Node Re-Ordering as a Means of Anomaly Detection in Time-Evolving Graphs

Lida Rashidi[1,2], Andrey Kan[2], James Bailey[2], Jeffrey Chan[3], Christopher Leckie[1,2], Wei Liu[4], Sutharshan Rajasegarar[5], and Kotagiri Ramamohanarao[2]

[1] Data61 Victoria Research Laboratory
[2] Dept. of Computing and Information Systems, The University of Melbourne
[3] Department of Computer Science and Information Technology, RMIT University
[4] Department of Engineering and IT, University of Technology Sydney
[5] School of Information Technology, Deakin University
lrashidi@student.unimelb.edu.au
{akan,baileyj,kotagiri,caleckie}@unimelb.edu.au
jchan@rmit.edu.au
wei.liu@uts.edu.au
srajas@deakin.edu.au

Abstract. Anomaly detection is a vital task for maintaining and improving any dynamic system. In this paper, we address the problem of anomaly detection in time-evolving graphs, where graphs are a natural representation for data in many types of applications. A key challenge in this context is how to process large volumes of streaming graphs. We propose a pre-processing step before running any further analysis on the data, where we permute the rows and columns of the adjacency matrix. This pre-processing step expedites graph mining techniques such as anomaly detection, PageRank, or graph coloring. In this paper, we focus on detecting anomalies in a sequence of graphs based on rank correlations of the reordered nodes. The merits of our approach lie in its simplicity and resilience to challenges such as unsupervised input, large volumes and high velocities of data. We evaluate the scalability and accuracy of our method on real graphs, where our method facilitates graph processing while producing more deterministic orderings. We show that the proposed approach is capable of revealing anomalies in a more efficient manner based on node rankings. Furthermore, our method can produce visual representations of graphs that are useful for graph compression.

1 Introduction

Dynamic graphs are becoming ubiquitous formats for representing relational datasets such as social, collaboration, communication and computer networks. One of the vital tasks for gaining an insight into the behavioral patterns of such datasets is anomaly detection. Anomaly detection in time-evolving graphs is the task of finding timestamps that correspond to an unusual event in a sequence of graphs [2]. For instance, a social network anomaly may correspond to the merging or splitting of its communities. Anomaly detection plays an important role in numerous applications, such as network intrusion detection [9], credit card fraud [11] and discontinuity detection in social networks [3].

However, there are many challenges associated with event detection in dynamic graphs. Networks such as Facebook or Twitter comprise billions of interacting users where the structure of the network is constantly updated. Moreover, there is often a lack of labels for normal and anomalous graph instances, which requires learning to be unsupervised. Due to these challenges, graph anomaly detection has attracted growing interest over time.

To address these challenges, many anomaly detection techniques use a pre-processing phase where they extract structural features from graph representations. These features may include node centrality [16], ego-nets [3] and eigenvalues [12]. They then apply well-known similarity measures to compare graph changes over a period of time [4]. In this scenario, the graphs are converted into feature sets and therefore they do not pose the complexities associated with the inter-dependencies of nodes, in addition to causing a considerable decrease in the time and space requirements for the anomaly detection scheme.

However, the process of generating structure-aware features for graphs can be challenging in itself. For instance, the eigenvalues of a graph can be a suitable representation for its patterns of connectivity, but they have high storage and time requirements. A common shortcoming between these approaches is the need to perform matrix inversions, where the graphs are too sparse to be invertible. Another property of graph summarization techniques should be their interpretability. Revealing structural information such as communities, node roles or maximum independent sets can be very useful in further analysis of graphs.

To address these issues we propose an approach for detecting graph anomalies based on the ranking of the nodes. The novelty of our method lies in a scalable pre-processing scheme that produces stable results. Our matrix re-ordering approach efficiently assigns ranks to each node in the graph, where the resulting ranks can be used directly as a basis for comparing consecutive graph snapshots. Our re-ordering approach reduces the input dimension of a graph from $O(n^2)$ to $O(n)$. We can easily use a rank correlation coefficient as a similarity measure over pairs of graphs. Another advantage of our approach is its capability to produce interpretable results that identify large independent sets. The compact representation of the graphs yields faster and simpler anomaly detection schemes.

We review some of the algorithms previously introduced in the domain of graph anomaly detection in Section 2. We then define our notation and outline the problem statement in Section 3. The details of the proposed method and its properties are summarized in Section 4. The benchmark datasets in addition to the baseline algorithms for comparison are discussed in Section 5. We then show the results of anomaly detection and discuss the scalability and stability of our algorithm in Section 6. Finally, we conclude the paper and present future directions for research in Section 7.

2 Related Work

One of the most valuable tasks in data analysis is to recognize what stands out in a dataset. This type of analysis provides actionable information and improves our knowledge of the underlying data generation scheme. Various approaches have been developed for detection of such abnormalities [5], however many of these techniques disregard relational datasets where data instances demonstrate complex inter-dependencies. Due to the abundance and cross-disciplinary property of relational datasets, graph-based anomaly detection techniques have received growing attention in social networks, web graphs, road map networks and so forth [3, 9].

We review some of the dominant techniques for the detection of anomalies. We focus on graphs that are plain where nodes and/or edges are not associated with attributes and the nodes are consistently labeled over time.

2.1 Graph-based Anomaly Detection

Several approaches to pattern mining in graphs stem from distance based techniques, which utilize a distance measure in order to detect abnormal vs. normal structures. An example of such an approach is the k-medians algorithm [10], which employs graph edit distance as a measure of graph similarity. Other approaches take advantage of graph kernels [17], where kernel-based algorithms are applied to graphs. They compare graphs based on common sequences of nodes, or subgraphs. However, the computational complexity of these kernels can become problematic when applied to large graphs.

Other graph similarity metrics use the intuition of information flow when comparing graphs. The first step in these approaches is to compute the pairwise node affinity matrices in each graph and then determine the distance between these matrices. There are several approaches for determining node affinities in a graph, such as Pagerank and various extensions of random walks [7]. Another recent approach in this category is called Delta connectivity, which can be used for the purpose of anomaly detection. This approach calculates the graph distance by comparing node affinities [18]. It measures the differences in the immediate and second-hop neighborhoods of graphs. These approaches also suffer from the curse of dimensionality in large graphs.

Moreover, there are approaches that try to extract properties such as graph centric features before performing anomaly detection. These features can be computed from the combination of two, three or more nodes, i.e., dyads, triads and communities. They can also be extracted from the combination of all nodes in a more general manner [1]. Many anomaly detection approaches [15] have utilized graph centric features in their process of anomaly detection. Since the graph is summarized as a vector of features, the problem of graph-based anomaly detection transforms to the well-known problem of spotting outliers in an n-dimensional space. Therefore standard unsupervised anomaly detection schemes such as ellipsoidal cluster based approaches can be employed [21]. A thorough survey of such techniques can be found in [5]. It is worth noting that the extracted features cause information loss that can affect the performance of the anomaly detection scheme.

Another approach for graph mining is tensor decomposition. These techniques represent the time-evolving graphs as a tensor that can be considered as a multidimensional array, and perform tensor factorization. Tensor factorization approximates the input graph, where the reconstruction error can highlight anomalous events, subgraphs and/or vertices [22].

Although this field of research has received growing attention in recent years, the problem of scalability and interpretability of results still remains. Graph-centric features can reduce the dimensionality of the input graphs, but they may not be able to provide visually interpretable results. On the other hand, decomposition-based methods provide meaningful representations of graphs but suffer from the curse of dimensionality. The trade-off between these two issues has motivated us to find a compact representation of graphs that preserves the structural properties of networks. This can help further analysis of the data to become computationally efficient. Specifically for the task of anomaly detection, we provide experiments that demonstrate the efficiency and utility of our approach.

3 Preliminaries and Problem Statement

We start by describing the basic notation and assumptions of our anomaly detection task. A graph $G = (V, E)$ is defined as a set of nodes V and edges $E \subseteq V \times V$, where an edge $e \in E$ denotes a relationship between its corresponding nodes v_i, v_j. The degree d_i of a vertex v_i is defined as the sum of the number of its incoming (in-degree) and outgoing (out-degree) edges. A Maximum Independent Set (MIS) is the largest subset of vertices $V_{MIS} \subseteq V$ such that there is no edge between any pair of vertices in V_{MIS}.

The maximum independent set problem is closely related to common graph theoretical problems such as maximum common induced subgraphs, minimum vertex covers, graph coloring, and maximum common edge subgraphs. Finding MISs in a graph can be considered a sub-problem of indexing for shortest path and distance queries, automated labeling of maps, information coding, and signal transmission analysis [20].

Graphs are often represented by binary adjacency matrices, $A_{n \times n}$, where $n = |V|$ denotes the number of nodes. An element of the adjacency matrix $a_{ij} = 1$ if there is an edge from v_i to v_j. The simultaneous re-ordering of rows and columns of the adjacency matrix is called matrix permutation.

We formulate the problem of anomaly detection as follows: Given a sequence of graphs $\{G\}_{1...m}$, where m is the number of input graphs, we want to determine the time stamp(s), $i \in \{1...m\}$, when an event has occurred and changed the structural properties of the graph G_i. We consider the following assumptions about the input graphs:

- The vertices and edges in the graph are unweighted.
- There is no external vertex ordering.
- The input graphs are plain, i.e., no attributes are assigned to edges or vertices.
- The number of nodes remains the same throughout the graph sequence.
- The labeling of nodes between graphs is consistent.

An important issue for the design of a scalable anomaly detection scheme is the number of input features or dimensions that are required to be processed. If a graph-based anomaly detection uses a raw adjacency matrix as input, then the input dimensionality is $O(n^2)$, which is impractical for large graphs. In order to address the issue of scalability, we need to find a compact representation for each graph. We propose a pre-processing algorithm that extracts a rank feature for each node that is associated with the maximum independent sets in each graph. Therefore, instead of storing and processing an adjacency matrix of size $n \times n$, we reduce the input dimensionality and computational requirements for our anomaly detector to n.

For each graph in the sequence $\{G_1 = (V_1, E_1), G_2 = (V_2, E_2), ..., G_m = (V_m, E_m)\}$, we determine the new matrix re-ordering vector $\{V_1^{'}, V_2^{'}, ..., V_m^{'}\}$. We then compute the rank correlation coefficient between every two consequent tuples, $(V_i^{'}, V_{i+1}^{'})$. We employ the Spearman rank correlation coefficient as shown in Equation 1 between two input rank vectors, $\overrightarrow{V}'_i$, $\overrightarrow{V}'_{i+1}$, where $d_i = v_i - v_{i+1}$:

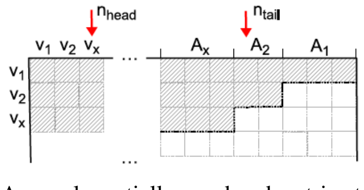
$$\rho = 1 - \frac{6 \sum {d_i}^2}{n(n^2 - 1)} \tag{1}$$

The computational complexity of Equation 1 is $O(n)$, where n is the length of the input vectors. The intuition behind our approach is to design a stable and scalable algorithm for determining the significance of each node and revealing structural information by manipulating the adjacency matrix $A_{n \times n}$. We need to find a matrix permutation that satisfies the following properties:

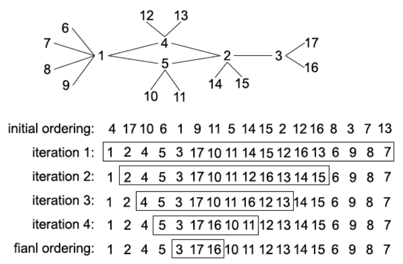
- *Locality*: Non-zero elements of the matrix should be in close vicinity in the ordering after the permutation.
- *Stability*: The initial ordering of the rows and columns should have no effect on the final outcome of the re-ordering.
- *Scalability*: The algorithm should have low computational complexity in order to handle large scale graphs.
- *Interpretability*: The permuted matrix should reveal structural information such as MISs about the graph.

4 Our Approach: Amplay

In order to achieve the above objectives, we propose an approach entitled Amplay (Adjacency matrix permutation based on layers). In each iteration, Amplay sorts vertices according to their total degree, and picks the vertex with the highest degree. Ties are resolved according to the ordering in the previous iteration. We then remove the vertex and its incidental edges, and recursively apply the algorithm. The outline of the re-ordering approach is given in Algorithm 1. In order to clarify the process of Amplay implementation, we have provided an example of Amplay operation in Figures a and b.



(a) A sample partially reordered matrix at the beginning of iteration 3 of the Amplay algorithm. In this iteration, v_x will be placed at position n_{head}, and A_x will be placed immediately before position n_{tail}. By definition, in each iteration, A_x are vertices that are only incidental to vertices placed before v_x or to v_x, which results in a zero area at the bottom right corner. White squares denote zero elements. Elements at gray squares can contain 0 or 1.



initial ordering:	4 17 10 6 1 9 11 5 14 15 2 12 16 8 3 7 13
iteration 1:	1 2 4 5 3 17 10 11 14 15 12 16 13 6 9 8 7
iteration 2:	1 2 4 5 3 17 10 11 12 16 13 14 15 6 9 8 7
iteration 3:	1 2 4 5 3 17 10 11 16 12 13 14 15 6 9 8 7
iteration 4:	1 2 4 5 3 17 16 10 11 12 13 14 15 6 9 8 7
fianl ordering:	1 2 4 5 3 17 16 10 11 12 13 14 15 6 9 8 7

(b) Amplay ordering for a sample graph. Each row shows an ordering at the end of each iteration. Rectangles outline sets V_i at the beginning of each iteration.

Algorithm 1: Amplay Permutation

```
Input  : Graph $G = (V, E)$ and $n = |V|$
Output: Node re-ordering $V \rightarrow V\prime$
1  $n_{head} = 1$; $n_{tail} = n + 1$;
2  $i = 1$; $V_i = V$; $E_i = E$; $G_i = (V_i, E_i)$;
3  while $n_{head} < n_{tail}$ do
4      Sort $V_i$ according to the degrees of vertices;
5      Resolve ties using previous ordering;
6      $v_x \in V_i \leftarrow$ a vertex with the maximum total degree;
7      $e_x \subseteq E_i \leftarrow$ edges incidental to $v_x$ in $G_i$;
8      $A_x \subseteq V_i \leftarrow$ vertices incidental only to $v_x$ in $G_i$;
9      $a_i = |A_x|$;
10     Place $v_x$ in position $n_{head}$;
11     Place $A_x$ in position $n_{tail} - a_i, ..., n_{tail} - 1$;
12     (preserving ordering of vertices $A_x$ from $G_i$);
13     $V_{i+1} = V_i \backslash v_x \cup A_x$($\backslash$ denotes set difference);
14     $E_{i+1} = E_i \backslash e_x$;
15     $n_{head} = n_{head} + 1$;
16     $n_{tail} = n_{tail} - a$;
17     $i = i + 1$
```

One of the interesting properties of Amplay is its capability to reveal MISs associated with each input graph. Figure 2 shows the permuted adjacency matrix of the Enron email dataset where the MISs are denoted as $S_1, S_2, ...$ The groupings of nodes into the MISs indicates that Amplay can be used as a heuristic to determine the MISs of a graph in various problem domains. A prominent feature of the matrices produced by the Amplay method is a front line such that all non-zero matrix elements are located above the line. Indeed, we can consider an adjacency matrix as a grid with integer coordinates. Here the first coordinate spans rows from top to bottom, the second coordinate spans columns from left to right. We define the front line as follows: $(1, n), (1, n - a_1 + 1), (2, na_1), (2, n - a_1 - a_2 + 1), ..., (s, s), ..., (n - a_1 + 1, 1), (n, 1)$, where $\{a_i\}$ is the sequence produced by Algorithm 1 and s is the number of iterations of the algorithm.

Lemma 1. *Every matrix element below the front line is zero.*

Proof. The front line spans intersections of vertices from sets A_x with their respective v_x. By definition, A_x are vertices that are only incidental to vertices placed before v_x or to v_x, which implies that matrix elements below and to the right from the intersections of A_x and v_x are zero.

As we explain below, the front line is important in visualization, because it allows us to grasp (1) the degree distribution of the graph, and (2) the relative size of the largest independent set revealed by Amplay. Note that the shape of the front line is defined by the sequence $\{a_i\}$, where a_i is closely related to the degree of the vertex placed at position i. As a consequence, the front line reflects the degree distribution in a graph. Here, the point of intersection of the front line with the matrix diagonal tends to shift in accordance to the ratio of the mean degree to the maximum degree. However, graphs with a similar ratio can still be visually distinguishable, if their structure is different. A key property of Amplay is multiple vertex sorting. Recall that at each iteration, vertices are sorted according to the total degree of the remaining graph, and ties are resolved using the ordering from the previous iteration. Such a sorting has two consequences. First, the resulting index of each vertex depends not only on the vertex degree, but also on a vertex connectivity pattern (e.g., the number of connections to high-degree nodes). This pattern is reflected in the positions of the vertex in subsequent sorting rounds. While many vertices can have the same degree, the vertices tend to differ in their connectivity patterns. As such, Amplay tends to produce a relatively deterministic ordering. This in turn results in a relatively small variance in the behavior of subsequent graph processing algorithms. Second, vertices that have a similar connectivity pattern will have similar positions during sorting across subsequent iterations, and thus have similar positions in the resulting Amplay ordering. This explains why Amplay tends to produce matrices with a smooth visual appearance.

Lemma 2. *Graph $G = (V, E)$ contains an independent set with at least $n - n_{tail}$ vertices, where n_{tail} is the value from Amplay at the moment of termination.*

Proof. At the end of each iteration of Amplay, vertices assigned to indices larger than or equal to n_{tail} are incidental only to vertices assigned to indices smaller than n_{head}. At the point of termination $n_{head} = n_{tail}$. Hence, vertices assigned to indices larger than n_{tail} are pairwise disjoint and form an independent set.

In addition to revealing structural properties of the graph, Amplay proves to be scalable. We describe the computational complexity of this re-ordering approach in Lemma 3.

Lemma 3. *The complexity of Amplay is $O(\sum_{i=0}^{s} n_i \log n_i)$ where $n_i = |V_i|$ defined in Amplay, and $s \leq |V|$ is the number of iterations.*

Proof. Each iteration of the algorithm operates on a subgraph with n_i vertices, and involves sorting (which can be performed in $O(n_i \log n_i)$ time), finding neighbors of the chosen vertex v_x (linear in n_i), and removing incidental edges (linear in n_i). As such the overall complexity of one iteration is bounded by $O(n_i \log n_i)$ and the total complexity is bounded by $O(\sum_{i=0}^{s} n_i \log n_i)$.

It is worth mentioning that in many real-world graphs, n_i rapidly decreases, which reduces the total running time. Moreover, we can improve the scalability of Amplay further, by choosing k vertices with the largest total degrees, place them, and advance the n_{head} pointer by k at each iteration (line 6 in Algorithm 1). Furthermore, in line 8 of Algorithm 1, we can define A_x as a set of vertices incidental only to the chosen k vertices. The front line is now defined as $(k, n), (k, n - a_1 + 1), (2k, n - a_1), (2k, n - a_1 - a_2 + 1), ..., (s.k, s.k), ..., (n - a_1 + 1, k), (n, k)$, and it is easy to verify that Lemmas 1 and 2 hold. If we increase k, we can see that the prominent structural features of the graph are preserved. Moreover the computational complexity of Amplay when $k > 1$ is $O(\sum_{i=0}^{s'} n_i' \times r_i)$ where $r_i = \max(\log n_i', k)$. Using $k > 1$ is beneficial because it reduces the number of iterations s', and sequence n_i' decreases faster than n_i.

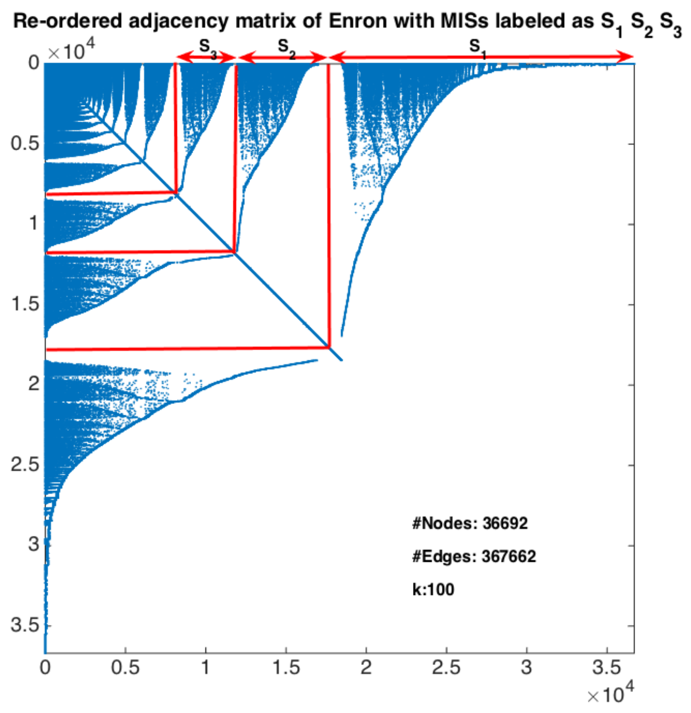


Fig. 2: The Amplay re-ordered adjacency matrix of the Enron email dataset.

5 Evaluation Methodology

In this section, we describe each dataset used in our experiments and elaborate on the baseline algorithms for comparison.

5.1 Benchmark Datasets

For the purpose of anomaly detection, we have selected a representative sample of sparse real-world datasets. The first real dataset that we have used is the Facebook wall posts data collected from September 26th, 2006 to January 22nd, 2009 from users in the New Orleans network [24]. The number of users in this graph is 90,269, however only 60,290 users exhibited activity.

The next real benchmark dataset is the Autonomous Systems (AS) dataset [19]. The graphs comprising the AS dataset represent snapshots of the backbone Internet routing topology, where each node is an AS that corresponds to a subnetwork in the Internet. The edges in this graph represent the traffic flows exchanged between two neighbors. The dataset consists of 733 days from November 8, 1997 to January 2, 2000 with nodes being added or deleted from the network.

Another real dataset is the Enron email network that gathers the email communications within the Enron corporation from January 1999 to January 2003 [8]. There are 36,692 nodes in this network, where each node corresponds to an email address. We have used the nodes with a minimum activity level and reduced the graph to 184 nodes.

The final real dataset is the DBLP [6] dataset that consists of co-authorship information in computer science. The number of nodes is 1,631,698 and the data is gathered from 1954 to 2010. The description of these datasets is summarized in Table 1. DBLP graphs are used to test the scalability of our approach.

Table 1: Benchmark Description.

Dataset	Type	Description	#Nodes	#Time Stamps
AS	Undirected	Communication	65,535	733
Facebook	Undirected	Social	60,290	1,495
Enron	Directed	Communication	184	893
DBLP	Undirected	Co-authorship	1,631,698	57

5.2 Baseline Algorithm

For the purpose of comparison, we have used a recent approach for computing graph similarity with applications in anomaly detection as our baseline. This algorithm is called delta connectivity (DeltaCon) [18], where the node affinity matrices for each graph are calculated using a belief propagation strategy shown in Equation 2. This approach considers first-hop and second-hop neighborhoods for calculating the influence of the nodes on each other and has been proven to converge.

$$S = [s_{ij}] = [I + \eta^2 D - \eta A^{'-1}] \tag{2}$$

After determining the node affinity matrices, they compare the consecutive graphs by calculating the root Euclidean distance shown in Equation 3, which varies in the range $[0, 1]$. We empirically have chosen $\eta = 0.1$ in our experiments.

$$sim(S_1, S_2) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{j=n} (\sqrt{S_{1,ij}} - \sqrt{S_{2,ij}})^2} \tag{3}$$

The computational complexity of this algorithm is reported to be linear in the number of edges of each graph, $O(|E|)$.

Another baseline algorithm is an approach called Random Projection (RP) that has shown to be effective in determining anomalous graphs in block-structured networks [23]. The intuition behind RP comes from the Johnson and Lindenstrauss lemma [13] as presented in Lemma 4. This lemma asserts that a set of points in Euclidean space, $P^{1\ldots n} \in \mathbb{R}^{n\times m}$, can be embedded into a d-dimensional Euclidean space, $P'^{1\ldots n} \in R^{n\times d}$ while preserving all pairwise distances within a small factor ϵ with high probability.

Lemma 4. *Given an integer n and $\epsilon > 0$, let d be a positive integer such that $d \geq d_0 = O(\epsilon^{-2} \log n)$. For every set P of n points in $\mathbb{R}^m$, there exists $f : \mathbb{R}^m \to \mathbb{R}^d$ such that with probability $1 - n^{-\beta}$, $\beta > 0$, for all $u, v \in P$*

$$(1-\epsilon)||u-v||^2 \leq ||f(u) - f(v)||^2 \leq (1+\epsilon)||u-v||^2 \tag{4}$$

[6] http://dblp.uni-trier.de/xml/

One of the algorithms for generating a random projection matrix that has been shown to preserve pairwise distances [13] is presented in Equation 5:

$$r_{ij} = \sqrt{3} \begin{cases} +1 & with\, probability\, 1/6 \\ 0 & with\, probability\, 2/3 \\ -1 & with\, probability\, 1/6 \end{cases} \tag{5}$$

6 Results and Discussion

In this section, we outline our experimental setup in five sections and report the observed results. We first demonstrate the effectiveness of Amplay and rank correlation in prioritizing nodes that can contribute the most to the structural change in consecutive graphs. We then investigate the capability of our algorithm in detecting anomalous graphs based on the produced similarity score. Thereafter, we discuss the scalability of our approach empirically by changing parameter k. We provide our empirical studies regarding the stability of the Amplay algorithm on static graphs.

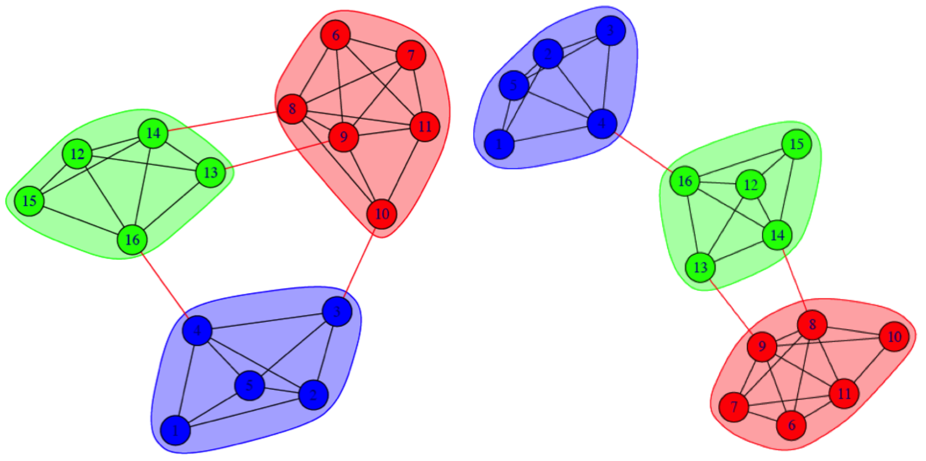
Experiment I: Gradual Change Detection The effectiveness of Amplay lies in its ability to reveal maximum independent sets. The nodes that comprise each set can be considered the most influential nodes collected from every community in the graph. Figure 3 shows the gradual change in the graph structure by removing the edge $e_{3,10}$ connecting v_3 and v_{10}. $e_{3,10}$ is the connecting bridge between two of the present communities in the graph and its elimination may lead to discontinuity in the entire graph structure. As can be seen, v_3 is the node that contributes the most to the dissimilarity between G_1 and G_2.

Experiment II: Anomaly Detection We have applied the proposed approach (with parameter $k = 1$) and the baseline algorithms on the benchmark datasets, and compared their computed similarity score between consecutive days. The implementations were run in Matlab using a machine with a 3GHz Processor and 8GB RAM. Due to the computational complexity of the random projection approach, we only use this algorithm as a baseline for comparing scalability.

Our proposed method and DeltaCon generate scores in the range $[0, 1]$. Figures 4, 5 and 6 demonstrate the graph similarity scores for the Autonomous Systems, Facebook and Enron datasets respectively. As can be seen, the trend of similarity scores is the same for DeltaCon and our proposed method.

Experiment III: Computational Scalability The reported results for anomaly detection were achieved by setting parameter $k = 1$, where k was defined at the end of Section 4 as the number of vertices that are processed and removed from the graph in a single iteration. We decided to increase k and investigate the performance of our anomaly detection scheme. It is worth recalling that we are using only a subset of nodes for the purpose of anomaly detection. We consider the top l elements in the rank vectors where $l = n_{head}$ after the termination of Amplay.

Increasing parameter k leads to an exponential decrease in computation time. This observation can be explained by the sparsity of real-world graphs, i.e., the small proportion of fully-connected cliques. Since k is the number of vertices that are processed and removed from the graph within a single iteration, increasing k leads to a more rapid



(a) G_1: Snapshot at $t = 1$ (b) G_2: Snapshot at $t = 2$

Fig. 3: Example of gradual change in the structure of the graph and the importance of each node in the overall similarity score.

Initial Node Ordering for G_1, G_2: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Amplay and Rank Correlation Node Importance: 3, 5, 13, 15, 6, 7, 1, 2, 4, 8, 9, 10, 11, 12, 14, 16
DeltaCon Node Importance: 3, 10, 14, 16, 12, 13, 2, 5, 15, 4, 6, 7, 11, 1, 9, 8

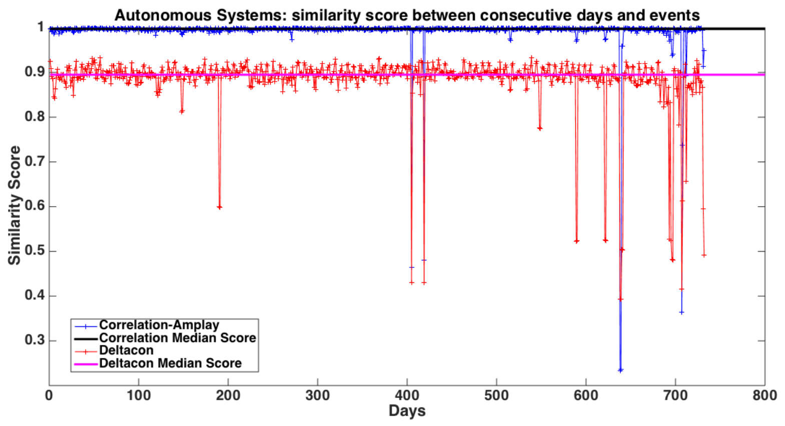


Fig. 4: Comparison of graph similarity scores based on the correlation score of the Amplay-permuted adjacency matrix and DeltaCon on the Autonomous Systems dataset.

graph reduction. However, at some value of k, all highly connected vertices are processed within a single iteration, and the remaining graph contains only vertices with low degrees. Therefore, subsequent increases of k do not lead to a significant performance improvement. Figure 7 demonstrates the effect of parameter k on the processing time of Amplay for the Enron dataset. Although the parameter k is increased to 100, we can still observe the maximum independent sets $S_1, S_2, ..., S_n$ as demonstrated in

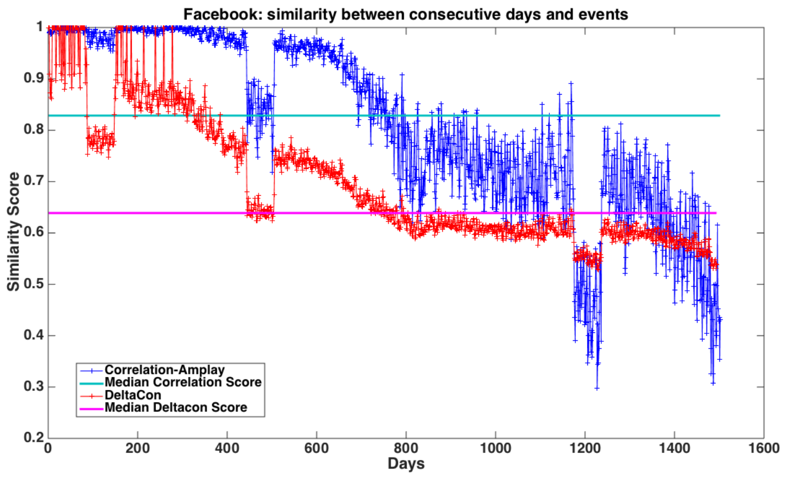


Fig. 5: Comparison of graph similarity scores based on the correlation score of the Amplay-permuted adjacency matrix and DeltaCon on the Facebook dataset.

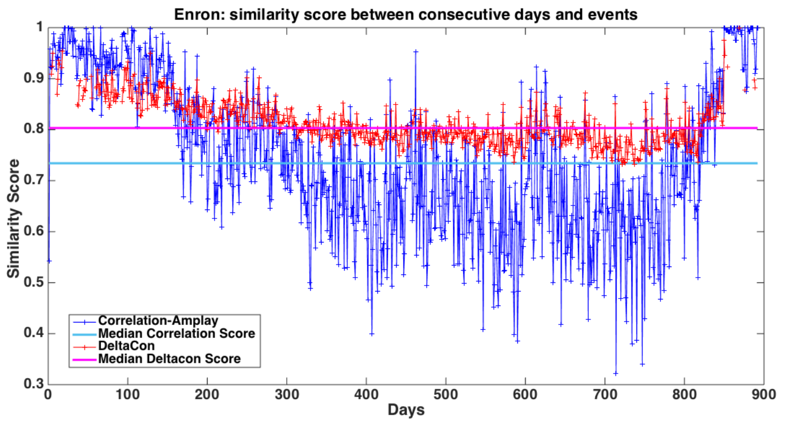


Fig. 6: Comparison of graph similarity scores based on the correlation score of the Amplay-permuted adjacency matrix and DeltaCon on the Enron dataset.

Figure 2. Another attractive property of our scheme is the compact representation of the graph produced by Amplay. This compact representation scales linearly in the number of input nodes n. The real-world graphs are mainly comprised of sets of dense cores and sparse periphery nodes. Therefore, the number of nodes to consider for graph similarity computation is only a fraction of the total number of nodes in a graph. Amplay discards the peripheral nodes that are connected to only a few vertices from the core. The influential nodes usually appear as $V_1^{'}, V_2^{'}, ..., V_{n_{head}}^{'}$, where $n_{head} << n$. The upper bound of n denotes the worst case scenario where the input graph is fully-connected. Table 2 demonstrates the computation time and number of considered nodes in calculating graph similarity. The upper bounds for time complexity of the embedding approaches is demonstrated in Table 3. As can be seen, our proposed method and DeltaCon outper-

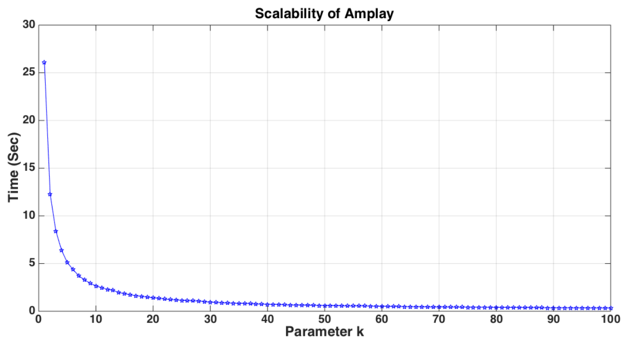


Fig. 7: Amplay computation time as the parameter k is increased in the Enron dataset where k is the number of vertices that are processed and removed from the graph in a single iteration.

Table 2: Computation time of Amplay on different datasets.

Dataset	Amplay Time	DeltaCon Time	#Nodes	#Nodes to Consider
AS	0.1956 ± 0.0049	$0.0872 \pm$ 3.6162e-04	65,535	1,913
Facebook	0.05376 ± 0.0029	$0.0721 \pm$ 4.4818e-04	60,290	1,316
Enron	$0.00093874 \pm 0.00077612$	$0.0029 \pm$ 8.2857e-07	184	41
DBLP	$29.7074 \pm$ 6.2685e+03	1.7174 ± 0.1998	1,631,698	38,903

form random projection, and both are scalable when the adjacency matrices are sparse. The advantage of our approach lies in its ability to generate an interpretable result where structural features of a graph, such as MISs, are revealed as shown in Figure 2.

Table 3: Computational Complexity for Baseline and Proposed Approaches.

Approach	Embedding Complexity + Similarity Complexity
Amplay - Rank Correlation	$O(\sum_{i=0}^{s} n_i \log n_i) + O(\lvert V \rvert)$
DeltaCon - Euclidean Distance	$O(\lvert E \rvert) + O(\lvert V \rvert)$
Random Projections - Euclidean Distance	$O(n^2 d) + O(\lvert V \rvert)$

Experiment IV: Amplay Stability We compare Amplay with other ordering methods, namely random, RCM [6], and SlashBurn [14]. Random permutation serves as a naive baseline; RCM is a classical bandwidth reduction algorithm [6]; and SlashBurn is a recent method that is shown to produce adjacency matrices with localized non-zero elements. This method is shown to be one of the best state-of-the-art methods [14].

We use a representative sample of sparse real-world graphs of different sizes for quantitative evaluation (Table 4) where all graphs were downloaded from the Stanford Large Network Collection [2]. The table shows graph names as they appear in the Collection, however in the paper we use simplified names (e.g., gnutella instead of p2p-Gnutella08).

We first load each graph as an adjacency matrix S and produce $N+1$ random permutations of the graph vertices $RND_i(S), i = 0, 1, ..., N$. We then take each random permutation as input and either leave it as it is (method Random), or apply RCM, SlashBurn

[2] snap.stanford.edu/data

or Amplay permutation -respectively, $RCM(RND_i(S))$, $SlashBurn(RND_i(S))$ and $Amplay(RND_i(S))$.

We then evaluate ordering stability by selecting one of the random permutations as a reference (e.g., $i_{ref} = 0$), and comparing the vertex ordering between each of the other permutations and the reference (e.g., compare $RND_0(S)$ with $RND_j(S)$). In this section, we use both Amplay and SlashBurn with $k = 1$. That is, we evaluate the basic forms of these algorithms, as opposed to more coarse scalable versions.

We compare two vertex orderings using the Kendall correlation coefficient. This coefficient takes values in $[-1, 1]$, where 1 is reached in the case of equivalence of the orderings. If the two orderings are independent, one would expect the coefficient to be approximately 0.

Intuitively, vertices with higher degrees tend to have a higher impact on matrix operations and visual appearance. Therefore, we also separately look at ordering stability for higher degree vertices only. Specifically, we compute the Kendall correlation while ignoring a certain proportion $(0, 80, 90, 95\%)$ of vertices with low degrees. Here 0% means that we compare orderings for all graph vertices. On the other hand, 95% means that we only consider the ordering of the top 5% of vertices with the highest degrees. We present our results in Figure 8 and Table 5 (permutations with $k = 1$ were slow for large graphs, therefore we have fewer runs for large graphs). Overall, Amplay outperforms the other methods by a large margin ($p < 0.01$, Wilcoxon signed rank test). In other words, Amplay tends to be less dependent on the input ordering.

Table 4: Real-world graphs used in our stability analysis. Asterisks mark undirected graphs.

Dataset	Vertices	Edges
ca-HepTh	9877	51971*
Wiki-V ote	7115	103689
p2p-Gnutella08	6301	20777
oregon1_010331	10670	22002*
email-Enron	36692	367662*
soc-epinions1	75879	508837
Email-EuAll	265214	420045
loc-Gowalla	196591	1900654*
flickr	105936	2300660*

Table 5: Stability measured with Kendall Tau at 90% for large graphs. The table shows the means for three comparisons.

Dataset	Random	RCM	SlashBurn	Amplay
Email-Eu	< 0.01	0.02	0.11	0.46
gowalla	< 0.01	0.41	0.78	0.89
flicker	< 0.01	0.27	0.05	0.99

7 Conclusion and Future Work

In this paper, we presented an unsupervised approach for detecting anomalous graphs in time-evolving networks. We created a compact yet structure-aware feature set for each graph using a matrix permutation technique called Amplay. The resulting feature set included the rank of each node in a graph and this rank ordering was used by rank correlation for comparing a pair of graphs. This simple yet effective approach overcomes the issues of scalability when handling large-scale graphs. We showed the low time complexity and structure-aware property of our re-ordering approach both empirically and theoretically. Moreover, we designed experiments for the purpose of anomaly

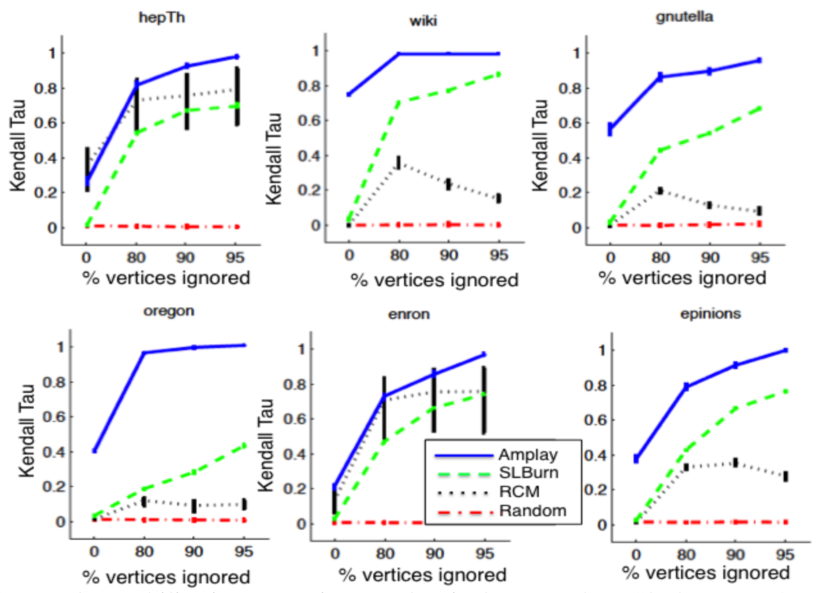


Fig. 8: Amplay stability in comparison to the rival approaches, SlashBurn [14], RCM [6] and Random ordering, as we vary the percentage of vertices with low degree that we ignored.

detection in four real datasets, where our approach was compared against an effective graph similarity method and proved to be successful in highlighting abnormal events. In future work, we will explore the possibilities of reducing the dimensionality of the graph even further by using a random projection approach. Since we reduce the dimensionality from $O(n^2)$ to $O(n)$, we can consider the rank vectors of each graph as a data stream. Thereafter, we will investigate a window-based approach for determining anomalous graphs given a history of past normal instances.

References

1. Leman Akoglu, Pedro OS Vaz de Melo, and Christos Faloutsos. Quantifying reciprocity in large weighted communication networks. In *Advances in Knowledge Discovery and Data Mining*, pages 85–96. Springer, 2012.
2. Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.
3. Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Netsimile: a scalable approach to size-independent network similarity. *arXiv preprint arXiv:1209.2684*, 2012.
4. Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.
5. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.
6. Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference ACM*, pages 157–172. ACM, 1969.

7. Gianna M Del Corso, Antonio Gulli, and Francesco Romani. Fast pagerank computation via a sparse linear system. *Internet Mathematics*, 2(3):251–273, 2005.
8. Jana Diesner, Terrill L Frantz, and Kathleen M Carley. Communication networks from the Enron email corpus it's always about the people. Enron is no different. *Computational & Mathematical Organization Theory*, 11(3):201–228, 2005.
9. Qi Ding, Natallia Katenka, Paul Barford, Eric Kolaczyk, and Mark Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 886–894. ACM, 2012.
10. Miquel Ferrer, Ernest Valveny, Francesc Serratosa, Itziar Bardají, and Horst Bunke. Graph-based k-means clustering: A comparison of the set median versus the generalized median graph. In *Computer Analysis of Images and Patterns*, pages 342–350. Springer, 2009.
11. Ulrich Flegel, Julien Vayssière, and Gunter Bitz. A state of the art survey of fraud detection technology. In *Insider Threats in Cyber Security*, pages 73–84. Springer, 2010.
12. Shunsuke Hirose, Kenji Yamanishi, Takayuki Nakata, and Ryohei Fujimaki. Network anomaly detection based on eigen equation compression. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1185–1194. ACM, 2009.
13. William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz Mappings into a Hilbert Space. *Contemporary Mathematics*, 26(189-206):1, 1984.
14. U Kang and Christos Faloutsos. Beyond'caveman communities': Hubs and spokes for graph compression and mining. In *IEEE 11th International Conference on Data Mining (ICDM)*, pages 300–309. IEEE, 2011.
15. U Kang, Duen Horng Chau, and Christos Faloutsos. Mining large graphs: Algorithms, inference, and discoveries. In *IEEE 27th International Conference on Data Engineering (ICDE)*, pages 243–254. IEEE, 2011.
16. U Kang, Spiros Papadimitriou, Jimeng Sun, and Hanghang Tong. Centralities in large networks: Algorithms and observations. In *SDM*, volume 2011, pages 119–130. SIAM, 2011.
17. U Kang, Hanghang Tong, and Jimeng Sun. Fast random walk graph kernel. In *SDM*, pages 828–838. SIAM, 2012.
18. Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *SIAM*, 2013.
19. Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 177–187. ACM, 2005.
20. Yu Liu, Jiaheng Lu, Hua Yang, Xiaokui Xiao, and Zhewei Wei. Towards maximum independent sets on massive graphs. *Proceedings of the VLDB Endowment*, 8(13):2122–2133, 2015.
21. Masud Moshtaghi, Christopher Leckie, Shanika Karunasekera, James C Bezdek, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Incremental elliptical boundary estimation for anomaly detection in wireless sensor networks. In *IEEE 11th International Conference on Data Mining (ICDM)*, pages 467–476. IEEE, 2011.
22. Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.
23. Lida Rashidi, Sutharshan Rajasegarar, and Christopher Leckie. An embedding scheme for detecting anomalous block structured graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 215–227. Springer, 2015.
24. Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.