Under consideration for publication in Knowledge and Information Systems

Discovering Correlated Spatio-Temporal Changes in Evolving Graphs

Jeffrey Chan[1], James Bailey[1] and Christopher Leckie [1]

[1]NICTA Victoria Research Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne
Parkville, Australia 3010

Abstract.

Graphs provide powerful abstractions of relational data, and are widely used in fields such as network management, web page analysis and sociology. While many graph representations of data describe dynamic and time evolving relationships, most graph mining work treats graphs as static entities. Our focus in this paper is to discover regions of a graph that are evolving in a similar manner. To discover regions of correlated spatio-temporal change in graphs, we propose an algorithm called cSTAG. Whereas most clustering techniques are designed to find clusters that optimise a single distance measure, cSTAG addresses the problem of finding clusters that optimise both temporal and spatial distance measures simultaneously. We show the effectiveness of cSTAG using a quantitative analysis of accuracy on synthetic data sets, as well as demonstrating its utility on two large, real-life data sets, where one is the routing topology of the Internet, and the other is the dynamic graph of files accessed together on the 1998 World Cup official website.

Keywords: Data mining, evolving graphs, dynamic graph analysis, spatio-temporal analysis, correlated spatio-temporal changes, clustering, event discovery.

1. Introduction

Graphs are powerful abstractions of relational data, hence their popularity in many fields, including network management, sociology (Chen 2005), webpage linkage analysis (Kleinberg, Kumar, Raghavan, Rajagopalan & Tomkins 1999)

Received May 24, 2007
Revised Oct 21, 2007
Accepted Nov 15, 2007

and bioinformatics (Kawaji, Yamaguchi, Matsuda & Hashimoto 2001). Previous work has concentrated on discovering patterns within and among a set of static graphs - graphs that are not considered to change with time. Examples include finding frequent subgraphs among a set of static graphs (Cook & Holder 1994)(Washio & Motoda 2003), discovering communities in social and information networks (Girvan & Newman 2002)(Kleinberg 1998), and using degree distributions, diameter and connectivity measures to characterise a computer science literature citation graph (An, Janssen & Milios 2004).

In contrast to the focus of this prior work, many of the relationships modelled by graphs evolve with time. The additional temporal dimension introduced by these evolving, dynamic graphs permits many new types of analyses. These include analysing how global properties of graphs, like their diameter, change with time (Leskovec, Kleinberg & Faloutsos 2005), mining dynamic frequent subgraphs (Borgwardt, Kriegel & Wackersreuther 2006), dynamically analysing links using a form of incremental pagerank algorithm (Desikan, Pathak, Srivastava & Kumar 2005), and detecting anomalous changes in an evolving graph (Shoubridge, Kraetzl, Wallis & Bunke 2002).

A novel but unconsidered problem in this context is how to group changes in a dynamic graph that are topologically and temporally related. In particular, how to group similar sequences of changes that occur over the same period of time (temporally correlated), as well as the same region of the graph (spatially correlated). We refer to subgraphs that are temporally and spatially correlated over a period of time as *regions of correlated spatio-temporal change*.

Changes to graphs can be structural, e.g., the appearance or disappearance of an edge between two snapshots of an evolving graph, changes to the weights of edges and vertices, or even changes to designated subgraphs. Although we focus on structural changes in this paper, the techniques we present to find these spatially and temporally correlated subgraphs are general, and can easily be adapted to analyse changes in other graph properties.

The problem of finding regions of correlated spatio-temporal change arises in a variety of contexts, such as:

– Fault diagnosis - When a fault occurs in a communications network, it can induce changes in the structure of the routing topology of the network (Steinder & Sethi 2004)(Lee & Poole 2006). For example, consider a packet-switched network such as the Internet, where IP routers maintain routes between end points in the network. If a fault occurs in an IP router, all routes that traverse that router will be affected. Similar problems can occur in optical networks (Kandula, Katabi & Vasseur 2005), where lightpaths traverse optical switches. The problem of fault diagnosis in these contexts is to find the root cause(s) in the lower level or underlying layers of the network that explains the observed changes in the higher lever layers. In cases such as optical networks and large networks like the Internet routing network, active probing (e.g., pinging) and Bayesian fault diagnosis approaches are either unavailable or very expensive for finding the root causes of faults (Steinder & Sethi 2001)(Tang, Al-Shaer & Boutaba 2005). In these cases, changes in the end-to-end routes or lightpaths can be used to infer faults in their underlying routers or switches (Kandula et al. 2005). By partitioning the set of changes into spatially and temporally correlated groups, each group is likely to be caused by a common underlying fault. Hence, finding regions of correlated spatio-temporal change plays an important role in identifying the scope of a problem and its root cause.

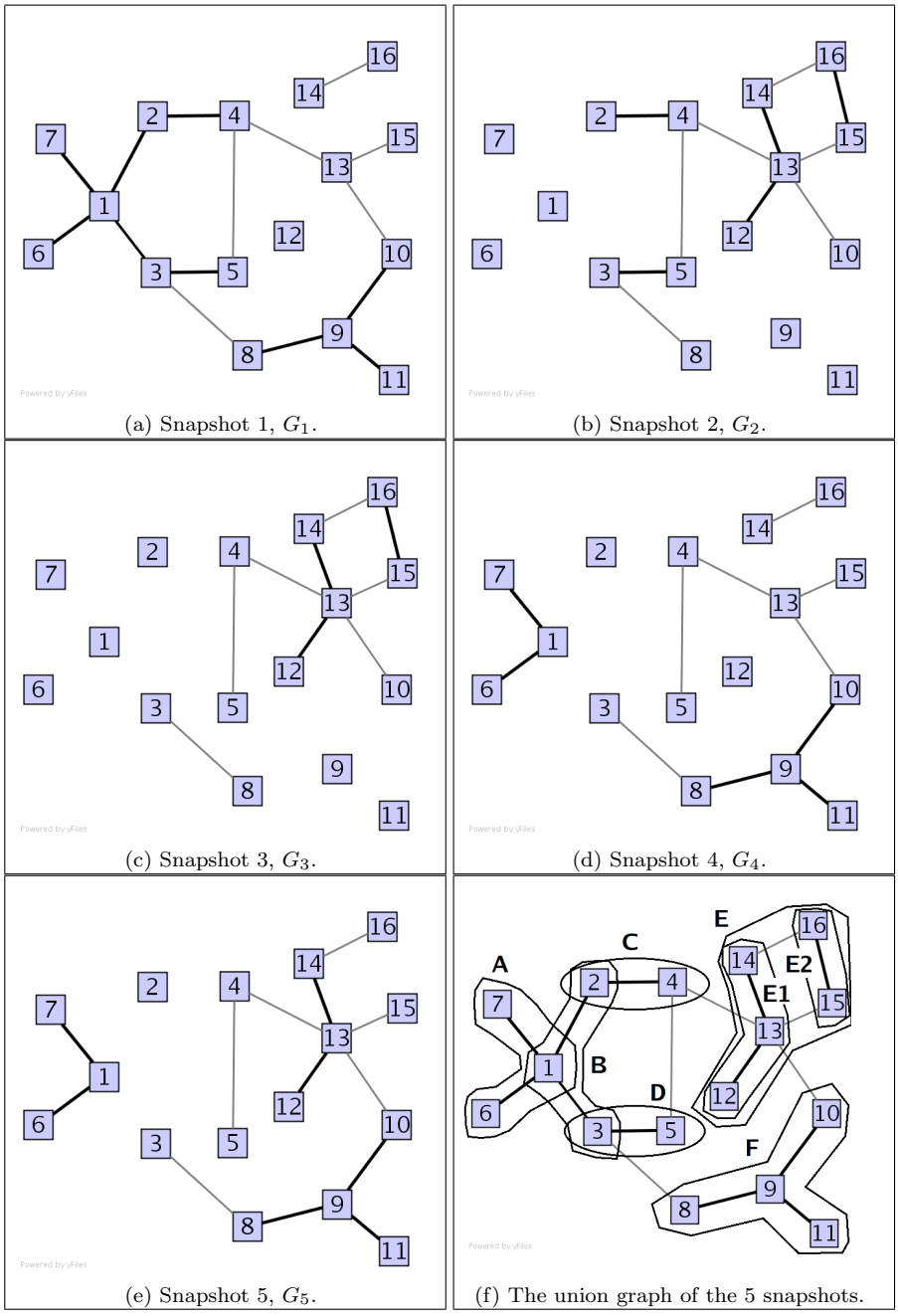
Region	Subgraph	Edge Set	Lifetime	Change Waveform
				G1 G2 G3 G4 G5
1	A	{1-6, 1-7}	1 – 5	
2	B	{1-2, 1-3}	1 – 5	
3	C	{2-4}	1 – 5	
4	D	{3-5}	1 – 5	
5	E	{12-13, 13-14, 15-16}	1 –4	
5a	E1	{12-13, 13-14}	1 – 5	
5b	E2	{15-16}	1 – 5	
6	F	{8-9, 9-10, 9-11}	1 – 5	

Table 1. Regions of correlated spatio-temporal change, their subgraph, lifetime, and associated change waveforms from Figure 1.

– Mining of spatio-temporal co-occurring items - In this context (Celik, Shekhar, Rogers, Shine & Yoo 2006), the aim is to find items that are near each other, or co-located, for a significant amount of time. Typically, spatial proximity is geographically based, and a distance metric can easily be defined. If the spatial distances are represented as edge weights, and the items as vertices, we can map the problem of mining of spatio-temporal co-occurrence to one of seeking subgraphs where all the vertices are topologically close (spatially correlated), and where the edge weights change in a similar manner (temporally correlated).
– Event detection - For example, discovering external events based on an analysis of the frequency of (query, page click) pairs in search engine logs (Zhao, Liu, Bhowmick & ng Ma 2006). Query-page pairs that are frequent and rare at similar times (temporally correlated), and semantically related in terms of sharing either a query or a page (spatially correlated) have been found to correspond to real events with reasonable accuracy. For example, over the period of the 2005 US Independence Day (June 21 - 27, 2005) there were correlated surges in firework related queries and clicked page pairs, which were most likely due to the Independence Day firework shows.

In this paper, we focus on the applications of fault diagnosis and event discovery in multi-layered computer networks. However, we stress that any application with data that has some relation defined over a set of entities, and exhibits correlated change behaviour, can make use of regions of correlated spatio-temporal change.

To illustrate what is a region of correlated spatio-temporal change, consider an example of five sequential snapshots of an evolving graph, shown in Figure 1. For example, this evolving graph could represent the routing topology of an IP network. Consider the set of edges in region A. Notice that for each of the five snapshots, the edges are either all present, or all absent. This is an example of correlated temporal behaviour. In addition, consider the shortest path distance between the edge pairs in region A - all the distances are very small. This is



(a) Snapshot 1, G_1. (b) Snapshot 2, G_2.

(c) Snapshot 3, G_3. (d) Snapshot 4, G_4.

(e) Snapshot 5, G_5. (f) The union graph of the 5 snapshots.

Fig. 1. An example of a dynamic graph with five snapshots. Bold edges highlight edges that have experienced change in the five snapshots. The changed edges belonging to each region are circled in Figure 1(f).

an example of spatial correlation. Hence, the edges in region A form a region of correlated spatio-temporal change. The same type of analysis can be performed for the other regions. Note that computing the shortest path distance and requiring all edges in a region to have small temporal and spatial distances is only one method to group the changes to edges. In Section 3, we shall highlight several other techniques. We use this example throughout the rest of the paper to illustrate our approach.

To discover the regions of correlated spatio-temporal change from evolving graphs, we have developed a framework named cSTAG (*C*lustering for Spatio-Temporal Analysis of Graphs). The framework takes as input a sequence of snapshots of a dynamic graph, and outputs the discovered regions of correlated spatio-temporal change. It represents (structural) changes to the graph as binary waveforms. Temporal correlation can then be measured using the distance between the associated binary waveforms. Similarly, topological distance measures are used to compute the spatial correlation, e.g., shortest path distance (Newman 2003). In addition, we introduce various clustering solutions that simultaneously incorporate temporal and spatial distance information to find regions of correlated spatio-temporal change. Finally, a region association method is presented to find regions with long term correlation.

To demonstate the utility of spatio-temporal correlation analysis on evolving graphs, we applied cSTAG to the discovery of events in two applications. One of these applications was the discovery of the root causes of routing changes in the Border Gateway Protocol (BGP) (Halabi & McPherson 2001) connectivity graph of the Internet during a known external event - the landfall of Hurricane Katrina (Cowie, Popescu & Underwood 2005). The BGP connectivity graph models the top-level routing topology of the Internet. In this application, cSTAG was able to discriminate regions that were spatially or temporally close, but corresponded to different events. The other application was detecting flash crowd events at the official 1998 World Cup website. Each flash crowd event corresponded to a particular match being played.

In addition, we have evaluated cSTAG on a number of synthetic datasets, in order to quantify the effectiveness of different clustering methods in our framework and the effect of the parameter settings on the accuracy and timing.

In summary, the main contributions of this work are as follows:

1. We propose a new pattern to mine from evolving graphs, namely regions of correlated spatio-temporal change, which groups changes based on their spatio-temporal correlation;
2. We provide a comprehensive evaluation of the accuracy and efficiency of cSTAG using a synthetic dataset, as well as demonstrating the utility of discovering regions of correlated spatio-temporal change by discovering reported events on the Internet routing topology during the 2005 Hurricane Katrina disaster, and identifying flash crowd events at the 1998 World Cup website during matches. The real dataset analyses demonstrate that cSTAG is able to accurately identify multiple simultaneous events occurring on a dynamic network.

The rest of the paper is organised as follows. In Section 2, the problem of discovering regions of correlated spatio-temporal change is formally presented. Section 3 describes the cSTAG algorithm in more detail. Accuracy and timing evaluation using synthetic datasets, and an analysis of the patterns mined from two real application domains, are presented in Section 4. In Section 5, related

Symbol	Description
$G(V_G, E_G)$	A graph, with vertex and edge set V_G and E_G.
$< G_1, G_2, \ldots, G_S >$	A sequence of S snapshots of a dynamic graph G.
$W^{ts,te}$	A subsequence of snapshots $< G_{ts}, G_{ts+1}, \ldots, G_{te} >$.
$E^{ts,te}_{union}$	Union of edge sets of snapshots in subsequence $W^{ts,te}$.
$E^{ts,te}_{C}$	Set of edges in $E^{ts,te}_{union}$ that have experienced at least one change.
$q^{ts,te}(e_i)$	The change waveform of edge e_i of subsequence $W^{ts,te}$.
$d^{ts,te}_{tem}(e_i, e_j)$	Temporal distance measure between edge e_i and e_j, over subsequence $W^{ts,te}$.
$d^{ts,te}_{spa}(e_i, e_j)$	Spatial distance measure between edge e_i and e_j, over subsequence $W^{ts,te}$.
$R^{ts,te}_r$	r^{th} region of correlated spatio-temporal change, defined over subsequence $W^{ts,te}$.
$R^{ts,te}_r.Q$	The set of (change waveform, frequency) pairs of region $R^{ts,te}_r$.
$\mathbf{R}$	Set of regions of correlated spatio-temporal change, of varying subsequences.
ω	Length of the sliding window used.
inc	Number of snapshots to slide for each iteration of the sliding window process.
ω_{spd}	Length of the union window used for computing shortest path distances.
d_{limit}	Maximum number of hops that a shortest path search can explore.

Table 2. Description of symbols used.

work is surveyed. Finally, Section 6 presents future work and concludes the paper.

2. Problem Statement

In this section, we formally define the problem of discovering regions of correlated spatio-temporal change. First, we introduce the notation we use. For ease of reference, Table 2 provides a description of the main symbols used in the paper.

A graph $G(V_G, E_G)$ consists of a set of vertices V_G, and a set of (unweighted) edges E_G, $E_G : V_G \times V_G$, representing the pairwise relationships over V_G. Where there is no ambiguity, we shall use G to denote $G(V_G, E_G)$. A **dynamic graph** is represented as a sequence of consecutive snapshots $< G_1, G_2, \ldots, G_S >$ of the graph, $1 \leq t \leq S$. We denote the subsequence of snapshots (or window) $< G_{ts}, \ldots, G_{te} >$ by $W^{ts,te}$, $1 \leq ts \leq te \leq S$. The **union of the edge sets**, $E^{ts,te}_{union}$, of the subsequence $W^{ts,te}$, $1 \leq ts \leq te \leq S$, is defined as $E^{ts,te}_{union} = \bigcup_{t=ts}^{te} E_{G_t}$. For example, Figure 1(f) represents the union graph of the subsequence $W^{1,5}$ from Figure 1.

A **structural change** of an edge e, $e \in E^{1,S}_{union}$, is defined as the appearance ($e \notin E_{G_t}, e \in E_{G_{t+1}}$) or disappearance ($e \in E_{G_t}, e \notin E_{G_{t+1}}$) of e between any two consecutive snapshots G_t, G_{t+1}, $1 \leq t \leq S-1$. Then the set of all edges that have experienced at least one structural change over the S time intervals, or the **set of changed edges** $E^{1,S}_C \subseteq E^{1,S}_{union}$, can be defined as $E^{1,S}_C = \{e | e \notin E_{G_t}, e \in E_{G_{t+1}}\} \cup \{e | e \in E_{G_t}, e \notin E_{G_{t+1}}\}$, $1 \leq t \leq S-1$.

Next, we define our notation for the **change waveforms**, which represent the temporal change behaviour of edges over a particular subsequence of snapshots.

Definition 1. For structural changes to edge e_i, over the subsequence $W^{ts,te}$, we can represent these changes by a binary valued **change waveform** $q^{ts,te}(e_i) =$

$q(e_i)[1]q(e_i)[2]\dots q(e_i)[te-ts+1]$, where

$$q^{ts,te}(e_i)[k] = \begin{cases} 0 & e_i \notin G_{ts+k-1} \\ 1 & e_i \in G_{ts+k-1} \end{cases}, 1 \leq k \leq te-ts+1$$

As an example, the change waveforms for each changed edge in Figure 1 are shown in Table 1. This representation of temporal behaviour can easily be extended to continuous entities, like real-valued edge weights. Where there is no ambiguity, we will use the notion $q(e_i)$ for $q^{ts,te}(e_i)$.

Finally, we present the notation used to describe the temporal and spatial distance measures between changed edges. We then give the formal definition of a region of correlated spatio-temporal change, and summarise the problem of discovering such regions of correlated spatio-temporal change.

The pairwise **temporal distance** between a pair of edges is defined as $d_{tem}(e_i, e_j, W^{ts,te}) = d(q^{ts,te}(e_i), q^{ts,te}(e_j))$, where $d(q^{ts,te}(e_i), q^{ts,te}(e_j))$ represents the distance between the two waveform representations. The temporal distance measures the difference in temporal change behaviour of the edges e_i and e_j over the subsequence $W^{ts,te}$. We shall elaborate on the measures we used to compute the distance in Section 3. We similarly define a **spatial distance** $d_{spa}(e_i, e_j, W^{ts,te})$. As mentioned in the introduction, the spatial distance is based on the topological properties of the dynamic graph. As the topological properties and hence the spatial distance may change with time, spatial distance is also defined over a subsequence.

Definition 2. A **region of correlated spatio-temporal change** $R_r^{ts,te}$ is defined as a subset of $E_C^{ts,te}$, for which the spatial and temporal correlation of its edges is valid over the subsequence $W^{ts,te}$. The set of changed waveforms associated with the changed edges in $R_r^{ts,te}$ is denoted by $R_r^{ts,te}.Q$, the set of (change waveform, frequency) pairs.

$R_r^{ts,te}.Q$ summarises the temporal behaviour of the edges in $R_r^{ts,te}$. We define the set of change waveforms for a region as the collection of the change waveforms of its member edges. Then we define the (relative) frequency of each unique change waveform in the collection of waveforms as the (relative) number of times it occurs. The frequency ranges from 0 (exclusive) to 1 (inclusive).

The **lifetime** of a region of correlated spatio-temporal change $R_r^{ts,te}$ is defined as the period of time the edges in the region are correlated. The correlation span is required to be continuous, i.e., there are no gaps in it. The lifetime is defined by the **start time** ts and **end time** te.

For example, Table 1 shows the regions of correlated spatio-temporal change from the dynamic graph in Figure 1. Of particular interest are regions 5, and 5a and 5b. Region 5 has a lifetime of 1 to 4, because its three edges $12-13$, $13-14$ and $15-16$ are only temporally correlated over the subsequence $< G_1 - G_4 >$. However, if correlation is sought over the whole sequence, then edges $12-13$ and $13-14$ form a region (region 5a), and edge $15-16$ forms another (region 5b), due to the difference in their change waveforms (i.e., their temporal behaviour).

Where there is no ambiguity, we shall use $Q_r.q_a$ and $Q_r.freq_a$ to denote a waveform-frequency pair $(q_a, freq_a)$ in $R_r^{ts,te}.Q$. Recall q_a denotes a change waveform in the set of change waveforms of a region, and $freq_a$ denotes the relative frequency of waveform q_a in the region. In addition, if it is clear from the context, we use R_r instead of $R_r^{ts,te}$.

With all the definitions in place, we can now define the set of regions of correlated spatio-temporal change and the problem of discovering these regions.

Definition 3. The **set of regions of correlated spatio-temporal change R** is a hard partition of $E_C^{1,S}$ into $\{R_1^{ts1,te1}, \ldots, R_L^{tsL,teL}\}$, where $R_r^{tsr,ter} \subseteq E_C$, $R_g^{tsr,ter} \cap R_h^{tss,tes} = \emptyset, g \neq h, 1 \leq g, h \leq L$, and L is the number of regions.

Definition 4. The **problem of discovering regions of correlated spatio-temporal change** involves finding a set of regions of correlated spatio-temporal change that maximise the following intra-region criteria simultaneously and across all regions:

1. $\max_{\mathbf{R}} \sum_{R_r \in \mathbf{R}} criteria_{tem}(R_r, d_{tem})$;
2. $\max_{\mathbf{R}} \sum_{R_r \in \mathbf{R}} criteria_{spa}(R_r, d_{spa})$,

where $criteria_{tem}$ and $criteria_{spa}$ are the criteria used to measure the spatial and temporal compactness of the edges in R_r, based on the distance measures d_{tem} and d_{spa} respectively. Given that there are two criteria to be optimised simultaneously, the exact interpretation of maximisation in this context can be approached in several different ways. We propose several specific approaches to this problem in Section 3.5.1.

To summarise, the **problem of discovering regions of correlated spatio-temporal change** can be summarised as follows:

Input:
The sequence of graph snapshots, $< \mathbf{G_1}, \mathbf{G_2}, \ldots, \mathbf{G_S} >$
The temporal and spatial distance measures, $\mathbf{d_{tem}}$, $\mathbf{d_{spa}}$
The temporal and spatial cohesion criteria, $criteria_{tem}$, $criteria_{spa}$
Process:
Find a set of regions, $\mathbf{R}$*, that maximises* $\sum_{R_r \in \mathbf{R}} criteria_{tem}(R_r, d_{tem})$ *and* $\sum_{R_r \in \mathbf{R}} criteria_{spa}(R_r, d_{spa})$ *simultaneously*
Output:
The set of regions of correlated spatio-temporal change, $\mathbf{R}$

3. cSTAG

In this section, we first outline the challenges in solving the problem of discovering regions of correlated spatio-temporal change. We then provide an overview of cSTAG, our solution to this problem, which discovers regions of correlated spatio-temporal change. It consists of a number of components, which we will describe in the following subsections.

3.1. Challenges and Overview

There are a number of challenges that need to be addressed by any solution to this problem:

- The number of regions of correlated spatio-temporal change is unknown a-priori.
- The length of correlation, or lifetime, of each region is unknown a-priori.
- The exact start time of each region is unknown a-priori.

Region	Subgraph	Edge Set	Lifetime	Change Waveform
				G1 G2 G3 G4 G5
6	F	{8-9, 9-10, 9-11}	1 – 5	
6a	F	{8-9, 9-10, 9-11}	1 – 4	
6b	F	{8-9, 9-10, 9-11}	2 – 5	
1	A	{1-6,1-7},	1 – 5	
1a	A	{1-6,1-7},	1 – 4	
1b	A	{1-6,1-7},	2 – 5	

Table 3. Example where region 6 from 1 has broken up into regions 6a and 6b, and region 1 into regions 1a and 1b.

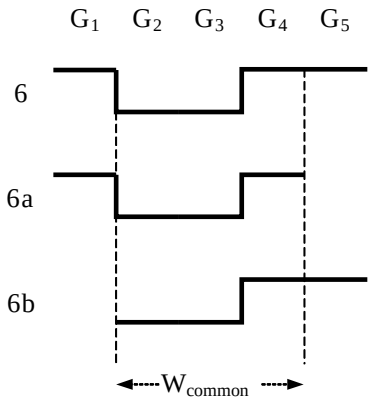


Fig. 2. An example of W_{common}, taken from region 6 and its sub-regions 6a and 6b.

A quick sketch of a naive method to find the set of regions **R** demonstrates the difficulty of the problem. A naive method would partition the sequence into a number of overlapping windows, of lengths 2 to S. Then for each window, it would enumerate all regions of correlated spatio-temporal change that exhibit temporal correlation over the window. From the set of all such regions, we find a subset, across all window lengths, that satisfies the constraints outlined in the problem statement and which maximises the temporal and spatial criteria. This step may require further splitting and merging of regions. The naive method would be complete, but intractable - this naive approach is similar to finding an optimal job/timetable schedule from a set of known jobs of different lengths and costs, which is known to be NP-Hard.

Hence, we have developed an approximate solution to the problem of discovering regions of correlated spatio-temporal change. As an illustration of the approach, consider an example from Figure 1. Consider the case where region 6 has been split into two regions, 6a and 6b, where region 6a is defined over the snapshots G_1 to G_4, and region 6b over snapshots G_2 and G_5. Refer to Table 3 for details about regions 6, 6a and 6b, and Figure 2 for the alignment of the change waveforms of region 6.

There are several important points to note:

1. Both regions 6a and 6b have the same set of edges as region 6, since if the edges 8-9, 9-10, 9-11 are spatially and temporally correlated over the subsequence $W^{1,5}$, then they are also correlated over $W^{1,4}$ and $W^{2,5}$.
2. For the snapshots common to both $W^{1,4}$ and $W^{2,5}$, which we denote W_{common} ($W_{common} = W^{2,4}$, and illustrated in Figure 2), the change waveforms for 6a and 6b are the same.
3. Even though regions 1a (1b) have the same change waveforms over W_{common} to regions 6b (6a) respectively, the two regions (1a and 6b, 1b and 6a) have different sets of edges.

This means that we can first find regions of correlated change over shorter correlation periods, then merge those that have high overlap in their set of edges and high similarity in their change waveforms (over the common subsequence of snapshots) to form regions with longer correlation periods. This can be repeated, until the region with the maximal correlation period is found.

From this illustrative example, it can be seen that we can find the regions of correlated change by first dividing the sequence of snapshots into a set of overlapping, subsequences of snapshots. Then for each subsequence, we find the set of regions of correlated change for each subsequence, using a clustering approach. This is analogous to finding R_{1a} for subsequence 1, then R_{1b} for subsequence 2. Up to this point, we can find regions with correlation up to the length of each subsequence. To find regions with correlation greater than the length of a subsequence, we merge regions discovered in adjacent windows, using the criterion that two regions should be merged across subsequences if both their sets of edges and their change waveform, over the common subsequence of snapshots, are highly overlapping and similar.

The general process is illustrated in Figure 3, and in more detail in Algorithm 1. Note that there are some terms and ideas in Algorithm 1 that are not introduced until later in this section. Nevertheless, the general direction of the algorithm can still be appreciated without understanding all the terms in detail.

The overall process of cSTAG can be broken down to the following steps. First, in the *windowing* step, the input sequence is split into a number of subsequences using a sliding window process. The set of changed edges $E_C^{ts,te}$, for subsequence $W^{ts,te}$, is then extracted by the function *extract()* (line 15). This can be easily done by examining if an edge appears or disappears between two snapshots in the subsequence. Then the temporal and spatial distances are computed in lines 19 and 21 respectively. Next, we discover the set of regions whose correlation length is the same as the length of the sliding window. We call this step the discovery of *regions of correlated spatio-temporal change*, represented by the *regDiscovery()* function (line 23). The final step, *region association*, merges regions discovered in the individual windows to find regions of correlated spatio-temporal change, whose lifetime is greater than the sliding window length (line 32).

In comparison with the naive method, cSTAG has the advantage that it enumerates only the regions of correlated spatio-temporal change for one given subsequence/window length. Although it is an approximate solution, we show in the evaluation section that it can achieve up to 100% accuracy in detecting a variety of regions of correlated spatio-temporal change. In the coming sections,

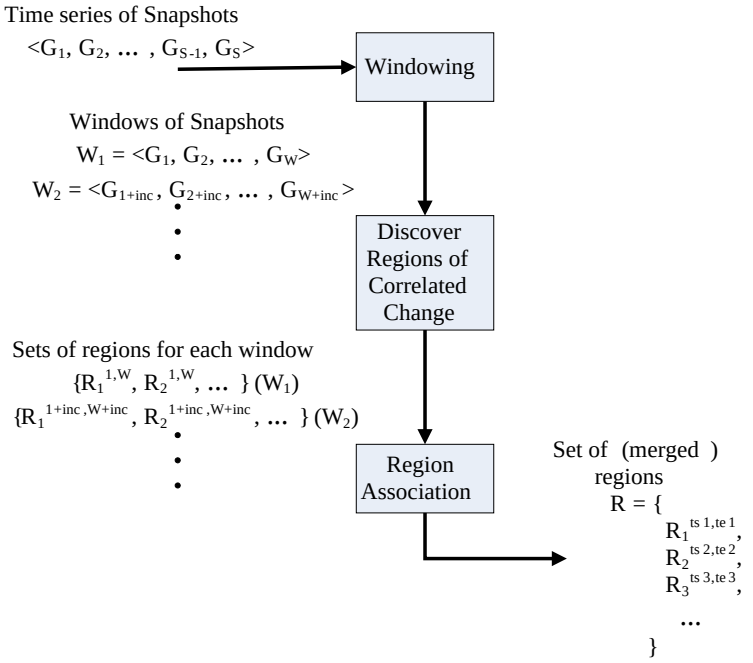


Fig. 3. Summary of the cSTAG Algorithm.

we describe each component in detail. We also discuss the temporal (d_{tem}) and spatial (d_{spa}) distance measures that we have applied to this problem.

3.2. Windowing

Sliding windows have been extensively studied and applied in the literature. Consequently, we provide only a brief description of the sliding window procedure. The sliding window procedure involves moving a window across a sequence, generating a number of overlapping windows.

More formally, let the length of the sliding window be denoted by ω, and the increment by which the sliding window is moved each time be denoted by $inc, 1 \leq inc \leq \omega - 1$. Then for sequence $< G_1, \ldots, G_S >$, a sliding window process of window length ω and increment inc will generate a set of $\lceil \frac{S-\omega+1}{inc} \rceil$ subsequences $\{W^{1,\omega}, W^{1+inc,\omega+inc}, \ldots, W^{ts,\omega+ts-1}, \ldots, W^{S-\omega+1,S}\}$.

The window length ω represents the minimum length of temporal correlation that a region of correlated spatio-temporal change must possess to be considered a region. A short window length tends to produce few regions of many edges, but most of the discovered regions are uninteresting and the changes are more likely to be grouped together by coincidence. A longer window length will avoid these problems, but tends to produce many regions of small size, which again may be of limited use. We shall empirically investigate the effect of window length in the evaluation section (Section 4.1.3).

Algorithm 1 Overview of the cSTAG Algorithm.

Input: $GS = < G_1, G_2, \ldots, G_S >$ - sequence of snapshots, ω - length of sliding window
Output: $\mathbf{R}$ - the regions of correlated spatio-temporal change

// Compute the set of union graphs, which is used to compute the spatial distance
$UG = computeUnion(GS)$

// Initialise the index of the initial snapshot, ts, and last snapshot, te, for the first window $W^{1,1+\omega}$ in the sliding window process
$ts = 1$
$te = ts + \omega$
// While there are more windows to extract in the sequence, find the regions $\mathbf{R^{ts,te}}$ for each window $W^{ts,te}$
while $ts \leq S - \omega$ **do**
  // Construct current window of snapshots
  $W^{ts,te} = < G_{ts}, G_{ts+1}, \ldots, G_{te} >$
  // Extract the set of changed edges $E_C^{ts,te}$
  $E_C^{ts,te} = extract(W^{ts,te})$
  // Compute the set of change waveforms Q for the current window
  $Q = constructWaveform(E_C^{ts,te})$
  // Compute the temporal distances $D_{tem}^{ts,te}$ for current window
  $D_{tem}^{ts,te} = computeTem(E_C^{ts,te}, Q)$
  // Compute the spatial distances $D_{spa}^{ts,te}$ for current window
  $D_{spa}^{ts,te} = computeSpa(E_C^{ts,te}, UG)$
  // Compute the set of regions for current window
  $\{R_1^{ts,te}, R_2^{ts,te}, \ldots\} = regDiscovery(E_C^{ts,te}, D_{tem}^{ts,te}, D_{spa}^{ts,te})$
  // Add discovered regions to $\mathbf{R^{ts,te}}$, which holds regions discovered for current window
  $\mathbf{R^{ts,te}} = \{R_1^{ts,te}, R_2^{ts,te}, \ldots\}$
  // Slide window by inc number of snapshots
  ts += inc
  te += inc
end while

// Merge the regions in each set $R^{x,y}$ (where x and y are valid window indices) across adjacent windows
$\mathbf{R} = regAssoc(\mathbf{R^{1,1+\omega}}, \mathbf{R^{1+inc,1+\omega+inc}}, \ldots)$

3.3. Temporal Distance Measures *(computeTem)*

The choice of what is the "best" temporal distance measure is dependent on the application context. For example, in the case of applications such as network fault diagnosis, where we wish to group routing changes induced by the same root cause, changes effected by the same root cause might exhibit delays as changes cascade through the network. In this context, two waveforms should only be

considered similar if they have the same shape, and the timing of the changes roughly coincides.

In this section, we shall outline several temporal distance measures that are potentially relevant in this context and discuss their advantages and disadvantages. We then describe the actual distance measure we have developed for use in network fault diagnosis. Note that in general, cSTAG can accept any temporal distance measure, as long as it is symmetric and produces a distance relation on the set of changed edges.

Two candidate distance measures that take waveform shape similarity into consideration are Dynamic Time Warping (DTW) (Cormen, Leiserson, Rivest & Stein 2001) and Longest Common SubSequence (LCSS) (Vlachos, Kollios & Gunopulos 2002). They can handle translations between the compared waveforms, and in the case of LCSS, are robust to noise. However, the main drawback of these measures is that they are computationally expensive for comparing simple binary waveforms. In particular, if $q(1)$ and $q(2)$ are the compared waveforms, where $l_1 = |q(1)|$ and $l_2 = |q(2)|$, and δ is a threshold specifying the maximum waveform stretching allowed, then an implementation of DTW that computes exact distances has $O(max(l_1, l_2)^2)$ (Salvador & Chan 2004) complexity, and it is $O(\delta^3(l_1 + l_2))$ for LCSS (Vlachos et al. 2002).

An alternative approach that addresses these complexity costs is the popular Euclidean distance measure (d_{euc}), which is simple and efficient to calculate ($O(W)$ where $W = |q|$).

Definition 5. For two binary sequences $q(e_i)$ and $q(e_j)$ of length ω, the (normalised) Euclidean distance can be defined as

$$d_{euc}(q(e_i), q(e_j)) = \frac{1}{\omega} \sum_{k=1}^{\omega} q(i)[k] \oplus q(j)[k] \tag{1}$$

where $\oplus$ represents exclusive OR ($0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$).

However, two of the known weaknesses of the Euclidean distance measure are that it is not robust to any delays or difference in scale between two compared waveforms, even if the two waveforms have the same shape. Since we are using binary valued waveforms, there are no scaling problems. To overcome the translation weakness, we additionally consider the shapes when comparing the (binary) waveforms. The approach we take is similar to DTW, particularly the Derivative DTW (DDTW) (Keogh & Pazzani 2001), but we do not need to use the relatively expensive dynamic programming technique to stretch or compress the time axis of the waveforms.

Before we present our modified Euclidean distance measure, we first introduce the concept of a **transition sequence**. In a transition sequence, we represent the changes in a binary waveform as a sequence of transitions. Let a $1 \rightarrow 0$ transition in a change waveform be denoted as $-$, and a $0 \rightarrow 1$ transition as $+$. We then define a sequence of transitions $trans(e_i)$ for waveform $q(e_i)$ as a sequence of alternating $-/+$ transitions. Using this definition, two waveforms with the same shape will have transition sequences of the same length and each of the $-/+$ transitions in the sequences will match. For example, consider Table 4, which shows the change waveforms and transition sequences for edges e_{1-6}, e_{1-3}, and e_{2-4} from Figure 1. The transition sequences of e_{1-6}, e_{1-3} are different, corresponding to the difference in the shape of their change waveform. However,

Edge	Change Waveform					Transition Sequence
	G1	**G2**	**G3**	**G4**	**G5**	
e_{1-6}						$< -+ >$
e_{1-3}						$< - >$
e_{2-4}						$< - >$

Table 4. Changed edges and their corresponding change waveforms and transition sequences

the transition sequences of e_{1-3} and e_{2-4} are the same, despite $q(e_{2-4})$ being a delayed version of $q(e_{1-3})$.

So, our modified distance measure is then defined as:

Definition 6. The modified Euclidean distance between waveforms $q(e_i)$ and $q(e_j)$ of lengths ω can be defined as

$$d_{m\_euc}(q(e_i), q(e_j)) = \begin{cases} 1, & trans(e_i) \neq trans(e_j) \\ d_{euc}(q(e_i), q(e_j)), & \text{otherwise} \end{cases}$$

where $0 \leq d_{m\_euc}(q(e_i), q(e_j)) \leq 1$.

Intuitively, this version of the Euclidean distance measure determines the number of snapshots in which two edges are either both present or both absent, while taking into consideration the shape of the waveforms. Two waveforms are similar if they have the same general shape (in terms of their transition sequences) and their unmodified edit distance is low (i.e., $d_{m\_euc}()$ is close to 0). For example, reconsider the waveforms in Table 4. The modified distance between $q(e_{1-3})$ and $q(e_{2-4})$ is 0.2, as they have the same transition sequence and $d_{euc}(q(e_{1-3}), q(e_{2-4})) = 0.2$, but $d_{m\_euc}(q(e_{1-6}), q(e_{1-3})) = 1$ since the transition sequences are different.

The complexity of computing the modified Euclidean distance is $O(|q|)$, which is the same or lower than the other alternatives. We further reduce the complexity of computing the modified Euclidean distance by implementing an incremental version of d_{euc} and $trans$.

We can incrementally compute the Euclidean distance $d_{euc}(e_i, e_j, W^{ts+\omega,te+\omega})$ for window $W^{ts+\omega,te+\omega}$ from the computed Euclidean distance for the previous window $W^{ts,te}$ by the following relation:

$$\begin{aligned} & d_{euc}(e_i,e_j, W^{ts,te}) \\ & \quad = d_{euc}(e_i, e_j, W^{ts-inc,te-inc}) - d_{euc}(e_i, e_j, W^{ts-inc,ts-1}) \\ & \qquad + d_{euc}(e_i, e_j, W^{te-inc+1,te}) \\ & \quad = d_{euc}(e_i, e_j, W^{ts,te-inc}) + d_{euc}(e_i, e_j, W^{te-inc+1,te}) \end{aligned} \tag{2}$$

The correctness of Equation (2) can be shown by writing both the left and right hand sides of the equality in their summation form.

Similarly, $trans(e_i, W^{ts,te})$ can be computed from $trans(e_j, W^{ts-inc,te-inc})$

by:

$$\begin{aligned} trans(e_i,&W^{ts,te}) \\ &= concat(truncate(trans(e_i, W^{ts-inc,te-inc}), ts-inc, ts-1), \\ &\qquad trans(e_i, W^{te-inc+1,te})) \\ &= concat(trans(e_i, W^{ts,te-inc}), trans(e_i, W^{te-inc+1,te})) \end{aligned} \tag{3}$$

where the function $truncate(trans(e_i, W^{ts-inc,te-inc}), ts-inc, ts-1)$ removes the changes or transitions associated with the snapshots $ts-inc$ to $ts-1$, inclusive, from the transition sequence $trans(e_i, W^{ts-inc,te-inc})$. The function $concat(trans(e_i, W^{ts,te-inc}), trans(e_i, W^{te-inc+1,te}))$ concatenates the transition sub-sequence $trans(e_i, W^{te-inc+1,te})$ of changes from snapshot $te-inc+1$ to te, to the transition sub-sequence $trans(e_i, W^{ts,te-inc})$, to form the the transition sequence for snapshots ts to te, $trans(e_i, W^{ts,te})$.

Using these Equations (2) and (3), d_{euc} can be incrementally computed by keeping the subsequence distances calculated in the previous window, $D_{euc}^{ts,te-inc}$, and just computing the distances of the snapshots that are newly arrived, $D_{euc}^{te-inc+1,te}$. For each change in the transition sequences, we introduce time markers to record the time the change occurred. This enables the truncation to be computed in constant time. Hence for transition sequences, we only need to maintain the set of transition sequences for the last window, $Trans^{ts-inc,te-inc}$.

In the remainder of the paper, in order to simplify the description of our algorithms, we describe cSTAG in terms of using a non-incremental temporal distance implementation. However, it is a straightforward matter to use the described incremental distance implementation.

3.4. Spatial Distance Measures (*computeSpa*)

Given that we are mining patterns in graphs, the spatial distance measures that we use are topology based. There are many topology based measures, but the most intuitive one for fault diagnosis is the shortest path distance. Two changing vertices that have low shortest path distance, and hence are spatially similar, are more likely influenced by the same change event, e.g., consider the router example from the introduction. In general, changes (to edges) caused by the same change event will have edges that possess small shortest path distances between them. Again, the exact definition of closeness depends on the specific application.

For example, in prior work (Chan, Bailey & Leckie 2006), two changed edges were considered to be close if there exists a path between the two edges that only consists of edges from the set of changed edges, i.e., among the changed edges, the sets of changed edges that form connected components are considered close to each other. Although this enabled very fast computation of spatial proximity and is generally well suited for network fault detection, it can be inaccurate in certain cases. For example, there can be two different faults occurring at the same time and inducing the same routing changes. The two affected sets of edges are also topologically connected by a single edge (i.e., a bridge). Then using the previous connected component method, the two sets of edges will be incorrectly identified as one spatially close region. However, if we use the shortest path distance and use a clique definition (i.e., a set of edges that are considered spatially close should all have low shortest path distances between them), then we can correctly

separate the edges into two separate regions, based on their spatial differences. In general, using shortest path distance can allow many definitions of spatial proximity, hence can increase the range of applications for which finding regions of correlated change can be useful.

Before we introduce how we compute the shortest path distance between two edges in a snapshot graph, we shall consider how to compute shortest path distances on a dynamic graph. As the underlying graph is dynamic, the shortest path distances themselves are dynamic too. A naive method would be to compute the shortest path distances for each snapshot, and then take the average across all the snapshots. However, this approach is expensive. Hence, a popular, alternative approach is to maintain the shortest path trees between vertices (or edges in our case) (Demetrescu & Italiano 2004)(King 1999)(Frigioni, Marchetti-Spaccamela & Nanni 1996) (Ramalingam & Reps 1996). By maintaining the shortest paths, only paths which are affected by changes need to be updated, and queries can be answered in constant time. However, due to the storage and maintenance of the shortest paths, this causes a substantial increase in memory usage. For example, an implementation of several well-known dynamic shortest path algorithms (Demetrescu & Italiano 2006) required over 1.5 GB of memory when tested on scale-free graphs with a few thousand vertices and edges. Hence, we do not consider dynamic shortest path algorithms to be a viable option for use in cSTAG.

The alternative method we used was to compute the shortest path distances over a subsequence of snapshots. Basically, we compute the union graph over the subsequence and compute the shortest path distances on these. This reduces the time cost of computing the distances for each snapshot, while avoiding the high memory cost of maintaining shortest paths. Computing the distances over a union graph is accurate if the shortest path distances themselves do not undergo significant change. This will occur when there are few changes. Even when there are many changes, if there exist many alternative shortest paths between any pair of edges (e.g., in a dense graph), most of the shortest path distances themselves do not vary much. We will observe this phenomenon in the evaluation of the synthetic datasets in Section 4.1.

We compute the shortest path distances for a subsequence of consecutive snapshots of length ω_{spd}, $1 \leq \omega_{spd} \leq S$. The sequence of snapshots is segmented into a number of disjoint subsequences, all of length ω_{spd}. The last window can be less than ω_{spd}. For each subsequence, we compute the union graph of the snapshots (performed by the function *computeUnion*) in Algorithm 1, then compute the shortest path distances on this union graph. For example, if the subsequence of snapshots is $<G_1, G_2, \ldots, G_5>$ and $\omega_{spd} = 2$, then there are three union graphs for these sequence of snapshots: $U_1^{1,2} = G_1 \cup G_2$, $U_2^{3,4} = G_3 \cup G_4$, and $U_3^{5,5} = G_5$. If we want to compute the shortest path distance for snapshot G_1, then we use the union graph $U_1^{1,2}$. But if we want to compute the distance over the subsequence $W^{1,3}$, then we compute the distance over $U_1^{1,2}$ and $U_2^{3,4}$, then take an average. In the evaluation, we have evaluated the effect of varying the union window size ω_{spd} on the running time and accuracy of cSTAG.

Algorithm 2 provides an outline of our approach to estimating the shortest path distances using a sequence of union graphs. Next, we describe the techniques we have used to speed up the shortest path distance calculations for a single graph snapshot.

Algorithm 2 Spatial distance computation - function *computeSpa()*.

Input: $E_C^{ts,te}$ - set of changed edges, UG - temporally ordered set of union graphs
Output: $D_{spa}^{ts,te}$ - set of spatial distances

// Find the earliest union graph with starting snapshot index $s \leq ts$ and ending snapshot index $t \leq te$
$U_i^{s,t} = find(UG, ts, te)$
// Compute the shortest path distance for each pair of changed edges
for each $e_i \in E_C^{ts,te}$ **do**
  for each $e_j \in E_C^{ts,te}, e_j \neq e_i$ **do**
    $d(e_i, e_j) = 0$
    $num = 1$
    // Compute the shortest path distance on all union graphs whose constituent snapshots have some overlap with window $W^{ts,te}$.
    while $s \leq te \leq t$ **do**
      $d(e_i, e_j)$ += $spd(e_i, e_j, U_i^{s,t})$
      i++
      num++
      // Get union graph with index i
      $U_i^{s,t} = UG[i]$
    end while
    // Insert the average distance into $D_{spa}^{ts,te}$
    $D_{spa}^{ts,te}[e_i, e_j] = \frac{d(e_i,e_j)}{num}$
  end for
end for

3.4.1. Computing the Shortest Path Distance for a Single Graph

For a pair of changed edges, the shortest path distance is defined as the smallest number of vertices needed to traverse from one edge to the other. Hence, with a slight, and simple modification, we can use existing algorithms for computing shortest path distances between vertices for the distance between edges.

A fast and popular algorithm for finding the exact shortest path distance is Dijkstra's algorithm (Ahuja, Magnanti & Orlin 1993), which is the equivalent to breadth-first search in unweighted graphs. This has a worst case complexity of $O(|E|)$ (Cook, Cunningham, Pulleyblank & Schrijver 1998). We have improved upon breadth-first search by using the following two techniques.

Bi-directional search We can improve this search by using a bi-directional search, starting at both the source and target edges. This is known (Kaindl & Kainz 1997) to decrease the average number of vertices (edges) visited for each distance computation, compared with a unidirectional search. From empirical testing on synthetic and scale free graphs, this produces a speed improvement of up to 70%.

Limiting search depth Let d_{limit} represent the maximum depth to which the shortest path distance search expands. For bi-directional, it is the total depth of both searches, i.e., the sum of the depths from each search. Another speed up

technique is to limit the depth to which the search expands to d_{limit}. When the depth limit is reached, the two edges in question are considered far apart and given a maximum distance of 1. Obviously, when the distance of two edges is actually more than d_{limit}, then this will incorrectly identify them as being far apart, i.e., it will overestimate the actual distance. However, our experiments have shown that setting a small d_{limit} does not have a large effect on accuracy. This is particularly true when the regions themselves do not have a diameter greater than d_{limit}.

3.5. Discovering Regions of Correlated Spatio-Temporal Change for each Subsequence (*regDiscovery*)

The aim of this component of cSTAG is to discover the set of regions of correlated spatio-temporal change for each subsequence of graphs. Recall that we seek sets of edges that have high temporal and spatial similarity, measured by the set of pairwise temporal and spatial distances computed earlier. To find these regions, we use clustering techniques to group the edges based on their temporal and spatial distances. Each cluster of changed edges represents a region of correlated spatio-temporal change with lifetime $[ts, te]$. The clustering methods used define the temporal and spatial grouping criteria $criteria_{tem}$ and $criteria_{spa}$ respectively. The clustering methods used also determine the shape that the edges in a region of correlated spatio-temporal change will form. For example, single linkage (Duda, Hart & Stork 2000) generally produces elongated clusters, hence the regions produced using single linkage clustering will tend to be elongated as well.

However, many existing clustering methods (Jain & Dubes 1998)(Duda et al. 2000) cannot be directly used to find the regions of correlated spatio-temporal change in a subsequence. The reason is that these existing methods are designed to find clusters that satisfy some optimality measure on **one** distance measure. However, the problem of finding regions of correlated change involves clustering and optimising both $criteria_{tem}(R_r, d_{tem})$ and $criteria_{spa}(R_r, d_{spa})$ **simultaneously**.

This problem has similarities to the co-clustering (also known as bi-clustering) problem (Cheng & Church 2000)(Dhillon 2001). In co-clustering, two sets of objects have one distance relation defined over them - e.g., document-term frequency analysis in information retrieval (Dhillon 2001) - and the aim is to find co-clusters (subsets of both object sets) that minimise or maximise some objective measure. However, the problem of finding regions of correlated spatio-temporal change is not exactly the same. There is only one object set (edge-edge), but two distance relations (spatial and temporal). Therefore, co-clustering cannot be used.

Hence, we propose a variety of clustering methods to solve this clustering problem. Each of these methods implement the *regDiscovery()* function from Algorithm 1 differently. These include (1) combining the two distance measures into a single one, and (2) clustering using one set of relations, then further refining the obtained clusters using the other set of relations (possibly using a different clustering method). In the rest of the section, we detail each of these **multi-relation** clustering methods. In addition, we describe a well-known single-relation clustering method that we have used.

3.5.1. Combining Distance Measures

One method to solve the multi-relation clustering problem is to combine the two distance measures into one, then use an existing single-relation clustering method to find the regions. We outline two proposals, one based on constrained clustering, the other combining the two distances using a weighted linear sum. After combining the two distance measures into one, the changed edges are clustered using the combined distances. The process for both proposals is the same, apart from the way the distance measures are combined. Algorithm 3 outlines the overall process.

Algorithm 3 Outline of *regDiscovery()* function for Hard and Soft Modification Methods.

1: **Input:** $E_C^{ts,te}$ - set of changed edges, $D_{tem}^{ts,te}$ - set of pairwise temporal distances, $D_{spa}^{ts,te}$ - set of pairwise spatial distances, γ - other parameters specific to the single-relation clustering algorithm
2: **Output:** $\mathbf{R^{ts,te}}$ - a set of regions for window $W^{ts,te}$
3:
4: *// Combine the temporal and spatial distances into a single distance measure*
5: $D_{comb}^{ts,te} = combine(D_{tem}^{ts,te}, D_{spa}^{ts,te})$
6: *// Cluster the changed edges using single-relation clustering on the combined distances*
7: $\mathbf{R^{ts,te}} = cluster(E_C^{ts,te}, D_{comb}^{ts,te}, \gamma)$

Hard Modification The *hard modification* multi-relation clustering method is similar in principle to constrained clustering (Tung, Ng, Lakshmanan & Han 2001)(Wagstaff & Cardie 2000). One of the relations is chosen as the main relational space (optimised space), while the other (constraint space) is used to constrain and modify the optimised distance relation.

Definition 7. Let T_{con} $(0 \leq T_{con} \leq 1)$ be a user defined constraint threshold, and $d_{opt}(e_i, e_j)$ and $d_{con}(e_i, e_j)$ be the optimised and constraint distance relations over edges $e_i, e_j \in E_C$. Then the constrained optimised distance $d_{comb}(e_i, e_j)$ for edges e_i, e_j is defined as:

$$d_{comb}(e_i, e_j) = \begin{cases} d_{opt}(e_i, e_j) & d_{con}(e_i, e_j) \leq T_{con} \\ 1 & \text{otherwise} \end{cases}$$

The advantage of this approach is that it is extremely simple to implement and comparatively efficient - it can be partially precomputed during the distance calculation step[1]. However, the disadvantages of this approach are that T_{con} needs to be tuned, and all object pairs with $d_{con}(e_i, e_j) \geq T_{con}$ are set to the same, maximum distance. This ignores the original distances, which may decrease the accuracy of the clustering - if two objects are close in the optimisation space, but their distance in the constraint space is just above the threshold, then this is regarded as being just as dissimilar as two objects that are far in both spaces. This motivates the next technique, which does not use hard cutoffs.

[1] Compute $d_{con}(e_i, e_j)$ first, using the distance values to avoid computing $d_{opt}(e_i, e_j)$ if $d_{con}(e_i, e_j) > T_{con}$

Soft Modification The second distance modification approach combines the two distances using a weighted linear sum.

Definition 8. Let $\alpha, 0 \leq \alpha \leq 1$ be a user defined parameter to determine the relative weight of d_{con} and d_{opt} in the combined distance measure. Then the combined distance is defined by

$$d_{comb}(e_i, e_j) = (1 - \alpha) \cdot d_{opt}(e_i, e_j) + \alpha \cdot d_{con}(e_i, e_j)$$

Again, the advantage of this method is that it is simple and relatively efficient to compute. However, there is a parameter α to tune again, and there is an implicit assumption that the relationship between the two distance relations is linear, which might not necessarily be the case.

3.5.2. Sequential Clustering

A disadvantage of the previous schemes was that one (modified) distance measure and one clustering method was used to cluster objects in two different spaces. Depending on the measures and the true distribution, clustering separately might produce more accurate results. Hence, another relatively simple approach is to cluster in one space first, then further partition each discovered cluster using the other distance relation. The resulting sub-clusters are the final clusters. Again, any clustering method can be used for clustering in both steps. The disadvantage of this is the sensitivity of the choice of which distance relation to use first. Such order dependence is less than in the distance modification approaches, but it is still important, as the initial clustering in the first domain constrains which sub-clusters can be found in the second clustering. Algorithm 4 outlines the sequential clustering process. In our cSTAG implementation, we cluster using the temporal distances first, then the spatial ones. We found that by using the temporal distances first, we obtained better accuracy. This is because in general, the temporal distances between two regions are more likely to be larger than the spatial distances. Hence, separating by temporal distance first enables the algorithm to make fewer mistakes in the first clustering step.

Algorithm 4 Outline of *regDiscovery()* function for Sequential Method.

Input: $E_C^{ts,te}$ - set of changed edges, $D_{tem}^{ts,te}$ - set of pairwise temporal distances, $D_{spa}^{ts,te}$ - set of pairwise spatial distances, γ - other parameters specific to the initial single-relation clustering, η - other parameters specific to the single-relation sub-clustering algorithm algorithm
Output: $\mathbf{R^{ts,te}}$ - a set of regions for window $W^{ts,te}$

// Initially cluster using the temporal distance and method cluster1
$\mathbf{R'} = cluster1(E_C^{ts,te}, D_{tem}^{ts,te}, \gamma)$
// For each cluster R_r, perform further clustering using method cluster2 on the spatial distances
for each $R_r \in \mathbf{R'}$ **do**
    $R_{sub} = cluster2(R_r, D_{spa}^{ts,te}, \eta)$
    $\mathbf{R^{ts,te}} = \mathbf{R^{ts,te}} \cup \{R_{sub}\}$
end for

3.5.3. Single Relation Clustering

To group the changed edges into clusters using a single distance relation, we require a single-relation clustering method. The number of clusters is not known a priori, hence we require a clustering method that does not require knowing the number of clusters beforehand. Therefore, we chose a partitional method, **leader-follower**, which we describe below. For comparison, we also tested two hierarchical methods - single and average linkage clustering. From our experiments, we have found that these hierarchical methods are, at best, slightly more accurate than using the leader-follower method. However, they run much slower, particularly for larger graphs. Since the results of hierarchical clustering do not show any significant difference from the results of using the leader-follower algorithm, we do not present them in this paper.

Leader-Follower: Leader-Follower clustering describes a generic clustering approach, where a threshold is used to decide whether each point should be included in an existing cluster, or a new cluster should be created to incorporate the point. In Duda et al (Duda et al. 2000), a basic leader-follower clustering algorithm is described. This algorithm assumes that a cluster centre can be computed and updated. However, the "points" we are dealing with are edges, and all the distances defined over them are relational, hence there is no concept of a cluster centre. Therefore, we modified the basic leader-follower algorithm described by Duda et al. This modified version is outlined in Algorithm 5.

The algorithm considers each edge and compares the average distance with existing clusters until it finds a cluster that has an average distance that is less than a cluster (region) width threshold ($regWidth$). If an edge cannot be incorporated into any of the existing clusters, a new singleton cluster, with the edge as its member, is created.

3.6. Region Association (*regAssoc*)

The final step in the cSTAG algorithm is region association. Recall that the aim of the region association step is to discover regions whose correlation is longer than the sliding window length ω. This is achieved by associating and merging similar regions that have been discovered in consecutive windows.

As described in the introduction to cSTAG in Section 3.1, the criteria for merging two regions are:

1. Both regions share a significant number of edges (we do not require strict set equality, as it is possible that one or two edges might lose correlation throughout the lifetime of a region, but the majority of edges in the discovered region remain correlated); **and**
2. The subsequences extracted from both regions for the common snapshot subsequence W_{common} are highly similar, i.e., d_{tem} is small.

More formally, we can express these criteria as two inter-region distance functions, d_S and d_Q, and two user specified thresholds, π_{merge} (spatial) and λ_{merge} (temporal). d_S measures the amount of overlap between the edges in the two regions, while d_Q measures the total weighted distance between the waveform sets of the two regions. Hence, given two regions $R_g^{ts,te}$ (extracted from subsequence

Algorithm 5 Modified Leader-Follower Algorithm.

Input: $E_C^{ts,te}$, $D_X^{ts,te}$ - set of distances (X can be either temporal or spatial), $regWidth$ - width threshold of a cluster
Output: $\mathbf{R^{ts,te}}$ - a set of regions for window $W^{ts,te}$

$R^{ts,te} = \{\}$
for each $e_i \in E_C^{ts,te}$ **do**
  inserted = false
  // Insert into existing cluster?
  for each cluster $R_r \in R^{ts,te}$ **do**
    // Average distance ≤ regWidth, so add e_i to cluster R_r
    if $\frac{\sum_{e_j \in R_r} d(e_i, e_j)}{|R_r|} \leq regWidth$ **then**
      $R_r = R_r \cup \{e_i\}$
      inserted = true
      break
    end if
  end for
  if !*inserted* **then**
    // e_i was not inserted in existing regions, so create a new region
    $R_{new} = \{e_j\}$
    $R^{ts,te} = R^{ts,te} \cup \{R_{new}\}$
  end if
end for

$W^{ts,te}$) and $R_h^{ts+1,te+1}$ (extracted from subsequence $W^{ts+1,te+1}$), we merge these two regions if

$$d_S(R_g^{ts,te}, R_h^{ts+1,te+1}) = 1 - \frac{|R_g^{ts,te} \cap R_h^{ts+1,te+1}|}{max(|R_g^{ts,te}|, |R_h^{ts+1,te+1}|)} \leq \pi_{merge} \tag{4}$$

and

$$d_Q(R_g^{ts,te}, R_h^{ts+1,te+1}) = \sum_{a=1}^{|R_g^{ts,te}.Q|} \sum_{b=1}^{|R_h^{ts+1,te+1}.Q|} Q_g.freq_a \cdot Q_h.freq_b \cdot d_{tem}(Q_g.q_a[2, \omega - 1], Q_h.q_b[2, \omega - 1]) \leq \lambda_{merge} \tag{5}$$

$0 \leq \pi_{merge} \leq 1$, $0 \leq \lambda_{merge} \leq 1$.

Recall that $R_r.Q$ denotes the set of change waveforms of region R_r. Equation 5 compares each waveform in $R_g^{ts,te}.Q$ with each waveform in $R_h^{ts+1,te+1}.Q$ over the common period W_{common}. The distance between the two waveforms is weighted by the relative frequency of each waveform in their respective Q sets.

Without loss of generality, we can extend merging $R_g^{ts,te}$ and $R_h^{ts+1,te+1}$, which are generated by a sliding window process with increment $inc = 1$, to merging $R_g^{ts,te}$ and $R_h^{ts+inc,te+inc}$ where $inc > 1$.

Algorithm 6 provides an overview of the region association procedure. Basically, an evolution graph is constructed from the set of regions discovered for each window. The vertices of the evolution graph are the regions discovered,

and there exists an edge between two vertices if their corresponding regions are discovered in adjacent windows and their spatial and temporal inter-region distances are below the respective thresholds. Then each connected component in the constructed evolution graph will represent discovered regions that are similar across windows. Hence, the final step in the region association step is to find the connected components of the evolution graph, and merge these regions. Merging two regions involves unioning the edges of the regions, and concatenating the waveforms. Concatenating the waveforms is non-trivial, as each region can consist of a set of waveforms. However, we can use the same rationale as used in computing d_Q, i.e., if two waveforms have a high relative frequency in their respective merged regions, then we should assign a high frequency to the resulting concatenated waveform, as the parent waveforms were frequent in their respective regions. Therefore the concatenated waveform should also be frequent in the merged region.

If the two regions to be merged are R_g and R_h, then the merged region R_{gh} has edge set $R_g \cup R_h$, and its set of (waveform, frequency) pairs $R_{gh}.Q$ is $\{(concat(q_a[1, \lceil \frac{\omega}{2} \rceil],\ q_b[\lceil \frac{\omega}{2} \rceil + 1, \omega]),\ freq_a \cdot freq_b) \mid (q_a, freq_a) \in R_g.Q,\ (q_b, freq_b) \in R_h.Q\}$.

At the end of the region association step, we have a set of merged regions, representing the regions of correlated spatio-temporal change of different correlation lengths.

4. Evaluation

In this section, we evaluate the performance of cSTAG using two types of datasets. First, we use synthetic datasets to test the accuracy of cSTAG. In addition, we demonstrate the practical benefits of using regions of correlated spatio-temporal change to detect local routing events in the BGP connectivity graph from the Internet, and also flash crowd events at the 1998 World Cup website. In each of the evaluations, we first describe the dataset and then present the results.

4.1. Synthetic Dataset Evaluation

4.1.1. Accuracy Evaluation Measure

We require a quantitative measure to evaluate the sensitivity and accuracy of cSTAG to its parameter settings. We have used a measure from cluster validation (Halkidi, Batisakis & Vazirgiannis 2001). Cluster validation involves comparing two sets of clusters - the reference clustering, and the actual clustering obtained from the evaluated clustering method. Although there are many proposed measures, we present results using only the Jaccard coefficient (Halkidi et al. 2001). We have considered other measures such as the *Rand index* (Halkidi et al. 2001) and *Minkowski measure* (Halkidi et al. 2001), but found that they yielded the same results as the *Jaccard coefficient*.

Let $C = \{C_1, \ldots, C_u\}$ be the actual clustering results, and $P = \{P_1, \ldots, P_v\}$ be the reference clustering. If both clusterings partition a data set X, then for each pair of objects, $(x_i, x_j), x_i \neq x_j, x_i, x_j \in X$, it is possible to classify them as follows (Halkidi et al. 2001): SS, if both objects belong to the same cluster in both clusterings; SD, if the objects belong in the same cluster for clustering

Algorithm 6 Procedure for region association step - function *regAssoc()*.

Input: $\mathbf{R}^{\mathbf{1,1+\omega}}, \mathbf{R}^{\mathbf{1+inc,1+\omega+inc}}, \ldots$ - sets of regions, one per window (there should be $winNum = \lceil \frac{S-\omega+1}{inc} \rceil$ sets in total)
Output: **R** - the regions of correlated spatio-temporal change

$\mathbf{R} = \{\}$
// Construct the empty evolution graph G_{evol}
G_{evol}
// Loop through each pair of adjacent region sets
for g := 1; $g \leq winNum - inc$; g += inc **do**
  // Add vertices to G_{evol} *for each region in the two adjacent region sets* $R^{g,g+\omega}$ *and* $R^{g+inc,g+inc+\omega}$
  for each $R_1 \in R^{g,g+\omega}$ **do**
    $G_{evol}.addVertex(R_1)$
  end for
  for each $R_2 \in R^{g+inc,g+inc+\omega}$ **do**
    $G_{evol}.addVertex(R_2)$
  end for

  // Compute temporal and spatial merge similarities between all regions in $R^{g,g+\omega}$ *and* $R^{g+inc,g+inc+\omega}$
  for each $R_1 \in R^{g,g+\omega}$ **do**
    for each $R_2 \in R^{g+inc,g+inc+\omega}$ **do**
      if $d_S(R_1, R_2) \leq \pi_{merge}$ and $d_Q(R_1, R_2) \leq \lambda_{merge}$ **then**
        // Add edge between the vertices corresponding to the regions
        $G_{evol}.addEdge(R_1, R_2)$
      end if
    end for
  end for
end for
// Find connected components among the vertices (i.e. regions) in G_{evol}
$connComp = getConnectedComponents(G_{evol})$
// Merge the regions in each connected component to form merged region R_X
for each $conn_i \in connComp$ **do**
  $R_X = merge(conn_i)$
  $\mathbf{R} = \mathbf{R} \cup \{R_X\}$
end for

C, but different clusters for clustering P; DS, if the objects belong to different clusters for clustering C, but in the same cluster for clustering P; DD, if the objects belong in different clusters for clustering C and P. Let n_{SS}, n_{SD}, n_{DS} and n_{DD} denote the number of SS, SD, DS and DD pairs respectively. Then the *Jaccard coefficient* is defined as (Halkidi et al. 2001):

$$JC = \frac{n_{SS}}{n_{SS} + n_{SD} + n_{DS}}$$

The *Jaccard Coefficient* ranges from 0 to 1. It measures the similarity between the clusterings. A score of 1 indicates a perfect match between the actual and reference clusterings, while a score of 0 indicates a complete mismatch.

Data set		Vertices (Edges)	Regions	Comments
closeDiff	different	100 (400)	3	All different types of change.
	combo	100 (400)	4	Flapping type of changes.
farSameCloseDiff		100 (400)	3	Close regions have different change, but distant regions have the same change.
cross		100 (400)	5	Cross shaped region, with four surrounding regions.
scale-free		339-5339 (1000-16000)	9	Graphs with scale-free properties.

Table 5. Synthetic datasets. *closeDiff, farSameCloseDiff, cross* datasets consist of 10 snapshots each, while the *scale-free* datasets are each 30 snapshots long.

To evaluate cSTAG using cluster validation, we simulated several datasets and manually introduced changes, so that we know the correct reference regions of change. We then applied the Jaccard Coefficient to measure the accuracy of cSTAG, using the discovered regions as the actual clustering to be evaluated, and the introduced regions as the reference clustering.

4.1.2. Dataset Description

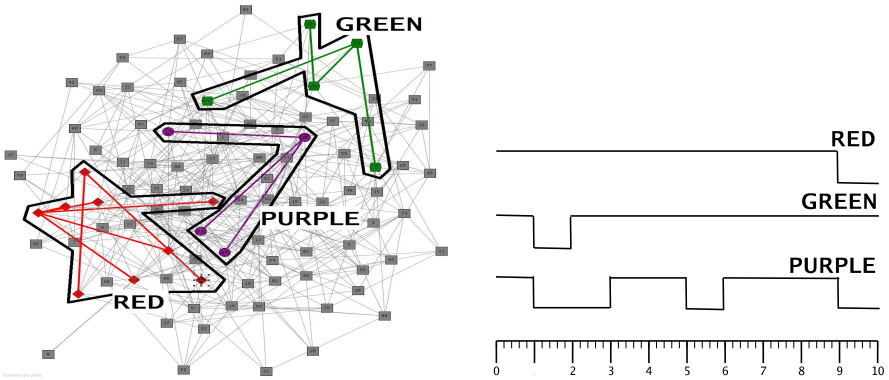
To create the reference clusters, we first constructed several synthetic graphs and manually introduced changes into them. The results are the set of four synthetic datasets in Table 5[2] (we shall call these the *manual synthetic datasets*). The aim of the *closeDiff* datasets is to test the accuracy with respect to temporal similarity. The first dataset, *different*, tests different sequences of changes, each with their own unique transition sequences. The second dataset, *combo*, tests multiple regions that have the same sequence of transitions, but where the timing of the actual changes are different.

The *farSameCloseDiff* dataset has several regions that have different types of change, but in close topological proximity. There are also pairs of regions that are far apart, but have the same type of change. This tests any potential tradeoffs between the spatial and temporal similarities.

There is the more challenging dataset *cross*. The *cross* dataset consists of a region resembling a cross, with four other regions separated by the cross region. The four regions have the same type of change. The *cross* dataset tests how well the algorithm detects non-spherical regions.

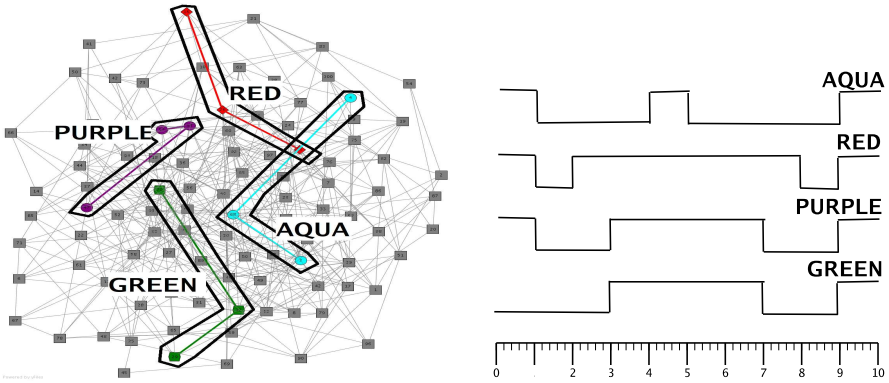
In addition, we generated a set of scale-free graphs, based on the Barabasi model (Barabasi & Albert 1999), and randomly introduced regions of change into them. Each region generally forms a connected component, though this is not guaranteed. We generated scale-free graphs to test the effect of ω_{spd}, the union window size, and d_{limit}, the maximum depth limit used when calculating the shortest path distance, on the running time and accuracy. We found that ω_{spd} had little effect on the accuracy for the four other synthetic datasets. We also tried generating graphs where we randomly generated regions, then added paths between them - again ω_{spd} did not affect the accuracy. In addition, we needed larger graphs to test the effect of d_{limit} and the scalability of cSTAG, which the Barabasi model allows us to easily do. Hence, we used scale-free graphs, which have been found to accurately model many real life graphs.

[2] Available at `www.csse.unimelb.edu.au/~jkcchan/`



(a) Graph. (b) Corresponding waveforms.

Fig. 4. *Different* dataset.



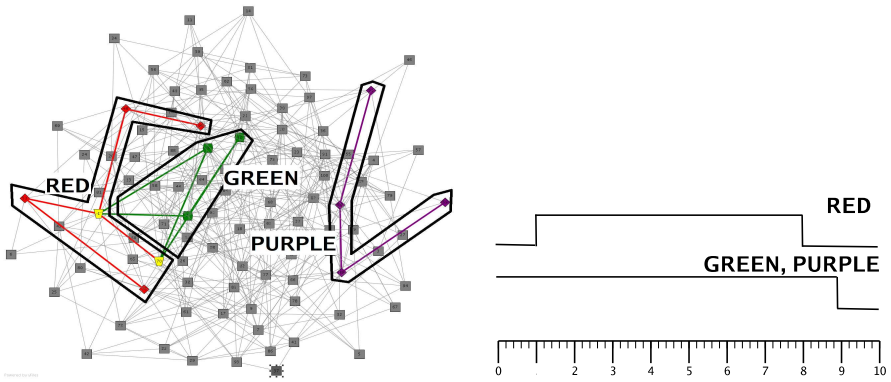
(a) Graph. (b) Corresponding waveforms.

Fig. 5. *Combo* dataset.

4.1.3. Manually Constructed Synthetic Datasets

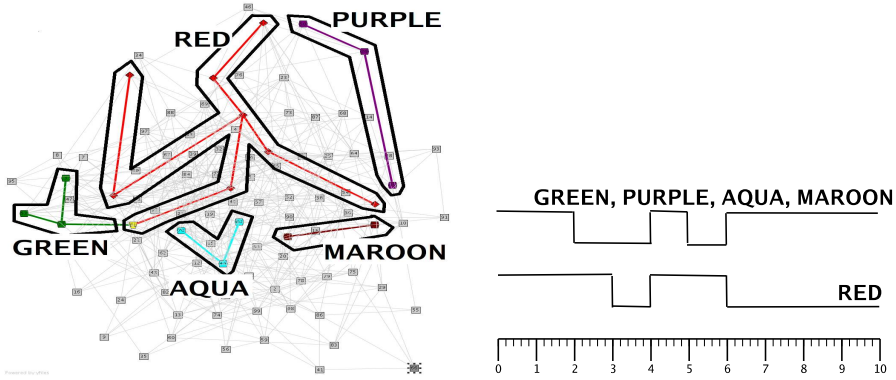
To evaluate the accuracy of our approach, we applied cSTAG to the first four datasets in Table 5, i.e., *different*, *combo*, *farSameCloseDiff* and *cross*. In each case, we tested the three different clustering techniques that we proposed in Section 3.5.1 and 3.5.2 for combining temporal and spatial distance measures, i.e., hard modification, soft modification and sequential clustering, combined with the leaderFollower algorithm. For each clustering method, we measured how the accuracy of the results varied as we changed the parameter settings in each algorithm. Note that in the hard and soft modification clustering techniques, *temporal* distance is used as the optimised criterion d_{opt}, and *spatial* distance is used as the constraint criterion d_{con}. Similarly, in sequential clustering, we first cluster using temporal distance, and then further partition each cluster based on spatial distance.

We do not report the timing results for these four datasets because their running time was in the tens of milli-seconds, which is too small to make any



(a) Graph. (b) Corresponding waveforms.

Fig. 6. *FarSameCloseDiff* dataset.



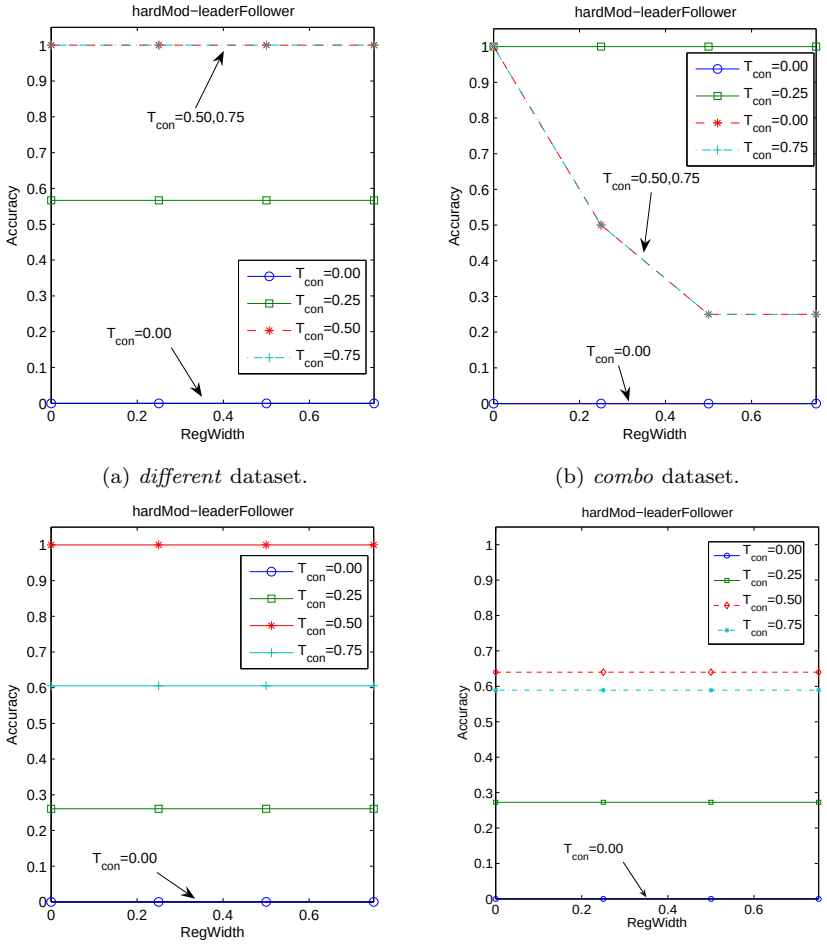
(a) Graph. (b) Corresponding waveforms.

Fig. 7. *Cross* dataset.

meaningful comparisons. We examine the effect of the significant parameters (d_{limit} and ω_{spd}) on the timing in the next subsection, using the scale-free graphs.

The accuracy results for each synthetic dataset are shown in Figures 8 to 10. For the hard modification method (Figure 8a to 8d), the results of varying T_{con} and $regWidth$ on the accuracy are shown. For the soft modification method (Figure 9a to 9d), the results of varying α and $regWidth$ on the accuracy are shown. For the sequential method (Figure 10a to 10d), we show the results of varying the sliding window size ω and $regWidth$ on the accuracy. Varying the window size ω tests the accuracy of the merging portion of cSTAG. Note that we have also evaluated the effect of varying ω on the other two clustering methods, but obtained similar trends as the results for sequential clustering, hence we do not present those results. In addition, we found that varying λ_{merge} and π_{merge} from 0.05 to 0.50 did not affect the accuracy over all window sizes ω and all combining methods. Hence, to avoid over reporting, we do not present the accuracy graphs when varying the two merging thresholds.

In each figure, we show the clustering accuracy for each clustering technique based on the Jaccard measure. Let us first consider each set of results in detail.



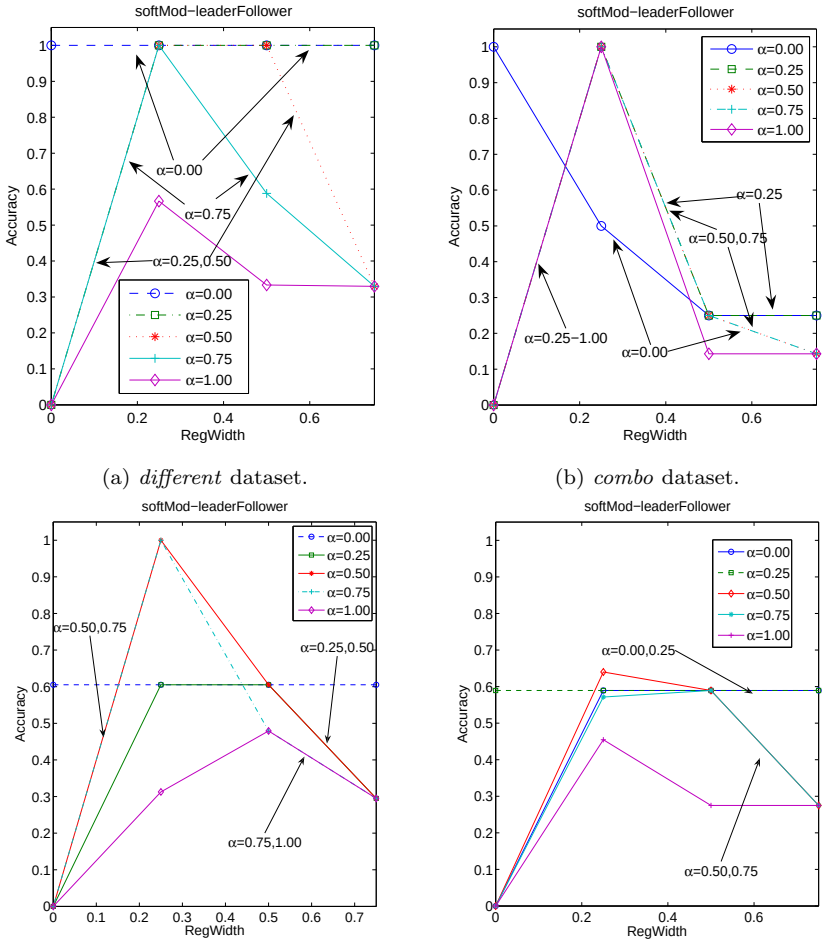
(a) *different* dataset. (b) *combo* dataset.

(c) *farSameCloseDiff* dataset. (d) *cross* dataset.

Fig. 8. Accuracy vs. regWidth results for the *hard* modification method on the manual synthetic datasets. The effect of varying the constraint threshold T_{con} is also presented.

Hard modification - When hard modification is used, if the threshold on distance in the constraint space is zero ($T_{con} = 0$), then spatial distance is not considered by the clustering algorithm, and the results only depend on the temporal similarity of the change waveforms. Otherwise, the clustering results are highly sensitive to the choice of T_{con}. As shown in Figures 8c and 8d, if T_{con} is too low, then the spatial distance is too tightly constrained, and the true regions are fragmented into smaller regions. Conversely, if T_{con} is too high, then the discovered regions can contain excessive variation in spatial distance, and distant regions with the same temporal change are incorrectly merged.

Soft modification - When soft modification is used, the accuracy of the discovered regions is highly sensitive to the choice of values for the cluster width parameter *regWidth* in the Leader Follower clustering algorithm, as well as the choice of value for α, which determines the relative importance of the constraint



(a) *different* dataset.

(b) *combo* dataset.

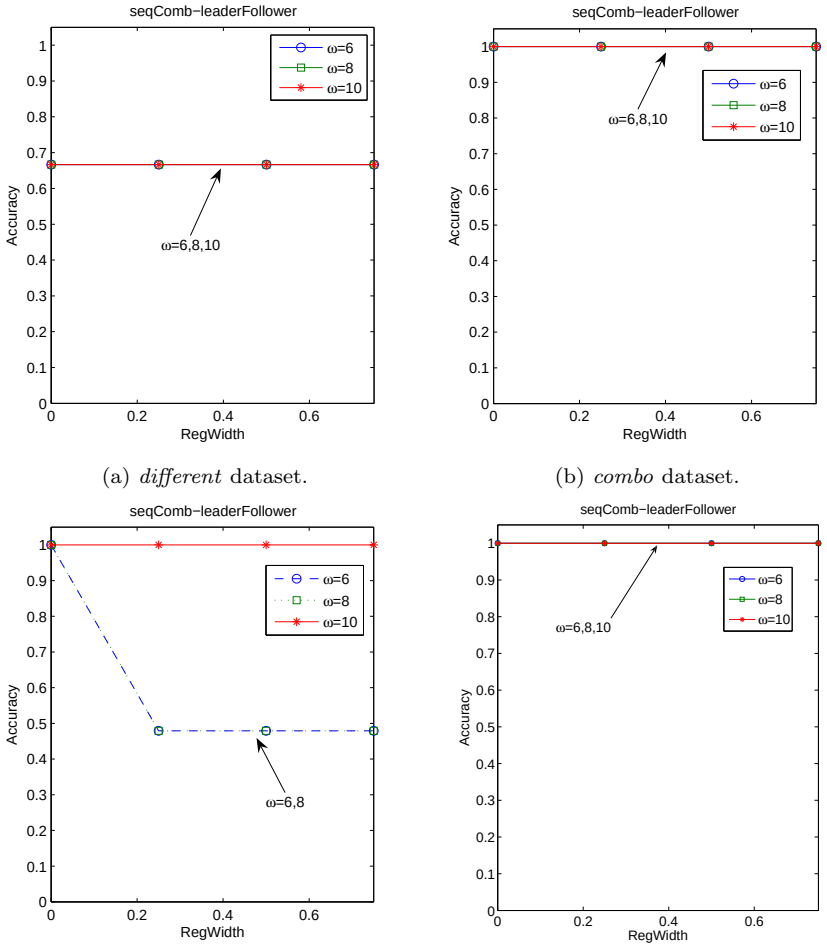
(c) *farSameCloseDiff* dataset.

(d) *cross* dataset.

Fig. 9. Accuracy vs. regWidth results for the *soft* modification method on the manual synthetic datasets. The effect of varying the weighting parameter α is also presented.

distance, i.e., spatial distance. When $\alpha = 0$ for the *different* and *combo* datasets (see Figure 9a, 9b), 100% accuracy can be achieved, since only temporal distances are considered. Otherwise, in all datasets, the maximum accuracy that can be achieved depends on how the settings of α and *regWidth* are fine-tuned. If *regWidth* is too small, then regions become fragmented. However, if *regWidth* is too large then distant regions can be erroneously merged together.

Sequential clustering - In comparison to hard and soft modification, sequential clustering is able achieve high accuracy, with little sensitivity to the choice of parameters. In sequential clustering, edges are first grouped based on their temporal similarity. Thus, dissimilar changes are kept separate, e.g., in the *combo* dataset (see Figure 10b). Each temporally coherent region is then further partitioned based on spatial distance. Thus, distant regions with similar change waveforms are separated, e.g., in the *farSameCloseDiff* and *cross* datasets (see



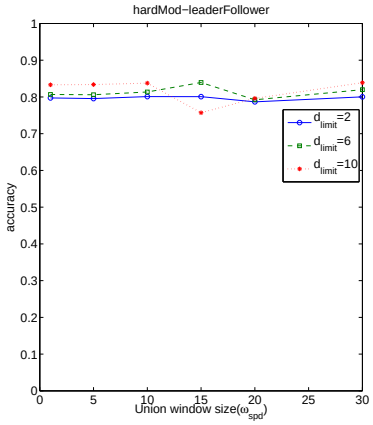
(a) *different* dataset. (b) *combo* dataset.

(c) *farSameCloseDiff* dataset. (d) *cross* dataset.

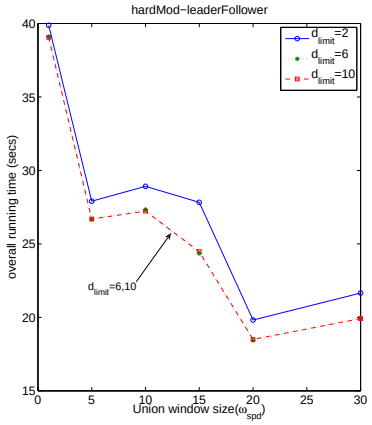
Fig. 10. Accuracy vs. regWidth results for the *sequential* method on the manual synthetic datasets. The effect of varying the sliding window size ω is also presented.

Figure 10c, 10d). Note that some errors can occur if the sliding window size is too small, e.g., when $winSize = 6$ for the *farSameCloseDiff* dataset (see Figure 10c). In this case, there are time windows when the change waveforms appear similar, and hence regions can be erroneously merged. However, in most cases high accuracy can be achieved.

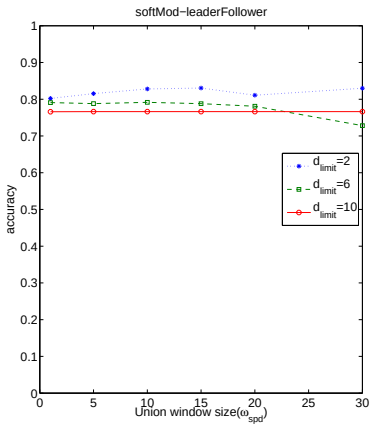
In summary, of the three clustering methods, sequential clustering generally provides the most accurate and robust results across all our synthetic datasets. While hard and soft modification can achieve good results, they are much more sensitive to the choice of parameter settings.



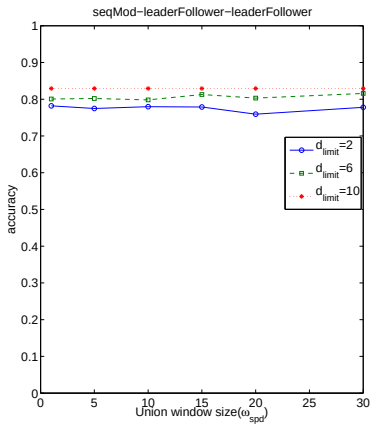
(a) Accuracy vs. union window size, ω_{spd}. This is for the *hard modification* method, with leaderFollower clustering.



(b) Total running time vs. union window size, ω_{spd}. This is for the *hard modification* method, with leaderFollower clustering.



(c) Accuracy vs. union window size, ω_{spd}. This is for the *soft modification* method, with leaderFollower clustering.



(d) Accuracy vs. union window size, ω_{spd}. This is for the *sequential* method, with leaderFollower clustering methods for both domains.

Fig. 11. Accuracy and timing results vs. union window size ω_{spd} for the scale-free graphs of 1,000 edges. 30% of the edges in the graphs experienced change. The depth limit d_{limit} is also varied, from 2-10. The results are for the hard and soft modification, and sequential methods, with leader-follower clustering.

4.1.4. Scale-free Graphs

We have found the shortest path distance calculations dominate the running time as the graphs increase in size. As the union window size ω_{spd} and the depth limit d_{limit} are the main factors that determine the running time of the shortest path distances, our aim is to test the specific effects of these parameters. In this section, we test the effect of ω_{spd} and d_{limit} on accuracy and running time using scale-free graphs.

We varied the size of the scale-free graphs from 1,000 to 16,000 edges, generating three different graphs for each size. We then introduced a different set of changes to each graph, and averaged the accuracy and timing results. The total percentage of edges in each graph that experienced change was 30%. We also tried varying 10%-60% of the edges in each graph, but the results exhibited the same trends as the 30% case, hence they are not presented. Figure 11 shows the effect of varying ω_{spd} and d_{limit} on accuracy and timing for the scale-free graphs of 1,000 edges.

From Figure 11, we can make the following observations:

1. As the union window size (ω_{spd}) approaches the length of the whole sequence of snapshots, the running time improves by up to a factor of two. Moreover the accuracy does not degrade, and in some cases actually improves. This shows that the short path distances on scale-free graphs are invariant to a significant number of random introduced changes (the introduced changes), hence having a large union window size does not affect accuracy. The results also demonstrate that using a large union window size can reduce the running time without decreasing the overall accuracy.
2. A small depth limit (d_{limit}) results in a slightly increased running time. This is because on average more regions are found per window, and hence the region association step requires more time to merge regions, even though less time is needed for the shortest path calculation when d_{limit} is small. However, as we shall discuss shortly, when the size of the graphs increases, the running time of the region association step becomes insignificant when compared to the running time of the shortest path calculations. In those larger graphs, having a smaller d_{limit} results in smaller total running times.

In order to measure the scalability of cSTAG in terms of running time on large networks, we varied 20% of the edges on each of the generated scale-free graphs and measured the time required for cSTAG to find the regions therein. We tested the leader-follower algorithm with hard modification, soft modification and sequential methods, and found they required similar running times. Hence, we only present the timing results for hard modification with leader-follower clustering as shown in Figure 12. In Figure 12 we separate the total running time into three components: (1) the time required to compute the temporal and shortest path distances (*distance*), (2) the time required for clustering (*clustering*), and (3) the time required for region association (*association*).

Figure 12 shows that the distance calculations, particularly the shortest path distance calculations, dominate the running time. The clustering and association steps constitute an insignificant portion of the running time. In future work, we plan to try and decrease the time taken for shortest path distance calculations by coarsening the graph into a smaller, approximate version (Wu, Garland & Han 2004), and using zones to measure the shortest path distances in the coarsened graph (Rattigan, Majer & Jensen 2006).

4.2. BGP Dataset Evaluation

In this section, we demonstrate the effectiveness of cSTAG on a practical problem by analysing the effect of the 2005 Hurricane Katrina on the US portion of the Border Gateway Protocol (BGP) connectivity graph. This analysis has partly been presented in (Chan et al. 2006). The BGP connectivity graph represents

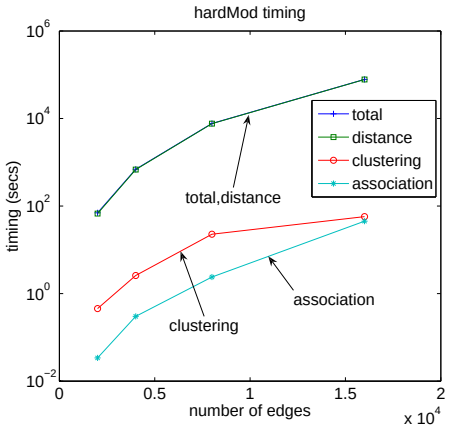


Fig. 12. Effect of graph size on total computation time of cSTAG, using scale-free graphs where 20% of edges have experience a change. Also shown is the portion of time spent on distance calculation, clustering and region association.

the top-level routing topology of the Internet. In this analysis, we compare the output of cSTAG on the BGP graph with known events of affected locations (Cowie et al. 2005). We also highlight the difficulty in identifying meaningful regions if cSTAG is restricted to using only spatial distances alone or temporal distances alone.

4.2.1. BGP Data Source

BGP is a routing protocol used to establish the forwarding tables between the routers of organisations, known as Autonomous Systems (ASs), on the Internet. The vertices in the BGP connectivity graph represent the ASs, and the edges represent the existence of a routing path between the ASs. An important challenge in managing the Internet is how to detect problems in the BGP routing topology and diagnose the event that caused each problem (Feamster, Balakrishnan & Rexford 2004).

In order to understand how the BGP graphs were built from routing tables, we briefly introduce how paths are stored in the tables. Each BGP routing table entry can be summarized as a network prefix and its AS PATH attribute. AS PATH lists the path of ASs that was used by the original announcement in reaching the current router and its AS. For example, AS1-AS2-AS3 means the prefix originated from AS3, and the announcement propagated from AS3 to AS2 to AS1, before reaching the current AS.

The RouteViews project (of Oregan n.d.) collects BGP routing information by passively peering with a number of distributed ASs. From each table obtained from RouteViews, we built a snapshot of the BGP connectivity graph using the AS PATH path entries. By using multiple path entries it is possible to construct an approximation of the true BGP connectivity graph.

4.2.2. Hurricane Katrina

We examine the Katrina event because it has been reported that its effect on the Internet was mostly localised around Louisiana and several other southern states.

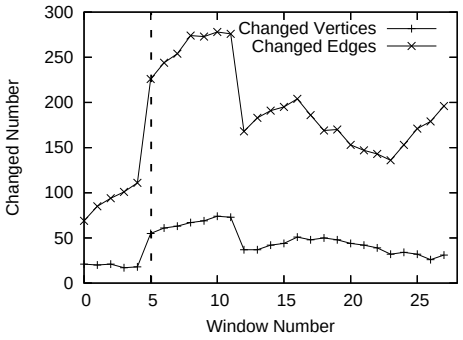


Fig. 13. Number of changed edges and vertices during each time window in the US portion of the BGP graph during the period 28 August to 31 August 2005. The dotted line signifies the landfall of Hurricane Katrina.

We concentrate on the ability of cSTAG to clearly show significant activity in the ASs around that region. We also compare the results found using an analysis reported in (Cowie et al. 2005), which is based solely on an analysis of global statistics of the network as a whole.

We concentrate on the US portion of the BGP graph, as this was large enough to hide very localised events, like the Hurricane Katrina event. In August 2005, the US BGP graph consisted of around 9,000-10,000 vertices and 45,000 edges. We analysed three and half days of snapshots, from 29 August, 13:19 to 31 August, 22:32. This period included the landfall of Hurricane Katrina (around 29 August, 10:00). This corresponds to the period between snapshots 11 and 12. As the synthetic dataset evaluation showed, sequential clustering was most stable and accurate, hence we use it for our BGP analysis. In addition, we found a *regWidth* of 0.25, window size ω of 10, window increment *inc* of 1, and π_{merge} and λ_{merge} of 0.2 produced the clearest results. In fact, window sizes from 6 to 10 produced very similar results, again highlighting the accuracy of the merging part of cSTAG.

Event Separation To demonstrate the difficulty of analysing the changes using only global statistics of the graph as a whole, we first consider how the total number of individual changes for each window fluctuated during the landfall of Katrina. We then compare and demonstrate the ability of cSTAG to separate specific events among this global set of changes.

Consider Figure 13, which shows the number of edges and vertices that have experienced a change in each window. It shows there is a significant number of changing edges and vertices that need to be analysed, even before the landfall of Katrina. This is partly due to general background changes, and Katrina's earlier effects on the Florida part of the network. So even before major events like the landfall of Katrina, there are 50-100 individual edge changes that need to be examined by users. During the window immediately after the landfall of Katrina, the number of changed edges rises to nearly 300. Given that the only knowledge available for each individual changed edge is whether it appeared or disappeared between adjacent snapshots, it is difficult to determine any pattern from the individual changes.

Contrast this with the regions of correlated spatio-temporal change discov-

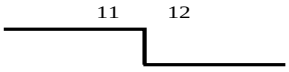
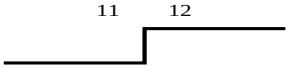
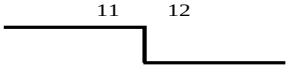
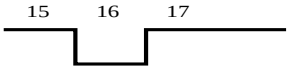
Region	No. of Edges	Change Waveform	Comments
A	41		Main failure region around Louisiana.
B	13		Recovery region spatially adjacent to region A.
C	12		DoD centred failure temporally similar to region A.
D	11		Failure and recovery event in a major ISP.

Table 6. Significant regions discovered by cSTAG in the US portion of the BGP graph during the period 28 August to 31 August, 2005, when Hurricane Katrina made landfall. Numbers in change waveforms indicate the snapshot number.

ered by cSTAG, displayed in Table 6. As the results demonstrate, the discovered regions correspond either to different events, or different local subgraphs affected by the same cause. For example, the merged region labelled A represents a large failure region. It has been reported (Cowie et al. 2005) that a significant percentage of the Louisiana (and Mississippi) network was knocked out by Katrina. To check if region A corresponds to this failure event, we obtained the registered, geographic states of each AS that are connected to the edges of region A, via the *whois* service of the American Registry for Internet Numbers (Ari n.d.). We found that of the 41 unique changed edges that are in region A, 32 of them are connected to an AS registered in Louisiana or Mississippi and seven to the ISPs Sprint and MCI, which experienced some connectivity problems due to Katrina. Therefore it is likely that region A corresponds to this reported disruption.

Region B is a recovery region. All the edges are centred around AS 701 (MCI), and this region appears to represent a recovery from failure problems that occurred before the start of our analysis - hence we only see the appearance of the edges. An important point to note is that the edges in region B are adjacent to region A (via AS 701) but are not incorrectly merged with them, due to the difference in temporal behaviour. This demonstrates the benefit of considering the temporal dimension as well as the spatial dimension, and highlights the ability of cSTAG to discriminate different events that are spatially close. In addition, only five of the edges are in reported connectivity blackout areas (Florida and Texas).

Region C is another failure region, with the time of failure transition also centred around snapshot 12. However, it is identified as a separate region from region A because the majority of the ASs in this region are associated with the Department of Defense and the military, and are spatially separate from each other. The cause of these failures could be due to Katrina, and hence could be merged with region A. However, we argue that it is more useful and interesting to separate the failure of the military network from the civilian organisations represented in region A, as they represent generally different types of organisa-

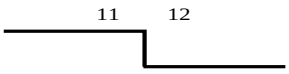
Region	No. of Edges	Change Waveform	Comments
TA	77		This region subsumes region A and B of Table 6.

Table 7. A significant region discovered by using temporal distances only (the *tem-only* strategy) in the US portion of the BGP graph during the period 28 August to 31 August, 2005, when Hurricane Katrina made landfall. Numbers in change waveforms indicate the snapshot number.

Region	No. of Edges	Change Waveform	Comments
SA	243	61 Different change waveforms	This region almost contains all the unique changed edges. It subsumes regions A, B, C and D from Table 6.

Table 8. A significant region discovered by using spatial distances only (the *spa-only* strategy) in the US portion of the BGP graph during the period 28 August to 31 August, 2005, when Hurricane Katrina made landfall.

tions and networks. Again we highlight the point that without considering the spatial dimension, regions A and C would have been merged together.

The changed edges in Region D, representing a failure followed by a recovery, are centred around AS 3356, another major ISP (Level 3). This is clearly a separate event from the other three, and again demonstrates the effectiveness of cSTAG at discriminating multiple simultaneous events.

To further illustrate why we use both temporal and spatial distances for region discovery, we also evaluated two naive strategies that used either temporal distances only (*tem-only*) or spatial distances only (*spa-only*).

Consider the *tem-only* strategy first. We applied this strategy using the same parameters as cSTAG (except that there are no spatial distance considerations). A significant region discovered by the *tem-only* strategy is highlighted in Table 7. As Table 7 shows, we have region TA, which has the same change waveform as the regions A and C discovered by cSTAG (Table 6). Therefore, due to its lack of spatial knowledge, the *tem-only* strategy has incorrectly grouped the edges of regions A and C as region TA.

Now consider the *spa-only* scheme. The results of analysing the Katrina data are shown in Table 8. As the table shows, region SA consists of many edges and a large number of different change waveforms. It has incorrectly grouped most of the changed edges together, including all the regions of Table 6. Hence, this mega-region SA is not very interesting or useful. Therefore, as these two naive schemes show, both temporal and spatial distances must be considered when trying to obtain interesting and meaningful results.

In summary, this analysis demonstrates cSTAG is able to separate events and their impact on the dynamic graph, even if the events, represented by regions, are either topologically adjacent or temporally similar to each other. In comparison to the high level temporal analysis of changes in Figure 13, cSTAG provides much greater insight into the underlying events that have caused the changes.

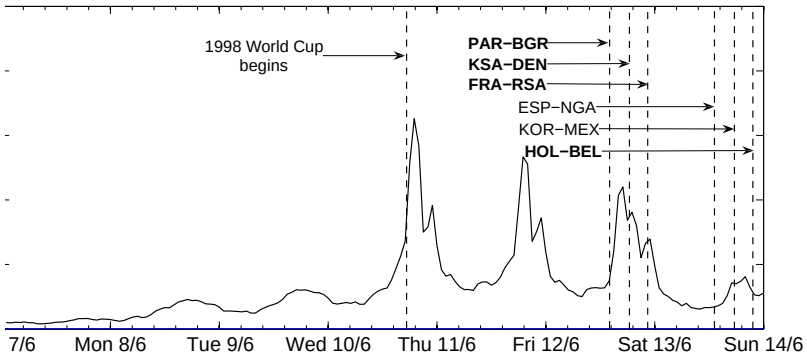


Fig. 14. Hourly Traffic Volume to the 1998 World Cup Website over the period Sunday, June 7 to Saturday, June 13. The matches that the regions in Table 9 correspond to are in bold. Based on Figure 2 (Arlitt & Jin, 1999).

4.3. 1998 World Cup Web Site Evaluation

In 1998, the 16^{th} FIFA World Cup was held in France. To study the workload characteristics of the official web site, *www.france98.com*[3], access logs[4] of the web site was analysed by Arlitt and Jin (Arlitt & Jin 1999).

It was reported by Arlitt and Jin that the website experienced flash crowds - sudden, large increases in the number of unique, legitimate clients accessing the website. This coincided with the time of weekday matches. Arlitt and Jin suggested that during weekdays most people are at work or school and cannot watch the matches on television. The 1998 World Cup was the first world cup where live scores were available online. Therefore, a significant number of fans, who cannot watch the matches on television, monitored the live scores via the website during the matches, producing the flash crowds. Figure 14, which shows the number of requests per hour over the period from June 7 to June 13, illustrates the aforementioned flash crowd effect coinciding with the times of the matches.

We wish to construct a dynamic graph of the website accesses and find regions of correlated change that correspond to the flash crowd events. By studying the change waveforms of the regions, we can corroborate which regions are most likely to be associated with each of the flash crowd events, and determine the set of affected web pages. This demonstrates another application where regions of correlated change can be used to infer underlying events.

4.3.1. Dataset

The access logs consist of a list of website accesses. Each access has a timestamp, client ID (corresponding to the IP address of computer accessing the website), object ID (where an object is any individual file requested by the clients, such as HTML, image, java files for example), and various other (irrelevant) information.

[3] As of August 2007, the address is still valid, but links to a general soccer promotion website.
[4] Available at (wc9 n.d.).

Each flash crowd should to have a set of objects that are uniquely associated with the flash crowd, i.e., objects associated with the team that was playing at the time of the flash crowd. This set of objects should have a sudden, large number of unique clients accessing them over the period of the associated matches, and after the match, this set of objects are no longer frequently accessed at the same time. For example, if Paraguay was playing, then we expect objects relating to Paraguay to be accessed by a large number of the same clients during that period.

Therefore, to infer the flash crowds/matches from the web logs, we construct snapshots of the object-object graph and find regions in the snapshot sequence. Each vertex in the object-object graph represents an object, and a (weighted) edge exists between a pair of objects if one or more clients accessed both objects during the period over which the snapshot is defined. The weights count the number of unique clients accessing the two incident objects. The regions of correlated change discovered over the snapshots represent groups of objects that have been accessed by the same set of clients. These clients should be predominately the fans of the teams playing, hence each flash crowd/match should produce a unique region.

To build the snapshots, we divide the list of accesses into a sequence of two hour snapshots, which roughly correspond to the length of a match, including half-time and regular extra-time. From each two hour bin, we build the object-object snapshots. We then convert the weighted snapshots to unweighted snapshots by setting a filter threshold - edges with weights less than the filter threshold are deleted, and all remaining edges are turned into unweighted edges. As Figure 14 shows, there is a high level of background activity and traffic which is not of interest. We plotted the number of edges with weight x vs. the weight x, and found that the distribution was heavy tailed. The objects involved in the flash crowds predominately have large edge weights between them. This suggests that there are many irrelevant edges with small weights that can be considered as noise and therefore should be filtered out. We set the threshold for filtering irrelevant edges to 500.

4.3.2. Findings on the 1998 World Cup Dataset

In this section, we present and discuss some of the regions of correlated change discovered by cSTAG, over the two day period from 0000 Friday, June 12th to 2359 Saturday, June 13th. This period included several matches/flash crowd events.

We shall discuss five of the discovered regions - three correspond to each of the matches on Friday, one corresponds to one of the matches on Saturday, and finally one corresponds to fans that are interested in multiple matches on Friday. The characteristics of the regions are presented in Table 9.

To obtain a better understanding what the regions represent, we extracted the list of incident objects in each region and obtained the actual files to which these objects correspond. As there are several hundred unique objects in total, we truncated the list of objects in each to the most interesting ones. These lists are presented in Tables 10 to 14.

As the website is no longer available, we do not know the content of the actual webpages or objects, but we can still distinguish the different regions and identify some interesting features of those regions. For example, we can infer that the files ***matchprogXXXX****.htm* refer to the webpages that display the live scores of

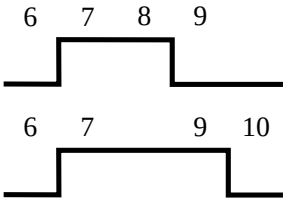
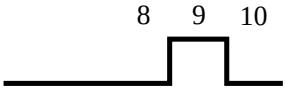
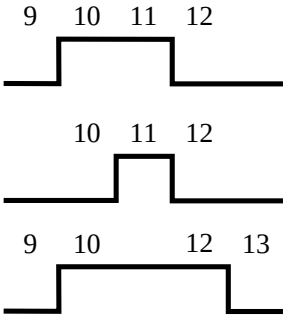
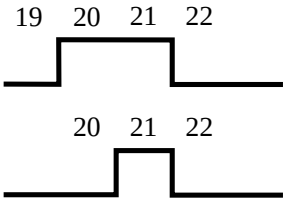
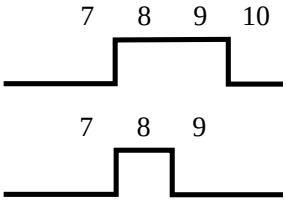
Region	No. of Edges	Change Waveform	Comments
Fri-1	160		Region corresponding to fans that monitored the live scores of match 8893 only, most likely to be the Paraguay vs. Bulgaria match (PAR-BGR).
Fri-2	42		Region corresponding to fans that monitored the live scores of match 8891, most likely to be the South Korea vs. Denmark match (KSA-DEN).
Fri-3	60		Region corresponding to fans that monitored the live scores of match 8892, most likely to be the France vs. South Africa match (FRA-RSA).
Sat-1	31		Region corresponding to fans that monitored the live scores of match 8895, most likely to be the Holland vs. Belgium match (HOL-BEL).
Fri-C	131		Region corresponding to fans that monitored multiple matches on Friday.

Table 9. Characteristics of some of the discovered regions of correlated change over the period 0000, June 12th to 2359, June 13th.

Object/File Name
/eng/teams/teambio128.htm
/eng/teams/teambio136.htm
/eng/competition/groupstandings163_76.htm
/eng/competition/matchstat8893.htm
/eng/competition/**matchprog8893.htm**
/fra/competition/matchstat8893.htm
/fra/competition/**matchprog8893.htm**
/fra/competition/groupstandings163_76.htm

Table 10. List of objects/files for region Fri-1.

Object/File Name
/eng/teams/teambio35.htm
/eng/teams/teambio141.htm
/eng/competition/groupstandings163_75.htm
/eng/competition/matchstat8891.htm
/eng/competition/**matchprog8891.htm**
/fra/competition/**matchprog8891.htm**
/fra/competition/matchstat8891.htm

Table 11. List of objects/files for region Fri-2.

match XXXX, and ***matchstatXXXX****.htm* refer to the webpages that display the statistics of the match. In addition, files of the form ***teambioYYY****.htm* probably refers to the biography of team YYY, and *groupstandings163_77.htm* refers to the group standings of group 164_77.

The four regions labelled Fri-1, Fri-2, Fri-3 and Sat-1 (Tables 10 to 14) are regions that we hypothesise refer to a unique match/flash crowd. Each of these regions have a unique matchprog88XX.htm object. In addition, most of them also have a unique matchstat88XX.htm object. Furthermore, the timing of the changes for all four regions does not coincide. For example, Fri-1, which appears in snapshot 7, is different from region Fri-2 (snapshot 9) and Fri-3 (snapshot 10). This coincides with the timing of each of the three matches on Friday. The fact that the numbering of the team biographies (and group standings) generally do not overlap between regions strengthens this hypothesis. There are some cases where the *matchstat* objects of other matches are in a region, but these are limited, and are likely to be fans checking the results of earlier matches in the day.

The region Fri-C (Table 13) is interesting. It is defined over the same period as regions Fri-1 and Fri-2 and consists of objects that belong to at least one of the regions. This region most likely represents fans that are interested in both the matches. This suggests that hierarchical relationships may exist between the regions.

In summary, we have used the synthetic datasets to show that cSTAG can accurately find regions of correlated spatio-temporal change. In addition, we have shown how the parameters settings in cSTAG affect accuracy and running time. Furthermore, we have analysed two real datasets. We found that the discovered regions for the BGP routing graph correlates with reported routing events during the landfall period of Hurricane Katrina. Finally, we have managed to find different user access patterns to the 1998 World Cup website based on the group of files that were accessed frequently together over time.

Object/File Name
/eng/teams/teambio121.htm
/eng/teams/teambio146.htm
/eng/teams/teambio83.htm
/eng/competition/matchstat8892.htm
/fra/competition/matchstat8892.htm
/eng/competition/groupstandings163_75.htm
/eng/competition/matchstat8891.htm
/eng/competition/**matchprog8892.htm**
/fra/competition/**matchprog8892.htm**

Table 12. List of objects/files for region Fri-3.

Object/File Name
/eng/teams/teambio35.htm
/eng/teams/teambio142.htm
/eng/teams/teambio141.htm
/fra/teams/teambio136.htm
/eng/teams/teambio128.htm
/eng/teams/teambio136.htm
/fra/teams/teambio146.htm
/eng/competition/matchstat8890.htm
/eng/competition/groupstandings163_76.htm
/eng/competition/groupstandings163_75.htm
/eng/competition/matchstat8893.htm
/eng/competition/**matchprog8893.htm**
/fra/competition/matchstat8893.htm
/eng/competition/matchstat8891.htm
/eng/competition/**matchprog8891.htm**
/fra/competition/**matchprog8891.htm**
/fra/competition/matchstat8891.htm

Table 13. List of objects/files for region Fri-C.

5. Related Work

The problem of analysing evolving graphs has been studied from a range of different perspectives. Desikan et al (Desikan & Srivastava 2004*b*) have proposed a framework to classify different approaches to this problem, in terms of whether we are analysing properties of (1) the whole graph, (2) specific subgraphs or (3) individual nodes. Whole graph analysis examines changes to global properties, like the diameter of the graph. Subgraph analysis investigates graph dynamics at the resolution of subgraphs. Single node analysis involves analysing the changes in node information. We consider our work to be in the subgraph analysis category. Although Desikan et al have not proposed any algorithm to perform subgraph analysis, recently they have used evolving graphs as a model

1 Object/File Name
1 /eng/teams/teambio76.htm
/eng/teams/teambio111.htm
/eng/teams/teambio22.htm
/eng/competition/matchstat8896.htm
/eng/competition/groupstandings163_77.htm
/eng/competition/matchstat8895.htm
/eng/competition/**matchprog8895.htm**
1

Table 14. List of objects/files for region Sat-1.

in computing an incremental page rank algorithm and in spam email detection (Desikan et al. 2005)(Desikan & Srivastava 2004*a*). In this section, we outline related work, using the framework of Desikan et al as a guide. We also summarise spatio-temporal approaches to clustering and pattern mining, and contrast work in data stream clustering with our own.

In terms of whole graph analysis, Leskovec et al (Leskovec et al. 2005) recently investigated how global properties of graphs, like node degree and the diameter of a graph, evolve with time. Using four large, real dynamic graphs, they found the average out-degree and number of edges to follow a power law distribution, as well as the diameter of graphs decreasing with time. Based on these observations, they proposed two probabilistic graph models to explain and generate the observed distributions. Using Desikan's framework, this can be regarded as global analysis, thus having a different focus to our work.

Similar to our work, Gaetler and Patrignani (Gaertler & Patrignani 2004) focused on summarising the temporal evolution of the BGP connectivity graph. They used spectral graph clustering to reduce the size of the BGP graph they analysed. However, they did not make simultaneous use of the topological and temporal information of the evolving BGP graph.

In terms of subgraph analysis, Kraetzl et al (Shoubridge et al. 2002) surveyed various techniques to detect abnormal changes, including two techniques for identifying the regions of greatest change between two snapshots. One technique was to construct and permute a change matrix. The other technique was to construct k-neighbourhoods for each vertex, and compare the neighbourhood graphs using a graph distance measure.

Recently, Borgwardt et al (Borgwardt et al. 2006) defined the novel problem of finding frequent subgraphs in dynamic graphs. In addition to the traditional definition of being topologically frequent, these subgraphs must also exhibit similar temporal evolution synchronously (over the same period of time), and asynchronously (can be over different periods of time). Although similar to our work, the frequent subgraphs sought by Borgwardt do not have any topological/spatial constraints, whilst in our work, we require changed edges to be topologically close.

Related to finding dynamic frequent subgraphs is mining minimal contrast subgraphs (Ting & Bailey 2006). Minimal contrast subgraphs are subgraphs that appear in one class of graphs, but not in another set, and are minimal, i.e., there are no proper subgraphs that are also contrast subgraphs. It can be applied to mining dynamic graphs if we consider a snapshot as one class, and the next snapshot as the other class. When applied in this context, the discovered set of minimal contrast subgraphs are the smallest subgraphs that explain the changes between the snapshots. In contrast to regions of correlated change, the edges and vertices in a minimal contrast subgraph do not have to be temporally correlated, or even topologically near each other, hence mining contrast subgraphs cannot be used as a solution to our problem.

In (Kumar, Novak, Raghavan & Tomkins 2003), Kumar et al explored the evolution of community structure and behaviour in several collections of weblogs. They introduced the notion of a time graph to model the evolution of a collection of weblogs and the links between them. Edges in the time graph are labelled with their creation time. Using these time graphs, they extracted the weblog communities and analysed the degree distribution, evolution of node distributions in communities, and burstiness of communities. In (Kumar, Novak & Tomkins 2006), they performed a similar analysis on the structure and evolution

of two online social networks, namely Flickr and Yahoo! 360. Similar to Leskovec et al (Leskovec et al. 2005), their focus is on how local structures evolve with time, rather than finding correlation in changes.

In (Ali, Mokbel, Aref & Kamel 2005), Ali et al introduced the idea of phenomena detection and tracking (PDT) in sensor networks. PDT involves detecting sensors that are geographically near to each other and have similar readings over a certain time period. The motivating example was to track oil spills from a sensor network deployed at sea. Although similar in aim to our work, the PDT algorithm involves optimising SQL queries in sensor network databases, and only uses single linkage to define a spatially close neighbourhood.

Lauw et al (Lauw, Lim, Tan & Pang 2005) used spatio-temporal co-occurring patterns to construct affiliation networks. The idea is that people or entities who are co-located frequently are mostly likely to be affiliated to each other. In an affiliation network, an affiliation is represented as an edge between two vertices (representing people or entities). Like spatio-temporal mining, which we survey next, our work uses graphical data, as opposed to time-stamped locations of different entities. In addition, our work seeks temporal correlation over longer periods than Lauw et al.

In spatio-temporal clustering and pattern mining, the main objective is to find objects or incidents that are in close spatial (usually geographical) proximity, and either occur at the same time (clustering), or occur frequently together (pattern mining). An example of spatio-temporal pattern mining is given in (Celik et al. 2006), where Celik et al define the problem of mining mixed-drove spatio-temporal co-occurrence patterns. These patterns are objects in spatio-temporal databases that are spatially near each other for many time instances. In spatio-temporal clustering (Hoebe, de Melker, Spanjaard, Dankert & Nlkerke 2004)(Neill, Moore, Sabhnani & Daniel 2005), clusters of disease, crime and other incidents of interest are grouped according to their geographical proximity and their temporal occurrence. Although both types of spatio-temporal mining are similar to our work, we are concerned with spatial distances that are based on topological distance in graphs, while in spatio-temporal mining, the spatial distance is geographically based. Furthermore, in spatio-temporal clustering, the focus is to seek dense groups of points, whereas density is not defined in relational data (i.e., graphs) - the closest definitions of density are vertex degree, and the number of triangles among triplets of vertices. Finally, spatio-temporal mining is concerned with clusters of incidents or objects that co-occur or are more frequent than normal, while our work is concerned with groups of entities that report strong temporal correlation over a window of time.

In the related area of data stream clustering (Aggarwal, Han, Wang & Yu 2003)(Zhou, Cao, Qian & Jin 2007), the aim is to cluster objects/points that arrive continuously. The emphasis of data stream clustering is to perform the clustering online, while keeping memory consumption within reasonable limits. Aggregate statistics of the objects are updated online, and clusters can be produced anytime from these aggregates. This is different from the region discovery problem, as there is no identifying how any particular object evolve with time, nor tracking of how the relationships between individual objects change with time. Hence, current stream clustering algorithms (Aggarwal et al. 2003)(Zhou et al. 2007) cannot be used to solve the region discovery problem.

6. Conclusion

In this paper, we have defined a novel problem in data mining, namely, how to find regions of correlated spatio-temporal change in evolving graphs. The problem of finding correlated regions of change arises in a variety of contexts, such as fault diagnosis and event discovery in computer networks. To address this problem, we have developed an algorithm called cSTAG, which can find clusters that simultaneously maximise both the temporal and spatial similarity of the regions of change. Using a quantitative analysis on a simulated dataset, we showed that cSTAG can accurately characterise a variety of different patterns of spatio-temporal change in networks. Furthermore, we have applied cSTAG to finding patterns in the Internet connectivity graph from the Border Gateway Protocol, and shown that it was able to identify reported failures and simultaneous events during the 2005 Hurricane Katrina disaster in an accurate and scalable manner. In addition, we have grouped different sets of accesses to the 1998 World Cup website based on the groups of files requested frequently together, and we were also able to corroborate the discovered regions with known flash crowd events.

Our research has stimulated a number of promising directions for future research. So far, we have used an incremental approach to merge regions of change from different time windows. A future challenge is to develop a more global approach to simultaneously combine related regions from consecutive time windows. In addition, one of the important post-processing steps in clustering is to use an measure to rank the obtained clusters. We currently use the size of regions as an indication of their interest to users, but there are other potential measures that could be extended and modified upon. For example, Bar-yossef et al (Bar-Yossef, Guy, Lempel, Maarek & Soroka 2007) recently proposed a cohesion based measure that ranks how good partitions of a graph are. Finally, there are many other types of application domains where cSTAG can be applied. We are currently investigating how cSTAG can be used to detect spatio-temporal regions of correlated activity in Magnetic Resonance Imaging snapshots of brain activity during different types of cognitive activity.

Acknowledgements.

We would like to thank Nikos Mamoulis for inviting us to submit this journal paper as an extended version of our SSTDM workshop paper, and the anonymous reviewers for their insightful and helpful comments. In addition, we would like to thank National ICT Australia for their support and funding of this research, Miro Kraetzl and his colleagues from DSTO for generously sharing their research with us, and the University of Oregon for making the RouteView data publicly available. Finally, we would like to thank Martin Arlitt and Tai Jin of Hewlett-Packard Laboratories for making the Workload characteristics to the World Cup website publicly available, and Lawrence Berkeley National Laboratory for making the traces available.

References

Aggarwal, C. C., Han, J., Wang, J. & Yu, P. S. (2003), A framework for clustering evolving data streams, *in* 'Proceedings of the 29th International Conference on Very Large Data Bases', pp. 81–92.

Ahuja, R., Magnanti, T. & Orlin, J. (1993), *Network Flows : Theory, Algorithms, and Applications*, Prentice Hall.

Ali, M. H., Mokbel, M. F., Aref, W. G. & Kamel, I. (2005), Detection and tracking of discrete phenomena in sensor-network databases, *in* 'Proceedings of the 17th International Conference on Scientific and Statistical Database Management', pp. 163–172.

An, Y., Janssen, J. & Milios, E. E. (2004), 'Characterizing and mining the citation graph of the computer science literature', *Knowledge and Information Systems* **6**, 664–678.

Ari (n.d.), 'American Registry for Internet Numbers'. `http://www.arin.net`

Arlitt, M. & Jin, T. (1999), Workload characterization of the 1998 World Cup website, Technical Report HPL-99-35R1, Hewlett-Packard Labs.

Bar-Yossef, Z., Guy, I., Lempel, R., Maarek, Y. S. & Soroka, V. (2007), 'Cluster ranking with an application to mining mailbox networks', *Knowledge and Information Systems* .

Barabasi, A.-L. & Albert, R. (1999), 'Emergence of scaling in random networks', *Science* **286**, 500–512.

Borgwardt, K. M., Kriegel, H.-P. & Wackersreuther, P. (2006), Pattern mining in frequent dynamic subgraphs, *in* 'Proceedings of the 6th International Conference on Data Mining', pp. 818–822.

Celik, M., Shekhar, S., Rogers, J. P., Shine, J. A. & Yoo, J. S. (2006), Mixed-drove spatio-temporal co-occurance pattern mining: A summary of results, *in* 'Proceedings of the 6th International Conference on Data Mining', pp. 119–128.

Chan, J., Bailey, J. & Leckie, C. (2006), Discovering and summarising regions of correlated spatio-temporal change in evolving graphs, *in* '1st Workshop on Spatial and Spatio-temporal Data Mining', pp. 361–365.

Chen, C. (2005), The centrality of pivotal points in the evolution of scientific networks, *in* 'Proceedings of the 10th International Conference on Intelligent User Interfaces', pp. 98–105.

Cheng, Y. & Church, G. M. (2000), Biclustering of expression data, *in* 'Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology', pp. 93–103.

Cook, D. & Holder, L. (1994), Substructure discovery using minimum description length and background knowledge, *in* 'AAAI-94: The 12th National Conference on Artificial Intelligence', Vol. 2, p. 1442.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R. & Schrijver, A. (1998), *Combinatorial Optimization*, Wiley-Interscience.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001), *Introduction to Algorithms*, MIT Press.

Cowie, J., Popescu, A. & Underwood, T. (2005), Impact of Hurricane Katrina on Internet infrastructure, Technical report, Renesys Corporation. `http://www.renesys.com/resource_library/Renesys-Katrina-Report-9sep2005.pdf`

Demetrescu, C. & Italiano, G. F. (2004), 'A new approach to dynamic all pairs shortest paths', *Journal of the ACM* **51**(6), 968–992.

Demetrescu, C. & Italiano, G. F. (2006), 'Experimental analysis of dynamic all pairs shortest path algorithms', *ACM Transactions on Algorithms* **2**(4), 578–601.

Desikan, P., Pathak, N., Srivastava, J. & Kumar, V. (2005), Incremental pagerank computation on evolving graphs, *in* 'Proceedings of 14th International Conference on World Wide Web', pp. 1094–1095.

Desikan, P. & Srivastava, J. (2004*a*), Analyzing network traffic to detect e-mail spamming machines, *in* 'ICDM Workshop on Privacy and Security Aspects of Data Mining'.

Desikan, P. & Srivastava, J. (2004*b*), Mining temporally evolving graphs, *in* 'KDD Workshop on Web Mining and Web Usage Analysis', Seattle.

Dhillon, I. S. (2001), Co-clustering documents and words using bipartite spectral graph partitioning, *in* 'Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 269–274.

Duda, R. O., Hart, P. E. & Stork, D. G. (2000), *Pattern Classification*, Wiley-Interscience.

Feamster, N., Balakrishnan, H. & Rexford, J. (2004), Some foundational problems in interdomain routing, *in* '3rd ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets)'.

Frigioni, D., Marchetti-Spaccamela, A. & Nanni, U. (1996), Fully dynamic output bounded single source shortest path problem, *in* 'Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms', pp. 212–221.

Gaertler, M. & Patrignani, M. (2004), Dynamic analysis of the autonomous system graph, *in* '2nd International Workshop on Inter-Domain Performance and Simulation', pp. 13–24.

Girvan, M. & Newman, M. E. (2002), Community structure in social and biological networks, *in* 'Proceedings of the National Academy of Science', Vol. 99, pp. 7821–7826.

Halabi, S. & McPherson, D. (2001), *Internet Routing Architectures*, 2 edn, Cisco Press, USA.

Halkidi, M., Batisakis, Y. & Vazirgiannis, M. (2001), 'On clustering validation techniques', *Journal of Intelligent Information Systems* **17**(2–3), 107–145.

Hoebe, C. J., de Melker, H., Spanjaard, L., Dankert, J. & Nlkerke, N. (2004), 'Space-time cluster analysis of invasive meningococcal disease', *Emerging Infectious Diseases* **10**(9).

Jain, A. K. & Dubes, R. C. (1998), *Algorithms for Clustering Data*, Prentice-Hall, Inc.

Kaindl, H. & Kainz, G. (1997), 'Bidirectional heuristic search reconsidered', *Journal of Artificial Intelligence Research* **7**, 283–317.

Kandula, S., Katabi, D. & Vasseur, J.-P. (2005), Shrink: A tool for failure diagnosis in IP networks, *in* 'ACM SIGCOMM Workshop on Mining Network Data (MineNet-05)', pp. 173–178.

Kawaji, H., Yamaguchi, Y., Matsuda, H. & Hashimoto, A. (2001), 'A graph-based clustering method for a large set of sequences using a graph partitioning algorithm', *Genome Informatics* **12**, 93–102.

Keogh, E. & Pazzani, M. (2001), Derivative dynamic time warping, *in* 'Proceedings of 1st SIAM International Conference on Data Mining'.

King, V. (1999), Fully dynamic algorithms for maintaining all-pairs shorest path and transitive closure in digraphs, *in* 'Proceedings of the 40th IEEE Symposium on Foundations of Computer Science', pp. 81–99.

Kleinberg, J. M. (1998), Authoritative sources in a hyperlinked environment, *in* 'Proceedings of the ACM-SIAM Symposium on Discrete Algorithms', pp. 668–677.

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S. & Tomkins, A. S. (1999), 'The Web as a graph: Measurements, models and methods', *Lecture Notes in Computer Science* **1627**, 1–17.

Kumar, R., Novak, J., Raghavan, P. & Tomkins, A. S. (2003), On the bursty evolution of blogspace, *in* 'Proceedings of the 12th International Conference on World Wide Web', pp. 568–576.

Kumar, R., Novak, J. & Tomkins, A. S. (2006), Structure and evolution of online social networks, *in* 'Proceedings of the 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (poster)'.

Lauw, H. W., Lim, E.-P., Tan, T.-T. & Pang, H.-H. (2005), Mining social networks from spatio-temporal events, *in* 'Workshop on Link Analysis, Couterterrorism and Security'.

Lee, G. J. & Poole, L. (2006), Diagnosis of TCP overlay connection failures using bayesian networks, *in* 'ACM SIGCOMM Workshop on Mining Network Data (MineNet-06)', pp. 305–310.

Leskovec, J., Kleinberg, J. & Faloutsos, C. (2005), Graphs over time: Densification laws, shrinking diameters and possible explanations, *in* 'Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining', pp. 177–187.

Neill, D. B., Moore, A. W., Sabhnani, M. & Daniel, K. (2005), Detection of emerging space-time clusters, *in* 'Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 218–227.

Newman, M. E. J. (2003), 'The structure and function of complex networks', *SIAM Review* **45**.

of Oregan, U. (n.d.), 'Route views project'. `http://www.routeviews.org`

Ramalingam, G. & Reps, T. (1996), 'An incremental algorithm for a generalisation of the shortest-path problem', *Journal of Algorithms* **21**, 267–305.

Rattigan, M. J., Majer, M. & Jensen, D. (2006), Using structure indices for efficient approximation of network properties, *in* 'Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 357–366.

Salvador, S. & Chan, P. (2004), Fastdtw: Toward accurate dynamic time warping in linear time and space, *in* 'KDD Workshop on Mining Temporal and Sequential Data'.

Shoubridge, P. J., Kraetzl, M., Wallis, W. D. & Bunke, H. (2002), 'Detection of abnormal change in a time series of graphs', *Journal of Interconnection Networks* **3**(1-2), 85–101.

Steinder, M. & Sethi, A. S. (2001), The present and future of event correlation: A need for end-to-end service fault localization, *in* 'Proceedings of World Multi-Conference on Systemics, Cybernetics, and Informatics', pp. 124–129.

Steinder, M. & Sethi, A. S. (2004), 'Probabilistic fault localization in communication systems using belief networks', *IEEE/ACM Transactions on Networking* **12**(5), 809–822.

Tang, Y., Al-Shaer, E. & Boutaba, R. (2005), Active integrated fault localization in communication networks, *in* 'Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005', pp. 543–556.

Ting, R. & Bailey, J. (2006), Mining minimal contrast subgraph patterns, *in* 'Proceedings of SIAM International Conference on Data Mining', pp. 639–643.

Tung, A. K. H., Ng, R. T., Lakshmanan, L. V. S. & Han, J. (2001), Constraint-based clus-

tering in large databases, *in* 'Proceedings of the 8th International Conference on Database Theory', pp. 405–419.
Vlachos, M., Kollios, G. & Gunopulos, D. (2002), Discovering similar multidimensional trajectories, *in* 'Proceedings of the 18th International Conference on Data Engineering', p. 673.
Wagstaff, K. & Cardie, C. (2000), Clustering with instance-level constraints, *in* 'Proceedings of the 17th International Conference on Machine Learning', pp. 1103–1110.
Washio, T. & Motoda, H. (2003), 'State of the art of graph-based data mining', *ACM SIGKDD Explorations Newsletter* **5**(1), 59–68.
wc9 (n.d.), '1998 World Cup website access traces'. `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`
Wu, A. Y., Garland, M. & Han, J. (2004), Mining scale-free networks using geodesic clustering, *in* 'Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 719–724.
Zhao, Q., Liu, T.-Y., Bhowmick, S. S. & ng Ma, W.-Y. (2006), Event detection from evolution of click-through data, *in* 'Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 484–493.
Zhou, A., Cao, F., Qian, W. & Jin, C. (2007), 'Tracking clusters in evolving data streams over sliding windows', *Knowledge and Information Systems* .

Author Biographies

Jeffrey Chan is currently a Ph.D candidate at the Department of Computer Science and Software Engineering, University of Melbourne, Australia. He received the B.Sc (Hons) degree and B.E. degree in electrical engineering (with first class honours) in 2005, both from the University of Melbourne. His research interests include graph mining, clustering analysis and validation, shortest path algorithms, and network intrusion detection.

James Bailey is currently a Senior Lecturer at the University of Melbourne and has made significant research contributions in database systems and data mining. In 2005 he won the best paper award at the IEEE ICDM Conference. He has published significant work on the fast mining of contrast patterns for high dimensional relational, sequence and graph data. He has also recently published work in the area of comparison and differentiation of clusterings. These results have appeared at leading conferences such as IEEE ICDM and ACM KDD. He was co-Program Chair for the Australasian Database Conference in 2006 and 2007 and is on the program committee for a number of top conferences, including ICDM and VLDB.

Christopher Leckie received the B.Sc. degree in 1985, the B.E. degree in electrical and computer systems engineering (with first class honours) in 1987, and the Ph.D. degree in computer Science in 1992, all from Monash University, Australia.

He joined Telstra Research Laboratories in 1988, where he conducted research and development into artificial intelligence techniques for various telecommunication applications. In 2000, he joined the University of Melbourne, Australia, where he is currently a Senior Lecturer with the Department of Computer Science and Software Engineering. His research interests include using artificial intelligence for network management and intrusion detection.