Efficient Comparison of Massive Graphs Through The Use Of 'Graph Fingerprints'

Stephen Bonner
School of Engineering and Computing Sciences
Durham University
Durham, UK
s.a.r.bonner@durham.ac.uk

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

Georgios Theodoropoulos
Institute of Advanced Research Computing
Durham University
Durham, UK

Ibad Kureshi
Institute of Advanced Research Computing
Durham University
Durham, UK

Andrew Stephen McGough
School of Engineering and Computing Sciences
Durham University
Durham, UK

ABSTRACT

The problem of how to compare empirical graphs is an area of great interest within the field of network science. The ability to accurately but efficiently compare graphs has a significant impact in such areas as temporal graph evolution, anomaly detection and protein comparison. The comparison problem is compounded when working with graphs containing millions of anonymous, i.e. unlabelled, vertices and edges.

Comparison of two or more graphs is highly computationally expensive. Thus reducing a graph to a much smaller feature set – called a fingerprint, which accurately captures the essence of the graph would be highly desirable. Such an approach would have potential applications outside of graph comparisons, especially in the area of machine learning.

This paper introduces a feature extraction based approach for the efficient comparison of large topologically similar, but order varying, unlabelled graph datasets. The approach acts by producing a 'Graph Fingerprint' which represents both vertex level and global level topological features from a graph. The approach is shown to be efficient when comparing graphs which are highly topologically similar but order varying. The approach scales linearly with the size and complexity of the graphs being fingerprinted.

CCS Concepts

•**Mathematics of computing** → **Graph algorithms;** •**Computing methodologies** → *Feature selection;*

Twelfth Workshop on Mining and Learning with Graphs (MLG) '16 San Francisco, California USA

© 2016 ACM. ISBN 0-12345-67-8/90/01...$15.00

DOI:

Keywords

Graph Similarity; Feature Extraction; Graph Fingerprints

1. INTRODUCTION

Network science is an interdisciplinary field for the study of detailed real-world phenomena by viewing them as a graph. In many domains, being able to compute the similarity between two graphs is extremely valuable. Such domains include: anomaly detection [2], protein comparisons [31] [24] and the study of temporal graph evolution and link predication [1]. As such, graph comparison and specifically similarity measurement is an area of increasing research interest within the field of network science.

The terms graph and network are often used interchangeably within the literature, however, in this work we shall use the term graph without loss of generality. We define a graph $G = (V, E)$ as a finite set of vertices (sometimes referred to as nodes) – V – and a finite set of edges – E. The elements of E are an unordered tuple $\{u, v\}$ of vertices $u, v \in V$. Two vertices are said be incident if there is an edge between them. The number of vertices $N_v = |V|$ and edges $N_e = |E|$ are often called the order and size of the graph. A directed graph G, where each edge has an associated direction, implying that $u, v \in E$ is distinct from $v, u \in E$. It is possible for a graph to have a set of labels associated with vertices, edges or both. In such cases we can define a graph $G = (V, E, L)$, where L is a set of labels. Such labels contain additional information about an edge, vertex or the graph itself, for example a person's name or age on a vertex within a social network.

There are many definitions of similarity between graphs [5] [14] [22], however, they can be broadly split into two categories – those which can be applied to labelled graphs and those which can be applied to graphs irrespective of labelling. When labels are available, similarity can be based on such metrics as the number or content of labels appearing in both graphs, however, when labels are not present similarity is based around comparison of graph topology. In this paper we are concerned with computation of the similarity between unlabelled graphs and thus we focus upon comparing graph topologies.

There are a number of considerations which need to be addressed when computing the topological similarity between graphs to ensure the most accurate comparison. For example, two graphs might appear very similar when considering the micro-level linking between vertices, yet be of completely different scales with regards to total graph sizes. A counter example being two graphs which are of comparable sizes, yet have vastly different degree distributions (The distribution of edges between the vertices within a graph). A further consideration is how sensitive should the comparison be between a graph and a new graph formed as a perturbation of that graph (vertices and edges). Likewise, any comparison approach should be able to scale to 'high volume' (vertices and edges) graphs. Graph processing techniques are being applied to a broader range of data driven fields, where data volumes are large and constantly increasing, resulting in more graphs of larger sizes [20]. The current Facebook social network, for example, is said to contain over one billion vertices and is still growing [9]. This dramatic increase in the quantity of data means that larger and larger graphs need comparing against one another. This has a compound effect upon graph similarity measures, as any such algorithm needs to be both accurate and computationally efficient.

In this paper we present a approach entitled Graph FingerPrint Comparison (GFPC) to compare the similarity of large graphs by extracting micro and global level features. We introduce the graph fingerprint as a compact but representative abstraction of a graph, with numerous potential applications within the field of machine learning. We demonstrate an application of the fingerprint approach for the comparison of graphs that is label and attribute independent as it exploits only the topology of a given graph in order to compare similarity. The contributions of this paper are as follows; Firstly, due to the nature of the features chosen for extraction, the generation of the feature vector is less computationally complex then other approaches [5] and thus can be completed in near linear time with respect to dataset size. Secondly, owing to the inclusion of the global features within the similarity score, the GFPC method is sensitive to both graph size and low-level vertex interconnectivity.

In Section 2 we discuss related work, Section 3 details the motivation for the work, Section 4 details the generation of graph fingerprints, Section 5 details the comparison of fingerprints, Section 6 presents empirical results and Section 7 draws conclusions and explores possible future research.

2. PREVIOUS WORK

It has been argued [5] [14] that the various label dependant and independent methods for graph comparisons can be further categorised into three major cross cutting classes: graph-isomorphism based methods, iterative methods and feature extraction based methods.

Graph-Isomorphism - Two graphs are said to be similar if they are isomorphic or if they contain isomorphic subgraphs [23]. Unfortunately computing sub-graph isomorphism is known to be an NP-Hard and thus not applicable to even moderately sized graphs [7] [19]. The graph isomorphism has been generalised by the Graph Edit Distance (GED) concept. The GED for two graphs is the number of operations needed to transform one graph into another [32], with the permitted operations being the addition/subtraction/rewiring of vertices and edges, along with the reversing of edge directionality. As with graph isomorphism, the GED between two graphs has been proven to be NP-Hard [32]. Linked to the isomorphism approaches is the GRAAL family of algthioms [15] which are designed to align two graphs and have been used to align small protein interaction graphs, but with an exponential runtime.

Iterative Methods - In iterative methods for comparing graphs, a vertex generates a local score, capturing some key information about itself which is then exchanged among other vertices within a graph. Perhaps the most widely know of the iterative methods is the SimRank algorithm, designed to measure how similar vertices are within a single graph by comparing the structure of a vertex's local neighbourhood [13]. Each vertex generates a score capturing information about its neighbourhood, the algorithm then exchanges this information among the vertices to look for similar scores. This work is concerned with self-similarity within a graph, not similarity to other graphs so is not suitable for the work in this paper. A more modern iterative method for graph comparison utilises belief propagation [30] as a way of finding possible correspondences between vertices in two graphs [4]. However the approach requires possible correspondences to be provided, as such the approach is not label independent.

Feature Extraction - Such methods extract a range of features from a given graph and use them to compare with other graphs, with the idea that the more similar two graphs, the more similar their features. Feature extraction based method have advantages over other approaches as they can be highly scalable and thus have more efficient runtimes [14]. However, which features need to be extracted to give the best, yet most compact, representation of a graph, is still an area of active research [22]. One such feature extraction method presented by Roy extracts a variety of centrality measures (used to rank the importance of a vertex with a graph [27]) for a graph and uses them to compare with other graphs [25]. This approach requires that the graph data be annotated with labels and the presented results only considered small graphs. An alternative feature extraction method presented by Papadimitriou has been explored to measure the similarity between snapshots of a graph of links between webpages [22]. In this approach several similarity measures are tested on time-series of graph data with the goal of detecting anomalies between time-steps. However many of the methods tested rely on labeled data to compute similarity.

The NetSimile algorithm [5], considered to be the most state of the art approach, relies upon extracting details about the 'EgoNet' (A vertex's EgoNet is every other vertex which is connected to it in it's local neighbourhood) for each vertex within a graph which is then compared, via a distance metric, with results from other graphs. In the presented results, NetSimile is shown to be independent of graph size when making the comparison and only considers the similarity of the underlying linking model, meaning that two graphs of vastly different scales could be identified as similar.

Feature extraction has been explored outside of similarity measurement as a way of classifying graphs based on comparisons between global features and labels [18]. Additionally feature extraction has been explored by the anomaly detection community as a way of detecting unusual elements or events within static and temporal graphs [2].

3. GRAPH COMPARISONS

3.1 Motivation and Requirements

The research behind the GFPC approach is part of a larger body of work investigating new machine learning techniques to study and predict the evolution of graphs. As such, an accurate way of comparing an empirical data source to a synthetically generated one, as predicated by the machine learning algorithm, was needed. Any difference between the empirical and generated datasets could be used to validate the generation method. Thus it was crucial that it was highly accurate and sensitive to small changes in graph structure. The development of the GFPC approach was required when we found existing methods for graph comparison to be insensitive to highly similar graphs, ignored a graph's size when making comparisons or had an unacceptably long runtime when processing large datasets. In this context, we define two graphs to be similar if they share similar global and micro level topological features. As such, the development of our approach was driven by the following requirements:

1. **Scalability** - The new approach should be highly scalable, to graphs of millions of vertices/edges, and capable of computing the similarity in a finite time. Ideally the approach should be portable to a many-core or distributed graph processing system to help with scalability.
2. **Sensitivity To Graph Size** - The similarity score should take the size and order of the graphs into consideration.
3. **Sensitivity To Similar Topologies** - It should be able to detect the difference between graphs which are highly structurally and topologically similar.
4. **Label Free** - Finally, it should be able to perform comparisons without the need for labeled datasets.

3.2 Approach Overview

Our approach is comprised of two distinct stages: The generation of a graph's fingerprint (the GFP approach) and the comparison of these fingerprints (the GFPC approach). The Graph FingerPrint (GFP) generation takes the high dimensional graph object and reduces the complexity down to two fixed length vectors. The GFP approach achieves this by extracting micro and global features from the given graph, allowing it to capture both the macro and microscopic topological features. The decision to extract both vertex level and global level features was driven by the desire to make the comparison between graphs more sensitive to small variations in the underlying graph topology and the overall size of the graph than the current state of the art methods [5]. The aim was to create a feature set which comprises of a wide spectrum of similarity scores when compared with other graph comparison techniques.

The GFP approach is broken down into three stages required to generate a feature set:

1) Vertex Level Feature Extraction 2) Vertex Level Feature Creation 3) Global Level Feature Extraction

These stages are executed for each graph to generate their fingerprint which can be used to compare graphs and can be stored for later use. After the vertex features have been extracted from the graph, they are then aggregated during the vertex feature creation stage. In addition global features are also extracted from each graph. It is worth noting that the GFP approach can be extended to include any vertex or global level feature, not just those detailed in this paper.

The GPFC approach then computes the similarity between any two graphs using the following stages:

1) Vertex Level Comparison 2) Global Level Comparison 3) Final Similarity Score Generation.

Once both the vertex and global level features have been prepared for each graph, a vector distance metric âĂŤ we use the Canberra distance metric [16] in this paper. This results in two separate similarity scores, one comparing the vertex level topology and one the global level similarity. The last stage is to combine these two scores to produce the final similarity score between two graphs. In the next two sections, both the GFP and GFPC approaches are described in greater depth.

4. GENERATING GRAPH FINGERPRINTS

4.1 Vertex Features

The GFP approach extracts features from each vertex within a graph. Although a wide selection of vertex feature metrics exist each exhibits different characteristics in terms of topological sensitivity and runtime. Through experimentation we have determined that the following six feature metrics gives the best balance between topological sensitivity and runtime. However, other metrics could also be used if other characteristics of a graph are important. For each of the six vertex features listed below, a value is extracted for each vertex $v \in V$.

Eigenvector Centrality Value ($\mathbf{Ax}$) - The Eigenvector centrality, as with all centrality measures, is used to calculate the importance of each vertex within a graph [6]. Eigenvector centrality calculates the importance of a vertex based upon the number of connections it has originating from other important vertices. Formally the Eigenvector centrality can be written as the following eigenvector equation, where λ is the largest eigenvalue, $\mathbf{A}$ is the graph in adjacency matrix from and $\mathbf{x}$ is the eigenvector centrality:

$$\mathbf{Ax} = \lambda \mathbf{x}.$$

PageRank Score ($PR(v)$) - The PageRank centrality method was originally developed by Google as a way of ranking web pages [21], however it is now commonly used to measure the local influence of a vertex within a graph [11]. The PageRank score for a given vertex v is:

$$PR(v) = \frac{1-d}{N} + d \sum_{u \in \Gamma^-(v)} \frac{PR(u)}{d^+(u)},$$

where $\Gamma^-(v)$ is the set of incoming neighbours of v, $d^+(u)$ is the out-degree of u and d is a constant damping factor (0.85 for this work).

Total Degree (td_v) - This is the sum of both the in and out degree for the vertex v:

$$td_v = \Gamma^-(v) + d^+(v).$$

Two-Hop Away Neighbours (th_v) - The number of two-hop away neighbours from the current vertex v gives an indication of how connected, and thus how important, a vertex's neighbourhood is within the graph:

$$th_v = \frac{1}{|N(v)|} \sum_{\forall j \in N(v)} d^+(j),$$

where $N(v)$ is every vertex incident on the current vertex v.

Local Clustering Score (c_v) - The local clustering score for vertex v represents the probability of two neighbours of v also being neighbours of each other [28]:

$$c_v = \frac{2\Phi}{d^+(v)(d^+(v)-1)},$$

where Φ is the number of pairs of $v's$ neighbours which are themselves connected.

Average Clustering of Neighbourhood (nc_v) - The average clustering score of the neighbourhood is taken for each vertex by taking the mean of all the local clustering scores for the vertex's neighbourhood:

$$nc_v = \frac{1}{|N(v)|} \sum_{\forall j \in N(v)} c_j,$$

where c_j is the local clustering score computing in the previous feature extraction step.

4.2 Global Features

In order to make the GFP approach sensitive to the overall features of a given graph, a selection of global features are extracted in addition to previously defined vertex features. GFP extracts a total of six global features from each graph. The global features, chosen to represent each graph, were selected due to their ability to capture key elements of global graph topology, whilst also being efficient to compute. A vector is used to represent these six global features:

Graph Order - Defined as: $|V|$.

Number of Edges - Defined as: $|E|$.

Number of Triangles - The number of triangles, α, for a given graph is the number of vertices which form a triangle, with a triangle being a set of three vertices with an edge between every member.

Global Clustering Coefficient - This feature is a representation of how connected the graph is overall, using the total number of possible vs complete triangles within a graph:

$$gc = 3\frac{\alpha}{\beta},$$

where β is the number of connected triplets of vertices (three vertices which are all connected, but not necessarily into a triangle) within the graph.

Maximum Total Degree Value - This represents the total number of edges the most connected vertex in the graph has to other vertices.

Number of Components - This is the total number of components within the graph, with a component being a subgraph in which there is a possible path between every vertex, whilst vertices in different components have no possible path between them.

4.3 Feature Creation

The matrix, $VF_{m,n}$, where $m = |V|$, contains all the vertex feature scores as defined in Section 4.1, and $n = |F|$ (F is the vector of features for each vertex):

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

In order to create the graph fingerprint, we need to reduce the dimensionality of the feature matrix down to a more compact vector. To perform this transformation, a series of metrics are taken for each of the feature columns in the matrix. The metrics chosen are the median, mean, standard deviation, variance, skewness, kurtosis, minimum value and maximum value. These are often used and well understood methods to capture numerical variation.

After this process has been completed, the resulting vertex feature vector $\overrightarrow{v_{g1}}$ for graph $G1$ has been created. The vertex feature vector contains the eight aggregation scores for each column in the feature matrix which are concatenated together:

$$\overrightarrow{v_{g1}} = (\bar{x}_1, Mo_1, \sigma_1, \sigma_1^2, Skew[x]_1, Kurt[x]_1, x(1)_1, x(n)_1, ...$$
$$, \bar{x}_n, Mo_n, \sigma_n, \sigma_n^2, Skew[x]_n, Kurt[x]_n, x(1)_n, x(n)_n).$$

5. COMPARISON OF GFPS

The GFPC approach must compare the fingerprints of two graphs to compute their similarity. After extensive experimental evaluation and similar to [5], the Canberra distance was selected to compare the numerical distance between the fingerprints. Other distance metrics tested including the Bray, Correlation, Chebyshev, Cosine and Manhattan but these were found to be insensitive when the feature vectors were highly similar, or produced unintuitive results such as a high similarity scores for highly dissimilar graphs.

The Canberra distance between two vectors is:

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

The Canberra distance is able to detect changes close to zero, which makes it ideal for detecting small variations in graph topology – one of the key goals for the GFPC approach. The Canberra distance is used to compare both the distance between the vertex feature vectors and the global feature vectors. The two graphs are more 'similar' the closer the result of the Canberra distance is to zero, with a score of zero indicating that the graphs are 'fingerprint' identical.

5.1 Final Similarity Score Generation

The GFPC approach returns two similarity scores, one for the distance between the vertex feature vectors vf_s and one for the distance between global vectors gf_s for the two graphs being compared. These two scores can be used independently to compare the topology and size as separate entities. However, the GFPC approach can produce a final similarity score between the two graphs, using the following aggregation - $FinalSimScore = vf_s(1 - \gamma + gv_s * \gamma)$. Where γ is a user controllable parameter to adjust the weighting of the difference between the global feature vectors in the final similarity score.

5.2 Implementation

The GFP and GFPC approaches are currently implemented in Python, utilising the Graph-Tool package [8] to perform

Table 1: Graph Datasets Used

Dataset	$\|V\|$	$\|E\|$	gc	α
Ca-HepPh	12008	118521	0.6115	3358499
Cit-HepTh	27770	352807	0.3120	1478735
com-dblp	317080	1049866	0.6324	2224385
Enron-Email	36692	183831	0.4970	727044
p2p-Gnutella04	10876	39994	0.0062	934
soc-Slashdot0811	77350	516575	0.0549	548054

the graph analytic stages of the algorithm. The Python and Graph-Tool combination was chosen at it offered the possibility for rapid prototyping, whilst also offering good computational performance. The Graph-Tool package is written in C++ using the Boost Graph Library [26] and with support for OpenMP, allowing it to be used on many-core, shared memory architectures. Graph traversal is known to be slow for massive graphs [20], therefore, GFPC has been designed to only traverse through the entire set of vertices once, collecting all required features for a certain vertex before proceeding to the next.

The GFP and GFPC frameworks have been open sourced under a GPLv3 licence and are available on GitHub[1]. In addition, the code used to run each experiment and the implementation of NetSimile used for the comparison, also implemented in Graph-Tool, is available in the same repository.

6. RESULTS

In this section, the GFPC approach is assessed against the four criteria as discussed in section 3.1. In each experiment GFPC is compared to the current state of the art feature extraction graph comparison method NetSimile. As both the GFPC and NetSimile approaches generate their final similarity scores using the Canberra distance, their results are comparable. As such, comparisons between GFPC and NS are based are based upon any differences between there distance scores. It is worth noting that other distance metrics would produce similar differences between the results.

6.1 Experimental Setup

All the experiments presented in this paper were performed on a system with 2 * 10C 2.3GHz Intel Xeon E5-2650 v3, 64GB RAM, CentOS 7.2, GCC 4.8.5, Boost 1.56, Python 2.7.5 and Graph-Tool 2.8. For all experiments, γ has been set to 2. All the random Barabási-Albert and Erdős-Rényi [8] graphs used for the experiments were generated from within Graph-Tool's random graph generation methods. For the random Barabási-Albert graphs generated, the out-degree of newly added vertices parameter was set to two. This means that for the newly created graph, $|E| = |V| * 2$. The empirical data used for some of the experiments was taken from the widely used Stanford Network Analysis Project (SNAP) datasets [17]. A summary of the datasets used can be seen in Table 1. The datasets are from a range of domains including citation, collaboration, communication and social networks.

6.2 Random Rewire Process

To demonstrate that the GFPC approach is highly sensitive to the underlying topology of a given graph, the edges in a Barabási-Albert preferential attachment graph were rewired in a random fashion. Figure 1 shows how the degree distribution of the original graph was altered by the random rewiring process, with the numbers representing the number of rewired edges within the graph. This Figures shows the probability of a vertex having a certain number of connections. This process alters a given source graph's degree distribution by randomly altering the source and target of a set number of edges according to the Erdős-Rényi random model. During this re-wire process, it is not guaranteed that the source or target of the edge will be altered, indeed it is not always possible due to the graphs topology. The rewiring process does not change the total number of edges or vertices within the graph.

6.3 Sensitivity to Variations in Topology

For this experiment the graph rewiring process (section 6.2) was used to transform an input Barabási-Albert graph with 200,000 vertices by rewiring a varying percentage of the edges. The degree distributions of the graph after varying the amount of rewiring can be seen in Figure 1.

For the results, the original graph was compared to each of the rewired graphs to measure similarity. Figure 2 shows that GFPC is sensitive to the changes in the topology of the graph, with an increase in the percentage of the graph rewiring always being detected as more dissimilar to the source graph.

6.4 Sensitivity to Variations in Size

The GFPC approach was tested for its sensitively to variations in the global graph size. For this experiment, a random Barabási-Albert graph G_o was generated with $|V| = 10,000$ and $|E| = 20,000$. To compare with the source graph, six new graphs were generated again using the Barabási-Albert method each with increasing numbers of vertices and edges. As we are utilising the Barabási-Albert method all the graphs will be highly structurally similar. The results comparing the GFPC and NetSimile method for sensitivity to variations in graph size are displayed in Figure 3. In the figure, graphs of varying sizes were compared to the original graph G_o to generate the similarity score.

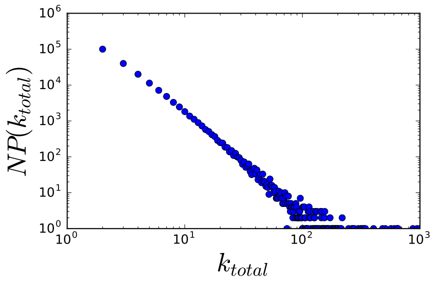
Figure 3 shows that the GFPC is more sensitive to variations in graph size than the NetSimile method, with an increase in the size of the graph always detected as more dissimilar to the source graph.

6.5 Runtime Analysis

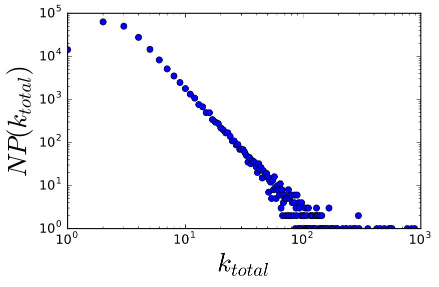
The final criteria upon which GFPC was evaluated was the the runtime of the feature extraction algorithm when generating a fingerprint across a range of empirical data sources, as well as comparing it to the NetSimile implementation. Figure 4 shows the runtime of the feature extraction stages for both GFP and NetSimile, across the datasets detailed in Table 1, with all experiments being repeated five times and the error bars being one unit of standard deviation. The figure shows that GFPC is constantly faster then NetSimile, often substantially. GFPC is able to process the largest graph, *com-dblp*, in under 90 seconds, with all other graph datasets being processed in less time.

In addition to testing on empirical datasets, the run time of GFP approach was evaluated across a range of synthetic Barabási-Albert graphs. These experiments were performed to assess the relationship between number of vertices / edges within a graph and the runtime of GFP. Again, all exper-

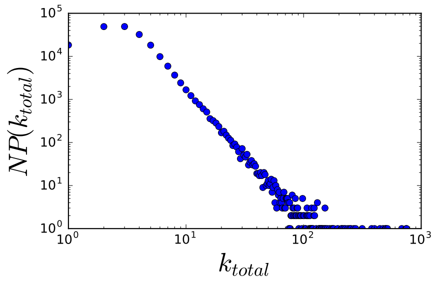
[1]https://github.com/sbonner0/GraphFingerprintComparison



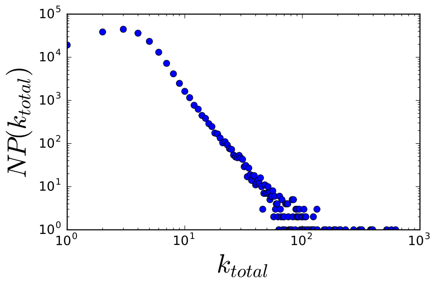
(a) Original Graph



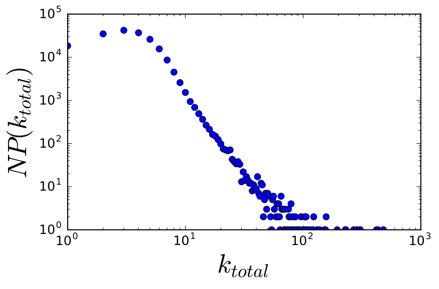
(b) Graph 1 (5000)



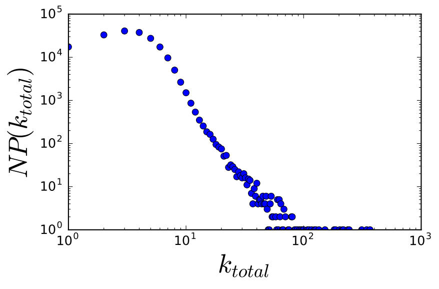
(c) Graph 2 (10000)



(d) Graph 3 (20000)



(e) Graph 4 (30000)



(f) Graph 5 (40000)

Figure 1: Change In Degree Distribution After Rewiring Process.

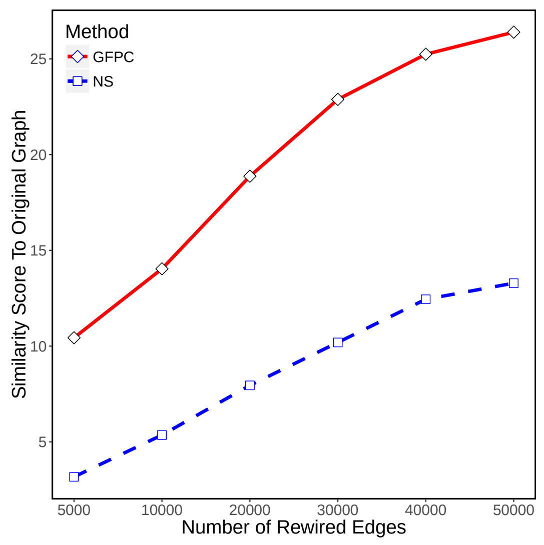


Figure 2: Sensitivity To Graph Topology

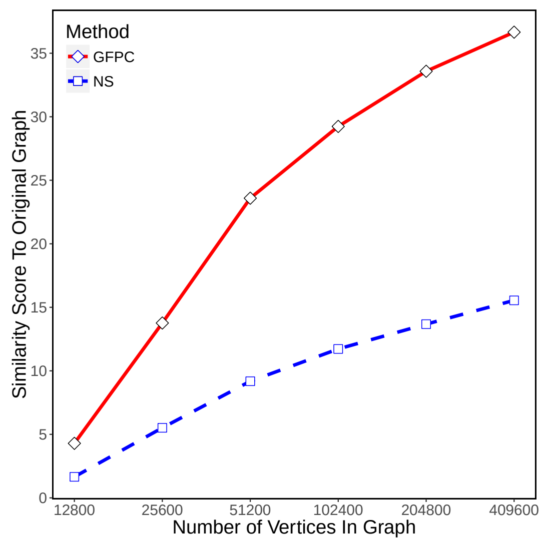


Figure 3: Sensitivity To Graph Size

iments were repeated five times and the error bars being presented as one unit of standard deviation.

Figure 5 shows how the runtime of the GFP approach responds to increases in the number of vertices. This figure shows that GFP responds in an almost linear fashion to increases in the number of vertices within a graph. The anomalous result at the start of the graph can be attributed to the constant time Python requires to create the data structures and perform other setup tasks.

Figure 6 shows how the GFP approach responds to increases in the number of edges within a graph. For this experiment, the number of vertices was kept constant at 10,000. This figure shows that the GFP approach also responds in a near linear fashion to increases in the total number of edges within a graph. However, the runtime appears to decrease when processing a graph with 100 millions edges. It is possible that when processing graphs of this scale, both of the sockets within the server are fully utilised.

6.6 Discussion

The evaluation stage for the GFPC approach has been positive and it has achieved all of the goals as laid out in section without the need for labeled datasets 3.1. The GFPC approach also outperforms the current state of the art feature based extraction methods on all metrics. The GFPC approach is sensitive to detecting small variations in graph

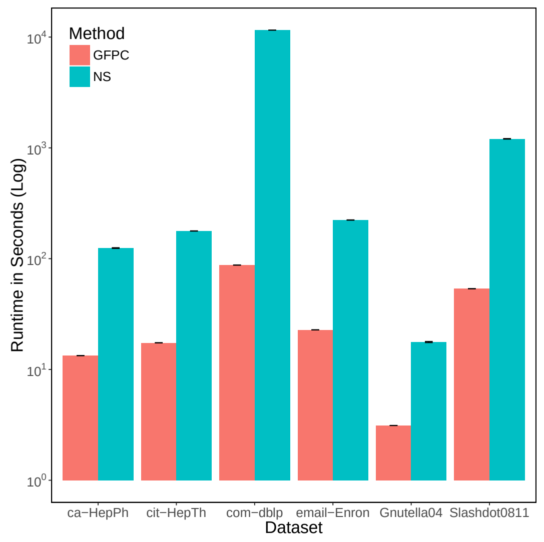


Figure 4: GFPC Runtime Performance

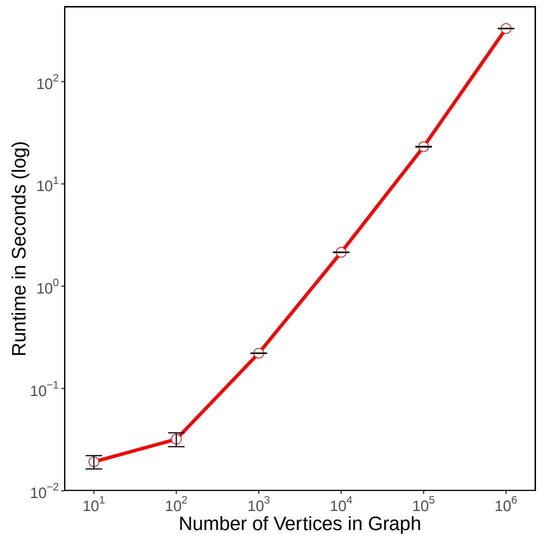


Figure 5: Runtime Across Number Of Vertices

topology and overall graph size. Due to the nature of the features extracted, the GFP approach requires no labels. However, perhaps the most promising result to arise is the near linear runtime of the approach when increasing dataset size. This has the potential to improve machine learning approaches of temporal graph analysis as there is now an efficient way to validate models against empirical data.

In addition, the results from this research have shown the GFP approach to be an accurate but compact representation of a graph. The GFP approach is effectively able to take the high-dimensional complexity inherent in graph datasets, and reduce it to a single fingerprint vector. There are numerous other applications, outside of similarity measures, that could massively benefit from a compact representation of a graph. The application of modern machine and deep learning techniques upon graph datasets is largely unexplored [12]. The compact representation of graphs offered by graph fingerprints could be a key aspect in unlocking the use of an extended range of these techniques. An example of a potential additional use for graph fingerprints is the classification

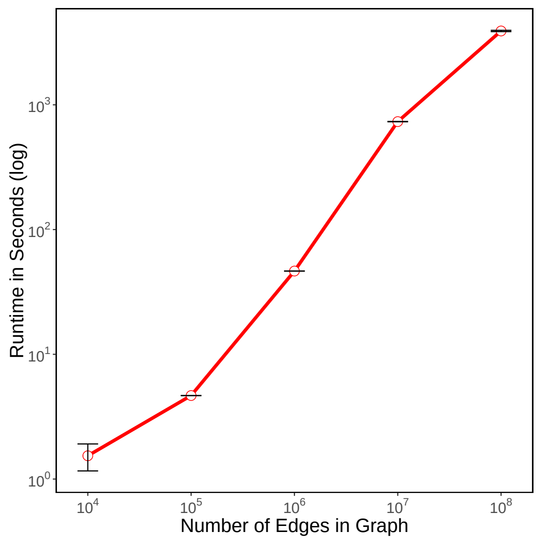


Figure 6: Runtime Across Number Of Edges

of unknown datasets by comparing their fingerprint vectors to labeled ground-truth datasets. For example, do datasets from a certain domain have fingerprints unique enough that they can be used to identify the domain? Previous work has suggested that social networks, for example, have many unique characteristics [3]. A graph fingerprint, taken from a dataset with an unknown domain class, could potentially be used to assign one based upon past training examples.

7. CONCLUSION AND FURTHER WORK

In this paper the Graph FingerPrint Comparison approach for assessing the similarity of two unlabelled graphs, based upon their global and micro features, has been presented. The GFPC approach is shown to be sensitive to small variations in graph topology, graph size, function without the use of labeled datasets and scale near linearly with dataset size. Thus the GFPC approach completes all of the goals established for it in section 3.1. The approach demonstrates some promising results and the concept of a compact but accurate representation of a graph has numerous potential additional applications in fields such as machine learning.

There is large scope for future research based upon the work presented in this paper. Currently the Python Graph-Tool implementation of the approach can only run upon a single compute node which limits the maximum size of graphs which can be compared. Further research is ongoing to port the feature extraction methods onto a graph-parallel framework which would improve speed and scalability. Currently the use of one of the Pregel [20] variants such as GraphX [29], or one of the emerging Graph GPU graph processing systems [10] are being explored. In addition to this the application of the extracted graph fingerprints to other use cases within network science will be explored. For example, could a graph's fingerprint be used to classify, via the use of machine learning, the graph as being a member of a certain domain class or to display properties consistent with one of the graph generation methods? Or could the variations in a graph's fingerprint over time be used to study and model it's temporal evolution? Hopefully the potentially useful applications of studying a graph's fingerprint will be numerous.

8. ACKNOWLEDGMENTS

The authors would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for funding.

9. REFERENCES

[1] C. Aggarwal and K. Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):1–36, 2014.

[2] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, may 2015.

[3] L. Backstrom and J. Leskovec. Supervised random walks: Predicting and Recommending Links in Social Networks. In *Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11*, page 635, New York, New York, USA, 2011. ACM Press.

[4] M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang. Message-Passing Algorithms for Sparse Network Alignment. *ACM Trans. Knowl. Discov. Data*, 7(1):3:1—-3:31, 2013.

[5] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos. NetSimile: A scalable approach to size-independent network similarity. *arXiv preprint*, pages 1–8, 2012.

[6] P. Bonacich. Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555–564, 2007.

[7] D. Conway. Modeling Network Evolution Using Graph Motifs. *Methods*, page 33, 2011.

[8] T. de Paula Peixoto. graph-tool: An efficient python module for manipulation and statistical analysis of graphs. , 2016.

[9] N. Doekemeijer and A. L. Varbanescu. A Survey of Parallel Graph Processing Frameworks. 2014.

[10] Y. Guo, A. L. Varbanescu, A. Iosup, and D. Epema. An Empirical Performance Evaluation of GPU-Enabled Graph-Processing Systems. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 423–432. IEEE, may 2015.

[11] M. Han, K. Daudjee, K. Ammar, M. T. Ozsu, X. Wang, and T. Jin. An Experimental Comparison of Pregel-like Graph Processing Systems. *Vldbe*, 7(12):1047–1058, 2014.

[12] M. Henaff, J. Bruna, and Y. LeCun. Deep Convolutional Networks on Graph-Structured Data. *arXiv*, pages 1–10, 2015.

[13] G. Jeh and J. Widom. SimRank : A Measure of Structural-Context Similarity. *Proceedings of the eighth ACM SIGKDD international conferfrnace on Knowledge discovery and data mining*, pages 1–11, 2001.

[14] D. Koutra, A. Parikh, A. Ramdas, and J. Xiang. Algorithms for Graph Similarity and Subgraph Matching. Technical report, 2011.

[15] O. Kuchaiev, T. Milenkovic, V. Memisevic, W. Hayes, and N. Przulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society, Interface / the Royal Society*, 7(50):1341–54, 2010.

[16] G. N. Lance and W. T. Williams. Mixed-data classificatory programs i - agglomerative systems. *Australian Computer Journal*, 1(1):15–20, 1967.

[17] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2014.

[18] G. Li, M. Semerci, B. Yener, and M. J. Zaki. Effective graph classification based on topological and label attributes. *Statistical Analysis and Data Mining*, 5(4):265–283, aug 2012.

[19] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo. Capturing topology in graph pattern matching. *Proceedings of the VLDB Endowment*, 5(4):310–321, dec 2011.

[20] G. Malewicz, M. Austern, and A. Bik. Pregel: a system for large-scale graph processing. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146, 2010.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking:bringing order to the web., 1998.

[22] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, 1(1):19–30, may 2010.

[23] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural computation*, 11(8):1933–1955, 1999.

[24] N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, jan 2007.

[25] M. Roy, S. Schmid, and G. Tredan. Modeling and measuring graph similarity. In *Proceedings of the 10th ACM international workshop on Foundations of mobile computing - FOMC '14*, volume 1, pages 47–52, New York, New York, USA, 2014. ACM Press.

[26] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. The Boost Graph Library (BGL), 2016.

[27] M. S. M. Vijay, M. Vijesh, S. Iyengar, S. R. Nayak, N. Shenoy, and R. Sundaram. Prediction of arrival of nodes in a scale free network. *Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012*, (1):517–521, 2012.

[28] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–2, 1998.

[29] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica. GraphX: A Resilient Distributed Graph System on Spark. *First International Workshop on Graph Data Management Experiences and Systems*, page 2, 2013.

[30] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding Belief Propagation and its Generalizations. *Intelligence*, 8:236–239, 2002.

[31] L. A. Zager and G. C. Verghese. Graph similarity scoring and matching. *Applied Mathematics Letters*, 21(1):86–94, 2008.

[32] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On Approximating Graph Edit Distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, aug 2009.