Mining Most Frequently Changing Component in Evolving Graphs[1]

AUTHORS: YAJUN YANG, JEFFREY XU YU, HONG GAO

WORLD WIDE WEB: 2014

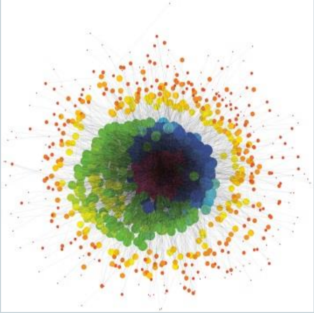
PRESENTED BY: MUHAMMAD AAMIR SALEEM

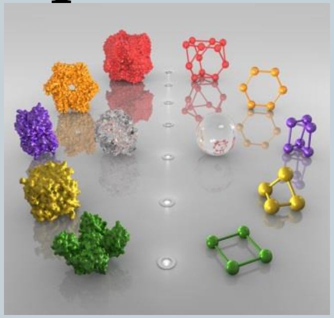
Agenda

- Motivation
- Introduction
- Related Work
- Modelling most frequently changing components
- Computation of Cumulated connectivity change
- Finding most frequently changing component
- Experiments
- Conclusion
- Q’s & A’s

Motivation

- Graphs are widely being used to model complex relationships
  - Road Networks
  - Social Networks
  - Author Collaborations etc.





- Graphs are dynamic and evolve with time
  - Changing relationships between users of social network
  - Changing road networks
    - Traffic jams causes
  - Changing financial network capture the unusual bursting / frauds etc.

Introduction

- Modelling frequently changing area in evolving graph is not trivial
  - Changing edges can be easily find
    - Not informative in analysing the graph
  - Changes occurs extensively in graph
    - Returning large area of graph is also not very useful

- Balance required between change frequency and area detected

Related Work

- Dynamic tensor analysis are proposed to incrementally summarize tensor streams[2]
- GraphScope [3] discovers communities in large and dynamic graphs as well as detects the changing time of communities
- Borgward [4] apply frequent-sub-graph mining algorithms to time series of graphs and extract sub-graphs that are frequent within set of graphs
- These existing work find a sub-graph sequence pattern
  - Embedding in a graph is frequent
  - Behaviour of these embedding are identical over time
- Liu et al. [5] proposed RWR to discover sub-graph that exhibit significant changes in evolving networks
  - Cannot quantify changes of a sub-graph in a time interval

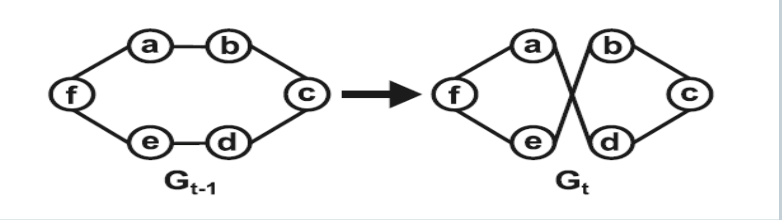
Modelling most frequently changing components (1/7)

- This study tackles the problem of discovering most frequently changing components (MFCC)
  - Dense connected components in evolving graphs

- Mining most frequently changing components
  - Evolving graph is series of undirected graph
    - $\mathsf{G} = (G_1, G_2, \cdots, G_{\|\mathsf{G}\|})$
    - $G_t=(V_t, E_t)$ is a snapshot at time t
    - Vertices remain constant in all of snapshot

Modelling most frequently changing components (2/7)

- **Measuring Changes between vertex pairs**
  - Changes between snapshots are measured by independent paths
  - Given $G_t=(V_t, E_t)$, u and v is said to b k-edge connected if removing any (k-1) cannot disconnect u and v, $(econ_t(u,v))$
  - Connectivity change between u and v from G_{t-1} to G_t is given by: $\delta_t(u,v) = |\mathrm{econ}_t(u,v) - \mathrm{econ}_{t-1}(u,v)|$
  - This relation does not capture all the changes
    - Following relation got zero connectivity change value

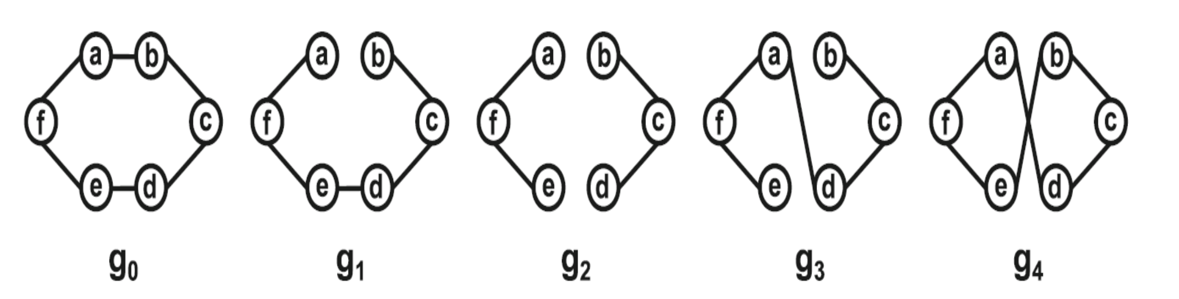


Modelling most frequently changing components (3/7)

- Cumulated frequency change
  - Keep track of edit operations i.e. edge insertions and deletions
  - Cumulated frequency change between two vertices is given by

$$\Delta_t(u, v) = \sum_{i=1}^{|\mathbb{g}_t|} \delta_i(u, v)$$

  - Example:



  - Cumulated frequency change in entire graph b/w 2 vertices is given by

$$\Delta(u, v) = \sum_{t=2}^{\|G\|} \Delta_t(u, v)$$

Modelling most frequently changing components (4/7)

- Modelling frequently changing components
  - Objective Function:

$$F(G_s) = \frac{\Delta(G_s) - \alpha(G_s)}{\beta(G_s)^{\gamma}}$$

where $0 \leq \gamma$ is parameter,

$$\Delta(G_s) = \sum_{u,v \in G_s, u \neq v} \Delta(u, v)$$

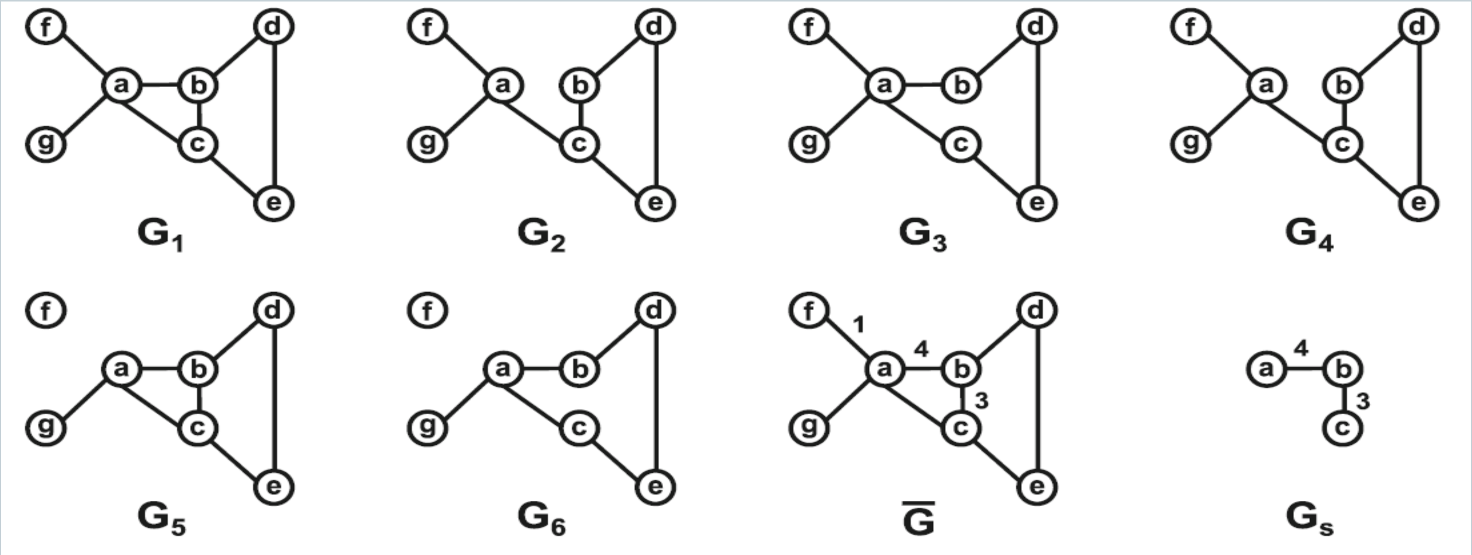
$$\alpha(G_s) = \sum_{(u,v) \in E_s, u \neq v} \alpha(u, v)$$

$$\beta(G_s) = |E_s| + \beta_c(G_s)$$

  - $\alpha(u, v) = \sum_{t=2}^{\|G\|} \alpha_t(u, v)$ is the number of times that the edge remains unchanged in G_s
  - $\beta_c(G_s)$: the number of pairs of vertices that cannot be connected by a path consisting of change edges
    - Less unchanged edges are expected to appear in G_s
  - cpath: path between two vertices so that every edge on it changed at least once
  - γ is to controll the size of G_s
    - *Importance of vertex pairs that cannot be connected*

Modelling most frequently changing components (5/7)

- Example:



  - Assuming $\gamma = 1$, $F(G_s) = ((a, b) - \alpha(a, b) + (b, c) - \alpha(b, c) + (a, c))/(|E_s| + \beta c(G_s)) = (3 + 1 + 4)/(2 + 0) = 4$
    - (a,b)= (1-2)+(2-1)+(1-2)+(2-1)+(2-2)=1+1+1+1+0=4
    - $\alpha(a, b)=1$

Modelling most frequently changing components (6/7)

- Properties of MFCC
  - cpath holds reflexive, transitive and symmetric property
  - All vertices of graph can be partitioned by cpath into equivalence classes called cpath components denoted by PC_i
    - *PC a subset of vertices in G in which every pair of vertices satisfies cpath*

$$\beta(G_s) = |E_s| + \sum_{i \neq j} |\mathrm{PC}_i| \times |\mathrm{PC}_j|$$

- So $F(G_s)$ objective function is given by

$$F(G_s) = \frac{\sum_{u,v \in V_s, u \neq v} \Delta(u,v) - \sum_{(u,v) \in E_s} \alpha(u,v)}{\left(|E_s| + \sum_{i \neq j} |\mathrm{PC}_i| \times |\mathrm{PC}_j|\right)^{\gamma}}$$

  - For a given subset of vertices in G, G_s with *max* $F(G_s)$ is a connected tree (*Lemma*)

Modelling most frequently changing components (7/7)

- Problem Statement:
  - Given an evolving graph and parameter the problem of discovering the most frequently changing component is to find a connected sub-graph such that $F(G_s)$ is maximized

Compute cumulated connectivity change (ccc)

- For every time stamp need to compute connectivity change of each of vertex pair edge operations time
  - Too costly, Naïve approach
- 2-way-ccc approach
  - k-edge connectivity is affected by 1 at most when an edge changes and order insensitive
  - For a snapshot compute k-edge-connectivity for every two difference vertices twice
    - G ->G_a(Graph at time *t-1*), G_b(Graph after deletion of vertices), G_c(Graph at time *t*)
    - Edges insertion + Edge deletion
    - P+q (single edge change p insertions + q deletions)>>2

Find Most Frequently Change Component(MFCC) (1/4)

- G_s is treated as weighted graph, every vertex (u , v) with $\alpha(u , v)$
  - A $G_s(V_s, E_s)$ is constructed as a tree for a given subset of vertices V_s
  - Maximizing $F(G_s)$ on V_s for a sub-graph is equivalent to minimizing $\alpha(G_s)$
  - Find minimum spanning tree on V_s in $\alpha(u , v)$
  - For a Given G_s to compute $F(G_s)$ we need
    - $\Delta(G_s)$, $\alpha(G_s)$, $\beta(G_s)$, (computed while ccc)

MFCC (2/4)

- Find-Max
  - Construct universal graph as a weighted graph
    - In addition we have connectivity change and PC components
  - Find connected sub-tree G_s of G with max $F(G_s)$
    - Start finding minimum spanning tree with min $\alpha(G)$
    - Start removing vertex from G, such that removed vertex should
      - Not miss the sub-tree with the max $F(G_s)$
      - Sub-tree must be connected
    - Partition tree is utilized to find minimum spanning tree

MFCC (3/4)

- Partition tree
  - Partition tree construction
    - Remove vertices that can not be included in final G_s with *max* $F(G_s)$
    - Maintain G_s that can possibly be the final answer
  - Partition tree traversal
    - Explore combination of maintained vertices and compute the final G_s with *max* $F(G_s)$

Experiments (1/2)

- Datasets
  - CAIDA anonymized internet traces datasets
    - Traffic traces of Chicago and Sanjose
      - Set of IP address as a subnet if they have the same p bits
      - Subnet as vertex
      - Edge if two IP address in two distinct subnets are connected
      - Evolving graph by generating graph at different time
  - Slahdost dataset
    - Technology related news website for specific user community
    - Users are nodes , edge represent user u agrees with user v's comment
  - Amazon dataset
    - Nodes represents products
    - Product co-purchased are linked by edge

Experiments (2/2)

- Parameters for Experiments
  - *$P(G_s)$: percentage of number of change edges over the total number of edges in G_s*
    - *Larger $P(G_s)$ the better resulting G_s found*
  - *$A(G_s)$: Average number of change of a edge in G_s*
    - *Larger $A(G_s)$ the better resulting G_s found*
  - *Ratio $\lambda = f(G_s)/f(G)$: density of cumulated connectivity change*
  - *Percentage τ : Number of edge changed in G_s over G*
    - The fraction of edge change time among vertices captured
  - $H(u) = \Sigma_v \Delta(u,v)$: the affected connectivity count of vertex *u*
    - Sort all vertices in descending order by *H(u)*
    - Measures that edge connectivity of vertices are affected frequently

Results (1/7)

- $P(G_s)$ and $A(G_s)$
  - When universal graph size is 20K, G_s is 422 & 388
    - $A(G_s)$ are always higher than other regions, $A(G - G_s)$
    - V'_s are always two vertices
      - Meaningless results

Dataset	$\lvert V_s \rvert$	$\mathcal{P}(G_s)$	$A(G_s)$	$A(\overline{G} - G_s)$	$F(G_s)$	$\lvert V'_s \rvert$
Chi-1	153	0.99	8.4	0.71	237.15	2
Chi-2	228	0.96	8.7	0.27	329.07	2
Chi-4	217	0.97	16.3	0.37	161.81	2
Chi-6	289	0.99	25.8	0.21	165.94	3
Chi-7	397	0.97	8.5	0.18	324.68	2
Chi-8	422	0.98	8.3	0.15	91.28	4
San-1	137	0.98	8.2	0.21	425.38	2
San-3	142	0.96	17.1	0.65	502.55	2
San-5	151	0.99	26.6	0.97	176.41	2
San-6	196	0.98	7.7	0.33	362.67	2
San-7	307	0.97	8.6	0.15	606.33	2
San-8	388	0.99	8.3	0.18	124.13	3
Slashdot	412	0.99	1.8	0.02	273.19	2
Slashdot	886	1	1.6	0.01	176.54	3
Amazon	537	0.99	3.7	0.06	338.98	2
Amazon	934	1	3.3	0.04	185.04	2

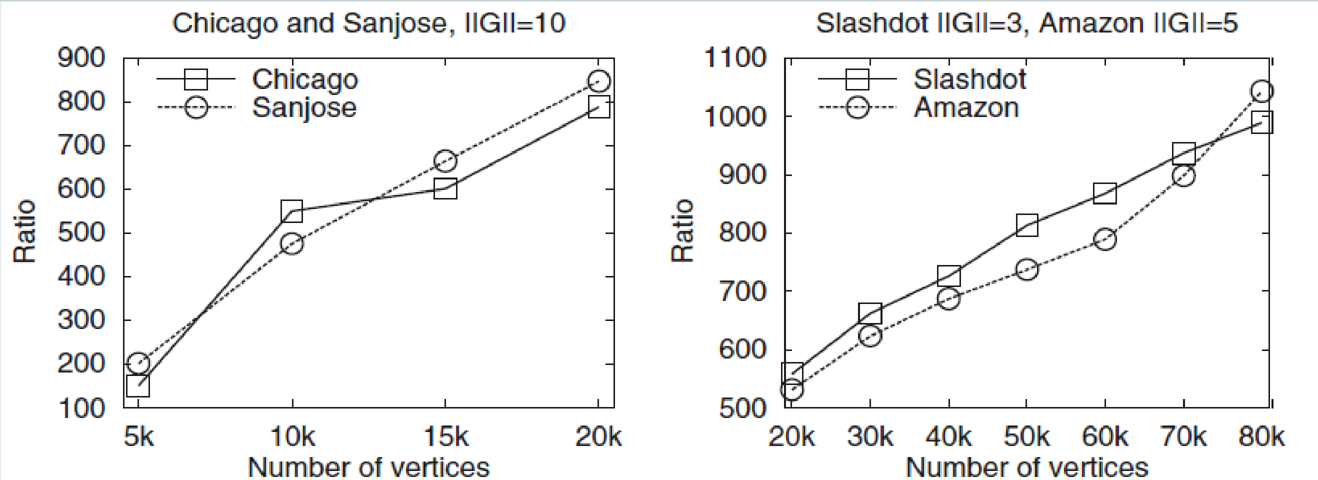
Results (2/7)

- *H(u)* order
  - Edge-connectivity among vertices in sub-graph G_s are affected most in whole graph G

	Top % of $\mathcal{H}(u)$ order				
Dataset	10 %	20 %	30 %	40 %	50 %
Chi-1-5K	0.85	0.93	0.99	1	1
Chi-2-10K	0.89	0.96	1	1	1
Chi-4-10K	0.88	0.96	1	1	1
Chi-6-10K	0.91	0.95	1	1	1
Chi-7-15K	0.95	0.99	1	1	1
Chi-8-20K	0.97	1	1	1	1
San-1-5K	0.82	0.94	0.98	1	1
San-3-5K	0.86	0.97	1	1	1
San-5-5K	0.87	0.95	1	1	1
San-6-10K	0.93	0.99	1	1	1
San-7-15K	0.94	0.97	1	1	1
San-8-20K	0.92	0.96	0.99	1	1
Slashdot-20K	0.96	1	1	1	1
Slashdot-80K	0.98	1	1	1	1
Amazon-20K	0.94	1	1	1	1
Amazon-80K	1	1	1	1	1

Results (3/7)

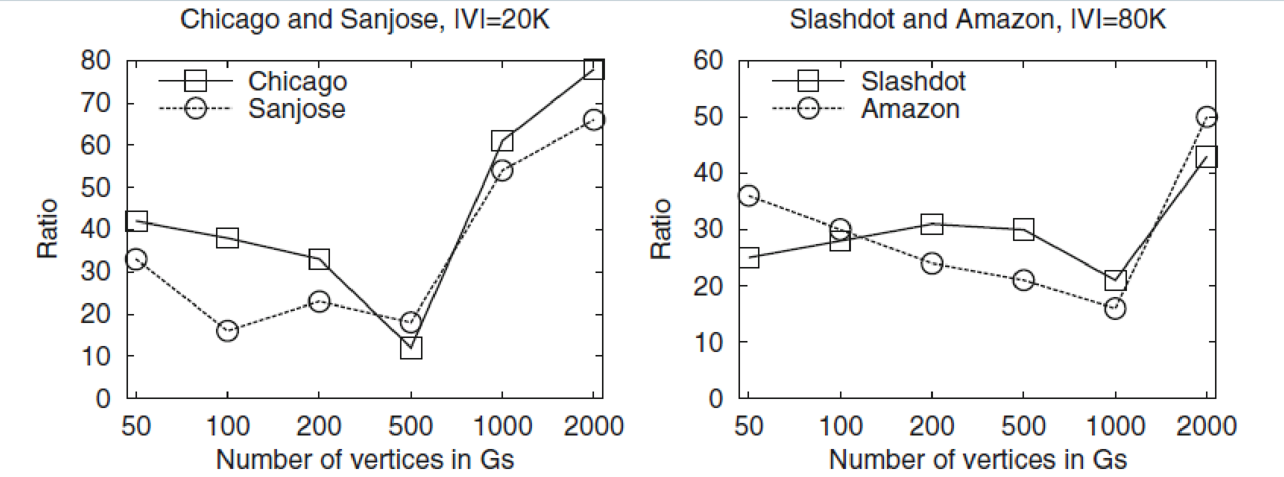
- Ratio λ *of* G_s *to* G
  - Density of commulated connectivity change
  - Ratio increases while the vertex size increases
    - Size of vertex reaches to 20k ratio increase up to 800 times
  - Confirms effectiveness of approach for large networks



(a) Chicago and Sanjose (b) Slashdot and Amazon

Results (4/7)

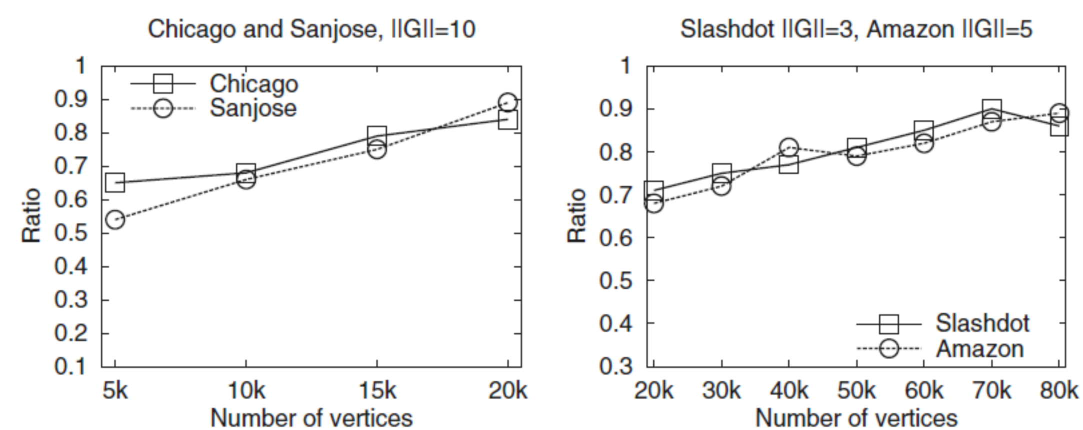
- λ' of G_s to random graph G_r
  - λ' is high when *r> 1000* and less when *r=500*
    - Size of G_r is close to G_s when *500>r>200*
  - Density of *ccc* of G_s is always *10* times larger than random graph



(a) Chicago and Sanjose (b) Slashdot and Amazon

Results (5/7)

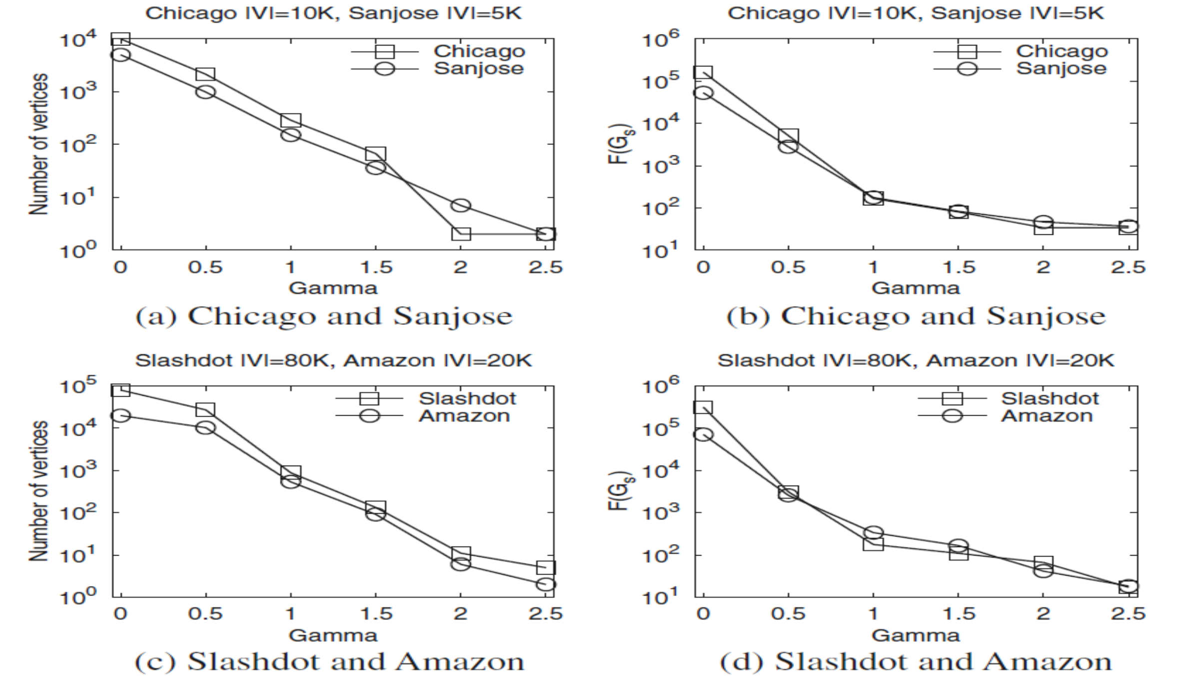
- *Percentage τ of G_s to G*
  - Curve *of τ* increas marginally while the size of vertices increase from *5K to 20K*
  - At size of *20K τ* is *80%* larger
    - *80% of edge change times in G is captured by a small fraction of vertices*



(a) Chicago and Sanjose (b) Slashdot and Amazon

Results (6/7)

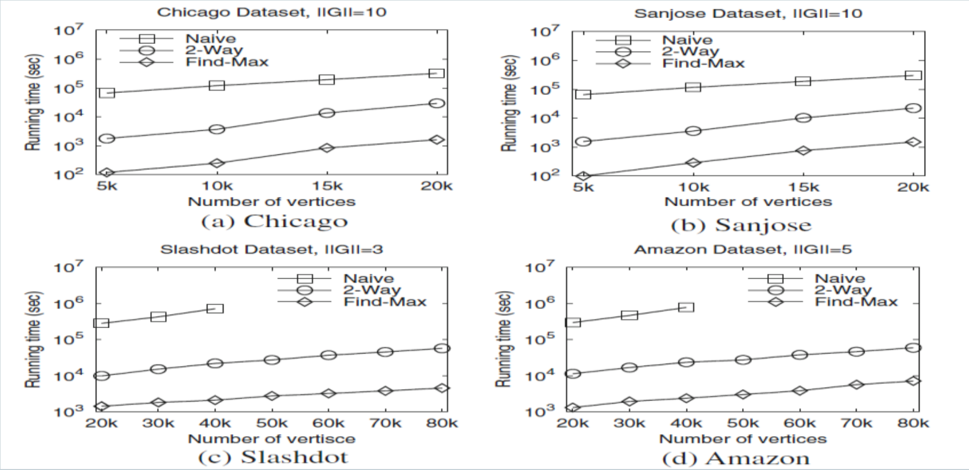
- Size of G_s decreases with increasing size of γ
  - $F(G_s)$ also decreases with increasing γ
  - Size of G_s can be controlled by γ



(a) Chicago and Sanjose (b) Chicago and Sanjose
(c) Slashdot and Amazon (d) Slashdot and Amazon

Results (7/7)

- Experiment 7
  - *2-way-ccc* is nearly 100 times faster than *Naïve-ccc*
  - *Naïve* cannot perform is size greater than *40k*
- Experiment 8
  - *Find-Max* time is order of *10²*
  - State excellent scalability with increasing size



Conclusion

- Most changing sub-graph in evolving graph is found
- Objective function to find a connected subgraph G_s in G with $maxF(G_s)$ is proposed
  - Cumulated connectivity change
  - Effectively used to identify the most changing sub-graph with small number of unchanged edges included
- Two algorithms are proposed to compute *ccc*
- Novel algorithm to identify the sub-graph with $max\ F(G_s)$
- Effectiveness and efficiency proved by experiments

References

[1] Mining most frequently changing component in evolving graph, Yajun Yang, Jeffrey Xu Yu, Hong Gao, World Wide Web Journal, volume17: pp 351-376 (2014)

[2] Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: dynamic tensor analysis. In: KDD (2006)

[3] Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: Graphscope: parameter-free mining of large time-evolving graphs. In: KDD (2007)

[4] Bifet, A., Gavaldà, R.: Mining frequent closed trees in evolving data streams. Intell. Data Anal. 15(1), 29–48 (2011)

[5] Liu, Z., Yu, J.X., Ke, Y., Lin, X., Chen, L.: Spotting significant changing subgraphs in evolving graphs. In: ICDM (2008)

- Questions ??
- Suggestions??
- Comments??