Investigation of graph edit distance cost functions for detection of network anomalies

K. M. Kapsabelis[1] P. J. Dickinson[2] K. Dogancay[3]

(Received 14 July 2006; revised 3 August 2007)

Abstract

Computer networks are becoming ubiquitous. Accurately monitoring and managing the behaviour of these complex and dynamic networks is a challenging task. It has become crucial to develop and employ good network monitoring techniques that assist in identifying and correcting abnormalities that affect network reliability, performance, security and future planning. There has been significant research in the detection of change and anomalous events in computer networks. A recent novel approach represents the logical communications of a periodically observed network as a time series of graphs and applies the graph matching technique, graph edit distance, to monitor and detect anomalous behaviour in the network. To date, only simple cost functions for graph edit operations have been used in application to computer network monitoring. This article investigates simple normalisation and non-linear techniques in the graph edit distance cost function, to improve detection of specific traffic related network anomalies in the computer network domain.

Contents

1 Introduction

Computer networks are becoming far more complex and dynamic in nature. From a management perspective, network reliability, quality of service and security are essential issues faced by network operators. Types of anomalous changes that affect a network include device failures and performance problems; network, service or user reorganisation; and malicious intrusions or attacks. Rapid identification and handling of such abnormalities has become a very important, but challenging task.

There has been significant research in the detection of anomalous events in computer networks. Statistical techniques are used to consider changes in traffic distribution [10, 12], topology variations [20] and anomaly detection [17]. Network visualisation techniques are also used to monitor changes specific to telecommunications networks [1]. Finite state machine techniques are used for fault detection [11, 13], as are signature based approaches [9].

A novel approach, to complement the aforementioned techniques, is to treat a periodically observed network as a time series of graphs, and apply graph matching techniques to monitor and detect anomalous changes in the network [5, 7, 8, 16]. In this approach a network observation is represented by a labelled and directed graph. Individual network devices or users correspond to uniquely labelled graph vertices. Logical communications or traffic, exchanged between nodes, correspond to directed edges. Edges can also be weighted. Edge weight model the amount of traffic or number of communications from one node to another.

By observing the network over arbitrary time intervals and describing each observation as a graph, we build a time series of graphs that characterise the logical behaviour of the network over a period of time. A measure of graph distance determines the difference between two consecutive pairs of graphs. When repeated across the complete time series of graphs this method produces a time series of numbers that represents the amount of change the network exhibits over time. In this way, the problem of detecting abnormal change in computer networks shifts to discovering good graph matching techniques for this domain.

This study focuses on the graph edit distance (`ged`) measure. The `ged` works to find the minimum number of edit operations required to transform one graph (network) to another, to achieve graph isomorphism [3, 4, 15, 16]. A cost function associates a cost with each edit operation. Simple `ged` cost functions have been successfully employed for the general case of network change detection [8, 16]. This study explores new `ged` cost functions, to facilitate the detection of specific traffic related anomalies, which would otherwise be overlooked by the simple cost functions. Matlab simulations explore the proposed cost functions experimentally. The techniques produced from this research can benefit the development of advanced warning systems used for network monitoring.

Section 2 provides an overview of the definition of the network described as a graph. Section 3 describes the `ged` measure and Section 4 overviews the cost functions considered in this investigation. Section 5 describes the experimental procedure used to simulate a computer network implanted with specific traffic anomalies. This is followed by an overview and discussion of the results obtained from applying the proposed cost functions to detect these anomalies. Finally, Section 6 concludes.

2 The network as a graph

Computer network communications at the logical network layer and above are represented as graphs [5, 8]. A single *graph* G represents the logical communications of a computer network over an arbitrary observation time interval. Network nodes, including network devices or users, represent vertices in the graph. Communications, or information exchange between nodes, represent edges in the graph. A graph, denoted by $G = (V, E)$, consists of a finite set of vertices V and finite set of edges E [19]. Two vertices $u, v \in V$ are said to be adjacent if they are endpoints of the same edge [19]. Edges in G can be *directed*, denoted by the ordered vertex pair $(u, v) \in E$ which is indicative of communications sent from vertex u and received by vertex v. In this case, graph G is said to be a *directed graph*, or a *digraph*.

Objects such as vertices or edges (or their combinations) associated with a graph are referred to as elements of the graph [5, 8]. A *weight* is a function whose domain is a set of graph elements in G. The domain can be restricted to that of edge or vertex elements only, where the function is referred to as 'edge weight' or 'vertex weight', respectively. *Values* of weight assigned to elements in G may be numerical (for example, the amount of traffic as values of edge weight), or symbolic (such as node identifiers as values of vertex weight). The set of possible values in the range of weight function are called attributes. A unique labelling is a one-to-one weight. A vertex weight in this computer network representation is assumed to be unique and serves to identify the individual users or devices (for example, computer Internet Protocol addresses). A graph with a unique labelling is called a *labelled* graph.

The graph $G = (V, E, \alpha, \beta)$ representing a computer network is vertex labelled with weight $\alpha : V \to L_V$ assigning vertex identifier attributes L_V to individual vertices, where the value of the assigned vertex weight is $\alpha(u)$ for vertex u. Furthermore, $\alpha(u) \neq \alpha(v)$, for all $u, v \in V$, $u \neq v$, since the graph possesses a unique vertex labelling in this case. Edges are also weighted to indicate the amount of traffic or number of communications observed between nodes over a finite time interval, with weight $\beta : E \to \mathbb{R}^+$. The number of vertices in $G = (V, E, \alpha, \beta)$ is denoted by $|V|$, and likewise the number of edges is denoted by $|E|$.

3 Graph edit distance

In computer networks, a number of graph distance measures have been applied to a time series of graphs to investigate network behaviour over time [5]. This article focusses on the use of the graph edit distance (`ged`). The concept of this distance measure is to find the minimum number of edit operations required to transform one graph (network) to another, to achieve graph isomorphism. Graph isomorphism is where there exists an exact structural and label correspondence between two graphs [3, 5]. Edit operations in the `ged` include edge or node deletions and insertions, and edge weight substitutions [4, 15]. A cost function associates a cost with each edit operation. A cost is used to attribute a relative weighting for each operation, in comparison to the other operations that occur. Typically, the more likely an edit operation is to occur the smaller is its cost [3]. The resultant minimum edit cost is a measure of difference, or change, between two graphs [5].

Generally, using `ged` can result in different combinations when matching one graph to another, as vertex labels do not necessarily need to be unique or even labelled [3, 5, 8, 16]. Such a matching problem is considered to be an NP-complete problem and requires exponential time and space to find the optimal solution [3]. As previously stated, in computer networks, each vertex is assumed to be uniquely labelled, to represent individual nodes within the network. As a result, there is only one possible combination to match one graph with another, and the computational complexity required to calculate the `ged` becomes a linear time problem [5, 8, 16]. Consequently, vertex substitution is not a required or valid `ged` edit operation for this application [5, 8, 16].

In computer networks, the `ged` $d_1(G,H)$, between two labelled graphs $G=(V_G,E_G,\alpha_G,\beta_G)$ and $H=(V_H,E_H,\alpha_H,\beta_H)$, representing two consecutive observations of the communications in a computer network, is

$$\begin{aligned} d_1(G,H) &= \sum_{n\in V_G\setminus(V_G\cap V_H)} C_{nd}(n) + \sum_{n\in V_H\setminus(V_G\cap V_H)} C_{ni}(n) + \sum_{e\in E_G\cap E_H} C_{es}(e) \\ &\quad + \sum_{e\in E_G\setminus(E_G\cap E_H)} C_{ed}(e) + \sum_{e\in E_H\setminus(E_G\cap E_H)} C_{ei}(e)\,, \end{aligned} \tag{1}$$

where $C_{nd}(n)$ is the cost of deleting a node n; $C_{ni}(n)$ is the cost of inserting a node n; $C_{es}(e)$ is the cost of substituting an edge weight for an edge e; $C_{ed}(e)$ is the cost of deleting an edge e; $C_{ei}(e)$ is the cost of inserting an edge e; and thus the cost function is $C_f(n,e)=(C_{nd}(n),C_{ni}(n),C_{es}(e),C_{ed}(e),C_{ei}(e))$.

The underlying cost function has a crucial influence on the performance of the `ged` matching technique and inferring the cost of edit operations is largely dependant on the problem domain [2].

Specific to network change detection, a number of simple `ged` cost functions have been proposed in past research. One such measure that considers both network topology and traffic [8] is

$$d_2(G,H) = c\left[|V_G|+|V_H|-2|V_G\cap V_H|\right] + \sum_{e\in E_G\cap E_H} |\beta_G(e)-\beta_H(e)| + \sum_{e\in E_G\setminus(E_G\cap E_H)} \beta_G(e) + \sum_{e\in E_H\setminus(E_G\cap E_H)} \beta_H(e)\,. \tag{2}$$

In this measure a unity cost for any node edit operation (deletion or insertion) is applied. Thus the total cost for all node edit operations is described simply as the difference between the total number of vertex elements in both graphs and all graph vertex elements in common. A positive constant c is applied to each node cost, which allows the importance to be placed on node operations relative to the weight changes on the edges [8]. To simplify notation this measure considers the graphs to be fully connected and an edge weight of zero is assigned to each edge where $e \in E_G$ ($e \in E_H$) does not exist in G (H) [8]. Thus, edge insertion, deletion, and edge weight substitutions are treated uniformly. The cost of edge weight substitution is defined by the absolute difference between weights for each edge in both graphs. As such, the cost $C_{es}(e)$ of changing weight $\beta_G(e)$ on edge $e \in E_G$ into $\beta_H(e)$ on edge $e \in E_H$ is defined as $|\beta_G(e)-\beta_H(e)|$ [8]. In the case of a deletion of edge $e \in E_G$ with weight $\beta_G(e)$, a cost of $\beta_G(e)$ is incurred, and alternately in the case of inserting an edge $e \in E_H$ there is a cost of $\beta_H(e)$.

This simple `ged` measure (d_2) performs quite successfully for the general case of detecting network change in both topology and traffic. It is reactive to any change in traffic that occurs in the network and does not measure the *relative* change experienced on each edge in the network.

4 Proposed `ged` cost functions

For the purpose of network anomaly detection, it is important that the `ged` measures are particularly sensitive to relatively significant traffic variation, but insensitive to random variations that can occur during normal network operation. Under normal operating conditions network devices communicate at different frequencies and traffic rates, depending on the nature of communications. As such, the observed traffic flows resultant from such communications have vastly different absolute traffic values (edge weights). Therefore is is important to consider the relative change observed in a traffic flow (edge), to more accurately measure significant or abnormal change over typical traffic variations.

Example anomalies of interest to network operators include device failures or malfunctions, such as a runaway process on a device paging across the network to a file server or a service failure [17]; and malicious intrusions such as Denial of Service attacks [6]. These types of anomalies of interest can manifest a significant change on one or more traffic flows (edges).

With this in mind, we investigated cost functions endeavouring to improve the existing `ged` measure outlined in d_2. As described for Equation (2), the measures developed assume the graphs to be fully connected to simplify notation. The application of normalisation and non-linear techniques to edge costs in the `ged` cost function were investigated for this purpose.

A simple method to realise relative change in edge cost is to employ normalisation techniques [5, 14, 18, e.g.]. The edge cost incurred for any edge edit (deletion, insertion or substitution) were normalised by the maximum of the two edge weights being compared; the sum of the two edge weights; or the minimum value of the two edge weights being compared.

A simple method to accentuate relatively large changes in edge weight and hide small changes due to typical traffic variations is to employ non-linear mathematical functions. Several simple non-linear functions have been explored, including exponential and power functions. An example of an investigated `ged` cost function is described in Equation (3). The edge costs in this measure have a power function applied, with normalisation by the minimum of the two edge weights. In order to avoid the occurrence of a divide by zero for an edge deletion or insertion, each edge weight considered in the cost function has a value of $\epsilon = 1$ added.

$$\begin{aligned} d_3(G,H) &= c\left[|V_G| + |V_H| - 2|V_G \cap V_H|\right] \\ &+ \sum_{e \in E_G \cap E_H} \frac{|(\beta_G(e)+\epsilon) - (\beta_H(e)+\epsilon)|^2}{(\min(\beta_G(e),\beta_H(e))+\epsilon)^2} \\ &+ \sum_{e \in E_G \backslash (E_G \cap E_H)} (\beta_G(e)+\epsilon)^2 + \sum_{e \in E_H \backslash (E_G \cap E_H)} (\beta_H(e)+\epsilon)^2 \,. \end{aligned} \tag{3}$$

5 Experimental results

The `ged` cost functions developed in this study were applied to a computer network simulated in Matlab, to investigate their performance in detecting specific anomalous changes in network traffic. Only one network anomaly scenario is detailed in this article.

A time series of 100 directed and weighted graphs, each with a size of 10 nodes, were generated. The topology of the first graph in the time series was randomly generated with an edge density of 40%. A random topological structure was used for this purpose, as the `ged` measure is independent of the network topology [7]. An identical network topology was imposed on all graphs in the time series, in order to investigate the response of the `ged` cost functions to network traffic variations alone. The traffic values assigned to each edge was computed using either of two Poisson distributions functions, to simulate traffic flows that operate at different rates or frequencies.

The experiment detailed in this article was the introduction of two outliers in the time series of graphs, to simulate two relatively large, anomalous changes in traffic. The first outlier was a large relative (and large absolute) increase in edge weight on an edge in graph 25. Such an outlier could describe the beginning of a Denial of Service attack, for example. The second outlier

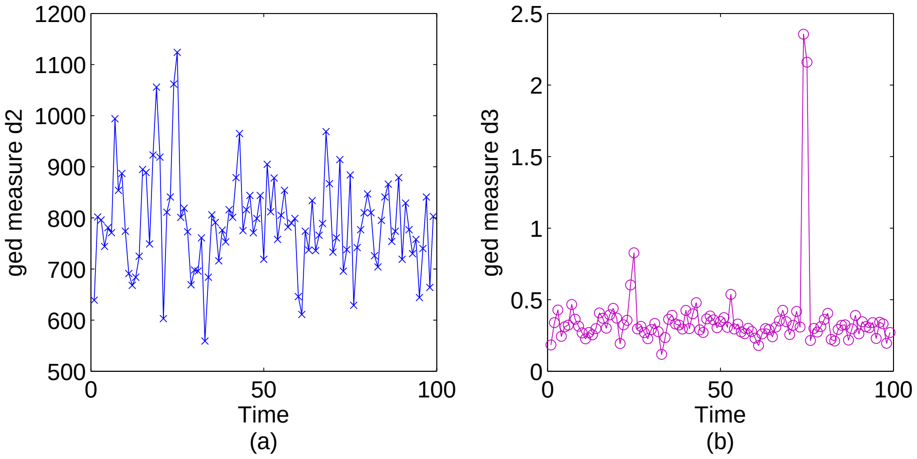

Figure 1: Response to the simulated outliers by (a) d_2, `ged` with simple cost functions, and (b) d_3, `ged` with non-linear cost functions.

was a large relative decrease in edge weight on an edge in graph 75. Compared to the first outlier, the second outlier had a smaller absolute change in edge weight. This might simulate the malfunction of a device or service.

The change detected by the original `ged` with simple cost functions, d_2, in response to the time series of graphs described in this experiment is demonstrated in Figure 1(a): the outlier anomalous change injected at graph 25 is detected, at time points 24 and 25. However, the outlier in graph 75 is not detected using this simple cost function. Since the absolute amount of change injected in graph 75 is low compared to that injected at graph 25, this change appears to be concealed by the typical variation of traffic occurring in the network.

Applying d_3 to the same simulated time series of graphs, results in the response also detailed in Figure 1(b): d_3 successfully detects the second outlier anomaly injected in graph 75. This is one example of a number of successful cost functions investigated in this study. By inspection, observe that employing ged cost functions with a combination of non-linear and normalisation techniques improves the detection of both outliers injected into the time series of graphs.

The responses of the improved ged cost functions have also proven to be scalable to anomalies characterised by change in network traffic flows on a number of links, either across the network or localised to one node.

6 Conclusions

Graph based matching techniques have been used to monitor and detect computer network anomalies, by treating periodically observed logical network communications as a time series of graphs. Specifically the graph edit distance (ged) is a graph matching distance measure to monitor and detect topology and traffic changes in a computer network. Past research has used only simple cost functions for the application of computer network change detection. This study shows that ged cost functions using normalisation and non-linear techniques can improve the detection of certain network anomalies deemed of high importance to network operators.

References

[1] R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–21, 1995. C438

[2] H. Bunke. Error correcting graph matching: on the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:917–922, 1999. C441

[3] H. Bunke. Recent developments in graph matching. In *Proceedings 15th International Conference on Pattern Recognition*, volume 2, pages 117–124, 2000. C438, C440, C441

[4] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, May 1983. C438, C440

[5] H. Bunke, M. Kraetzl, P. J. Shoubridge, and W. D. Wallis. Detection of abnormal change in time series of graphs. *Journal of Interconnection Networks*, 3(1&2):85 – 101, 2002. C438, C439, C440, C441, C443

[6] CERT Coordination Centre. Denial of service attacks. June 2001. Available at `http://www.cert.org/tech_tips/denial_of_service.html`. C443

[7] P. Dickinson. *Graph Based Techniques for Measurement of Intranet Dynamics*. PhD thesis, Institute for Telecommunications Research, University of South Australia, Adelaide, August 2005. C438, C444

[8] P. Dickinson and M. Kraetzl. Novel approaches in modelling dynamics of networked surveillance environment. In *Proceedings of the Sixth International Conference of Information Fusion*, volume 1, pages 302–309, 2003. C438, C439, C441, C442

[9] F. Feather and R. Maxion. Fault detection in an ethernet network using anomaly signature matching. In *Proceedings ACM SIGCOMM*, volume 23, pages 279–288, San Francisco, CA, September 1993. C438

[10] G. N. Higginbottom. *Performance Evaluation of Communication Networks*. Artech House, Massachusetts, 1998. C438

[11] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Network Journal*, 7(6):52–59, 1993. C438

[12] J. L. Jerkins and J. L. Wang. A close look at traffic measurements from packet networks. In *Proceedings of the IEEE GLOBECOM*, volume 4, pages 2405–2411, 1998. C438

[13] I. Katzela and M. Schwartz. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking*, 3(6):753–764, 1995. C438

[14] D. D. Parkes and W. D. Wallis. *Graph Theory and the Study of Activity Structure.* Timing Space and Spacing Time, vol. 2: Human Activity and Time Geography. Edward Arnold, London, 1978. C443

[15] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):353–362, 1983. C438, C440

[16] P. Shoubridge, M. Kraetzl, and D. Ray. Detection of abnormal change in dynamic networks. In *Proceedings of the International Conference on Information, Decision and Control*, pages 557–562, Adelaide, 1999. C438, C441

[17] M. Thottan and C. Ji. Anomaly detection in IP networks. *IEEE Transactions on Signal Processing*, 51(8):2191–2204, August 2003. C438, C443

[18] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988. C443

[19] D. B. West. *Introduction to Graph Theory.* Prentice Hall, New Jersey, 1996. C439

[20] C. C. White, E.A. Sykes, and J. A. Morrow. An analytical approach to the dynamic topology problem. *Telecommunication Systems*, 3:397–413, 1995. C438

Author addresses

1. **K. M. Kapsabelis**, Defence Science & Technology Organisation, Australia.
   mailto:kelly.kapsabelis@dsto.defence.gov.au
2. **P. J. Dickinson**, Defence Science & Technology Organisation, Australia.
3. **K. Dogancay**, University of South Australia, Australia