GFP-X: A Parallel Approach To Massive Graph Comparison Using Spark

Stephen Bonner, John Brennan, Georgios Theodoropoulos, Ibad Kureshi and Andrew Stephen McGough

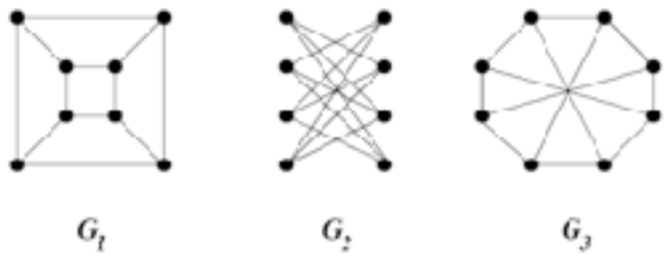
School of Engineering and Computing Sciences
Durham University, Durham, UK

Motivating Examples

- **Graph Comparison:** Given a set of *k* Graphs, of different sizes and no overlaps in vertices or edges, how can we quickly assess 'similarity' between them, without resorting to solving the vertex-correspondence problem?

- We need a definition of 'similarity' between two graphs. Needs to be more than graph isomorphism!

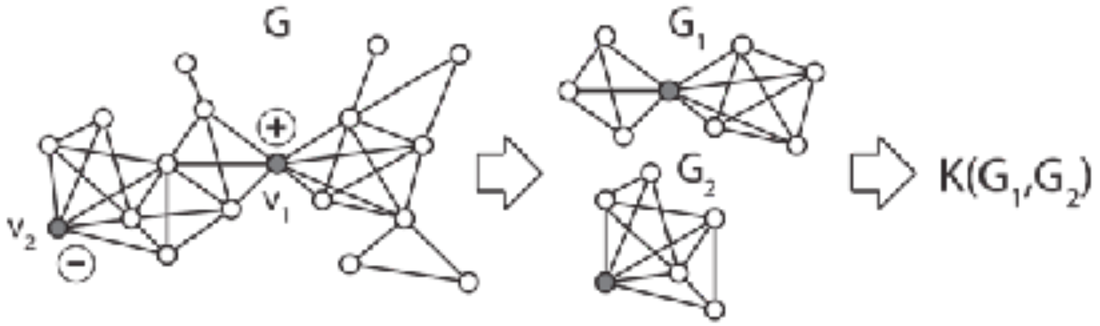
- For this work as we assume no common vertex labels or correspondence, so we consider the topological similarity between two, or a series of graphs.



Previous Approaches

- **Previous Approaches -**
- **Graph Kernels:** The traditional way for graph comparison, broadly a family of methods which compute an inner product on a pair of graphs. There are a large family of Graph Kernels including: Random Walks, Shortest Path, Sub-Tree and Graphlet (SubGraph) Kernels.
- These graph kernel methods have been used for graph comparison very successfully, but there are questions about their scaleability to truly massive graphs.

- **Feature Extraction Methods:** Approaches that extract discriminative features from graphs and using them for comparison. Less work in this area than Graph Kernels, but they have advantages over GK methods as they can scale better. The challenge comes in the correct feature selection.
- The current approach from the literature which we found scales best is *NetSimile.*



Approach Requirements

- **The GFP-X approach is driven by the following requirements:**
- **1) Scalability -** Highly scalable to massive graphs of millions of vertices/edges, and capable of computing the similarity in a finite time.
- **2) Sensitivity to Graph Size** - Taking the size and order of the graphs into consideration.
- **3) Sensitivity to Similar Topologies** - Detecting the difference between graphs which are highly structurally and topologically similar.
- **4) Label Free -** Able to perform comparisons without requiring labeled datasets, although the approach should still function when they are available.
- **5) Low Number of User Defined Parameters -** A minimum number of user defined parameters should be required to measure graph similarity.

Approach Overview

- **We introduce the Graph Fingerprint method for massive graph comparison.**

- The approach is comprised to two distinct phases: The graph fingerprint generation and the comparison of these fingerprints.

- The fingerprint generation is further comprised of the following stages:

- **Vertex Level Feature Extraction**
- **Vertex Level Feature Creation**
- **Global Level Feature Extraction**

- The fingerprint comparison is also broken down into further stages:

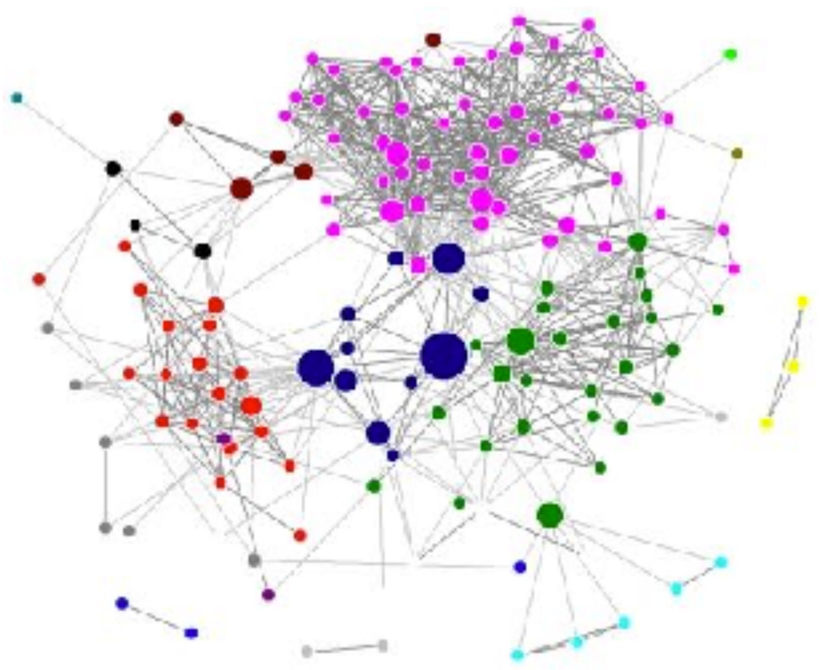
- **Vertex Level Comparison**
- **Global Level Comparison**
- **Final Similarity Score Generation**

Approach Overview

- So what features are we extracting from each graph?

- Firstly we extract a range of global features to capture details about the size and connectivity of each graph.

- *Graph order and number of edges*
- *Number of triangles*
- *Global clustering coefficient*
- *Maximum total degree*
- *Number of components.*

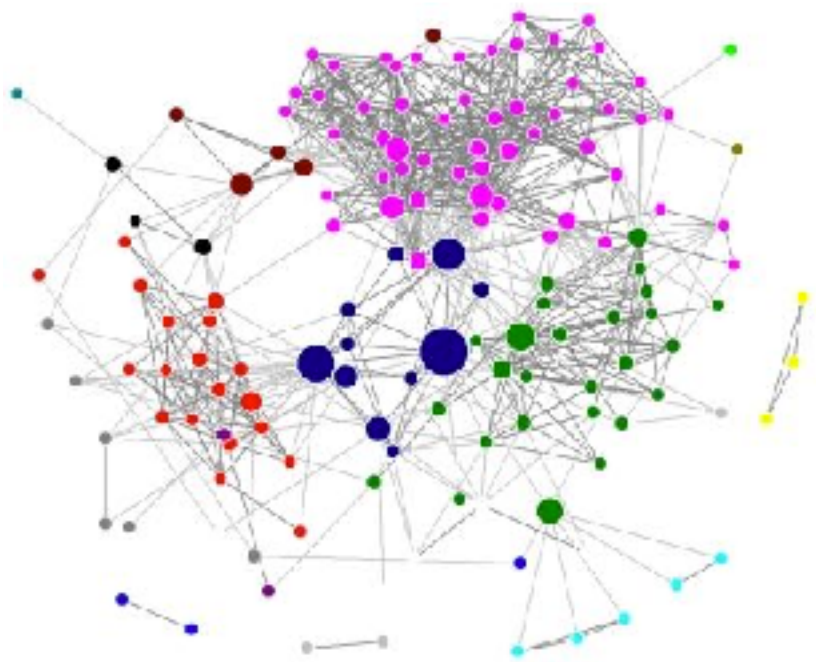


Approach Overview

- Secondly, a range of micro level features are extracted from each vertex within a graph.

- *Eigenvector Centrality Value*
- *PageRank Value*
- *Total Degree*
- *Number Of Two HopAway Neighbours*
- *Local Clustering Score*
- *Average Clustering of Neighbourhood*

- Tested a variety of vertex features from the literature and the above features resulted in the best combination of discriminative power and computational speed.



Durham University

Approach Overview

- This results in the vertex * feature matrix VF:

$$VF_{n,m} = \begin{pmatrix} \boxed{\begin{matrix} f_{1,2} \\ f_{2,2} \\ \vdots \\ f_{m,2} \end{matrix}} & \begin{matrix} \cdots \\ \cdots \\ \ddots \\ \cdots \end{matrix} & \begin{matrix} f_{1,n} \\ f_{2,n} \\ \vdots \\ f_{m,n} \end{matrix} \end{pmatrix}$$

$$\downarrow$$

$$\overrightarrow{V_G 1} = (\bar{x}_1, Mo_1, \sigma_1, \sigma_1^2, Skew[x]_1, Kurt[x]_1, x(1)_1, x(n)_1, \ldots \\ , \bar{x}_n, Mo_n, \sigma_n, \sigma_n^2, Skew[x]_n, Kurt[x]_n, x(1)_n, x(n)_n)$$

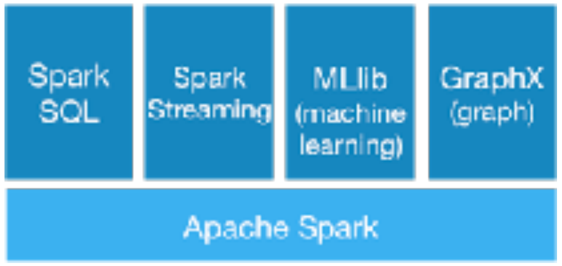
- This ‘fingerprint’ can be compared with others using the Canberra distance:

$$CD(p,q) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

Background Technologies

- **We use a variety of Big Data technologies for the creation of GFP-X, including Spark and Graph-X.**

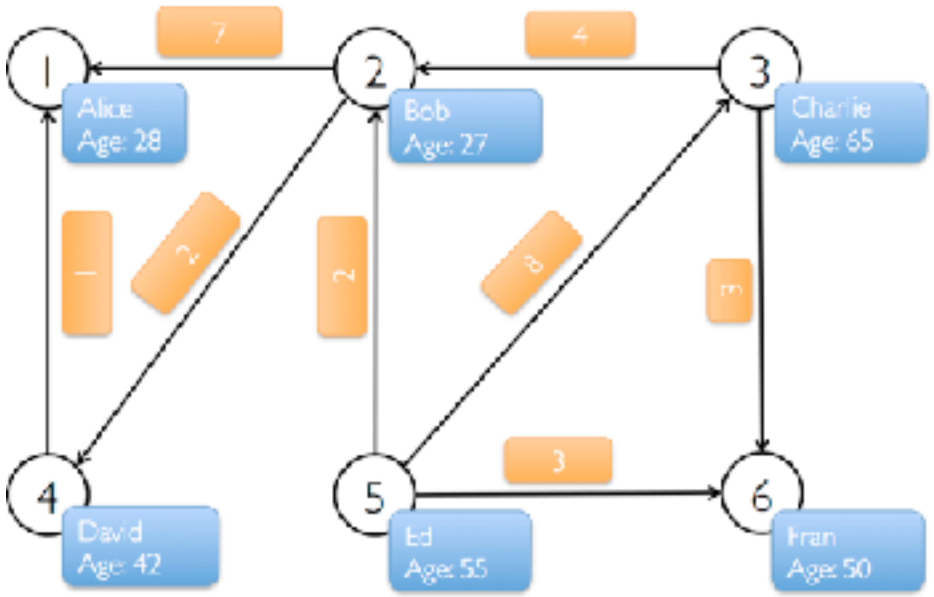
- **Apache Spark** - An in memory computation layer for the Hadoop ecosystem. It has a variety of domain-specific computation libraries including GraphX, Spark Streaming, MLlib and SparkSQL.
- **GraphX** - A domain specific library for parallel graph processing on Spark. Has been shown to have excellent performance and scalability.



GFP-X Implementation

- GFP-X has been implemented completely end to end in GraphX and Spark.
- Where possible we use the included GraphX primitives for the global features.
- For the vertex level features we used the ‘Aggregate Messages API’. This allows for arbitrary messages to be passed to a vertex from all of the vertices incident upon it. For example the local clustering score is passed to a vertex from it’s neighbourhood, which is then aggregated using a vertex level function.

- We then aggregate the feature matrix down using the data frames abstract with careful memory management.



Experimental Evaluation and Results

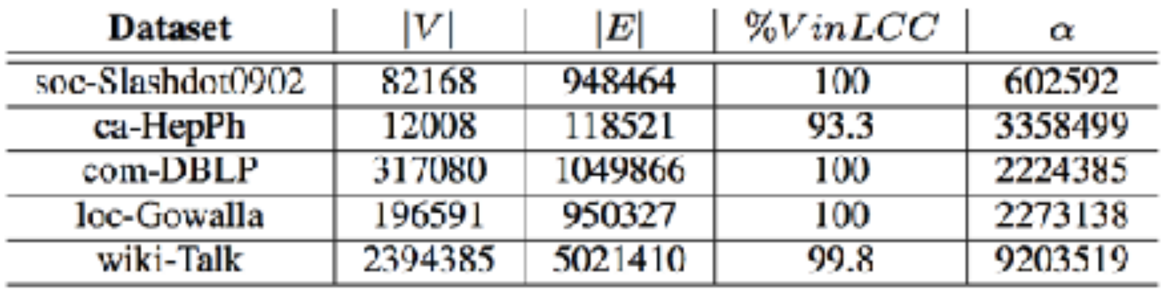
- **Testing Environment:**

- Software stack of CentOS 7.2, Java 1.8, Scala 2.10.5, Apache YARN 2.7.1 and Apache Spark 1.6.1.

- All nodes had identical hardware – 2 * 8C Intel Xeon E5-2630v3, 64GB RAM and 1TB of SSD storage. All nodes communication via a dedicated 10Gb SFP+ network.

- YARN was used to allocate cluster resources to Spark.

- **Datasets -** We used common benchmark datasets from SNAP.

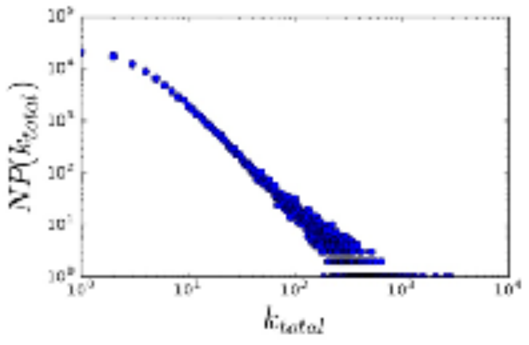


Dataset	$\|V\|$	$\|E\|$	$\%V in LCC$	α
soc-Slashdot0902	82168	948464	100	602592
ca-HepPh	12008	118521	93.3	3358499
com-DBLP	317080	1049866	100	2224385
loc-Gowalla	196591	950327	100	2273138
wiki-Talk	2394385	5021410	99.8	9203519

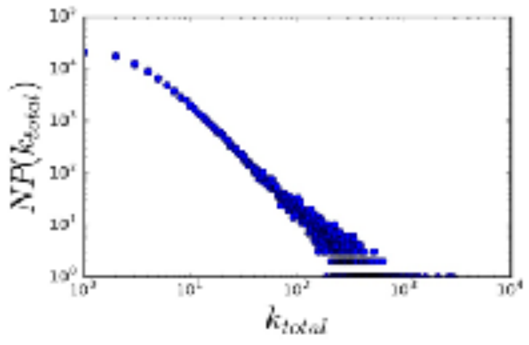
Experimental Evaluation and Results

- **Datasets:**

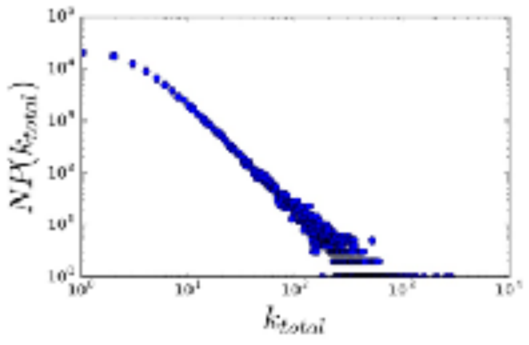
- We also generated synthetic graphs of varying sizes using the Forest Fire generation method.
- We also created perturbations of these graphs using the random rewire method.



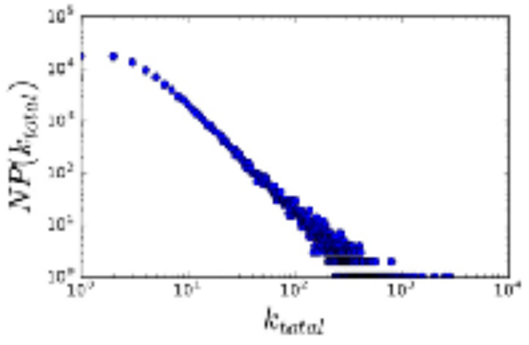
(a) Original Graph



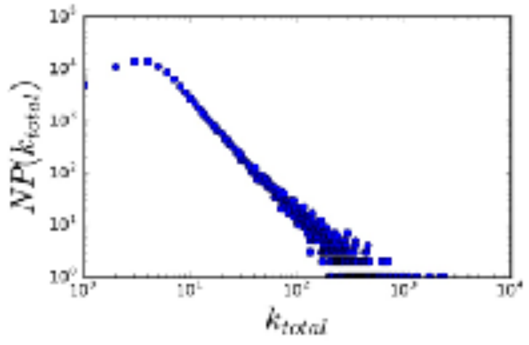
(b) Graph 1 (10^2)



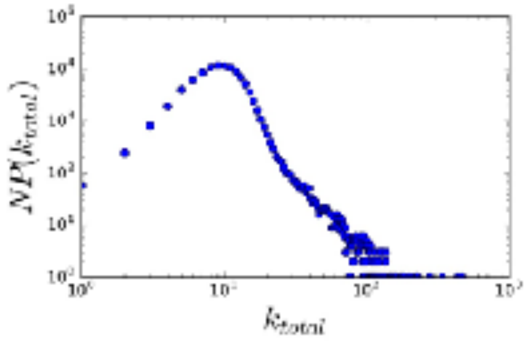
(c) Graph 2 (10^3)



(d) Graph 3 (10^4)



(e) Graph 4 (10^5)



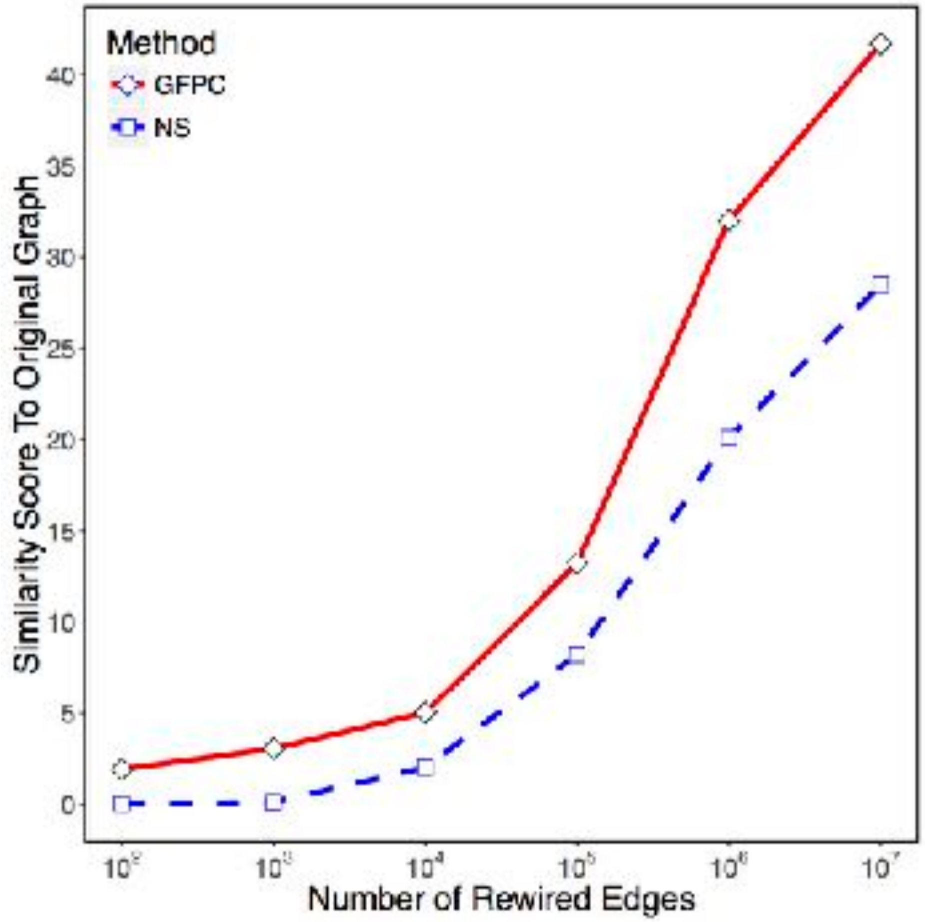
(f) Graph 5 (10^6)

Experimental Evaluation and Results

- This Figure shows how our new approach compares with the current state of the art: NetSimile. For both methods, we used the Canberra distance so the results are directly comparable.

- For the results presented here, an original Forest Fire graph (with 10^5 vertices) was compared to each of the rewired graphs.

- The results show that a change in graph topology is always detected as more different to the source graph.

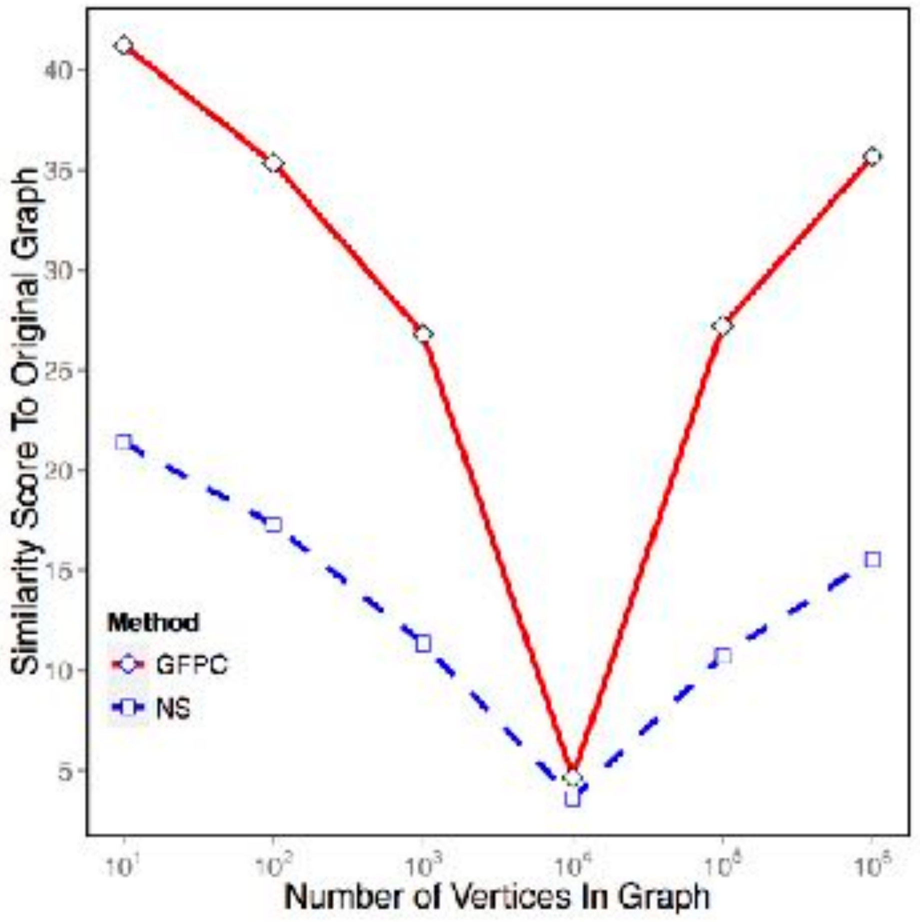


Experimental Evaluation and Results

- This Figure shows how the approaches respond to changes in the global size of the graph.

- For the results presented here, an original Forest Fire graph (with 10^4 vertices) was compared to other forrest fire graphs of varying sizes.

- It can be seen that the GFP-X approach is much more sensitive to global graph size then NS.



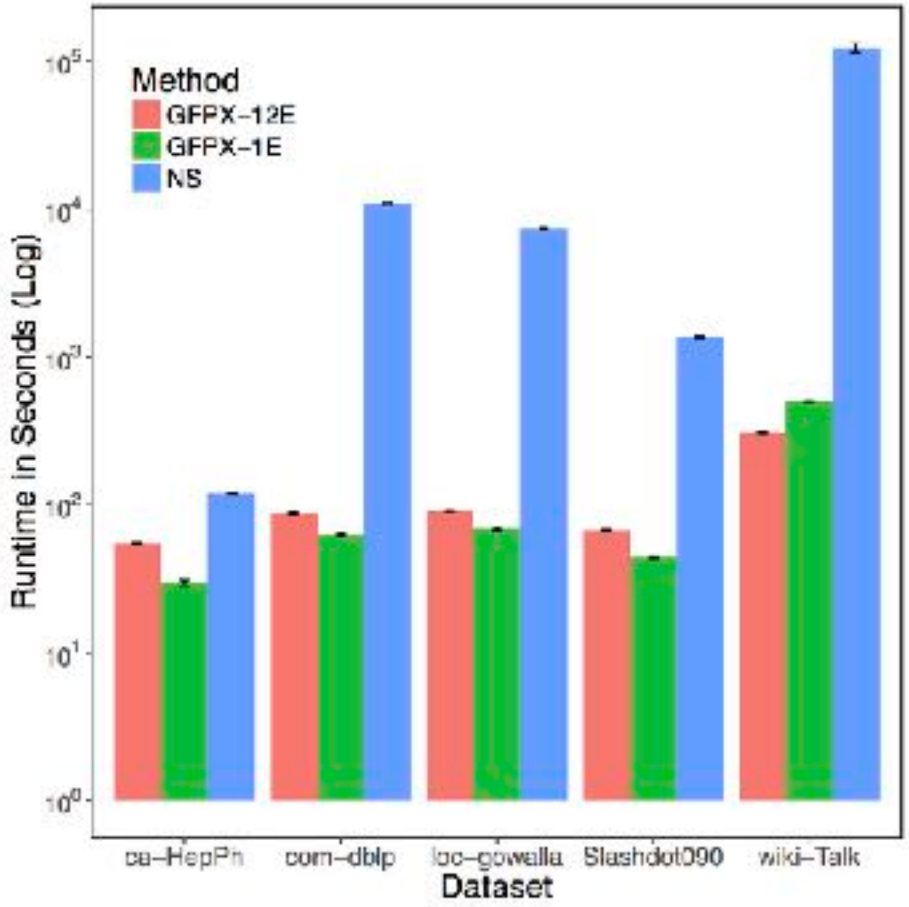
Experimental Evaluation and Results

- Now we have shown that the sensitivity results are comparable with the current state of the art approach, we want to investigate how the performance of GFP-X scales across dataset size.

- We compare GFP-X running on 1 executor, 12 executors and a NS implementation in C++.

- Interesting to note that even on a single executor, GFP-X is faster than NS.

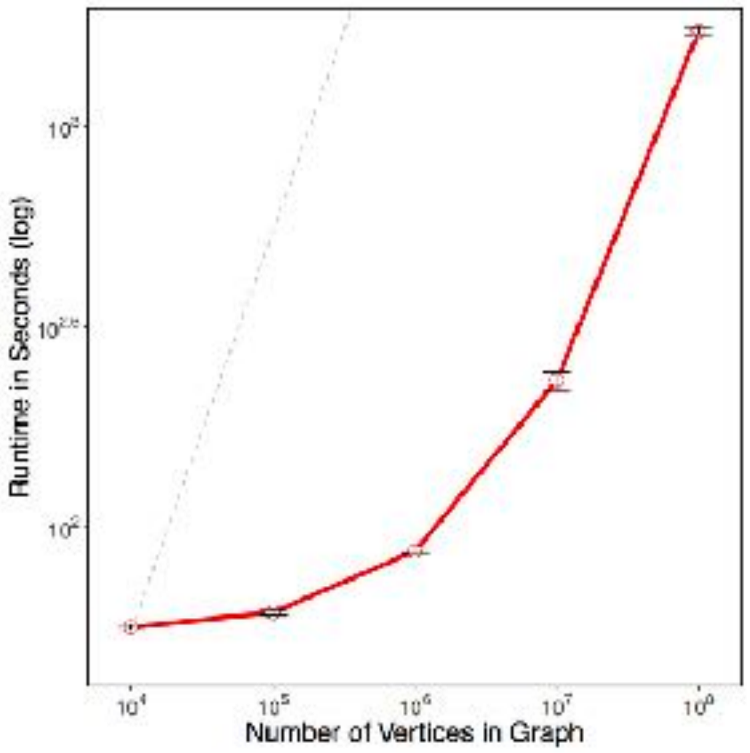
- It can also be seen that on these comparably small datasets, the inherent communication cost with the distributed computation are not always worthwhile.

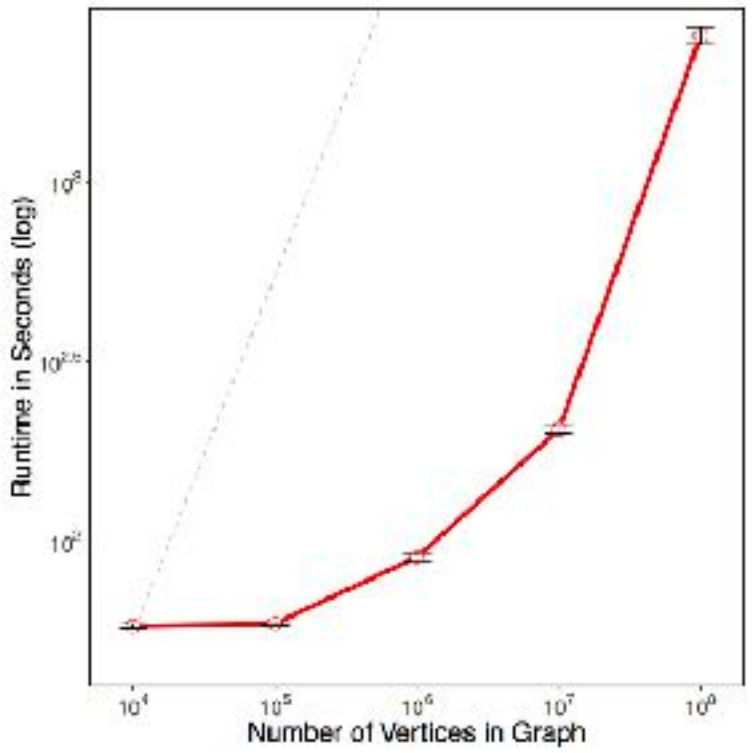


Experimental Evaluation and Results

- Lastly we tested the runtime of the fingerprint extraction one range of synthetic graphs, BA on the left and ER on the right.

- It can be seen that up until approximately a dataset size of $10^{7,}$ It can be said that the approach scales sub-linearly to dataset size.





Conclusions and Further Work

- **Conclusions:**

- Presented a parallel approach for massive graph comparison using feature vectors.
- With some caveats, GraphX is good at graph feature extraction, but needs careful tuning to achieve best performance.
- **Future Work:**

- We hope to truly test the scaleability of the GFP-X approach by deploying in into the cloud and running on even larger graph datasets.
- Investigate the use of the new GraphFrames API from Spark to further improve the runtime performance.
- Currently working on studying the temporal variation of a graphs fingerprint.
- We plan to perform further experimentation comparing with exciting new work on parallel Graphlets.

- Please note that all code and experiment scripts are open sourced under GPLv3 and is available on GitHub -

https://github.com/sbonner0/GFPX-GraphSimilarity