Pipeline design & implementation for the analysis of Natural Graphs

Graph partitioning and processing framework

Presented by,
Kunal Arora
Amritansh Sharma

Agenda

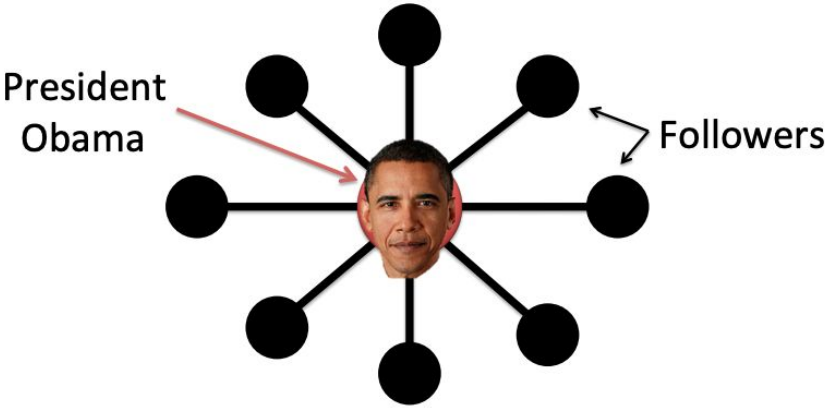
- ❖ **Introduction**
  - ➢ What are Natural Graphs
  - ➢ Power-Law degree distribution
- ❖ **Research**
  - ➢ Motivation
  - ➢ Goal/Problem definition
  - ➢ Tools
  - ➢ Methodology
  - ➢ Experiments
- ❖ **Conclusion**
  - ➢ Summary of Contribution
  - ➢ Future work
- ❖ **References**

Introduction

Introduction: What are Natural Graphs

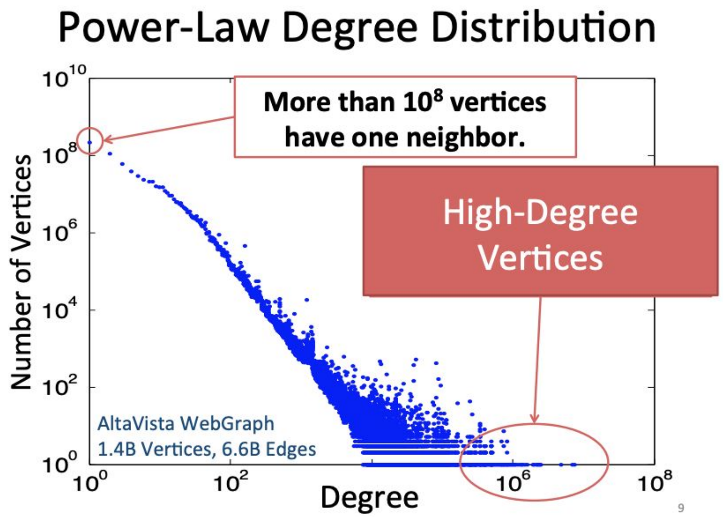
- ❖ Graphs derived from natural phenomenon

- ❖ Presence of High-Degree vertices
  E.g. Twitter Follower graph



Introduction: Power-Law degree distribution

- Few vertices with a very high-degree of vertices.
- Large number of vertices with only one or few neighbors
- Scaling factor : **alpha**



$$p(x) = \frac{\alpha - 1}{x_{\min}} \left(\frac{x}{x_{\min}} \right)^{-\alpha}$$

Research

Research: Motivation

- Natural Graphs represent an important class of graph topology given their presence in real-world graph dataset (particularly social network datasets).

- As a result, it is important to develop a set of methods to study and analyse Natural graphs to uncover their properties experimentally.

Research: Goal/Problem definition

- ➔ Study of **real-world Natural graphs** like, Twitter, Facebook social graphs
- ➔ Implement strategy to calculate the **Power-Law factor, alpha** (α) for these graphs
- ➔ Implement strategy to **artificially generate Natural graphs** with same power-law distribution as real-world graphs
- ➔ **Benchmark** the artificial and real-world graphs:
  - ◆ Partitions
  - ◆ Edgecut
  - ◆ Visualization of graph partitions
  - ◆ Graph processing algorithms (Page rank and Triangle counting)

Research: Tools

- **NetworkX-Metis**: NetworkX-Metis is a Python package for the creation, manipulation and study of the structure, dynamics, and functions of complex graphs

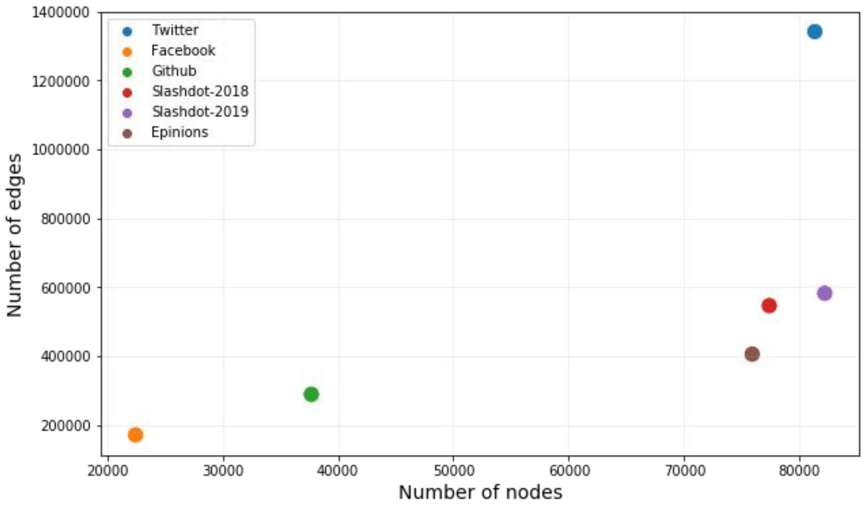
- **GraphX**: GraphX run on top of spark and uses spark configuration. This system is used to run mainly very large graphs in a distributed framework.

Research: Methodology (Real-world datasets information)

Real-world graphs

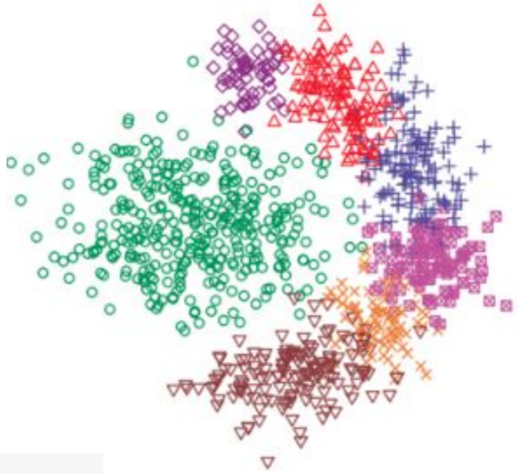
- Twitter
- Facebook
- Github
- Slashdot-2018
- Slashdot-2019
- Epinions

	Twitter	**Facebook**	**Github**	**Slashdot-2018**	**Slashdot-2019**	**Epinions**
Type	Directed	Undirected	Undirected	Directed	Directed	Directed
Nodes	81,306	22,470	37,700	77,360	82,168	75,879
Edges	1,768,149	171,002	289,003	905,468	948,464	508,837



Research: Methodology (Artificial graph generation method)

- Generate Power law sequence: **sequence**
- Use **sequence** as basis to define Random Partition Cluster graph



```python
def generate_synthetic_powerlaw_graph(alpha, p_in, p_out):
    sequence = nx.utils.powerlaw_sequence(100, alpha)
    sequence = [int(x) for x in sequence]
    synthetic_graph = nx.random_partition_graph(sequence, p_in, p_out)
    return synthetic_graph

sample_graph = generate_synthetic_powerlaw_graph(alpha=3.0, 0.25, 0.01)
```

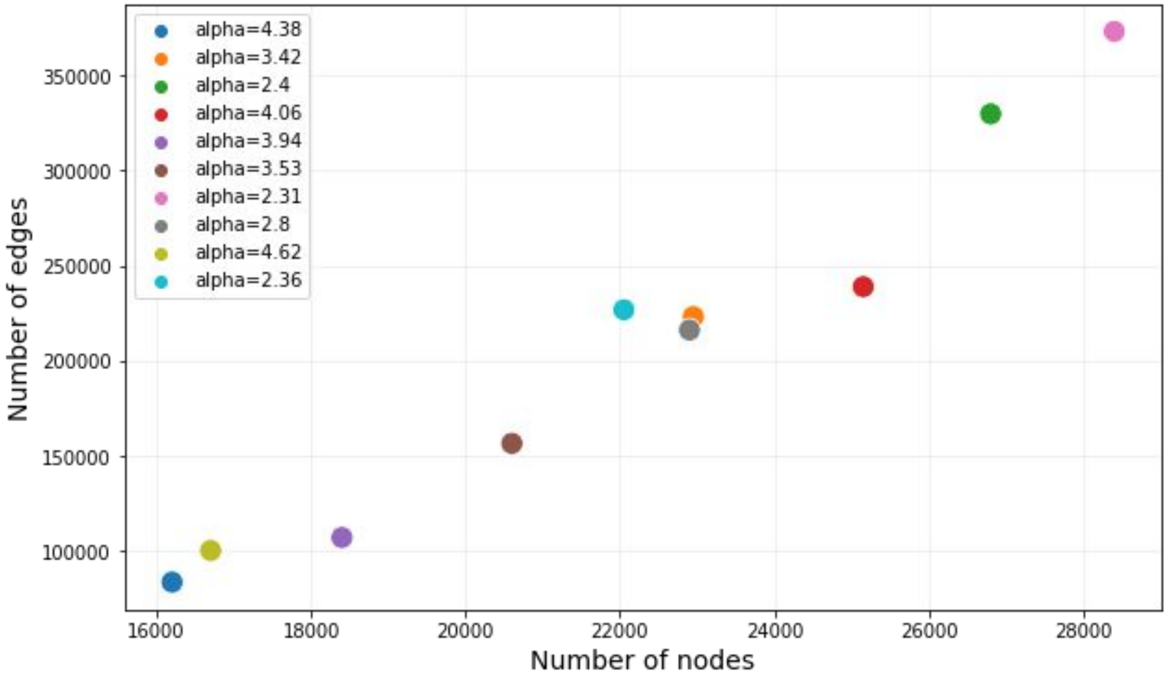
Research: Experiments (Generated artificial graph datasets information)

	4.38	**3.42**	**2.4**	**4.06**	**3.94**	**3.53**	**2.31**	**2.8**	**4.62**	**2.36**
Nodes	16,200	22,950	26,800	25,150	18,400	20,600	28,400	22,900	16,700	22,050
Edges	83,900	223,200	329,600	238,900	107,300	156,700	372,900	216,200	100,500	226,800

Artificially-generated graphs:

- Number of nodes, edges and the structure of graphs was varied
- Graphs generated with resulting alphas in the range of 2 to 5

Research: Experiments (Generated artificial graph datasets information)



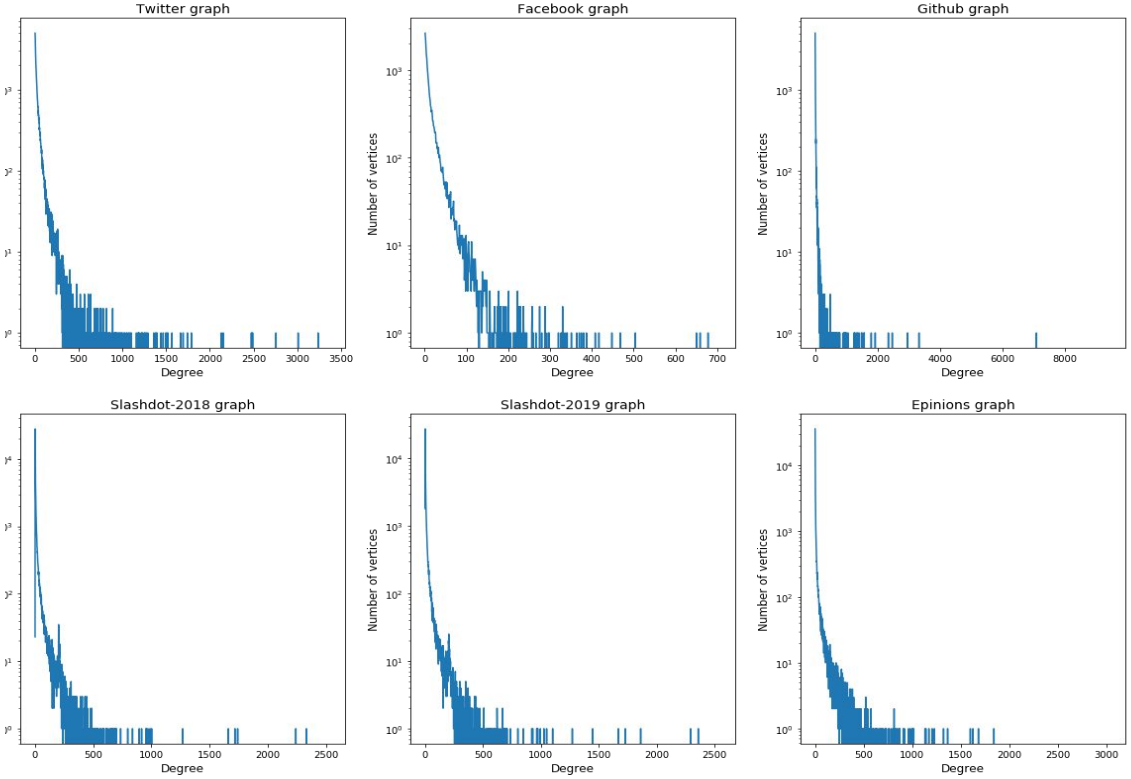
Scatter plot between number of nodes and edges for artificially generated graphs

Research: Methodology (Scaling exponent alpha(α) calculation)

- Estimating the scaling exponent (alpha) for Power-law distribution
- Maximum likelihood estimates are most reliable
- Python **powerlaw** package implementation

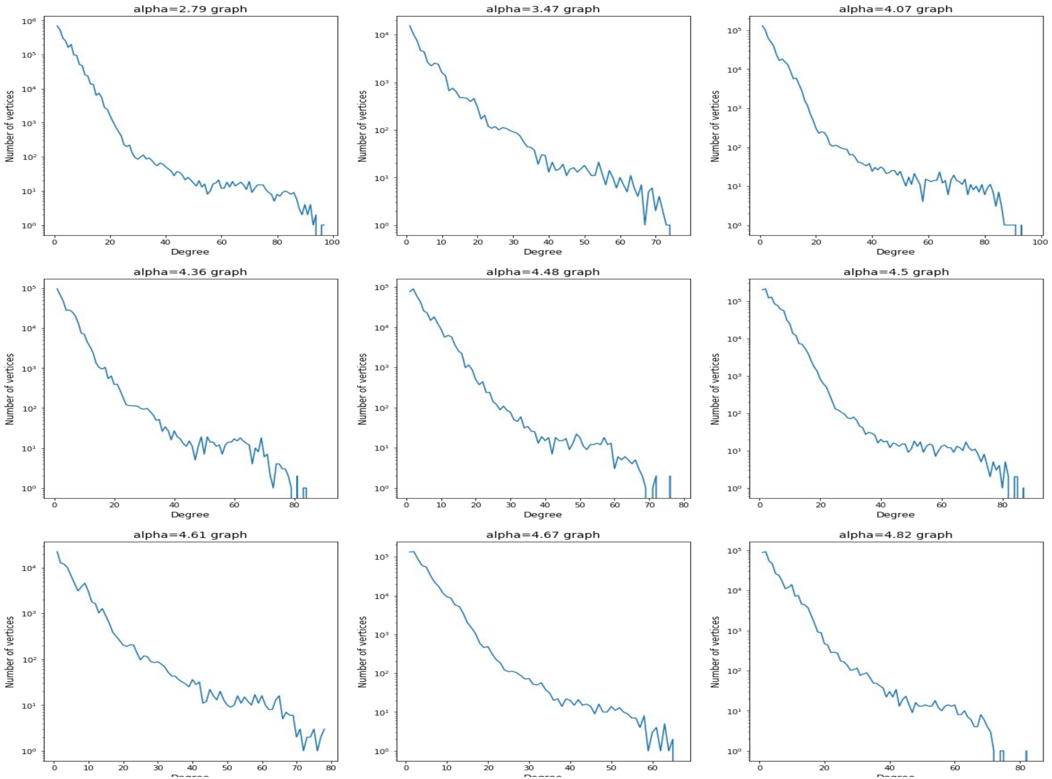
```
def get_power_law_coefficient(graph):
    # obtain degree distribution
    degrees = []
    for node in graph.nodes_iter():
        degrees.append(len(graph.neighbors(node)))
    # fit power law distribution
    dist = powerlaw.Fit(num_nodes)
    # obtain scaling factor
    return dist.power_law.alpha
```

Research: Experiments (Power-law degree distribution for real-world graphs)



Power-Law degree distribution for real-world graphs

Research: Experiments (Power-law degree distribution for artificial graphs)



Power-law degree distribution for artificially generated graphs

Research: Methodology (Partitioning strategy & edge-cut)

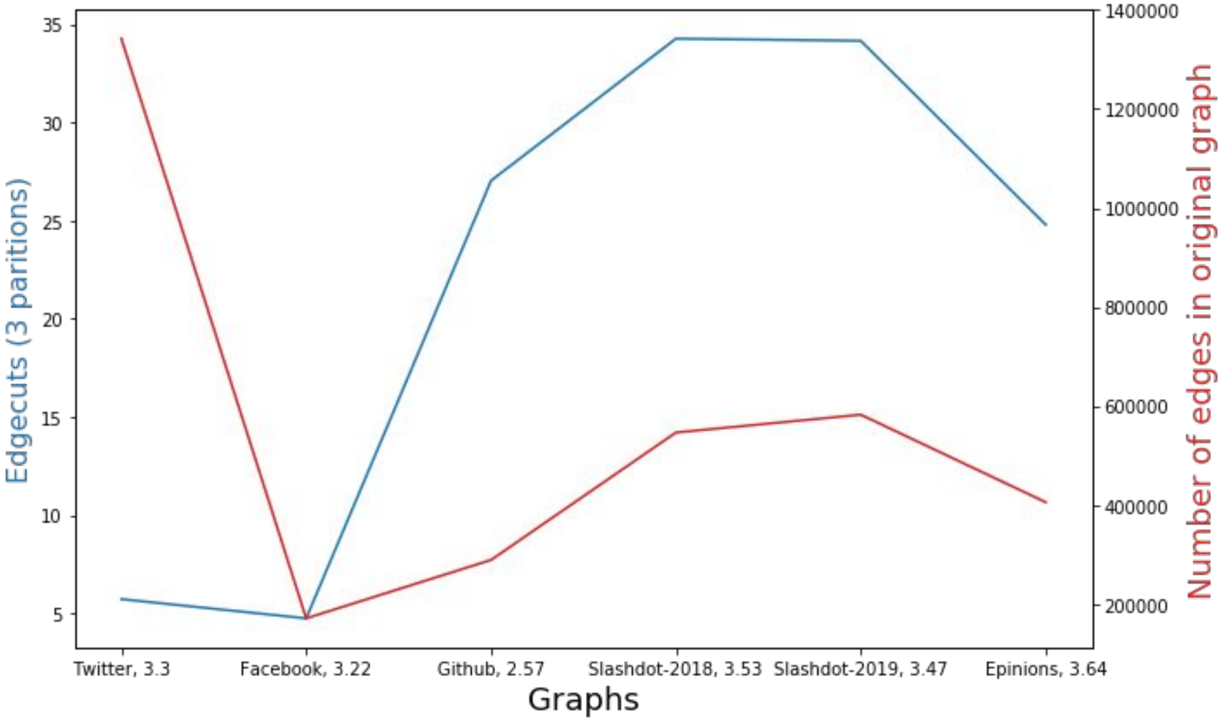
- Multilevel paradigm for partitioning (**METIS**)
- Visualisation using **networkx**

```
def partition_graph(G, num_partitions):
    (edgecuts, parts) = metis.part_graph(G, num_partitions)
    color_map = []
    for part, node in zip(parts, G):
        color_map.append(colors[part])
    pos = nx.spring_layout(G)
    nx.draw_networkx(G, pos, node_color=color_map, with_labels=False,
                     node_size=25, width=0.1)

    return round((float(100*edgecuts))/G.number_of_edges(),2),
     G.number_of_edges()
```

Research: Experiments (Edgecut for real-world graphs)

- Plotted Edgecut (for 3 partitions) along with total number of edges for the real-world graphs

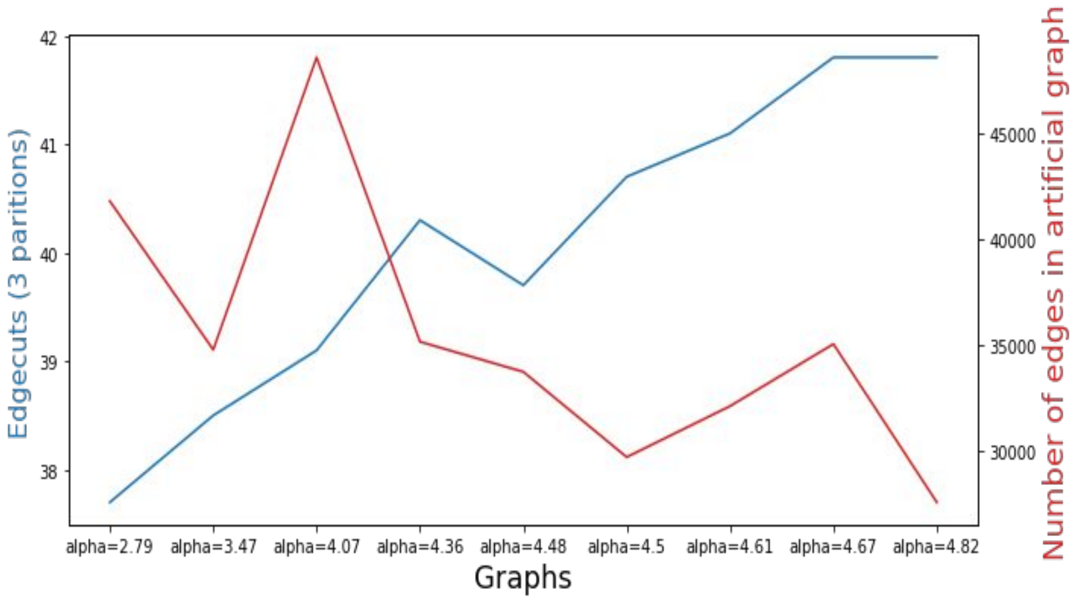
- Higher the number of edges, higher is the Edgecut except, Twitter graph.



Research: Experiments (Edgecut for artificial graphs)

- Plotted Edgecut (for 3 partitions) along with total number of edges for the artificial-graphs

- For higher alpha (means less natural the graph), with less number of edges, the edgecut might be high as observed for alpha=4.82



Research: Methodology (Sampling natural graphs)

- Problems with Visualising Large Graphs
  - Overlapping nodes (if more than **1000 nodes**)
  - System Out-of-Memory (if more than **100000 edges**)

- Solution: Representative Sampling of Natural Graphs

- Our Sampling Algorithm:
  - Choose largest connected component
  - Perform depth-limited breadth first traversal of this component
  - Return sub-graph induced on visited nodes

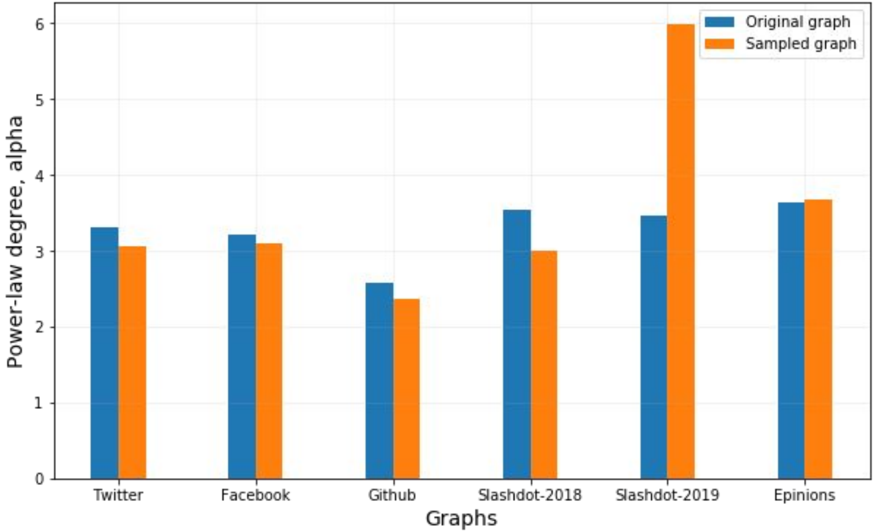
Research: Methodology (Sampling graphs: implementation)

```
def sample_graph(graph, graph_name, max_nodes=1000):
    # get largest connected component
    graph = max(nx.connected_component_subgraphs(graph), key=len)

    # perform bfs
    source_node = list(graph.nodes_iter())[0]
    bfs_result = nx.bfs_successors(graph, source=source_node)
    subgraph_nodes = set()
    q = Queue.Queue()
    q.put(source_node)
    while not q.empty() and len(subgraph_nodes)<=max_nodes:
        current_node = q.get()
        subgraph_nodes.add(current_node)
        for neighbour in bfs_result.get(current_node, []):
            q.put(neighbour)

    # obtain induced subgraph
    sampled_graph = graph.subgraph(subgraph_nodes)
    return sampled_graph
```

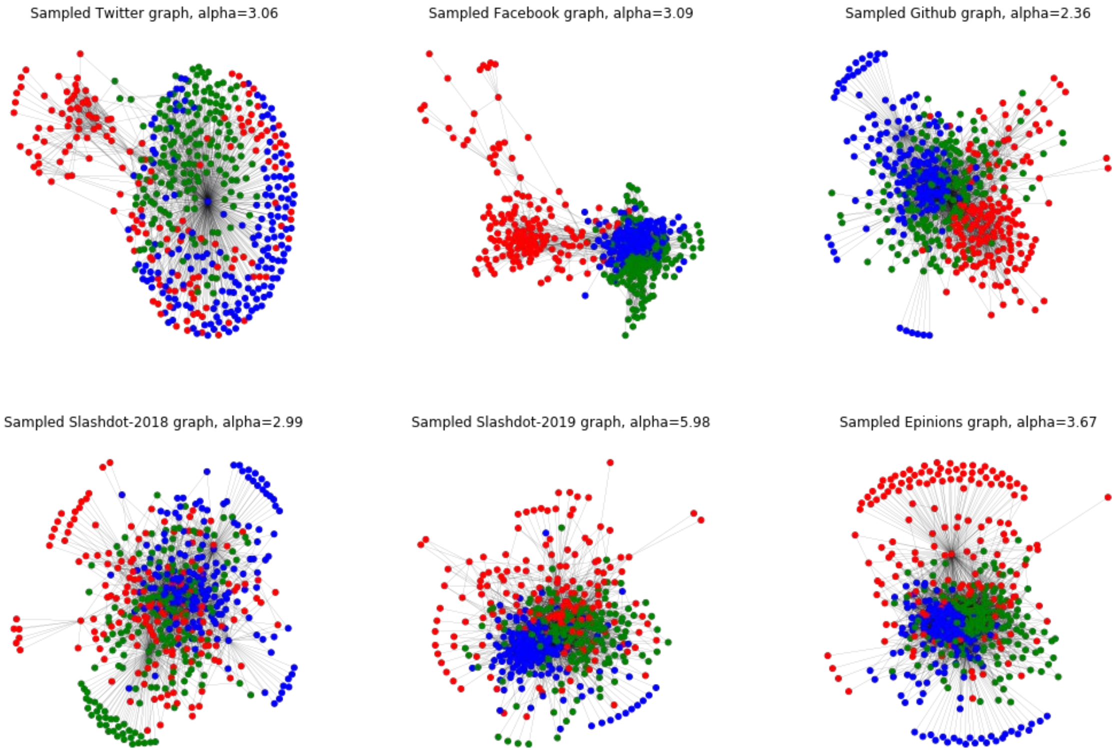
Research: Experiments (Sampling natural graphs)

	Twitter	Facebook	Github	Slashdot-2018	Slashdot-2019	Epinions
Original alpha(α)	3.3	3.22	2.57	3.53	3.47	3.64
Sampled alpha(α)	3.06	3.09	2.36	2.99	5.98	3.67

Significant results as the sampled alpha is almost the same as original for most of the graphs except, Slashdot-2019



Research: Experiments (Visualization of sampled real-world graphs)



Graph visualization of sampled real-world natural graphs for 3 partitions

Research: Experiments (Visualization of sampled artificial graphs)

Graph visualization of sampled artificial graphs for 3 partitions

Research: Methodology (Graph processing algorithms)

- PageRank algorithm: PageRank is an algorithm that measures the transitive influence or connectivity of nodes. For pagerank, not only the number of incoming links that is important, but also the importance of the pages behind those links.

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

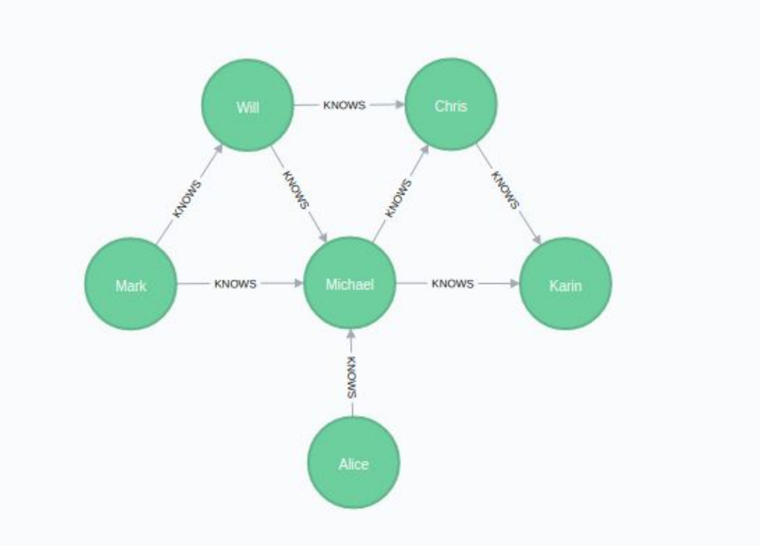
where,

- we assume that a page A has pages T1 to Tn which point to it (i.e., are citations.
- d is a damping factor which can be set between 0 and 1. It is usually set to 0.85.
- C(A) is defined as the number of links going out of page A.

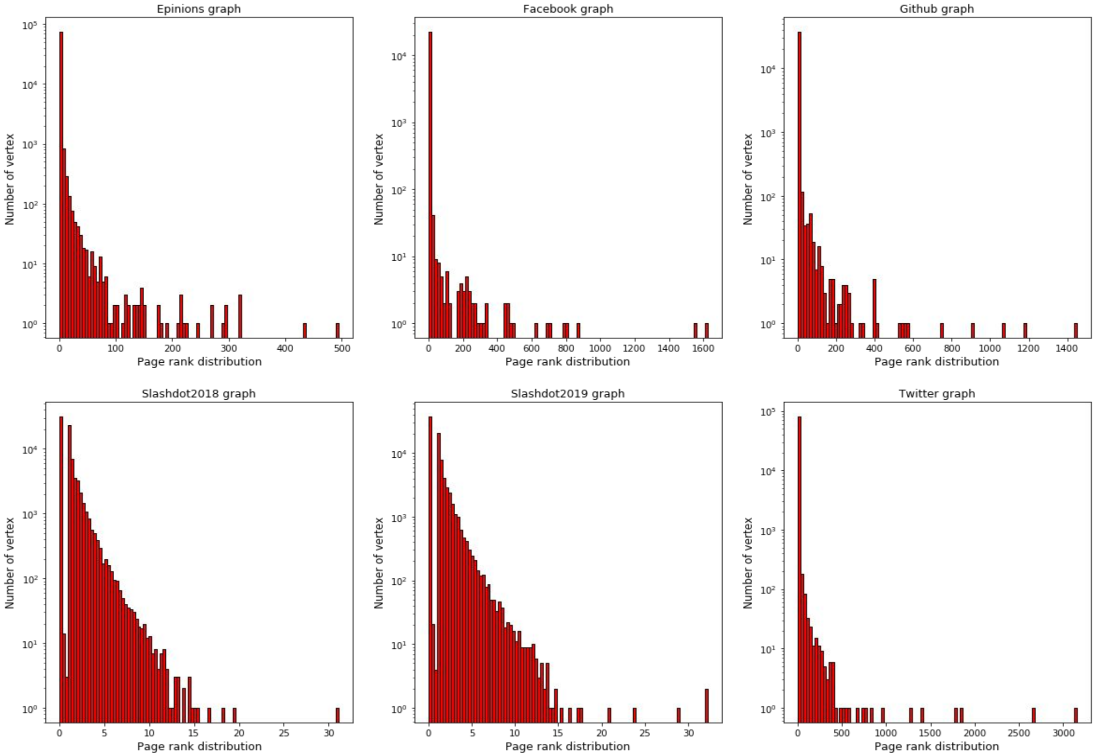
Research: Methodology (Graph processing algorithms)

- Triangle Counting: Triangle counting is a community detection graph algorithm that is used to determine the number of triangles passing through each node in the graph. It can also be used to determine the stability of a graph, and is often used as part of the computation of network indices, such as the clustering coefficient.

Michael is part of 3 circles



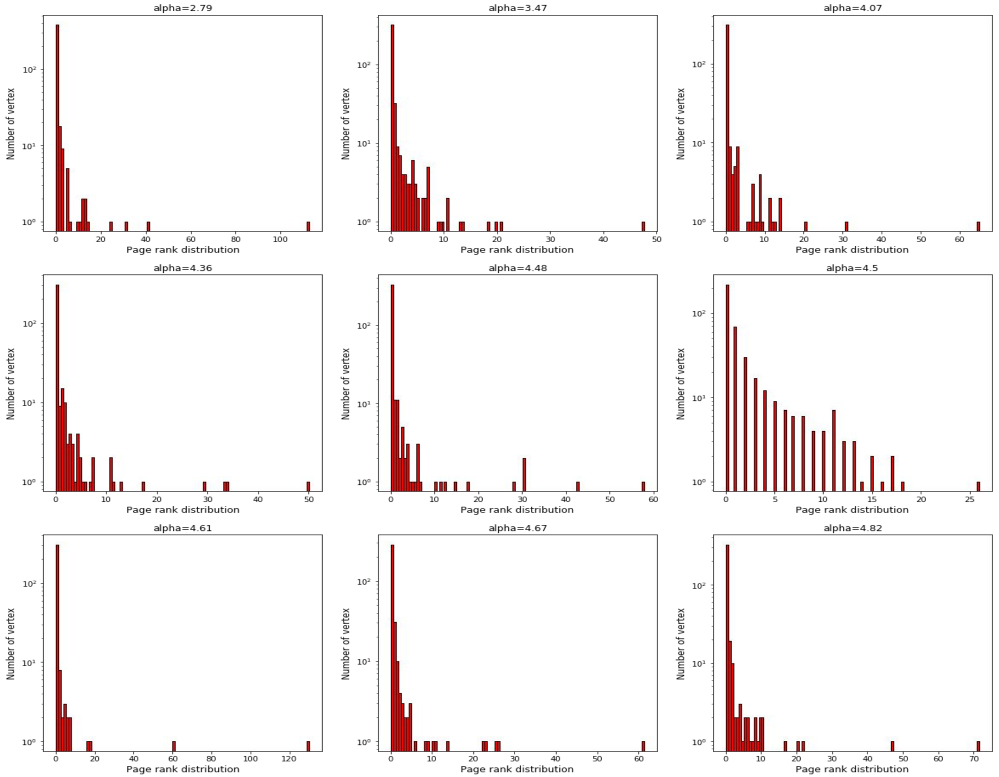
Research: Experiments (PageRank for real-world graphs)



Page rank distribution of real-world graphs

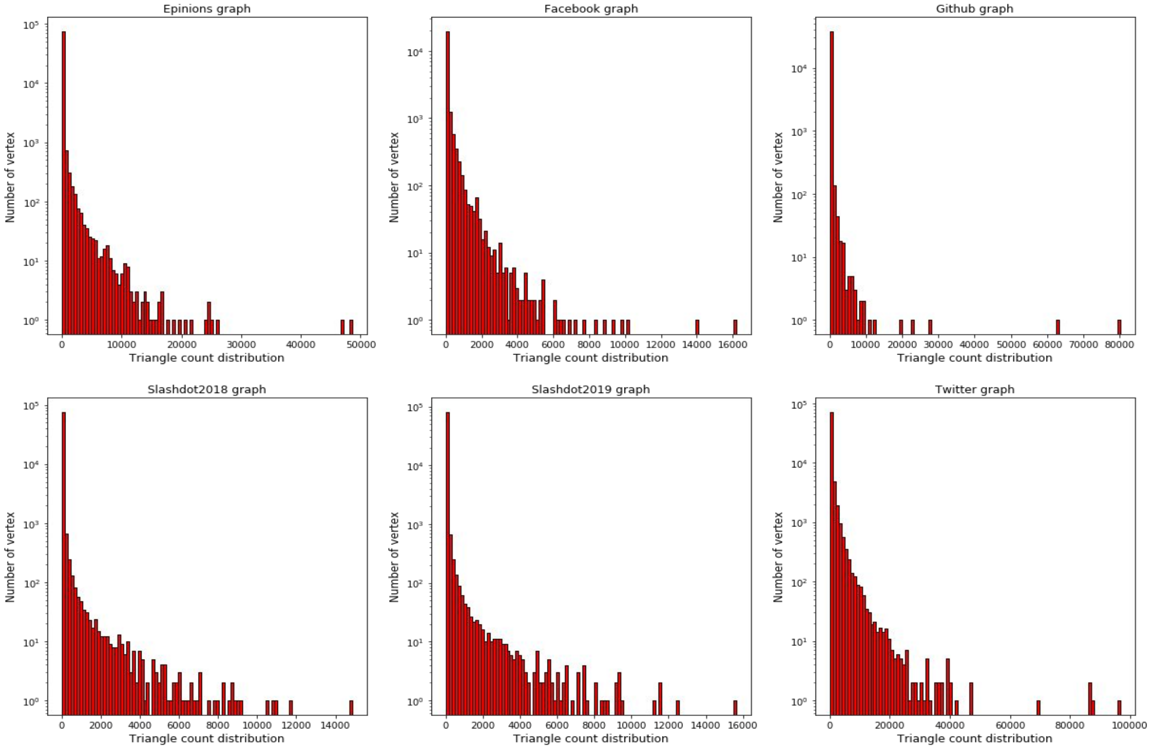
Research: Experiments (PageRank for artificial graphs)

Significant results have been achieved for artificial graphs, comparable to real-world graphs for similar alphas.



Page rank distribution of artificial graphs

Research: Experiments (Triangle counting for real-world graphs)

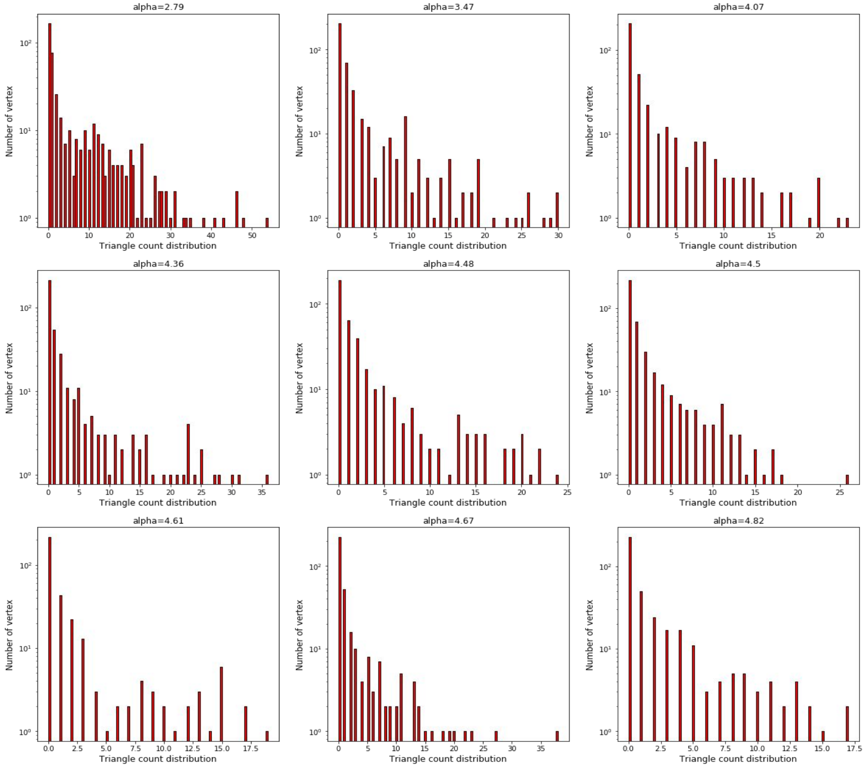


Triangle counting distribution for real-world graphs

Research: Experiments (Triangle counting for artificial graphs)

Triangle counting results are not very satisfactory for artificial graphs.

This is because there is not a few high-degree vertices but, rather a lot of vertices who have similar degrees and hence, all the triangle counts are well distributed.



Triangle counting distribution for artificial graphs

Conclusion

Conclusion: Summary of Contribution

We have designed and developed a pipeline aiding the study of Natural graphs

- **testing of Natural graphs** through scaling factor calculation and degree distribution plots
- **artificially generating Natural graphs** of different power-law scaling factors as well as edge-density
- **partitioning** Natural graphs (we provide implementation with METIS it can be replaced by any available graph partitioning algorithms)
- **visualisation** of partitioned Natural graphs
- a **sampling method** for visualising large Natural Graphs
- running **graph processing algorithms** on Natural Graphs on GraphX.

Conclusion: Future Work

- Confidence of Power Law degree distribution
  - Currently visually determined
  - Implement loss function between best fit and observed degree distribution
  - Normalize loss function by graph size

- Distributed graph processing
  - Extension of current pipeline

References

[1] Angles, Renzo. (2012). A Comparison of Current Graph Database Models. Proceedings - 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW 2012. 171-177. 10.1109/ICDEW.2012.31.

[2]Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., & Czajkowski, G. (2010, June). Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 135-146). ACM

[3] Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., & Hellerstein, J. M. (2012). Distributed GraphLab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, *5*(8), 716-727.

[4] Gonzalez, J. E., Low, Y., Gu, H., Bickson, D., & Guestrin, C. (2012). Powergraph: Distributed graph-parallel computation on natural graphs. In *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)* (pp. 17-30).

[5] Kim, J., & Wilhelm, T. (2008). What is a complex graph?. *Physica A: Statistical Mechanics and its Applications*, *387*(11), 2637-2652.

Thank You
Q & A

Overleaf link of report: https://www.overleaf.com/3263735487wwdntzwmfypn