million songs and can be linked with Last.fm data.
  - An edge between each song on an album.
  - Each song as a subgraph contains acoustic features and users who listened to the song augmented as vetices into the graph.
  - 50000 songs modelled as graphs with each graph having potentially over 100 vertices.

Evaluation

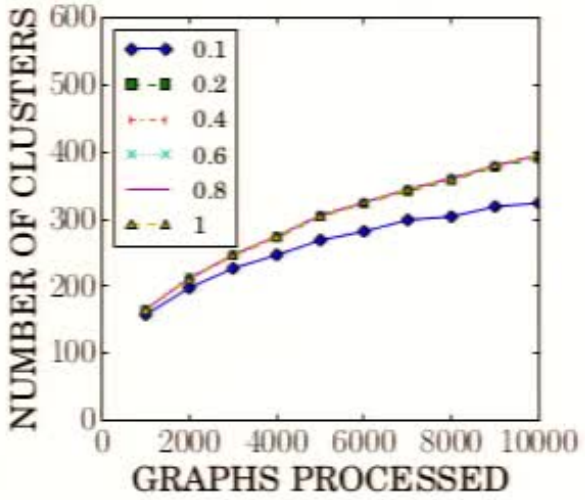
- We evaluate the quality of the clusters found by measuring average purity of the clusters.
- Cora: we evaluate purity of each cluster using authors most frequent publishing area.
  - 11 topics in total.
- MSD: We evaluate purity of each cluster using songs most common crowdsourced genre tag on Last.fm.
  - 10 of the most popular genres in total.
- To get an baseline of a performance we apply the Louvain method to the same augmented graphs.
  - The Louvain method is an offline community detection technique for graphs.
  - We found the average purity of clusters was comparable.

Results (Cora 1)

Effects of the merge threshold on the Cora graph.



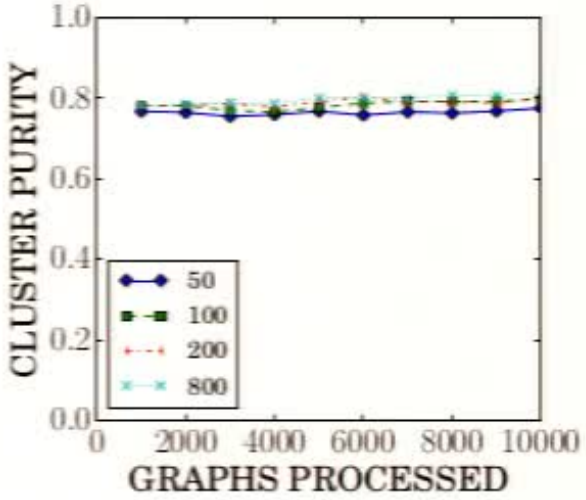
(a) Average Cluster Purity



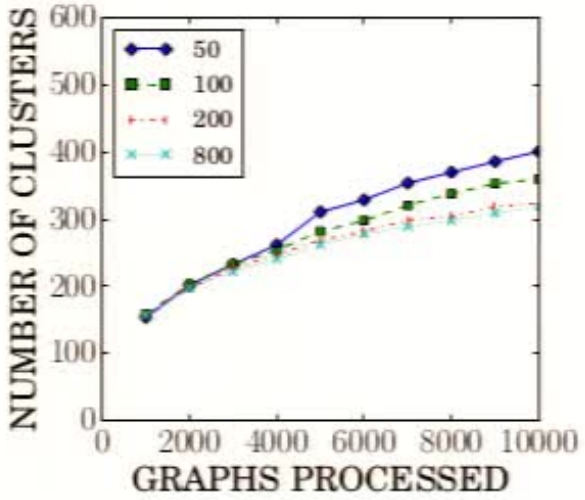
(b) Cluster Count

Results (Cora 2)

Effects of the *SoB* size on the Cora graph.



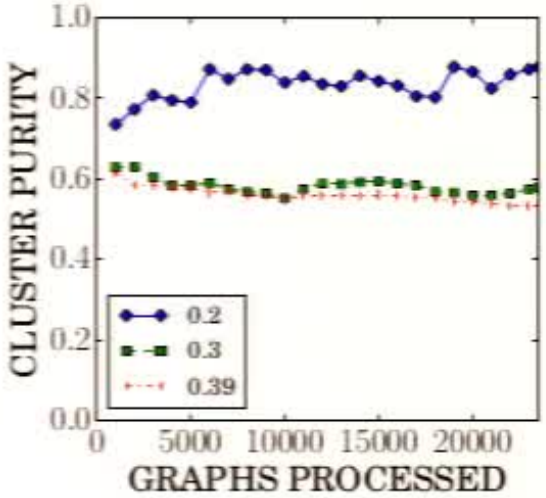
(a) Average Cluster Purity



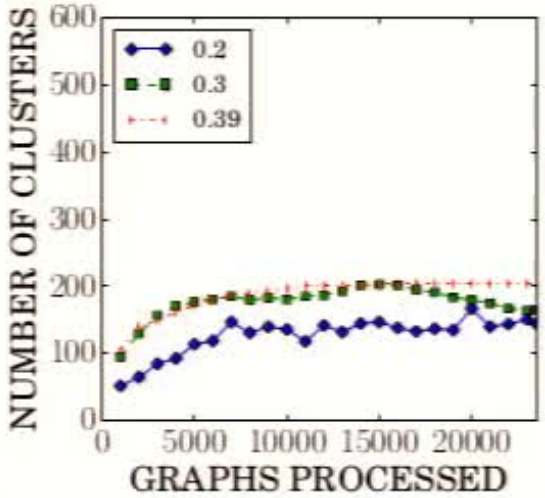
(b) Cluster Count

Results (MSD 1)

Effects of the merge threshold on the MSD graph.



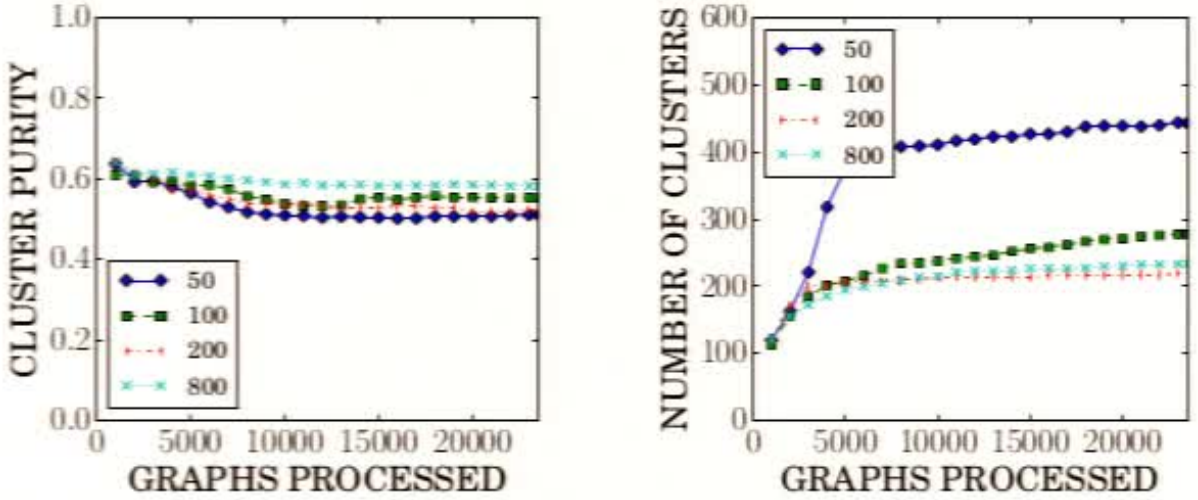
(a) Average Cluster Purity



(b) Cluster Count

Results (MSD 2)

Effects of the *SoB* size on the MSD graph.



(a) Average Cluster Purity (b) Cluster Count

Conclusion

- Results show promising clusters found from two real world graph streams.
- Integrated side information into graph stream for clustering using unified measure.
- Graph stream clustering without specifying number of clusters a priori.
- Merging algorithm which reduces the number of clusters over a similarity threshold.
- Vertex centric approach allows fine grained data stream mining.

Future work

- Efficiency is a concern with complex graphs and large numbers of clusters.
  - Each vertex and edge needs to be queried for each cluster. Time consuming with many clusters and large graphs!
  - Locality Sensitive Hashing as a means of prefiltering clusters for full search.
- Our clusters are limited to clusters formed on all of the attributes.
  - Can we find clusters across many subspaces in the graph stream?

Questions?

Thank you!