Using Graph Partitioning to Discover Regions of Correlated Spatio-Temporal Change in Evolving Graphs

Jeffrey Chan*, James Bailey, Christopher Leckie

NICTA Victoria Research Laboratory

Department of Computer Science and Software Engineering

University of Melbourne, Victoria 3010, Australia

{jkcchan, jbailey, caleckie}@csse.unimelb.edu.au

Abstract

There is growing interest in studying dynamic graphs, or graphs that evolve with time. In this work, we investigate a new type of dynamic graph analysis - finding regions of a graph that are evolving in a similar manner and are topologically similar over a period of time. For example, these regions can be used to group a set of changes having a common cause in event detection and fault diagnosis. Prior work [6] has proposed a greedy framework called cSTAG to find these regions. It was accurate in datasets where the regions are temporally and spatially well separated. However, in cases where the regions are not well separated, cSTAG produces incorrect groupings.

*Corresponding author. Department of Computer Science and Software Engineering, University of Melbourne, Victoria 3010, Australia. Telephone Number: (613) 8344 1267. Fax Number: (613) 9348 1184.

In this paper, we propose a new algorithm called regHunter. It treats the region discovery problem as a multi-objective optimisation problem, and it uses a multi-level graph partitioning algorithm to discover the regions of correlated change. In addition, we propose an external clustering validation technique, and use several existing internal measures to evaluate the accuracy of regHunter. Using synthetic datasets, we found regHunter is significantly more accurate than cSTAG in dynamic graphs that have regions with small separation. Using two real datasets - the access graph of the 1998 World Cup website, and the BGP connectivity graph during the landfall of Hurricane Katrina - we found regHunter obtained more accurate results than cSTAG. Furthermore, regHunter was able to discover two interesting regions for the World Cup access graph that cSTAG was not able to find.

Keywords: Dynamic graphs, graph mining, regions of correlated spatio-temporal change, graph partitioning, BGP connectivity graph, Web access graph.

1 Introduction

Graphs are powerful abstractions of relational data, hence their popularity across many fields [28][25]. In recent years, there has been growing interest in analysing how relational data, represented as graphs, evolve with time. Some examples include analysing how global properties of graphs change with time [21], detecting anomalous changes in an evolving graph [25], and analysing community evolution in social networks [19].

In previous work [6], we introduced the novel and interesting problem of discovering regions of correlated spatio-temporal change in dynamic graphs. This region discovery problem involves grouping similar sequences of changes that occur over the same period of time (temporally correlated), as well as within the same region of the graph (spatially correlated). The subgraphs that are both temporally and spatially correlated over a period of time are referred to as **regions of correlated spatio-temporal change**. The region discovery problem is to find these regions of correlated change in order to summarise changes to dynamic graphs.

Changes to graphs can occur in various forms, 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. In this paper, we focus on structural changes to edges[1], but the methodology can be easily extended to weight changes.

The problem of finding regions of correlated spatio-temporal change arises in a variety of contexts, including fault diagnosis in communications networks [26], identifying flash crowd events from web access graphs, and identifying active brain regions from functional Magnetic Resonance Imaging [7]. As a fault diagnosis example, we might be interested in detecting correlated connection failures at the IP layer to infer an underlying router failure. We wish

[1]We do not need to consider changes to vertices as well, since changes to vertices will induce changes on their incident edges.

to group all IP connection failures that have the same root cause (i.e., router failure), to help narrow down the possible root causes. A router failure will induce correlated failures on all of its dependent IP connections. This will induce a region of correlated change in the routing topology at the IP layer. Hence by finding such regions in the evolving network of the IP layer, we can assist with the discovery of problems in the underlying routers.

Illustrative Example To further illustrate the concept of regions of correlated spatio-temporal change, consider an example of five sequential snapshots from an evolving graph, shown in Figure 1. 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 edges in region A - all the distances are very small. This is 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.

We presented a greedy algorithm to discover these regions of correlated change in [6]. This algorithm, called cSTAG, first found regions that are optimally correlated over a (set length) subsequence of snapshots of the dynamic graph. Then it merges similar regions that were discovered in adjacent subsequences. cSTAG can achieve high accuracy when the regions are temporally or spatially well separated from each other. However, this is not the case when the regions are not well separated, or when they have characteristics that confuse a greedy approach. Consider the example of a two layered dynamic graph with 17 snapshots, illustrated in Figure 2, with the associated change waveforms presented in Table 2. Figure 2a represents the underlying physical link graph, while Figure 2b represents the IP connection graph built over it. As explained earlier, we are trying to infer underlying router instability events at the lower physical layer from changes in the upper layer. The edges of subgraph

A (solid dark edges), represent IP connections that transit through router A. Similarly for edges in subgraph B (dashed dark edges) for router B, and subgraph C (dotted dark edges) for router C. The gray edges in Figure 2 have not experienced any change over the sequence of snapshots, hence we are not interested in grouping them into regions of correlated change.

In this example, router A experiences two independent periods of instability, cause by two different instability events (see Table 2). Router B is an unstable, flapping router, forming one instability event. Router C just experiences one period of instability, caused by an event starting from snapshot 11. Each of these events at the physical layer causes one group of correlated changes in the IP connection layer. Hence, we wish to group the changing edges at the IP connection layer into four regions of correlated change (one per underlying event) - corresponding to the first failure of router A (region A1), the second failure of A (region A2), the flapping behaviour of router B (region B), and to the changes of router C (region C).

If analysing correlation over a short period of time (e.g., window length ω_1 in Table 2), regions A1 and B, and regions A2, B and C, will be temporally and spatially similar over snapshots 1–7, and 13–17. If analysing correlation over a longer period of time (e.g., window length ω_2 in Table 2), regions A1 and A2 will appear to be one event. Therefore, using a greedy algorithm such as cSTAG, with both limited look ahead and a fixed window length, will result in either regions A1 and B incorrectly associated over the snapshots 1-7, and regions A2, C and B over the snapshots 11-17 (if a short window length ω_1 is used), or regions A1 and A2 being considered as one region (if a longer window length ω_1 is used). *This example demonstrates that a more global approach is needed.*

Therefore, to handle datasets like the previous example, we have developed a new algorithm called regHunter. regHunter can lookahead an arbitrary number of snapshots to

find the regions, hence it is able to distinguish regions that are very close together, as well as regions that exhibit correlation over time windows of different lengths. It represents the evolving graph as two evolving dissimilarity graphs, and discovering regions of correlated change is equivalent to finding the optimal partitions of the dissimilarity graphs. Although regHunter can potentially be more computationally intensive than cSTAG, this is not an issue for off-line analysis applications, where accuracy is of paramount importance. For example, in an application such as fault diagnosis, the ability to accurately identify the root cause of a problem can greatly assist and save time for network administrators [13][26].

Contributions:

- We introduce a new model to represent evolving graphs, that is particularly suitable for finding regions of correlated change.
- We propose a new algorithm, called regHunter, that employs a graph partitioning approach to discovering regions of correlated spatio-temporal change. We have found that regHunter is substantially more accurate than the existing method (cSTAG [6]) for regions that are temporally and spatially close to each other. In addition, regHunter requires fewer parameters to be set than cSTAG (5 vs. 8), two of which are virtually invariant to a change of dataset, making regHunter simpler to use.
- We thoroughly analyse the worst case time complexity of regHunter, finding it to be polynomial in the number of changed edges.
- We introduce an external validation method to evaluate the accuracy of regHunter and cSTAG.

- We provide a quantitative framework to evaluate two real datasets, that were only qualitatively evaluated in previous work [6].

The rest of the paper is structured as follows. Section 2 surveys related work. In Section 3, we formally introduce the region discovery problem. Section 4 describes our new algorithm regHunter, including a description of the model regHunter adopts to solve the region discovery problem. In Section 5, we evaluate the accuracy and running time of regHunter on synthetic and real datasets, and compare it with cSTAG. Finally, in Section 6 we conclude and present possible directions for future work.

2 Related Work

We first summarise the cSTAG approach, and then outline relevant work in the related problems of dynamic graph analysis and spatio-temporal mining.

The cSTAG framework [6] converts the problem of finding regions of correlated region in a dynamic graph to one of clustering the set of edges that experienced change, using a temporal and topological dissimilarity measure defined over the edges. To discover regions of different correlation lengths, cSTAG partitions the initial sequence of snapshots into overlapping windows of snapshots. For each window of snapshots, (local) regions are discovered, using a variety of single-objective clustering algorithms. Then similar regions are merged across windows, to form regions whose correlation length are longer than a single window. In this greedy approach of cSTAG, if a mistake was made in the discovery of local regions, it is not possible for cSTAG to correctly identify that two regions discovered in consecutive windows actually describe a single region of longer correlation. The clustering methods used include single and average linkage hierarchical clustering, and an incremental method called

leaderFollower (see [6] for details).

The problem of analysing evolving graphs has also been studied from a number of other different perspectives.

Leskovec et al. [21] investigated how global properties of graphs, like node degree and the diameter of a graph, evolve with time. Based on an analysis of four large, real dynamic graphs, they proposed two probabilistic graph models to explain and generate the observed distributions. This example of global graph analysis is different from the more localised region discovery problem.

Recently, Borgwardt et al. [2] 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 over a period of time. Although very similar to our work, the frequent subgraphs sought by Borgwardt do not have any topological/spatial constraints, while in our work, we require changed edges to be topologically close.

In [19], 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 labeled with their creation time. Using these time graphs, they extracted weblog communities and analysed the degree distribution, the evolution of node distributions in communities, and the burstiness of communities. In [20], 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. [21], their focus is on how local structures evolve with time, rather than finding correlation among changes.

In [27], Sun et al. proposed an information theoretic based algorithm called GraphScope

to discover communities in evolving, relational, bipartite graphs. Communities are defined as groups of objects (vertices) that have the same connections among them over a period of time. Using lossless encoding schemes, groups of objects are encoded, and those with the smallest entropy are chosen as communities. The main difference between the problem of finding communities and the problem of finding regions of correlated change is that the communities problem is restricted to bipartite graphs. They do not have a notation of topological proximity. Hence, GraphScope cannot be used to find regions of correlated changes on graphs that are not bipartite.

In spatio-temporal clustering and pattern mining, the main objective is to find objects or incidents that are in close geographical proximity, and occur frequently together. An example of spatio-temporal pattern mining is given in [5], 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. Although spatio-temporal mining is similar to our work, it is concerned with clusters of incidents that co-occur frequently, while our work is concerned with groups of entities that report strong temporal correlation over a window of time, which may or may not co-occur frequently.

In summary, the related work either focuses on discovering patterns that co-occur at the same time, which is different from finding regions of correlated change, or target non-relational datasets. These problems and solutions cannot be extended to finding regions of correlated change, for the reasons outlined previously. In contrast, the cSTAG framework introduces a local approach to finding regions of correlated spatio-temporal change. For less well separated regions, cSTAG has difficulty in finding them. regHunter is designed to find these regions.

3 Problem Statement

In this section, we formally define what is a region of correlated spatio-temporal change and the problem of discovering these regions. For ease of reference, Table 3 provides a summary of the main symbols used in this paper.

A graph $G(V_G, E_G)$ consists of a set of vertices V_G, and a set of edges E_G, $E_G : V_G \times V_G$, representing the pairwise relationships over V_G.

Definition 1 *A **dynamic graph** is represented as a sequence of snapshots* $< G_1, G_2, \ldots, G_S >$ *of the graph* G, $1 \leq t \leq S$. *We denote the subsequence (or window)* $< G_{ts}, \ldots, G_{te} >$ *by* $W^{ts,te}$, $1 \leq ts \leq te \leq S$. *Let* wn *be the number of windows. Then, where it is unambiguous to do so, we denote a window by* W_k, $1 \leq k \leq wn$, *and if the first snapshot of* W_{k1} *is later than the first snapshot of* W_{k2}, *then* $k1 > k2$.

For example, the window $W^{1,2}$ for the graph illustrated in Figure 1 represents the sequence $< G_1, G_2 >$. If $wn = 4$, and the set of windows is $\{W^{1,2}, W^{2,3}, W^{3,4}, W^{4,5}\}$, then W_1 is $W^{1,2}$, W_2 is $W^{2,3}$, and so on.

Definition 2 *A **structural change** of an edge* e, $e \in \bigcup_{t=1}^{S} E_{G_t}$, *is defined as the appearance or disappearance of* e *between any two consecutive snapshots. We call an edge that has experienced structural change a **changed edge**, and* $E_C^{ts,te}$ *(*E_C^k*) the set of changed edges over subsequence* $W^{ts,te}$ *(*W_k*).*

For example, the edge e_{1-7} for the graph shown in Figure 1 is a changed edge over $W^{1,2}$. The set of changed edges for $W^{1,2}$, $E_C^{1,2}$, is $\{e_{1-7}, e_{1-6}, e_{1-2}, e_{1-3}, e_{13-14}, e_{15-16}, e_{8-9}, e_{9-10}, e_{12-13}, e_{9-11}\}$.

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

Definition 3 *For structural changes to edge e_i, over the subsequence $W^{ts,te}$, we 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*

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

As an example, the change waveforms for each changed edge in Figure 1 are shown in Table 1.

A **region of correlated spatio-temporal change** $R_r^{ts,te}$, $R_r^{ts,te} \subseteq E_C^{1,S}$, is a set of changed edges that have the following characteristics:

- Over the snapshots spanning *ts* to *te*, all edges are highly correlated in their change behaviour, as well as being topologically close.
- The temporal and spatial distances between different regions are maximised.

Intuitively, the edges within a region should have minimal temporal and spatial distances between them, while edges in different regions should have maximal temporal and spatial distances between them. Hence, a set of regions discovered for a dynamic graph should maximise the temporal inter-region/intra-region distance and the spatial inter-region/intra-region distance ratio simultaneously.

Definition 4 *The temporal relation $d_{tem} : E \times E \times W_k \rightarrow [0, 1]$ measures the difference between the temporal change behaviour of pairs of edges over the subsequence W_k. The spatial relation $d_{spa} : E \times E \times W_k \rightarrow [0, 1]$ measures the topological distance between pairs of edges over the subsequence W_k.*

As both temporal and spatial objective functions have a similar form, we shall just define the temporal one for brevity.

Definition 5 *Let* $\mathbf{R}$ *be the set of regions of correlated change. The temporal (spatial) objective function,* $f_{tem}(\mathbf{R}, d_{tem})$ *(*$f_{spa}(\mathbf{R}, d_{spa})$*) measures how temporally correlated (spatially close) the regions* $\mathbf{R}$ *are, under the temporal (spatial) distance relation* d_{tem} *(*d_{spa}*). It is formally defined as*

$$f_{tem}(\mathbf{R}, d_{tem}) = \sum_{R_r \in \mathbf{R}} \frac{tem\_inter(R_r, E_C^{1,S} - R_r)}{tem\_intra(R_r)}$$

where tem_inter *(*spa_inter*) and* tem_intra *(*spa_intra*) are the total temporal (spatial) inter-region distance and the total temporal (spatial) intra-region distance respectively.*

Definition 6 *The* ***problem of discovering regions of correlated spatio-temporal change*** *involves partitioning* $E_C^{1,S}$ *into a* ***set of regions of correlated spatio-temporal change*** $\mathbf{R} = \{R_1^{ts1,te1}, \ldots, R_L^{tsL,teL}\}$*, such that the objective functions* $f_{tem}(\mathbf{R}, d_{tem})$ *and* $f_{spa}(\mathbf{R}, d_{spa})$ *are simultaneously maximised, and* $R_r^{ts,te} \subseteq E_C^{1,S}$*,* $R_g^{tsg,teg} \cap R_h^{tsh,teh} = \emptyset$*,* $g \neq h$*,* $1 \leq g, h \leq L$*.*

The region discovery problem is a bi-objective optimisation problem. In fact, finding the optimal partition for either f_{tem} or f_{spa} is NP-Complete, as optimising either objective can be easily decomposed to the NP-Complete maximum k-cut problem [4]. Hence, we seek approximate solutions for the region discovery problem.

4 regHunter: Discovering Regions of Correlated Spatio-Temporal Change

In this section, we describe how regHunter solves the region discovery optimisation problem. We first describe how regHunter models an evolving graph as two evolving dissimilarity

graphs. Then we describe the two temporal and spatial distance measures used, and how they are used to build the model. Then we present the graph partitioning approach used to discover the optimal partitions/regions. Finally, we present the worst case time complexity of regHunter. A summary of the regHunter algorithm is provided in Algorithm 1. We shall refer back to it in the following subsections, as we explain each of the steps in regHunter.

Algorithm 1 Outline of regHunter.

Inputs:
$< G_1, G_2, \ldots, G_S >$ - sequence of snapshots;
ω - window length;
η - lookahead length;
ψ - maximum time between changes;
$stop_{tem}$ - temporal stopping threshold;
$stop_{spa}$ - spatial stopping threshold.
Outputs:
R - set of regions of correlated spatio-temporal change.

Segment the sequence into a number of overlapping subsequences (each of length ω), $W_1, W_2, \ldots W_{wn}$.
for $1 \leq k \leq wn$ **do**
    Extract E_C^k for W_k.
    Compute temporal intra-window distances $d_{intra\_tem}(e_i, e_j, W_k)$ between edges $e_i, e_j \in E_C^k$. Add distances and changed edges to the temporal evolution dissimilarity graphs G_{tem_evol}.
    for each $W_p \in$ adjacent(W_k, η) **do**
        Compute temporal inter-window distances $d_{inter\_tem}(e_i, e_j, W_p, W_k)$ between edges $e_i \in E_C^k$ and $e_j \in E_C^p$ and over windows W_p and W_k, $p < k$. Add distances and changed edges G_{tem_evol}.
    end for
end for
Apply multi-level graph partitioning to partition G_{tem_evol} into partitions **P**, using $stop_{tem}$ as a halting criterion.
for each partition $P_k \in$**P do**
    Compute the intra-window and inter-window spatial distances, $d_{intra\_spa}(e_i, e_j, W_k)$ and $d_{inter\_spa}(e_i, e_j, W_p, W_k)$, for edges e_i and $e_j \in P_k$, and construct the partial spatial evolution dissimilarity graph $G_{spa\_evol}^k$.
    Apply graph partitioning to partition $G_{spa\_evol}^k$ into a set of R_k regions, using $stop_{spa}$ as a halting criterion.
end for
R $= \bigcup R_k$

4.1 Model Formulation

In this subsection, we describe the model regHunter uses to represent evolving dissimilarity graphs. This, along with the graph partitioning algorithm, forms the regHunter approach.

A dynamic graph can be visualised as an evolving set of changed edges. Between each pair of changed edges, the temporal and spatial distances are varying with time. Conceptually, this evolving set of changed edges and distances can be represented as an evolving temporal dissimilarity graph and evolving spatial dissimilarity graph. The vertices of these evolving dissimilarity graphs are drawn from the time varying sets of changed edges, and the temporal or spatial pairwise distances are modeled as weighted edges between the associated vertices (i.e., the changed edges associated with the distance). For example, the evolving temporal and spatial dissimilarity graphs for some of the changed edges of the example given in the introduction (see Figure 1) are shown in Figure 3. The vertices are labeled as <changed edge, window id> pair - the window id identifies in which subsequence of snapshots the changed edge occurs, and are labeled in chronological order. This id can be thought of as a time label. The edge weights of the evolving temporal dissimilarity graph are the temporal distances (Figure 3a), and the edge weights of the evolving spatial dissimilarity graph are the spatial distances (Figure 3b).

Note that a region of correlated change is a set of changed edges, labeled with a starting and ending time. A partition of the vertices of the dissimilarity graphs also results in a set of changed edges, with a starting and ending time. Hence, finding a partition of the dissimilarity graphs that simultaneously maximises the inter-partition/intra-partition distance ratio is equivalent to maximising f_{spa} and f_{tem} (i.e., finding the optimal set of regions of correlated change for the dynamic graph). For example, the partitions/regions R_1, R_2, and R_3 in Figure 3 form a possible set of partitions/regions.

In order to find the set of regions, we need to clarify what it means to simultaneously maximise the two objectives. There are many maximisation definitions in multi-objective optimisation [12]. We use lexicographical optimisation [12] to optimise the two objective functions. In lexicographical optimisation the objective functions $f^i(S) = (f^1(S), f^2(S), \ldots, f^Q(S))$ are ordered according to their importance, in decreasing order. Each objective is optimised in turn, starting with f^1. Intuitively, assuming we are maximising and if S_x and S_{x+1} are the optimal solutions for objectives f^x and f^{x+1} respectively, then solution S_{x+1} should not decrease the objective values for the objectives f^1 to f^x.

The advantages of lexicographical optimisation are that we can reduce the total number of distance calculations (we shall describe how this is achieved in the next section), and in previous work[2] [6], it has been found to have the best accuracy and consistency out of three different multiobjective optimisation schemes. The other two are i) using a weighted linear sum to combine the two distance relations (named *soft* in [6]); and ii) thresholding one of the distance relations and using that as a constraint on the remaining distance relation (*hard*).

Compared to the spatial distances, we found that the temporal distances within a region are more consistent, while across regions it varies more. This is particularly true of graphs with small diameters - like scale-free graphs, of which Internet routing topologies are an example. We shall see in Section 5.7 that the spatial distances within and across regions show little variation for a routing topology graph. Therefore, the temporal distances are better measures for distinguishing regions, so we choose to optimise f_{tem} first, than f_{spa}.

[2]Named sequential clustering in [6].

4.2 Determining the Set of Evolving Changed Edges

Analysing the temporal and spatial correlation of the set of changed edges over the whole sequence of snapshots can miss local correlation, like regions A1 and A2 in the example of Figure 2. Therefore, we segment the sequence of snapshots into a number of fixed length subsequences, or windows, and extract the changed edges over each of the windows. This enables analysis of correlation over the length of a window. By using the evolving dissimilarity graphs, we can also analyse correlation over multiple windows. This is an important advantage over the previous work in cSTAG [6], which only searches for longer term correlation after local regions have been found for each window.

The actual implementation of regHunter uses a sliding window to segment the sequence into a number of overlapping windows of fixed length ω. For each window, we extract the set of changed edges over that window. This is described in line 11 of Algorithm 1.

4.3 Temporal Distance Measures

Before we introduce the distance measures, we shall define the concepts of intra and inter window distances. Recall that each vertex in an evolving dissimilarity graph represents a <changed edge, window id> pair. The weights of edges between vertices with the same window id represent **intra-window** temporal or spatial distances between changed edges of the same window. The weights of edges between vertices with different window ids represent **inter-window** distances between changed edges of different windows. Inter-window distances measure the distance between the two changed edges spanning multiple, consecutive, windows; i.e., it measures longer term correlation. For example, in Figure 3a, vertices (1-3,0) and (2-4,0) are two changed edges of window 0 (W_0), where edge <(1-3,0), (2-4,0)> repre-

sents their intra-window distance, and edge <(1-3,0), (2-4,1)> represents their inter-window distance over windows W_0 and W_1.

We shall now describe the intra-window temporal distance measure. The aim of a temporal distance measure is to measure the dissimilarity in temporal change behaviour over a window of snapshots. As a suitable distance definition is application dependent, we demonstrate one suitable for network fault diagnosis, first defined in [6]. Note that other formulations can easily be incorporated. Correlated network changes do not always coincide exactly in time, but usually the sequence of changes are still similar. Hence, two change waveforms should only be considered similar if they have the same shape, and the timing of the changes roughly coincides. To accommodate this similarity definition, we extended the popular edit distance measure to additionally take shape into consideration. This formulation has the advantage that it is relatively cheap to compute, and can clearly distinguish between different change waveform shapes. Refer to [6] for further details.

We first need to define the concept of a **transition sequence** $t_{seq}(q(i))$, which is used to represent the shape of a change waveform $q(i)$. In a transition sequence, the changes in a binary waveform are represented as a sequence of changes. Using this definition, two waveforms having the same shape will have the same transition sequence.

We define the intra-window temporal distance d_{intra_tem} in terms of the modified edit distance d_{med}, which is defined as:

$$d_{med}(e_i, e_j, W_k) = \begin{cases} 1, & tseq(q(i)) \neq tseq(q(j)) \\ d_{ed}(q(i), q(j)), & \text{otherwise} \end{cases}$$

where $0 \leq d_{med} \leq 1$, $q(x) = q^{ts,te}(e_x)$, and d_{ed} is the unmodified edit distance ($0 \leq d_{ed} \leq 1$).

As an example, reconsider the waveforms in Table 1. Let the waveforms of regions A, B,

and C be represented by q_A, q_B, and q_C respectively. The modified distance between q_B and q_C is 0.2, as they have the same transition sequence and $d_{ed}(q_B, q_C) = 1$, but $d_{sh\_ed}(q_A, q_B) = 1$ since the transition sequences are different.

For inter-window distances d_{inter_tem}, we wish to measure the correlation over subsequences that span multiple windows, i.e., the correlation over the union of multiple windows. We can reuse d_{intra_tem} for this purpose. More formally, $d_{inter\_tem}(e_i, e_j, W_k^{tsk,tek}, W_{k+x}^{tsx,tex}) = d_{intra\_tem}(e_i, e_j, W^{tsk,tex})$ where $x > 0$, and $e_i \in E_C^k$, $e_j \in E_C^{k+x}$. For example, consider edge <(1-3,0), (2-4,1)> in Figure 3a. It has weight 0.75 because over the four snapshots of $W_0 \cup W_1$, the edges have three snapshots in which their values are different.

A naive implementation for computing these distances is computationally expensive. If we perform a new computation for each of the intra and inter window distances, the time complexity to compute the intra-window distances is:

$$\sum_{k=1}^{wn} \binom{E_C^k}{2} \cdot \omega \leq \binom{E_C^{1,S}}{2} \cdot wn \cdot \omega = O((E_C^{1,S})^2 \cdot wn \cdot \omega)$$

For the inter-window distance, the complexity is:

$$\sum_{k=1}^{wn-\eta} \binom{E_C^k \cup E_C^{k+1} \cup \ldots \cup E_C^{k+\eta}}{2} \sum_{l=1}^{\eta} O(\omega+l) \leq \binom{E_C^{1,S}}{2} (wn-\eta) \sum_{l=1}^{\eta} O(\omega+l) = O((E_C^{1,S})^2 \cdot wn \cdot \eta(\omega+\eta))$$

One advantage of using the modified Euclidean distance is that it can be computed incrementally (in this paper, we analyse the advantage of the incremental computation, but see [6] for implementation details). Given the computed intra-window distances for window $W^{ts,te}$, we can subtract the effect of G_{ts} and add the effect of G_{te+1} to obtain the distances for window $W^{ts+1,te+1}$. We can perform a similar incremental computation for obtaining inter-window distances for $W^{ts,te+1}$. Using this incremental formation, the time complexity

for computing the intra-window distances is:

$$(1) \qquad \binom{E_C^1}{2} O(\omega) + \sum_{k=2}^{wn} \binom{E_C^k}{2} O(1) \leq \binom{E_C^1}{2} O(\omega) + (wn-1)\binom{E_C^{1,S}}{2} = O((E_C^{1,S})^2 \cdot wn)$$

and for inter-window distances it is:

$$(2) \quad \sum_{k=1}^{wn-\eta} \binom{E_C^k \cup E_C^{k+1} \cup \ldots \cup E_C^{k+\eta}}{2} \sum_{l=1}^{\eta} O(1) \leq \binom{E_C^{1,S}}{2} \cdot (wn-\eta) \cdot O(\eta) = O((E_C^{1,S})^2 \cdot wn \cdot \eta)$$

In the intra-window case, there is a saving of a factor of O(ω), and O($\omega + \eta$) for the inter-window case. In our evaluation, we found the incremental implementation can reduce running time by 50% to 300%.

4.3.1 Event Detection

An important application of finding regions of correlated change is to detect and localise the underlying root causes of a set of changes. Events usually occur in bursts - i.e., there are significant periods of activity, separated by periods of stability, or quiet periods [3][14]. To detect these types of events, we need to modify our definition of a region of correlated change as follows,

Definition 7 *A region $R_h^{ts,te}$ will additionally have the constraint that*

- *The time between changes of each edge in a region $R_h^{ts,te} \in \mathbf{R}$ must be $\leq \psi$,*

where ψ is the maximum number of snapshots between changes.

From this definition, if two edges have the same change behaviour and waveform, but the time between any two changes is greater than ψ snapshots, then the two edges are treated as two separate regions. Intuitively, groups of correlated edges with consecutive changes that

are separated by more than ψ snapshots are considered to be separated by a quiet period, and therefore represent two events. Note that this additional definition is optional, but we found it improved accuracy in event detection applications.

4.4 Spatial Distance Measures

The spatial distance measure we use is the shortest path distance between a given pair of edges. For many types of graphs, this is a natural measure of topological proximity. This is especially true for event detection and fault diagnosis, where changed edges that are close in terms of shortest path distance are more likely to be caused by the same event or fault.

The intra-window spatial distance (d_{intra_spa}) between two changed edges is formulated as the average distance over all snapshots in the window where both edges exist. If two edges never exist in the same snapshot over the window, we consider them to be far apart, so we assign a distance of 1 between them. More formally:

$$d_{intra\_spa}(e_i, e_j, W^l) = \begin{cases} \frac{1}{C} \sum_{k=ts}^{|W^l|} d_{spa}(e_i, e_j, G_k), & C > 0 \\ 1, & C = 0 \end{cases}$$

where $C = |\{(e_i, e_j) | e_i \in E_{G_k}, e_j \in E_{G_k}, G_k \in W^l\}|$ (i.e., the number of snapshots in which both edges exist) and $d_{spa}(e_i, e_j, G_k)$ is the shortest path distance (normalised by the graph diameter) between edges e_i and e_j in snapshot G_k. As an example, consider the edge <(2-4,0), (13-14,0)> in Figure 3b, representing $d_{intra\_spa}((2-4,0),(13-14,0),W_0)$. Over the three snapshots of W_0, only in G_1 do both edges exist, hence $C = 1$, $d_{spa}((2-4,0),(13-14,0),G_1) = 0.4$, and $d_{intra\_spa}((2-4,0),(13-14,0),W_0) = 0.4$.

The start and end times of regions are usually accompanied by large changes in spatial

and/or temporal inter-window distances. Hence, we wish to define the spatial inter-window distance d_{inter_spa} such that it has large changes at the boundaries of regions. We cannot exactly reuse the union idea of d_{inter_tem}, because if $d_{inter\_spa}(e_i, e_j, W_k^{tsk,tek}, W_{k+x}^{tex,tex})$ is defined as $d_{intra\_spa}(e_i, e_j, W^{tsk,tex})$, then over smoothing from averaging will occur as the length of $W^{tsk,tex}$ increases. But we can reduce the effect of over-smoothing and still detect large changes in the spatial distances by defining $d_{inter\_spa}(e_i, e_j, W_k^{tsk,tek}, W_{k+x}^{tex,tex})$ as follows:

$$(3) \qquad d_{inter\_spa}(e_i, e_j, W_k, W_{k+x}) = 0.5(d_{intra\_spa}(e_i, e_j, W_k) + d_{intra\_spa}(e_i, e_j, W_{k+x}))$$

The advantages of this approach are that it does not over-smooth, it is cheap to compute, particularly when most of the intra-window spatial distances would have been computed already, and it can still detect changes in spatial proximity between different regions.

4.5 Construction of Evolving Dissimilarity Graphs

In this subsection, we outline the construction of the dissimilarity graphs. We first construct the evolving temporal dissimilarity graph for all changed edges (lines 13-17 in Algorithm 1). Then for each subgraph resulting from the partitioning of the temporal dissimilarity graph, we construct an evolving spatial dissimilarity graph, which can then be further partitioned (lines 20–23). An important benefit of computing the spatial dissimilarity graph after the partition of the temporal dissimilarity graph is a reduction in the computational complexity of spatial distance calculation. This is because only the spatial distances between edges with similar temporal change behaviour are computed. This reduces computation time as most of the spatial distances for edges with similar temporal behaviour will form a region, and

hence is likely to have small spatial distances between its member edges. Constructing all the inter-window distances can be expensive, and is mostly unnecessary, as almost all regions do not span the whole sequence of snapshots. Hence, we introduce a maximum lookahead bound, η, which limits the amount of lookahead to η snapshots. There is a tradeoff between accuracy against speed. We shall explore this further in Section 5.4.1, but generally the η values that produced the most accurate regions are dependent on the window size.

4.6 Multi-level Multi-objective Graph Partitioning

We now describe how region discovery is achieved in regHunter by using spectral graph partitioning on the evolving dissimilarity graphs. One of the key challenges in this graph partitioning problem is how to manage its complexity on larger graphs. We describe how we address this challenge by using a form of multi-level graph coarsening and refining to increase the efficiency of partitioning without significant loss in accuracy.

4.6.1 Spectral Graph Partitioning

Spectral graph partitioning approaches have been successfully employed for circuit layout and graph partitioning [24][11], particularly for problems that require a more global approach. In spectral partitioning, the optimising properties of the eigendecomposition of graphs are used to find an optimal partition. There have been many different objective criteria proposed for spectral partitioning. One such criterion is the MinMaxCut criterion [11], which tries to optimise the inter-region/intra-region ratios across all regions. This is actually the same objective formulation as f_{tem} and f_{spa}. However, like the authors of MinMaxCut [11], we found that such an objective function tends to favour evenly sized regions, which is not the optimal solution for all sets of regions. Hence, we will modify f_{tem} and f_{spa} to form an

objective function that does not favour evenly sized regions - the Normalized Cut criterion [24]. The only difference is the denominator in the ratio. Rather than just the intra-region distance, in the Normalized Cut it is intra-region + inter-region distance. Due to space considerations, we only present the reformulated f_{tem}, as f_{spa} is similarly defined:

Definition 8

$$f_{tem}(\mathbf{R}, d_{tem}) = \sum_{R_r \in \mathbf{R}} \frac{tem\_inter(R_r, E_C^{1,S}/R_r)}{tem\_intra(R_r) + tem\_inter(R_r, E_C^{1,S}/R_r)}$$

Spectral Bipartitioning Due to the fact that neither the number of regions, nor their sizes, are known beforehand, our approach is to recursively bipartition the existing partitions into two, until a stopping threshold is reached. In the rest of this section, we first describe how spectral methods are used to find the optimal bipartitions. Then we outline the recursive bipartition process, and introduce a lemma that shows the recursive process will always terminate.

Optimising f_{tem} involves the same process as optimising f_{spa}, hence we shall limit the following discussion to f_{tem}. Let R_1 be an existing partition, and R_2 and R_3 be two disjoint regions, where $R_2 \cup R_3 = R_1$, $R_2 \cap R_3 = \emptyset$. Let $\mathbf{x}$ be a indicator vector of length $|R_1|$. It indicates whether each edge e_i either belongs to partition R_2 or R_3. It is defined as a) $x_i = -1$, if $e_i \in R_2$; and b) $x_i = 1$ if $e_i \in R_3$. The partitioning is performed by solving $max_x f_{tem}(R_2, R_3)$, i.e., finding the $\mathbf{x}$ that minimises $f_{tem}(R_2, R_3)$. In this form, the problem is NP-Complete[3] [24], hence a relaxed version of the problem is solved, where $\mathbf{x}$ can take real values (denoted as $\mathbf{x}'$). The process involves transforming the objective f_{tem} into the form of Rayleigh's quotient, which can be solved using a generalised eigenvalue system [23][24].

[3]In [24], the authors partition a *similarity* graph, hence solve $min_x f_{tem}(R_2, R_3)$. However, we can easily convert our dissimilarity graph to a similarity graph by using the fact similarity = 1 - dissimilarity.

Because we require a discrete solution, $\mathbf{x}'$ needs to be discretised back to the binary $\mathbf{x}$. Let i_{split} denote the optimal split point. Then we determine whether an edge e_i belongs in R_2 or R_3 by $R_2 = \{e_i | i \leq i_{split}\}$ and $R_3 = \{e_i | i > i_{split}\}$. We tried several approaches to determine the best method for determining i_{split}, including $i_{split} = 0$, $i_{split} = avg(\mathbf{x}')$, and $i_{split} = median(sort(\mathbf{x}'))$. We found that dividing the indicator vector into a number of splitting points, then choosing the point which resulted in the best $f_{tem}(R_2, R_3)$ value, produced the most accurate results.

Recursive Bipartitioning Process The recursive bipartition process selects the best region to bipartition, performs the bipartition, then checks with a stopping criterion to see if the process should continue. Algorithm 2 outlines the recursive bipartition graph partitioning algorithm. We shall next discuss how the process chooses the next region to bipartition and how it determines when to terminate the recursive process.

Algorithm 2 Outline of recursive bipartition graph partitioning algorithm.

Inputs: - Evolution distance graph $G_{dis}(V, E)$.
Outputs: - Set of regions of correlated spatio-temporal change, $\mathbf{R}$

leaf clusters $leaf = \{\{V\}\}$
$f_{opt} = f_{tem}(leaf)$
while $f_{opt} <$ stop_threshold **do**
    Choose next region $P_{par} \in leaf$ to bipartition.
    Obtain children P_A, P_B by performing $bipartition(P_{par})$.
    Remove P_{par} from $leaf$, and insert P_A, P_B into $leaf$.
    $f_{opt} = f_{tem}(leaf)$
end while
$\mathbf{R} = leaf$.

We choose the next region to bipartition based on its average dissimilarity. That is, the next region to split P_{par} among the current leaf regions *leaf* can be defined by: $P_{par} = argmax_{P \in leaf} avg\_dissim(P)$[4]. This criterion is based on the intuition that regions with the

[4] $argmax f(P)$ returns the value of P that maximises function f

highest average dissimilarity are most likely to cause the largest reduction in the overall objective if further partitioned. We also evaluated schemes where we choose P_{par} to be the region with the highest and lowest objective function values. We found both schemes to produce less accurate regions than using the average dissimilarity.

The stopping criterion tests if the total objective value of the current set of partitions (*leaf*) exceeds a user-determined threshold. When the objective value exceeds the threshold, the partitioning algorithm stops, to prevent the regions from becoming too fragmented.

The following lemma states that with the bipartitioning process, f_{tem} (and f_{spa}) is monotonically increasing with the number of regions in $\mathbf{R}$, and hence the bipartitioning process is guaranteed to reach the finite stopping threshold and terminate.

Lemma 1 *Let* $R^{(x)} = \{R_1, \ldots, R_x\}$ *and* $R^{(x+1)} = \{R_1, \ldots, R_{y-1}, R_{y+1}, \ldots, R_x, R'_y, R_{x+1}\}$ *be the sets of regions at iterations* x *and* $x+1$*, respectively. Let* R_y *be bisectioned into disjoint partitions* R'_y *and* R_{x+1}*,* $1 \leq y \leq x$*. Then*

$$f_{tem}(R^{(x)}) \leq f_{tem}(R^{(x+1)})$$

Proof 1 *See Appendix A.*

4.6.2 Multi-level Graph Partitioning

The motivation behind our use of multi-level graph partitioning [18] is to reduce computational complexity while maintaining similar accuracy. Computational complexity can be reduced by partitioning a simplified version of the original graph. It consists of three steps: a) reducing the original graph to a much smaller version (**coarsening**); b) performing the partitioning on the coarsened/compressed graph (**partitioning**); and c) iteratively expand-

ing the discovered partitions by one level and locally optimising the expanded partitions (**refinement**). At the end of the process, the refined partitions will form a disjoint partition of the vertices in the original graph. Algorithm 3 and Figure 4 provide overviews of the multi-level partitioning process. Next, we describe the coarsening and refinement steps (the partitioning step was described in Section 4.6.1).

Algorithm 3 Multi-level algorithm for partitioning the temporal evolution graph into regions.

Input: $G^0_{t\_evol}$ - Temporal evolution graph, $T_{coarsen}$ - Threshold to indicate when to stop coarsening
Output: R^0 - Regions of correlated temporal change

// Coarsen Graph
$l = 0$;
while $|G^l_{t\_evol}| > T_{coarsen}$ **do**
    // Find maximum matching among vertices
    $M^l = neighMatch(G^l_{t\_evol})$;
    // Merge matched vertices
    $G^{l+1}_{t\_evol} = merge(G^l_{t\_evol}, M^l)$;
    $l = l + 1$;
end while

// Partition reduced graph $G^l_{t\_evol}$ into initial set of regions, R^l
$R^l = bipartition(G^l_{t\_evol})$;

// Uncoarsen - expand and refine after each expansion step
while $l > 0$ **do**
    $R^{l-1} = expand(R^l)$;
    $R^{l-1} = refine(R^{l-1}, G^l_{t\_evol})$;
    $l = l - 1$;
end while

Coarsening As a compromise between having many levels of refinement against the speed of coarsening, we used the popular method of finding the maximum matching at each level of coarsening (i.e., the number of vertices will approximately be halved after each step of coarsening). We considered two methods to find the best matching among the vertices.

The first method was to pick a random vertex, and match it with a neighbouring vertex

that is most similar (i.e., has minimum edge weight). This produced very inaccurate results, as this approach is similar to cSTAG.

The second method that we considered, and implemented in regHunter, is to view the adjacency matrix of the evolving dissimilarity graphs as a set of row[5] vectors. Each row vector r_i describes the neighbourhood of vertex i. Given two vertices i and j, we can compare their similarity by the inner product of their row vectors $< r_i \cdot r_j^T >$. Small values for the inner product between the vectors mean that the vertices are similar to the same set of vertices (i.e., changed edges), which indicates these two vertices are likely to belong to the same region of correlated spatio-temporal change. Note that although computing the inner product can be computationally expensive, many of the values in the row vectors are zero (the adjacency matrix is very sparse), hence there are only a few non-zero multiplications.

Refinement In the refinement step, the condensed graph (and its associated partitions) is progressively expanded back to its original size. After each expansion, refinement, (i.e., local optimisation), is performed.

Kernighan-Lin (KL) based refinement strategies have been found to work well when refining bipartitions [18]. However, when this is extended to k partitions, where $k > 2$, the complexity increases significantly, as calculations are needed to determine if a set of points should be swapped among k different partitions [17]. In addition, KL based strategies attempt to keep partitions balanced in size, which is not necessarily what is desired in the region discovery problem.

The second method, which is implemented in regHunter, is the *kernel k-means* refinement strategy, first implemented in [10]. We use the number of partitions at the start of the refinement process as k, the number of clusters, and use k-means to move the objects around

[5]We can use column vectors as well, as the adjacency matrix is symmetric.

until we reach a minima. The difference with *greedy search* lies in the method to determine the distance - a kernel distance is used instead of the typical Euclidean distance. By using the higher-dimensional kernel space, the kernelised k-means algorithm can find partitions that are not spherical in shape. According to Mercer's Theorem [8], any positive semi-definite matrix can be converted to a Gram matrix, which is used to represent the distances between points in the kernel space. The matrix used in spectral partitioning is non-negative semi-definite, but can be made positive semi-definite by adding a small constant to it. This does not change the objective in theory (see [9]), but does slightly reduce the optimality of the solutions in practice. In [9], Dhillon et al. showed how to construct a kernel for the kernel k-means algorithm that has the same theoretical result as spectral partitioning. In theory, this means kernel k-means can be used to find partitions rather than the spectral partitioning methods. In practice, we found that kernel k-means produces less optimal regions than spectral methods. However, it can approach the accuracy of spectral methods if the initial set of partitions given to it are reasonably optimal. The starting partitions at each level of refinement certainly fit this condition, hence we chose to use kernel k-means for refinement. For brevity, we do not describe the kernel and its equivalence with f_{tem} here. Please refer to [9] for details.

4.7 Time Complexity

The steps in regHunter that make the largest contribute to its time complexity are i) computing the temporal and spatial distances; ii) performing the multilevel graph partitioning on the temporal evolving dissimilarity graph; and iii) partitioning the spatial evolving dissimilarity graphs. Let $m_c = |E_C^{1,S}|$ be the total number of changed edges. Let m be the number of edges in the graph resulting from the union of all the snapshots.

The complexity of computing the temporal distances is $O(m_c^2 \cdot wn \cdot \eta)$ (see Equation 1 and 2). The complexity for computing the spatial distances is more complex, because it depends on the accuracy of the temporal-based partitioning. In the highly unlikely worst case, where all the changed edges are at the fringe of the graph snapshots, and each changed edge spans the whole sequence, then the complexity is $O(S \cdot m_c \cdot m)$. In practice, this rarely occurs, and we can easily place bounds on the search depth to guarantee a lower bound.

The complexity of the graph partitioning step can be broken down into three parts a) coarsening; b) spectral partition; and c) refinement. The number of vertices in the evolving graphs is $O(m_c)$.

The nearest neighbourhood coarsening step can be measured in terms of the number of comparison it takes. Let L be the most coarse level reached (the levels start from 0). First, consider the number of comparisons needed at level l, $0 \leq l \leq L$. Let $m_{c,l}$ be the number of vertices in the adjacency matrix at the level l that is being partitioned. When $m_{c,l}$ is even, then the number of comparisons required are $(m_{c,l} - 1) + (m_{c,l} - 3) + \ldots + 5 + 3$ and the total sum of the sequence is $\frac{(\frac{m_{c,l}}{2}-1)(3+(m_{c,l}-1))}{2} = \frac{m_{c,l}^2}{4} - 1$.

Let the comparison cost of the inner product be ic. Then the total cost of coarsening at level l is $O(\frac{ic \cdot m_{c,l}^2}{4} - ic)$. Hence, the total cost across $L + 1$ levels is

$$\begin{aligned}\sum_{l=0}^{L} O(\frac{ic \cdot m_{c,l}^2}{4} - ic) &= O(\frac{ic \cdot m_c^2}{4} - ic) + \ldots + O(\frac{(\frac{m_c}{2^L})^2}{4} - \frac{ic}{2^L}) \\ &= O((\frac{1 - (\frac{1}{4})^{L+1}}{4})ic \cdot m_c^2 - (2 - (\frac{1}{2})^L)ic) = O(m_c^2 \cdot ic - ic)\end{aligned}$$

When the row vectors are sparse, then $ic << m_c$, hence, the complexity of coarsening is $O(m_c^2)$. The odd case can be solved similarly and has the same worst case complexity.

The spectral partitioning step can be performed using an iterative eigensolver like the

Lanczos technique [15]. Because of the sparsity of the adjacency matrix, the complexity of partitioning is approximately $O(m_{c,L}^{1.5})$.

The kernel k-means refinement step requires the computation of the kernel matrix and the iterative refinement of the k-means algorithm. The computation of the kernel matrix at level l is $O(m_{c,l}^2)$. The computation of the k-means algorithm at level l is $O(\kappa(m_{c,l}) \cdot k_l + m_{c,l}))$ $= O(\kappa(m_{c,l}) \cdot k_l)$, where k_l is the number of partitions at level l and κ is the number of iterations for k-means to converge. Therefore, the total complexity for the kernel k-means refinement over $L + 1$ levels is:

$$\sum_{l=0}^{L} O(m_{c,l}^2 + \kappa \cdot m_{c,l} \cdot k_l) = O(\frac{4 - (\frac{1}{4})^L}{3} m_c^2 + (2 - (\frac{1}{2})^L)\kappa \cdot k \cdot m_c) = O(m_c^2 + \kappa \cdot k \cdot m_c)$$

In general, $k << m_c$, and most of the refinement occurs in the first few iterations, hence κ can be upper-bounded by a small value and the k-means refinement will still produce accurate results. Hence, the computation of the kernel is the dominant factor, particularly for small l, where $m_{c,l}$ is largest. Fortunately, the computation of the normalised cut kernel is based on the adjacency matrix of the evolving dissimilarity graphs, which are generally very sparse (the limited lookahead, η, ensures many of the values are zero) for low l. If the number of non-zero entries is $nz_{c,l}$, then the complexity to compute the kernel is only $O(nz_{c,l})$ for the lower levels. Since $m_c = m_{c,0}$, then the cost to compute the refinement is $O(nz_c + \kappa \cdot k \cdot m_c)$.

The analysis of the partitioning of the spatial evolving dissimilarity graphs is more difficult, as it depends on the size and number of partitions produced in the temporal partitioning step. Consider two cases to obtain some intuition about the complexity. Let there be k partitions. The spatial evolving dissimilarity graphs are generally sparse. Hence, the complexity to partition one of these evolving graphs is $O(n^{1.5})$, where n is the number of vertices. First,

consider the case where the m_c changed edges are evenly divided into k partitions. The complexity of partitioning these are $k \cdot O((\frac{m_c}{k})^{1.5}) = O(\frac{m_c^{1.5}}{k^{0.5}})$. This uniform distribution of the partition sizes is actually the lower bound on the complexity among k partitions. On the other extreme, we can have the scenario where $k-1$ of the partitions are singletons, and the remaining partition contains the rest of the edges. In this case, the complexity is $(k-1)(1)+O((m_c-(k-1))^{1.5})$. If $m_c >> k$, which is usually the case, then the complexity is $O(m_c^{1.5})$. The complexity lies somewhere between the two extremes, and we can generally say that the complexity is $O(m_c^{1.5})$, given $k << m_c$ in general.

Therefore, the complexity of regHunter, k partitions and assuming the worst case for partitioning the spatial evolving graphs, is:

$$\begin{aligned} & O(m_c^2 \cdot wn \cdot \eta + S \cdot m_c \cdot m + m_c^2 + \\ & \quad m_{c,L}^{1.5} + nz_c + \kappa \cdot k \cdot m_c + m_c^{1.5}) \\ & = O(m_c^2 \cdot wn \cdot \eta + S \cdot m_c \cdot m + nz_c) \end{aligned} \tag{4}$$

In practice, most of the steps do not take the worst case times. For example, computing the temporal distances takes a small fraction of $O(m_c^2 \cdot w_n \cdot \eta)$ time. This is also the case for computing the shortest paths, particularly when many of the distances computed are short.

5 Evaluation of regHunter

In this section we evaluate the accuracy and running time of regHunter. Accuracy was evaluated using external and internal cluster validation methods. In the external methods, we compare the obtained set of regions with a known set of true regions, while in the internal methods, we compare against a set of measures that are based on other partitioning

objectives. External methods are used to evaluate the effect of the parameter settings on the accuracy and timing of regHunter on synthetic datasets (with known clusterings), and to compare the results of regHunter with those of cSTAG. We also use a set of internal measures to evaluate the regions found by regHunter on two real datasets, where there is no known set of clusterings - the access graph of the 1998 World Cup website, and the BGP connectivity graph during the landfall of Hurricane Katrina.

In the rest of this section, we first describe the external and internal accuracy validation measures used. Then we outline the synthetic dataset generation process, and present the accuracy and running time results of cSTAG and regHunter. Finally, we introduce the World Cup and BGP connectivity datasets, and present the internal validation results in each case, as well as the regions discovered for the World Cup data.

5.1 External Accuracy Validation Measure

In external accuracy validation, we have a set of true (generated) regions, R^{tru}, and a set of detected regions R^{det}. We wish to quantify how closely the two sets of regions match. The number of regions in each set can differ, and the matching between two regions can be partial, hence traditional notions of true and false positives cannot be used. This problem has been partially studied in the field of external cluster validation [16]. Existing validation methods can be divided into three groups: a) those that count the number of pairs; b) those that compare set membership; and c) those that compare membership distribution. Pair counting methods are based on comparing the number of pairs of objects that belong in the same cluster in one or both clusterings. Some examples include the Rand and Jaccard indices [16]. Methods that compare set membership are based on finding the best matching between the clusters, using set overlap as the criterion [32][16]. Finally, membership distribution

methods compute the distance between the cluster membership distributions of the two clusterings. Examples include ADCO [31] and [30].

However, in the region comparison problem, there is also the need to consider the temporal behaviour of the regions. Consider the case where two regions consist of the same set of changed edges but are defined over different periods of time, like regions A1 and A2 from Table 2. Existing cluster comparison measures will incorrectly indicate they are the same, since they have the same set of edges, but in fact the measure should indicate that they are different because of their temporal differences.

Recently, Zhou et al. [30] posed the comparison problem as a mass transportation problem [22], trying to minimise a set of matching weights between two clusterings. A significant advantage of this formulation is that it allows for partial matches among clusters and uses a region distance formulation that can incorporate both membership and temporal considerations. Therefore, we extend Zhou's method to consider temporal behaviour of regions and membership similarities.

Consider Figure 5, which illustrates the general region comparison problem. Each region in R^{tru} is matched to one or more regions in R^{det}. The relative importance of a region R_z^{tru} in R^{tru} is measured by β_z, $\beta_z \geq 0$, $\sum_{z=1}^{Z} \beta_z = 1$. It indicates the relative emphasis the matching algorithm places on matching region R_z^{tru} with regions in R^{det}. For example, $\beta_z = 0$ indicates that the matching of R_z^{tru} should have no bearing on the total distance between the two sets of regions, while $\beta_z = 1$ indicates R_z^{tru} must be matched with all the regions in R^{det}. We similarly define the relative importance of region R_y^{det} as α_y, $\alpha_y \geq 0$, $\sum_{y=1}^{Y} \alpha_y = 1$.

Weight w_{yz} indicates the extent of matching between R_y^{tru} and R_z^{det}. There are two constraints on the matching weights: 1) $w_{yz} \geq 0$; 2) $\sum_{y=1}^{Y} w_{yz} = \beta_z$, $\sum_{z=1}^{Z} w_{yz} = \alpha_y, \forall y, z$. The second constraint uses the perceived relative importance of each region to control the

amount of matching. As we wish all regions to be matched, and we do not know a priori their relative importance, we set $\alpha_y = \frac{1}{Y}, \forall y$ and $\beta_z = \frac{1}{Z}, \forall z$ for the experiments.

The distance between two regions has a temporal and spatial component. If the temporal and spatial components are matched separately, then it is likely that two different sets of matchings will result. To prevent this occurring, the two inter-region distances are combined into one measure. More formally, the distance between region R_y^{tru} and region R_z^{det} is defined as:

$$d(R_y^{tru}, R_z^{det}) = \mu \cdot d_S(R_y^{tru}, R_z^{det}) + (1 - \mu) \cdot d_W(R_y^{tru}, R_z^{det})$$

where $0 \leq \mu \leq 1$. Distances $d_S(R_y^{tru}, R_z^{det})$ and $d_W(R_y^{tru}, R_z^{det})$ are the set and waveform distance between R_y^{tru} and R_z^{det}, respectively. The waveform distance measures the temporal correlation. The terms μ and $1 - \mu$ represent the relative contribution of each distance to the overall distance.

Definition 9 *The set distance between regions is defined as:*

$$d_S(R_y^{tru}, R_z^{det}) = 1 - \frac{2 \cdot |R_y^{tru} \cup R_z^{det}|}{|R_y^{tru}| + |R_z^{det}|}.$$

It is computed as 1 - the (normalised) set intersection between the edges of the two regions. Note that the normalisation avoids cases where a small distance can occur between a large and small region, where the large region is a superset of a smaller region.

For notational convenience, we denote the set of changed waveforms (and their relative frequencies) associated with the changed edges in region $R_r^{ts,te}$ by $R_r^{ts,te}.Q$. $R_r^{ts,te}.Q$ is a set of (change waveforms, frequency) pairs, where the change waveforms are drawn from the collection of change waveforms of its member edges, and the frequency defines the relative number of times it occurs among the collection of waveforms. The frequency ranges from 0

to 1.

Definition 10 *The waveform distance between regions is defined as:*

$$d_Q(R_y^{tru}, R_z^{det}) = \sum_{a=1}^{|R_y^{tru}.Q|} \sum_{b=1}^{|R_z^{det}.Q|} freq_a \cdot freq_b \cdot d(q_a, q_b) \tag{5}$$

This distance measure compares each waveform in $R_y^{tru}.Q$ with each waveform in $R_z^{det}.Q$, weighted by the relative count of the waveforms. The waveform measure we use, d, is the Euclidean distance $d_{eq}()$. However, the waveforms in the region comparison problem might not have the same length, and could have different starting and ending times. We wish to penalise these cases, as similar sequences of change should be defined over similar periods of time. Hence, $d(q_a, q_b)$ incorporates this by penalising misalignment among the two waveforms, and is formally defined as:

$$d(q_a^{tsa,tea}, q_b^{tsb,teb}) = (tsc - \min(tsa, tsb)) + d_{ed}(_a^{tsc,tec}, q_b^{tsc,tec}) + (tec - \max(tea, teb))$$

where (tsc, tec) denotes the common starting and ending times of the two waveforms, and d'_{ed} is the unnormalised Euclidean distance measure.

With the distance between two regions defined, we can now define the distance measure between two sets of regions.

Definition 11 *The total distance between the sets of regions R^{tru} and R^{det} is given by*

$$d(R^{tru}, R^{det}) = min_{w_{yz}} \sum_{y=1}^{Y} \sum_{z=1}^{Z} w_{yz} d(R_y^{tru}, R_z^{det}) \tag{6}$$

subject to $w_{yz} \geq 0, \sum_{z=1}^{Z} w_{yz} = \alpha_y, \sum_{y=1}^{Y} w_{yz} = \beta_z, \forall y, z,$ *and* $0 \leq d(R^{tru}, R^{det}) \leq 1$.

Solving this equation involves finding the set of matching weights that minimises the total distance between the matched regions. This is equivalent to finding a matching that minimises the distance between the regions. Equation 6 is in the form of a mass transportation problem [22]. Mass transportation problems are known to always yield an optimal solution [22], and can be solved using a technique such as the simplex method [22].

In our presentation of the results, we converted the distance scores to an accuracy score (1 - distance). Henceforth, we refer to our method as *extRegCompare*.

5.2 Internal Accuracy Validation Measure

The two internal measures we use are the average intra-region distance and the average inter-region distance. Average intra-region distance measures how compact the regions are. Average inter-region distance measures how well separated the regions are.

Both of these measures depend on an intra or inter-region distance measure. We use separate spatial (in terms of shortest path distances) and temporal distances. So effectively, there are four different measures (two for temporal, two for spatial). More formally, the average intra-region spatial distance is defined as:

Definition 12

$$d_{spa}(R_y, R_z) = \frac{1}{|R_y||R_z|} \sum_{e_i \in R_y} \sum_{e_j \in R_z} d_{intra\_spa}(e_i, e_j, W^{1,S})$$

where R_y and R_z are two of the discovered regions.

The average inter-region spatial distance is defined as $\frac{1}{|\mathbf{R}-1|} d_{spa}(R_y, R_z),\ R_y, R_z \in \mathbf{R}$, $R_y \neq R_z$. The average intra and inter region temporal distances are defined similarly, except d_Q (see Equation 5) replaces d_{intra_spa}.

5.3 Synthetic Region Dataset

5.3.1 Synthetic Graphs

The accuracy of regHunter is dependent on the separability of the regions. The separability is measured by three factors: **minSpaSep**, the minimum spatial separation between any pair of edges that are in different regions; ii) **minTemSep**, the minimum temporal separation between the change waveforms of any pair of edges in different regions; and iii) **minEvtSep**, the minimum temporal separation between consecutive, but independent and separate windows of changes affecting the same set of edges. By varying these factors, we can generate a variety of synthetic dynamic graphs and introduce known regions of correlated change.

Two methods were used to generate the synthetic datasets. The first method generates the regions, then generates the path between them to form the dynamic graph. The second method generates the graph, then randomly selects subgraphs to become regions. The first method allows more control over the separation criteria between the regions. However, it is more difficult to control the size of snapshots generated using the first method. Hence, the introduction of the second method.

Region-then-link Method To generate the datasets using this method, a pseudo random graph is first generated, with a number of random subgraphs extracted from the random graph. Each subgraph must be at least minSpatSep from all the other extracted subgraphs. Next, a random sequence of changes are generated for each subgraph, such that the distance between any pair of sequences is at least minTemSep. In addition, each sequence consists of a random number of subsequences of high change, separated by at least minEvtSep of periods of no change. Each subgraph, along with a subsequence of change, constitutes a region. A simulation is used to generate the snapshots and regions by applying the sequences of

changes to the original random graph. The names of datasets generated by this method have the prefix *synGen*.

Graph-then-region Method To generate datasets using this method, a graph with random edges and vertices is first generated. Any graph model generator can be used to generate this graph. Next, non-overlapping subgraphs are selected for each region. Finally, random change sequences are assigned to each region, and additional graph snapshots are generated based on the original graph, the set of regions, and their associated change sequences. The names of the datasets generated by this method have the prefix *introGen*.

5.4 Synthetic Dataset Evaluation

In this section, we evaluate the sensitivity of regHunter to its parameters: the stopping thresholds, window size, lookahead and maximum quiet period. We also compared regHunter against several methods from cSTAG, namely *hard* and *soft* techniques with {single-linkage, average-linkage and leaderFollower} clustering algorithms, and *sequential* with leaderFollower + singleLinkage (apply leaderFollower using the temporal distances, then singleLinkage using the spatial distances) and leaderFollower + averageLinkage. For each type of dataset, three different instances were generated. The results reported are an average over the three instances. Finally, cSTAG was implemented completely in C++, while most of regHunter was implemented in Matlab^{TM}. All synthetic tests, except the tests involving the variation in graph size, were conducted on an Intel Duo Core 2.8GHz PC with 2GB of memory and running Fedora Core 7. The graph size variation tests and the real dataset evaluations were performed on a server with 16 Intel Xeon 3.0GHz processors, 32 GB of memory and running RedHat Enterprise Linux 5.

5.4.1 Parameter Sensitivity

We used datasets generated with the parameters **minTempSep** = 3, **minSpatSep** = 3 and **minEvtSep** = 3 to evaluate the effect of regHunter's parameters. We have also conducted this evaluation using other generated datasets, but the results from these datasets exhibit similar trends, hence we do not report them here.

Figure 6a and 6b compares the effect of varying the temporal and spatial stopping thresholds ($stop_{tem}$ and $stop_{spa}$), with different window sizes (ω), on the accuracy and timing of regHunter. The results show that regHunter is more sensitive to $stop_{spa}$ than $stop_{tem}$. As $stop_{spa}$ increases, the accuracy generally increases, hits a maximum around the values of 0.8 to 0.9, then falls away. For low stopping thresholds, the algorithm is prematurely stopped, while for high thresholds, too much partitioning has occurred. This phenomena occurs across all window sizes.

In general, the smaller window sizes produced more accurate results. This can be attributed to the fact that smaller windows give better comparison granularity, which usually results in less false negative matches but more false positive matches. However, combined with sufficient lookahead and graph partitioning, the number of false positive matches is reduced, hence the smaller window lengths are more accurate than the longer window lengths.

It can be seen from Figures 6e and 6f that the running time also plateaus around $stop_{tem}/stop_{spa} = 0.8$ to 0.9. The reason this occurs is that as the stopping threshold increases, the total number of iterations for refinement increases. However, when the thresholds hit 1.0, there are actually fewer iterations, because the regions are much smaller in size. Hence, the optimal stopping thresholds occur over the range of 0.8–0.9. This holds over all the other synthetic datasets that we generated and evaluated.

Figure 6c shows the results of varying the lookahead (η) and window size(ω). The results

indicate that the optimal lookahead value depends on the window size used. For a small window, greater lookahead improves the accuracy, as it allows regHunter to look for longer correlation spans that a small window does not allow it to do. If there is too much lookahead, similar regions that should be separate can be accidentally merged.

It is difficult to identify a trend or pattern for the running time (Figure 6g). After manually examining the steps of the algorithm and results, changing the lookahead and window size affects both the coarsening and refinement time, and also the time needed to partition the spatial evolving dissimilarity graphs. For example, $\omega = 9$ and $\eta = 3$ had triple the running time of $\omega = 9$ and $\eta = 6$ because there were some larger spatial evolving dissimilarity graphs to be partitioned, which greatly increased the running time.

Figures 6d and 6h show the results from varying the maximum quiet time (η) and four configurations of $stop_{tem}$ and $stop_{spa}$. The results show that there are no monotonicity properties associated with the maximum time between changes. For example, there is a dip in accuracy when $\psi = 5$. After studying the results, it seems that some regions discovered at $\psi = 4$ were further incorrectly partitioned at $\psi = 5$, while other regions that should have been further partitioned were not. These results suggest that a range of ψ values should be evaluated for different datasets, but generally ψ should not be too small compared to the sequence length.

5.4.2 Comparison with cSTAG

To ensure a fair comparison, we tested up to 360 different parameter configurations for each of the multi-objective approaches of cSTAG (120 for each of the single-relation clustering methods, including multiple window sizes (3–12)) for each dataset. In contrast, we only tested a total of about 100 parameter configurations ($\omega = \{3, 6, 9, 12\}$, $\eta = \{3, 6, 9, 12\}$,

$stop_{tem} = stop_{spa} = \{0.7, 0.8, 0.9, 1.0\}$, $\psi = \{3, 4, 5, 6\}$) for regHunter.

We evaluated regHunter and cSTAG on five different types of datasets, and their details are summarised in Table 4. *exampleDS* is the motivating example given in the introduction (Figure 2, Table 2). *synGen003* is the set of synthetic datasets used to evaluate the parameter sensitivity of regHunter. *synGen002* and *synGen001* are additional sets of datasets used for comparison. *synGen002* are generated with **minTempSep** = 2, **minSpatSep** = 2 and **minEvtSep** = 2, while *synGen001* are generated with **minTempSep** = 1, **minSpatSep** = 1 and **minEvtSep** = 1. The region separation of *synGen002* and *synGen001* are smaller than *synGen003*. In addition, *introGenSize* is a set of dynamic scale-free graphs that varied from 2000-16,000 edges. 20% of the edges experienced change over a sequence of 30 snapshots. In terms of scalability, *introGenSize* is particularly difficult as: 1) it has scale-free graph properties, greatly increasing the number of vertices visited in any shortest path search, and 2) each region does not necessarily form a connected component.

We present the results in terms of the average of the k most accurate results, where k takes values from the set $\{1, 10, 30, 50\}$. Then we use these top k results to compute the appropriate average running time. This presentation format allows, on one hand, the spread of the results to be displayed. On the other hand, it kept the comparison fair, as it eliminated some of the extremely inaccurate results (single digit accuracy) for cSTAG.

First, we shall examine the results of analysing the example dataset (Figure 7a). The results show that even though cSTAG consists of a variety of different methods that in theory should perform well for a large spectrum of datasets, and we also varied the window size over a large range, the best the algorithms of cSTAG can achieve is to find the three regions when a large window size is used. This corresponded to regions B, C and incorrectly identifying regions A1 and A2 as one region. On the other hand, regHunter could find the correct

regions, with fewer parameters to tune. Even the average accuracy of the top 50 results of regHunter was significantly more accurate than any version of cSTAG. Both algorithms took less than a second to find their set of regions.

regHunter also had superior accuracy for the other datasets. Consider the results for *synGen003* (Figure 7b and 7c). The regions generated are reasonably separated, hence cSTAG is able to achieve almost 90% accuracy for the leader-single *sequential* method. However, even for this well separated dataset, regHunter still has superior accuracy (almost 95%). Equally as important, the spread of the accuracy for regHunter is less than cSTAG (the top 50 result for regHunter is on par with the best results of cSTAG), which indicates that regHunter is less sensitive to the choice of parameters when evaluating datasets with reasonably separated regions.

Figure 7d and 7e shows the results for *synGen002*. Although the top 10, 30 and 50 accuracy results are comparable to cSTAG, the best results of regHunter are at least 10% better than the best results of cSTAG.

Consider the results for *synGen001* (Figure 7f and 7g). Again, regHunter has better accuracy across all the results. In particular, the best results are approximately 13-15% better than those of cSTAG.

Finally, consider the results for *introGenSize* (Figure 8). Although the running time for cSTAG is faster than regHunter for smaller graphs, as Figure 8 and other previous synthetic dataset evaluation showed, regHunter was about twice as fast as cSTAG for larger graph sizes. As the graph size increases, the time to compute the region association for cSTAG starts to become more and more significant. Around a graph size of 8000 edges, the region association time overtakes the time to compute the shortest path distances, resulting in regHunter being faster than cSTAG as the graph size increases.

Although regHunter is slower than cSTAG in smaller datasets, it must be remembered that most of regHunter was implemented in Matlab^{TM}, which is slower than a C++ implementation. Furthermore, a vast amount of time (up to 30%) was spent copying data structures between C++ and Matlab. Furthermore, the total time required to find the best regions for cSTAG using all parameter settings was magnitudes longer than for regHunter – days compared to hours. Most importantly, regHunter is faster than cSTAG as the size of the graph increases.

This synthetic dataset evaluation indicates that regHunter is more accurate than cSTAG for datasets with regions that have small temporal and spatial separation. The difference in accuracy increases as the regions in the datasets become less separated (*synGen003*, *synGen002*, *synGen001*, *exampleDS*). In addition, regHunter is generally less sensitive to parameter selection, reducing the time needed for parameter tuning, hence actually making it easier and faster to find the best regions.

5.5 Internal Accuracy Evaluation

In this section, we analyse the normalised intra and inter-region distances on the synthetic dataset *synGen003*, and compare them to the results obtained by using the external validation measure. This provides an indication of the relative accuracy of the two internal validation measures. In terms of these two measures, low intra-region distance and high inter-region distance is preferable.

Figure 9 shows the {temporal, spatial} {intra, inter} region distance for the results obtained when regHunter was evaluated on the dataset *synGen003*. As there are four different measures, we plotted different combinations of measures against each other.

For the four plots (Figure 9), we can see there is a group of results (blue ellipse, la-

beled '1') that have low spatial and temporal intra-region distances. These results have low *extRegCompare* scores, because they consist of many small sized regions, which have low intra-region distances but also low inter-region distances (for example, as shown by the results in the blue ellipse labelled '1' in Figure 9d).

The best group of results suggested by *extRegCompare* consist of the group of results circled by the red ellipses, labeled '2'. As can be seen, these results have low intra-region distances, but also low to medium spatial and temporal inter-region distances. Contrast this with the group of results (green ellipses, labeled '3') with high inter-region distances. These also have low to medium intra-region distances, but very low *extRegCompare* scores – within the bottom 30% of results.

The results (Figure 10) for the most accurate algorithm of the cSTAG framework, *sequential-leaderFollower-singleLinkage*, have similar trends to the results of regHunter. The singleton regions, highlighted by the blue ellipses, labeled '1', have low temporal and spatial intra-region distance, but relatively poor spatial inter-region distance. Again, these single region results had an accuracy of about 5% when evaluated by the *extRegCompare* measure. The results highlighted by the green ellipses, labeled '3', had high spatial inter-region distance, but relatively low temporal inter-region distances. They also had average spatial, but low temporal intra-region distances. These results had on average 65% accuracy when evaluated by the *extRegCompare* measure. Finally, the results that had higher *extRegCompare* measured accuracy are highlighted by the red ellipses, labeled '2'. Similar to the regHunter results, they have relatively low intra-region distances, high temporal and moderate spatial inter-region distances.

For comparison, the distances for the reference set of regions for the *synGen003* were: temporal intra-region = 0.000, spatial intra-region = 0.177, temporal inter-region = 0.923

and spatial inter-region = 0.447. Note that both the best results (red ellipse, labelled '2') of regHunter and sequential-leaderFollower-singleLinkage have similar values for the distances.

In summary, the results indicate that it is not possible to achieve both low intra-region and high inter-region distances. However, as the evaluation suggests, the set of intra and inter-region distances, when used together, provide an adequate guide to accuracy. Hence, in the next subsections, we shall use these as measures of the accuracy obtained for the two real datasets.

5.6 1998 World Cup Website Access 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*[6], access logs[7] of the web site were analysed by Arlitt and Jin [1]. 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. 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 football matches on television, monitored the live scores via the website during the matches, producing the flash crowds. Figure 11, 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 different types of access - e.g., a group of accesses relating to online viewing of a particular match. In previous work [6], we had manually matched discovered regions with the matches, based on the time the regions were defined and the websites

[6]As of August 2007, the address is still valid, but links to a general soccer promotion website.
[7]Available at [29].

contained in them. To extend our evaluation, we use the four described internal measures to evaluate the obtained regions. In addition, we present two new interesting regions found by regHunter that cSTAG was not able to find.

5.6.1 Construction of the Snapshots

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

Each flash crowd should 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 11 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.

The resulting snapshots varied greatly in size, ranging from 4692 to 3871 vertices and 110 to 530 edges. The size of the snapshot corresponds directly to the amount of traffic to the website (see Figure 11).

5.6.2 Internal Accuracy Evaluation

The *leaderFollower-singleLinkage* algorithm was used as a comparison benchmark against *regHunter* for the World Cup dataset evaluation, as it produced the best result among the algorithms of the cSTAG framework. We tested 15 diverse parameter configurations for both algorithms, and present the internal accuracy results for the best intra-region distance (Table 5), and best inter-region distance (Table 6) for each algorithm.

Consider the best intra-region results first (Table 5). Although the result for regHunter had higher temporal intra-region distance, it also had lower spatial intra-region distance, as well as significantly better temporal inter-region distance. The best inter-region results (Table

6) show similar trends. regHunter had slightly worse spatial results, but its temporal results were more accurate than *leaderFollower-singleLinkage*, particularly the temporal intra-region result. In summary, for both sets of results, *regHunter* had better overall accuracy.

5.6.3 New Findings on the 1998 World Cup Dataset

In this section, we present and discuss two new regions of correlated change discovered by regHunter. These regions were not detected by the *leaderFollower-singleLinkage* algorithm of cSTAG, nor in the analysis of the World Cup dataset in previous work [6]. The analysis was performed over the same two day period as in [6], from 0000 Friday, June 12th to 2359 Saturday June 13th.

We shall present the regions in two ways. The first method is to present the size, dominant change waveforms, and some comments about the two regions. The second method is to extract the list of incident objects in each region, and truncate the list to the most interesting objects, similar to the approach taken in [6].

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 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 refer to the biography of team YYY, and *groupstandings163_77.htm* refers to the group standings of group 163_77.

Consider the information about the two regions, presented in Tables 7, 8 and 9. The two regions identified by regHunter are different to all the regions identified by cSTAG. The newly identified regions only have matchstat objects for Saturday matches, but no matchprog

objects. This suggests the purpose of the accesses that make up these regions are to view the match summaries and team information, not to watch the live match updates (matchprog sites). This is significantly different from the regions found by cSTAG [6], where the accesses are more likely from fans who watch the live matches, then log off the website. Those regions had change waveforms that appear then disappear. The two regions however, have change waveforms that just appear, and are likely to represent a constant stream of fans who check the match summaries throughout Saturday.

The main difference between the two regions is the access of object teambio138.htm and the time of the appearance of the regions. Region Sat-C2 is likely to represent fans in a different time zone to Region Sat-C1 (hence the delay in the appearance of Sat-C2). They are also more likely to be fans of team 135.

We analysed why cSTAG could not detect these regions, and we found that cSTAG would distribute the edges discovered in these two regions to regions that represent live match viewing (i.e., with matchprog objects), or to noisy regions that are difficult to interpret. From this analysis, it can be seen that regHunter is not only quantitatively more accurate than cSTAG, but can also identify new regions that cSTAG misses.

5.7 BGP Dataset Evaluation

In this section, we use the internal validation measure to compare the accuracy of regHunter and cSTAG on the US portion of the Border Gateway Protocol (BGP) connectivity graph during the landfall of Hurricane Katrina. In prior work [6], we have qualitatively analysed this dataset, but did not perform any internal validation.

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. The BGP connectivity graph represents the top-level routing topology of the Internet.

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[8] at the University of Oregan 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. Please refer to [6] for details on how the connectivity graphs can be built from the table entries.

In [6], we examined the Katrina event because it has been reported that its effect on the Internet was mostly localised around Louisiana and several other southern states. This enabled us to demonstrate the ability of cSTAG to show significant activity in the ASs around that region. We also 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 28 August, 13:19 to 31 August, 22:32. This period included the landfall of Hurricane Katrina (around 29 August, 10:00). Figure 12 shows the number of edges and vertices that have experienced a change in each window.

[8] `http://www.routeviews.org`

5.7.1 Internal Accuracy Evaluation

Similar to the World Cup analysis, we tested 15 different parameter settings for the two algorithms, and presented the results with the best intra-region distances, and the best inter-region distances (Tables 10 and 11 respectively).

Consider the best intra-region distances (Table 10). In terms of temporal intra-region distance, the results for regHunter are about 10% more accurate than the *leaderFollower-singleLinkage* algorithm. For the other internal measures, they are similar. The best inter-region distance results (Table 11) again indicate that regHunter is much more accurate in terms of temporal intra-region distances, and has comparable results for the other measures. Note that the spatial distances for this dataset are generally high, no matter what algorithm is used to extract the regions, because the BGP network has scale-free properties, which means it has a small diameter, and the shortest path between any pair of vertices is small. Hence, the temporal aspect of the regions is the more discriminating dimension for this particular dataset.

These results indicate that regHunter can generally produce regions that are more accurate than the best algorithm of the cSTAG framework.

6 Conclusion

In this paper, we formulated the region discovery problem as a multi-objective optimisation problem. We then proposed a new algorithm, regHunter, to solve this optimisation formulation using multi-level spectral graph partitioning. In addition, we have developed a new dataset generator, presented a new comparison method to quantitatively measure the accuracy of the set of discovered regions and applied internal validation analysis to two

real datasets. We found regHunter was able to substantially outperform all methods proposed in cSTAG while being only slightly slower than cSTAG. Furthermore, we showed that regHunter is more accurate than cSTAG when analysing real datasets like the World Cup access graph and the BGP connectivity graph data. We also demonstrated that the increased accuracy of regHunter meant it was able to find interesting regions in the World Cup data that cSTAG did not find.

There are a number of future directions we would like to pursue. One of these is to use ideas from the constraint programming community to help the recursive bipartition process to focus on more promising areas in the search space. In addition, we would like to adapt the multi-level graph partitioning approach to an incremental formulation. We want to keep the accuracy obtained from the batch mode of regHunter, but adapt it to an incremental formulation for multilevel structures.

7 Acknowledgements

We would like to thank National ICT Australia for their support and funding of this research. In addition, we are in debt to Martin Arlitt and Tai Jin of Hewlett-Packard Laboratories for making the workload characteristics to the World Cup website publicly available, and the Lawrence Berkeley National Laboratory for making the traces available.

References

[1] Martin Arlitt and Tai Jin. Workload characterization of the 1998 World Cup website. Technical Report HPL-99-35R1, Hewlett-Packard Labs, September 1999.

[2] Karsten M. Borgwardt, Hans-Peter Kriegel, and Peter Wackersreuther. Pattern mining in frequent dynamic subgraphs. In *Proceedings of the 6th International Conference on Data Mining*, pages 818–822, 2006.

[3] Matthrew Caesar, Lakshminarayanan Subramanian, and Randy H. Katz. Towards localizing root causes of BGP dynamics. Technical Report UCB/CSD-04-1302, University of California, Berkeley, November 2003.

[4] M.R. Carey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, 1979.

[5] Mete Celik, Shashi Shekhar, James P. Rogers, James A. Shine, and Jin Soung Yoo. Mixed-drove spatio-temporal co-occurance pattern mining: A summary of results. In *Proceedings of the 6th International Conference on Data Mining*, pages 119–128, 2006.

[6] Jeffrey Chan, James Bailey, and Christopher Leckie. Discovering correlated spatio-temporal changes in evolving graphs. *Knowledge and Information Systems*, 2007.

[7] Stuart Clare. *Functional MRI : Methods and Applications.* PhD thesis, University of Nottingham, 1997.

[8] Nello Cristianini and John Shawne-Taylor. *An Introduction to Support Vector Machines and other Kernel-based methods.* Cambridge University Press, 2000.

[9] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[10] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kullis. A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[11] C. Ding, X. He, H. Zha, and H. Simon. A minmaxcut spectral method for data clustering and graph partitioning. Technical Report 54111, LBNL, 2003.

[12] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Kluwer, 2002.

[13] Nick Feamster, Hari Balakrishnan, and Jennifier Rexford. Some foundational problems in interdomain routing. In *3rd ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets)*, November 2004.

[14] Anja Feldmann, Olaf Maennel, Z. Morley Mao, Arthur Berger, and Bruce Maggs. Locating internet routing instabilities. In *SIGCOMM '04*, pages 205–218, New York, NY, USA, 2004. ACM Press.

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins Press, 3 edition, 1996.

[16] Maria Halkidi, Yannis Batisakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2–3):107–145, 2001.

[17] George Karypis and Vipin Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 1(3):285–303, 2000.

[18] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1998.

[19] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew S. Tomkins. On the bursty evolution of blogspace. In *Proceedings of the 12th International Conference on World Wide Web*, pages 568–576, 2003.

[20] Ravi Kumar, Jasmine Novak, and Andrew S. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (poster)*, 2006.

[21] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 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*, pages 177–187, 2005.

[22] D. Luenberger. *Linear and Nonlinear Programming*. Kluwer Academic Publishers, 2003.

[23] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, 2000.

[24] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (8), 2000.

[25] Peter J. Shoubridge, Miro Kraetzl, Walter D. Wallis, and Horst Bunke. Detection of abnormal change in a time series of graphs. *Journal of Interconnection Networks*, 3(1-2):85–101, 2002.

[26] Malgorzata Steinder and Adarshpal S. Sethi. Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM Transactions on Networking*, 12(5):809–822, 2004.

[27] Jimeng Sun, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. Graphscope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 687–696, 2007.

[28] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *ACM SIGKDD Explorations Newsletter*, 5(1):59–68, 2003.

[29] 1998 World Cup website access traces. `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`

[30] Ding Zhou, Jia Li, and Hongyuan Zha. A new mallows distance based metric for comparing clusterings. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[31] Eric Kyoo Han Bae, James Bailey and Guozhu Dong. Clustering Similarity Comparison Using Density Profiles. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence*, pages 342–351, 2006.

[32] Marina Meila. Comparing Clusterings. Technical Report 418, University of Washington, 2002.

A Monotonicity Proof of Normalized Cut

Let V be the set of vertices to be partitioned into regions. Let $R^x = \{R_1, \ldots, R_x\}$ be the set of regions at iteration x. Let R_y be the partition chosen to be bisectioned into disjoint partitions R'_y and R_{x+1}, $1 \leq y \leq x$. Let $inter(R_A, R_B) = \sum_{i \in R_A, j \in R_B} d(i,j)$ be the inter-region distance,

$intra(R_A) = \sum_{i \in R_A, j \in R_A} di, j$ be the intra-region distance, and $assoc(R_A) = inter(R_A, V - R_A) + intra(R_A)$ be the inter-region + intra-region distances.

Lemma 2

$$inter(R_y, V - R_y) \leq inter(R'_y, V - R'_y) + inter(R_{x+1}, V - R_{x+1})$$

Proof 2 *Consider the sum of the inter values of* R'_y *and* R_{x+1}.

$$\begin{aligned} &inter(R'_y, V - A'_y) + inter(R_{x+1}, V - R_{x+1}) \\ &\quad = \sum_{i \in R'_y, j \in V - R_y} d(i,j) + \sum_{i \in R_{x+1}, j \in V - (R_y \cup R_{x+1})} d(i,j) + \sum_{i \in R'_y, j \in R_{x+1}} d(i,j) + \sum_{i \in R_{x+1}, j \in R'_y} d(i,j) \\ &\quad = inter(R_y, V - R_y) + \sum_{i \in R'_y, j \in R_{x+1}} d(i,j) + \sum_{i \in R_{x+1}, j \in R'_y} d(i,j) \end{aligned}$$

The result follows since $\sum_{i \in R'_y, j \in R_{x+1}} d(i,j) + \sum_{i \in R_{x+1}, j \in R'_y} d(i,j) \geq 0$.

Lemma 3

$$assoc(R_y) = assoc(R'_y) + assoc(R_{x+1})$$

Proof 3 *Consider the association values of* A'_y + A_{x+1}.

$$
\begin{aligned}
&assoc(R'_y) + assoc(R_{x+1}) \\
&\quad = \sum_{i \in R'_y, j \in V} d(i,j) + \sum_{i \in R_{x+1}, j \in V} d(i,j) \\
&\quad = \sum_{i \in R'_y, j \in R'_y} d(i,j) + \sum_{i \in R'_y, j \in R_{x+1}} d(i,j) + \sum_{i \in R'_y, j \in V-(R'_y \cup R_{x+1})} d(i,j) \\
&\quad + \sum_{i \in R_{x+1}, j \in R_{x+1}} d(i,j) + \sum_{i \in R_{x+1}, j \in R'_y} d(i,j) + \sum_{i \in R_{x+1}, j \in V-(R'_y \cup R_{x+1})} d(i,j) \\
&\quad = \sum_{i \in R_y, j \in R_y} d(i,j) + \sum_{i \in R_y, j \in V-R_y} d(i,j) \\
&\quad = \sum_{i \in R_y, j \in V} d(i,j) = assoc(A_y)
\end{aligned}
$$

Lemma 4

$$
f_{tem}(R^x) \leq f_{tem}(R^{x+1}) \tag{7}
$$

Proof 4 *We can rewrite Equation 7 as:*

$$
f_{tem}(R_1, \ldots, R_y, \ldots, R_x) \leq f_{tem}(R_1, \ldots, R'_y, \ldots, R_x, R_{x+1})
$$

Expanding the terms, we get:

$$
\begin{aligned}
\frac{inter(R_1, V-R_1)}{assoc(R_1)} + \ldots + \frac{inter(R_y, V-R_y)}{assoc(R_y)} + \ldots + \frac{inter(R_x, V-R_x)}{assoc(R_x)} \\
\leq \frac{inter(R_1, V-R_1)}{assoc(R_1)} + \ldots + \frac{inter(R'_y, V-R_y)}{assoc(R'_y)} + \\
\ldots + \frac{inter(R_x, V-R_x)}{assoc(R_x)} + \frac{inter(R_{x+1}, V-R_{x+1})}{assoc(R_{x+1})}
\end{aligned}
$$

Canceling terms on either side of the inequality gives:

$$\frac{inter(R_y, V - R_y)}{assoc(R_y)} \leq \frac{inter(R'_y, V - R_y)}{assoc(R'_y)} + \frac{inter(R_{x+1}, V - R_{x+1})}{assoc(R_{x+1})}$$

From Lemma 2 and 3, we have

$$\begin{aligned}
\frac{inter(R_y, V - R_y)}{assoc(R_y)} &\leq \frac{inter(R'_y, V - R'_y) + inter(A_{x+1}, V - R_{x+1})}{assoc(R_y)} \\
&= \frac{inter(R'_y, V - R'_y) + inter(R_{x+1}, V - R_{x+1})}{assoc(R'_y) + assoc(R_{x+1})} \\
&\leq \frac{inter(R'_y, V - R'_y)}{assoc(R'_y)} + \frac{inter(R_{x+1}, V - R_{x+1})}{assoc(R_{x+1})} \\
&= \frac{inter(R'_y, V - R'_y)}{assoc(R'_y)} + \frac{inter(R_{x+1}, V - R_{x+1})}{assoc(R_{x+1})}
\end{aligned}$$

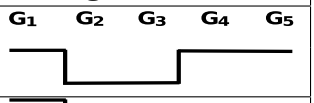
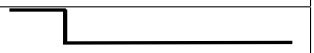
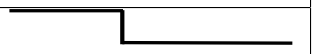
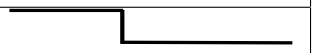
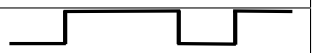
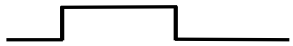
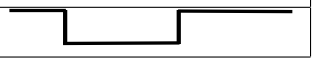
Region	Edge Set	Change Waveform
A	$\{e_{1-6}, e_{1-7}\}$	
B	$\{e_{1-2}, e_{1-3}\}$	
C	$\{e_{2-4}\}$	
D	$\{e_{3-5}\}$	
E1	$\{e_{12-13}, e_{13-14}\}$	
E2	$\{e_{15-16}\}$	
F	$\{e_{8-9}, e_{9-10}, e_{9-11}\}$	

Table 1: Regions of correlated spatio-temporal change and their associated change waveforms of the dynamic graph from Figure 1.

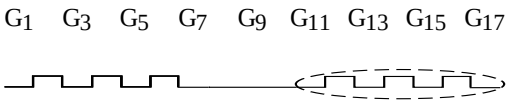
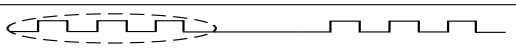
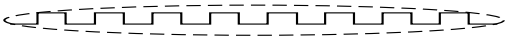
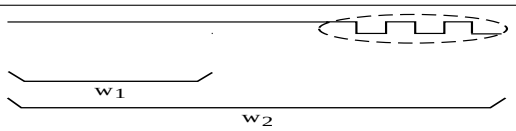
Region	Edge Set	Change Waveform
A1	$\{e_{1-3}, e_{2-3}, e_{2-6}, e_{3-5}, e_{3-4}, e_{4-11}\}$	
A2	$\{e_{1-3}, e_{2-3}, e_{2-6}, e_{3-5}, e_{3-4}, e_{4-11}\}$	
B	$\{e_{2-12}, e_{5-12}, e_{11-12}, e_{12-13}\}$	
C	$\{e_{6-8}, e_{7-8}, e_{8-9}, e_{8-10}\}$	

Table 2: Regions of correlated change and their associated change waveforms of the graph from Figure 2b. The dashed circle of each waveform illustrate the subsequence over which the respective regions of Figure 2 are defined. Two example window lengths, ω_1 and ω_2, are provided to highlight that no single window length can adequately find the correct regions.

Symbol	Description
$G(V_G, E_G)$	A graph, with vertex and edge set V_G and E_G.
$W^{ts,te}(W_k)$	A subsequence of snapshots $< G_{ts}, \ldots, G_{te} >$.
$E_C^{ts,te}$ (E_C^k)	Set of changed edges over $W^{ts,te}$ (W_k).
$q^{ts,te}(e_i)$	The change waveform of edge e_i over $W^{ts,te}$.
$R_r^{ts,te}$	A region of correlated spatio-temporal change, defined over $W^{ts,te}$.
$R_r^{ts,te}.Q$	The set of (waveforms, frequency) pairs of $R_r^{ts,te}$.
$\mathbf{R}$	A set of regions of correlated spatio-temporal change.
$d_{tem}(e_i, e_j, W_k)$	The temporal distance relation between edges e_i and e_j, over W_k.
$d_{spa}(e_i, e_j, W_k)$	The spatial distance relation between edges e_i and e_j, over W_k.
ω	Window size.
η	Maximum number of snapshots to look ahead.
ψ	Maximum allowed time between changes within a region.
$stop_{tem}$	Temporal stopping threshold for graph partitioning.
$stop_{spa}$	Spatial stopping threshold for graph partitioning.

Table 3: Summary of the main symbols and parameters used in this paper.

Name	**Vertices (Edges)**	**Parameters**
exampleDS	13 (17)	Example from introduction
synGen003	497–529 (1368–1507)	minTempSep = minSpatSep = minEvtSep = 3
synGen002	315–335 (1183–1333)	minTempSep = minSpatSep = minEvtSep = 2
synGen001	124–128 (309–313)	minTempSep = minSpatSep = minEvtSep = 1
introGenSize	670–5339 (2000–16,000)	Introduce regions into scale-free graphs

Table 4: Summary of the synthetic datasets used for comparison purposes.

Algorithm	**Intra**		**Inter**		**Timing**
	Temporal	**Spatial**	**Temporal**	**Spatial**	**(secs)**
regH	0.184	**0.119**	**0.951**	0.158	1583
ld-sn	**0.138**	0.145	0.810	**0.168**	296

Table 5: Results for the best intra-region distance, 1998 World Cup website access graph. Lower intra-region and higher inter-region distances are more accurate.

Algorithm	**Intra**		**Inter**		**Timing**
	Temporal	**Spatial**	**Temporal**	**Spatial**	**(secs)**
regH	**0.299**	0.164	**0.962**	0.167	513
ld-sn	0.442	**0.145**	0.932	**0.168**	1148

Table 6: Results for the best inter-region distance, 1998 World Cup website access graph. Lower intra-region and higher inter-region distances are more accurate.

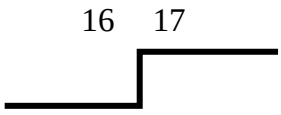
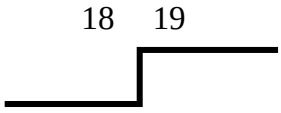
Region	No. of Edges	Change Waveform	Comments
Sat-C1	101		Region corresponding to fans that read the statistics summary of matches 8895 and 8896, but not match 8894.
Sat-C2	131		Region corresponding to fans that read the statistics summary of all matches played on Saturday (8895, 8896, 8894).

Table 7: Characteristics of the two new discovered regions of correlated change over the period 0000, June 12th to 2359, June 13th.

Object/File Name
/eng/teams/teambio160.htm
/eng/teams/teambio76.htm
/eng/teams/teambio111.htm
/eng/competition/matchstat8896.htm
/eng/competition/groupstandings163_77.htm
/eng/competition/matchstat8895.htm

Table 8: List of objects/files for region Sat-C1.

Object/File Name
/eng/teams/teambio160.htm
/eng/teams/teambio76.htm
/eng/teams/teambio111.htm
/eng/teams/teambio135.htm
/eng/competition/matchstat8894.htm
/eng/competition/matchstat8896.htm
/eng/competition/groupstandings163_77.htm
/eng/competition/matchstat8895.htm

Table 9: List of objects/files for region Sat-C2.

Algorithm	**Intra**		**Inter**		**Timing**
	Temporal	**Spatial**	**Temporal**	**Spatial**	**(secs)**
regH	**0.170**	0.405	**0.970**	0.616	5932
ld-sn	0.271	**0.397**	0.963	**0.621**	34402

Table 10: Results for the best intra-region distance, for the effect of Hurricane Katrina on BGP connectivity graph. Lower intra-region and higher inter-region distances are more accurate.

Algorithm	**Intra**		**Inter**		**Timing**
	Temporal	**Spatial**	**Temporal**	**Spatial**	**(secs)**
regH	**0.184**	**0.353**	**0.973**	0.586	6555
ld-sn	0.469	0.514	0.963	**0.647**	35209

Table 11: Results for the best inter-region distance, for the effect of Hurricane Katrina on BGP connectivity graph. Lower intra-region and higher inter-region distances are more accurate.

List of Figures

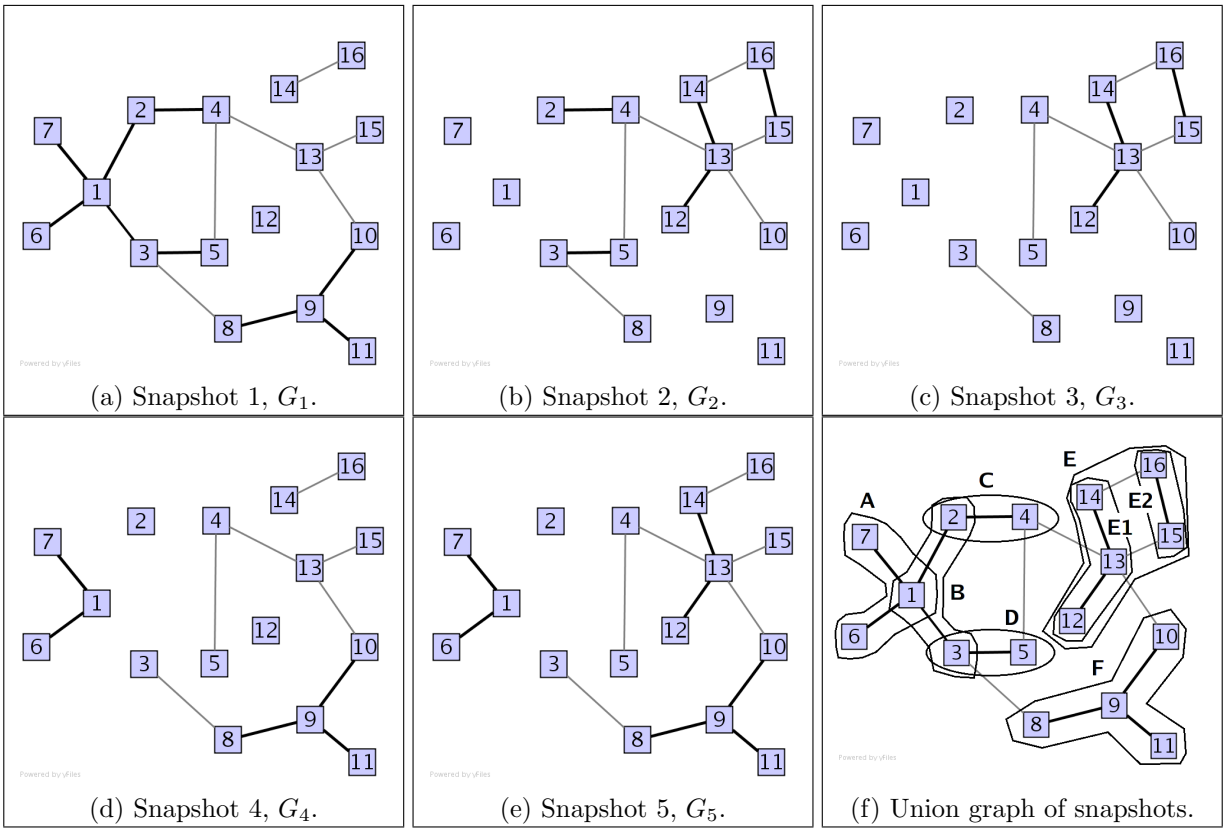
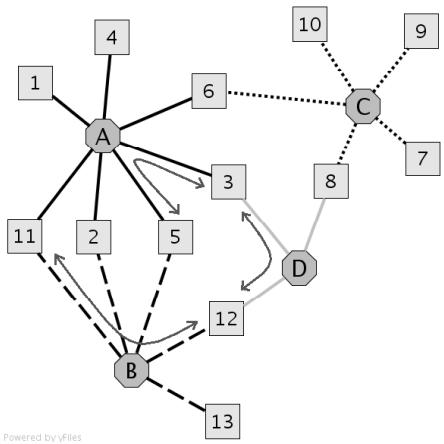
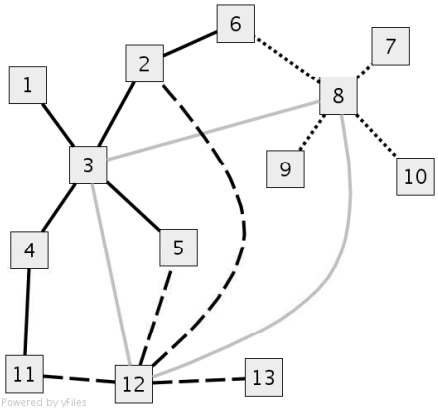


Figure 1: An example of a dynamic graph with five snapshots. Bold edges highlight edges that have experienced change in the five snapshots. The changed edges belonging to each region are circled in Figure 1f.

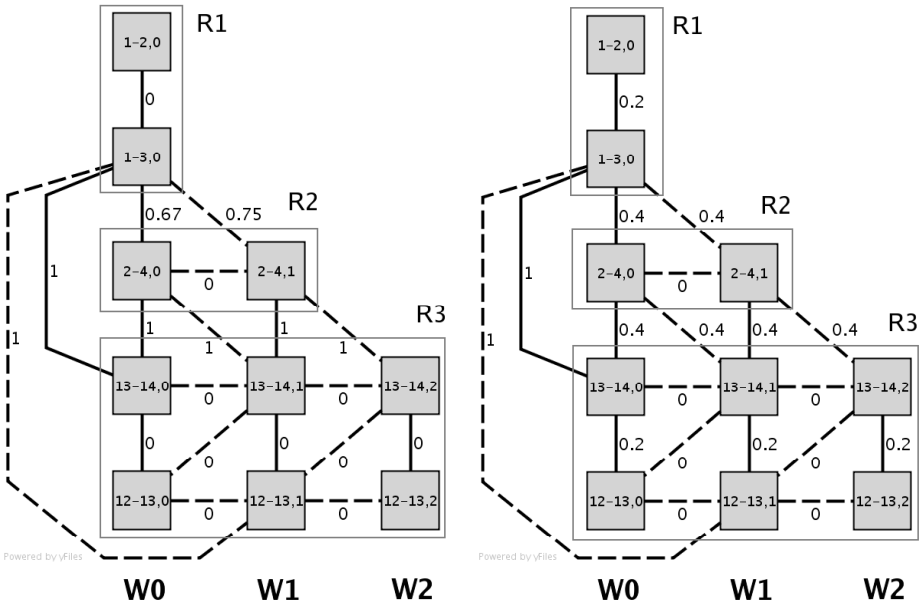


(a) Physical layer graph. The hexagons represent routers, and squares represent hosts. For ease of reference, all links connected to the same router have the same line style. Red, double headed arrowed lines are examples of the IP connections displayed in Figure 2b.



(b) Connection layer graph. Subgraph A is drawn with solid edges, subgraph B with dashed edges, and subgraph C with dotted edges. Gray coloured edges represent edges that have not experienced change.

Figure 2: A two layered evolving graph with changes in routers A, B, and C in the physical layer inducing changes in subgraphs A, B, C in the connection layer, respectively.



(a) Temporal. (b) Spatial.

Figure 3: Evolving dissimilarity graphs for some of the changed edges in the dynamic graph given in Figure 1. Solid and dotted edges represent intra-window and inter-window distances respectively. For clarity, not all intra/inter window edges are shown. These graphs were generated using a window size ω of 3, and window increment of 1. There are three actual regions, labeled R_1, R_2, and R_3. Note how across region boundaries, temporal and/or spatial distances are high, but within a region, they are generally low.

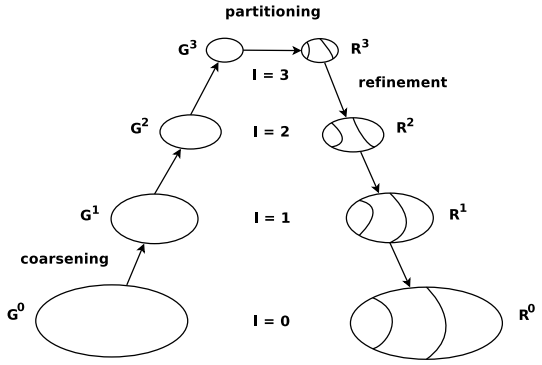


Figure 4: Illustration of the general multi-level graph partitioning process. The initial graph is labeled G^0, and the final partitions labeled R^0. There are four levels in this example (l = 0 to 3).

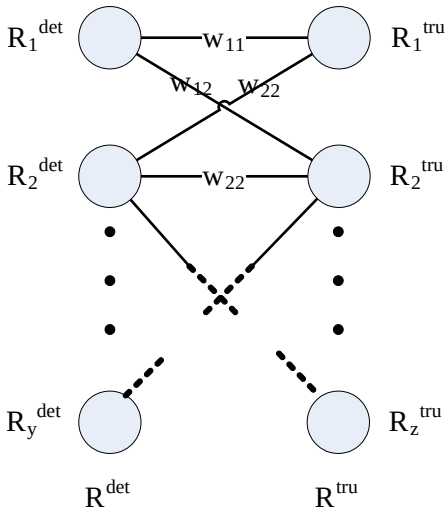
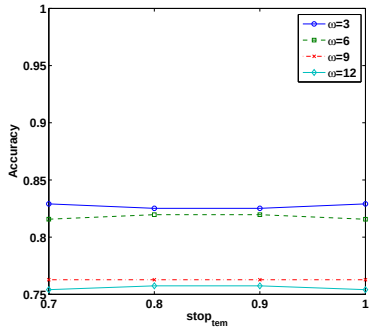
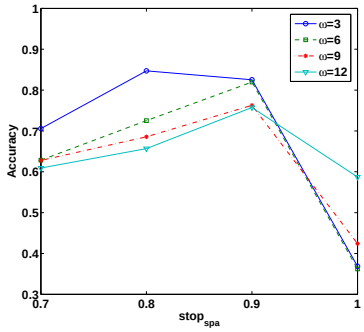


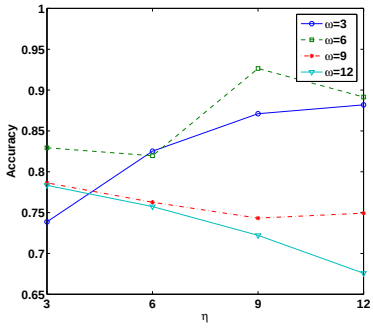
Figure 5: The general region comparison problem.



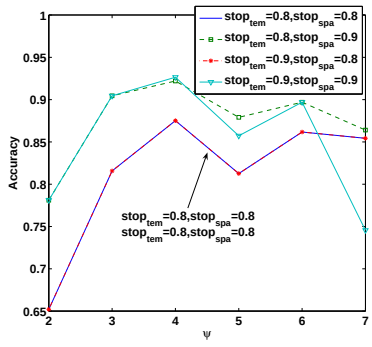
(a) $stop_{tem}$ vs accuracy. $stop_{spa} = 0.9$, $\eta = 6$, $\psi = 4$.



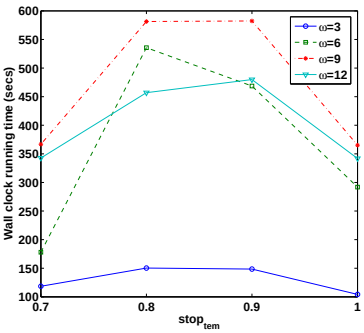
(b) $stop_{spa}$ vs accuracy. $stop_{tem} = 0.9$, $\eta = 6$, $\psi = 4$.



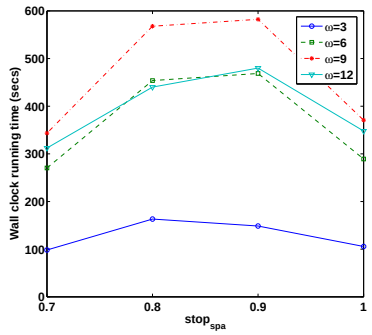
(c) η vs accuracy. $stop_{spa} = 0.9$, $stop_{tem} = 0.9$, $\psi = 4$.



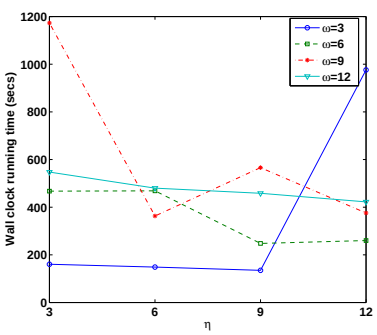
(d) ψ vs accuracy. $\omega = 6$, $\eta = 9$.



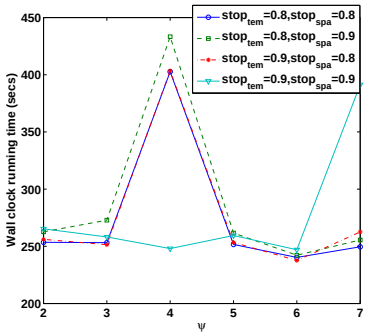
(e) $stop_{tem}$ vs accuracy. $stop_{spa} = 0.9$, $\eta = 6$, $\psi = 4$.



(f) $stop_{spa}$ vs accuracy. $stop_{tem} = 0.9$, $\eta = 6$, $\psi = 4$

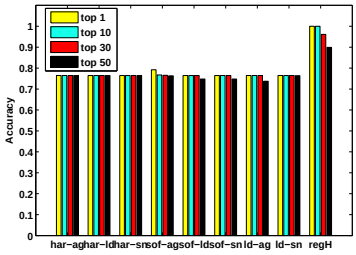


(g) η vs accuracy. $stop_{spa} = 0.9$, $stop_{tem} = 0.9$, $\psi = 4$

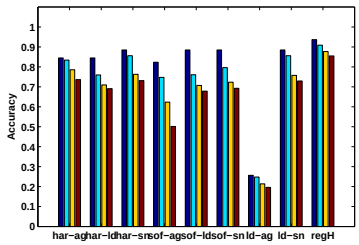


(h) ψ vs accuracy. $\omega = 6$, $\eta = 9$.

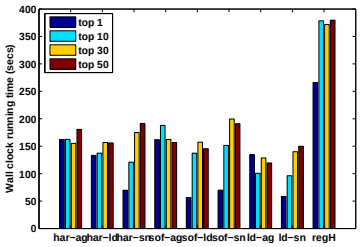
Figure 6: Parameter sensitivity evaluation results. Datasets used had the parameters **minTempSep** = 3, **minSpatSep** = 3 and **minEvtSep** = 3.



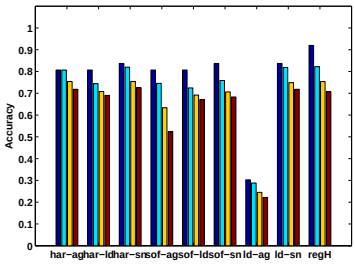
(a) Accuracy comparison for the *exampleDS* dataset.



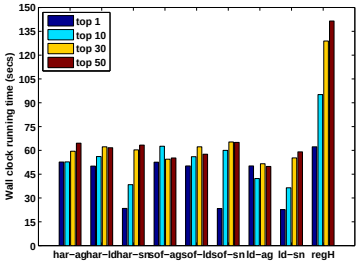
(b) Accuracy comparison for the *synGen003* datasets.



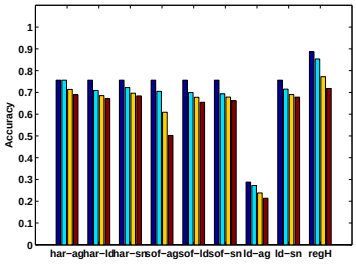
(c) Running time comparison for the *synGen003* datasets.



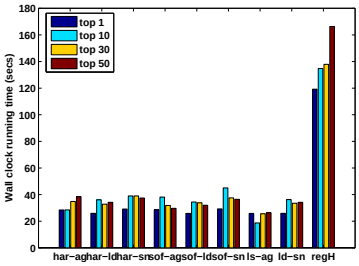
(d) Accuracy comparison for the *synGen002* datasets.



(e) Running time comparison for the *synGen002* datasets.



(f) Accuracy comparison for the *synGen001* datasets.



(g) Running time comparison for the *synGen001* datasets.

Figure 7: Accuracy and timing comparison of regHunter (*regH*) vs. algorithms of cSTAG. The algorithms labeled *har-ag, har-ld, har-sn* is the hard approach with the {averageLinkage, leaderFollower, singleLinkage} methods; *sof-ag, sof-ld, sof-sn* is the soft approach with the any of {averageLinkage, leaderFollower, singleLinkage} methods; *ld-ag, ld-sn* is the sequential approach with {leaderFollower + singleLinkage, leaderFollower + averageLinkage}; and *regH* is regHunter.

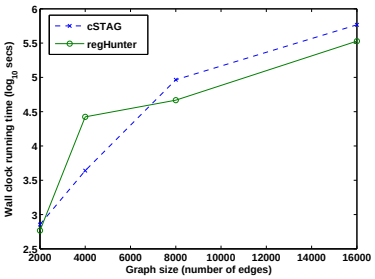
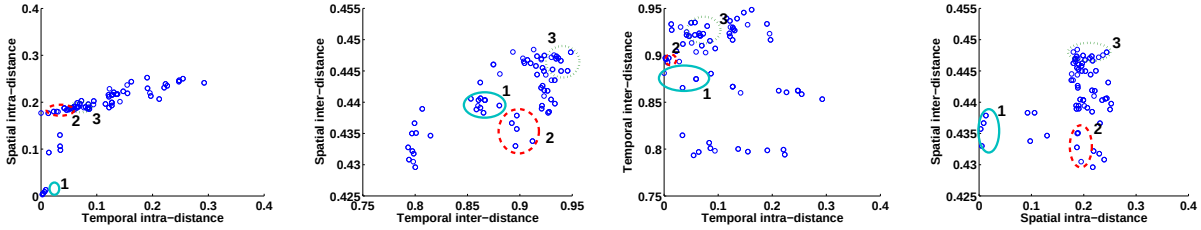


Figure 8: Timing comparison of regHunter against cSTAG, for *introGenSize* datasets.



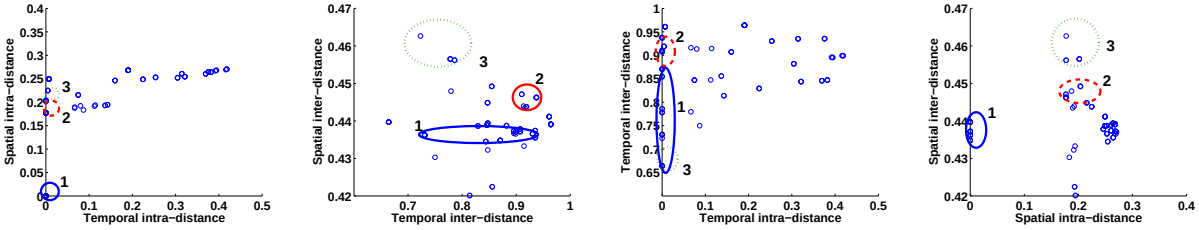
(a) Temporal intra-region distance vs. Spatial intra-region distance.

(b) Temporal inter-region distance vs. Spatial inter-region distance.

(c) Temporal intra-region distance vs. Temporal inter-region distance.

(d) Spatial inter-region distance vs. Spatial inter-region distance.

Figure 9: The temporal and spatial intra-region and inter-region validation results, for regHunter, *synGen003* dataset. Results with low intra-region distances are labeled with 1. Results with high inter-region distances are labeled with 3. Results with the most accurate *extRegCompare* scores are labeled as 2.



(a) Temporal intra-region distance vs. Spatial intra-region distance.

(b) Temporal inter-region distance vs. Spatial inter-region distance.

(c) Temporal intra-region distance vs. Temporal inter-region distance.

(d) Spatial inter-region distance vs. Spatial inter-region distance.

Figure 10: The temporal and spatial intra-region and inter-region validation results, for *seqComb-leaderFollower-singleLinkage*, *synGen003* dataset. Results with low intra-region distances are labeled with 1. Results with high inter-region distances are labeled with 3. Results with the most accurate *extRegCompare* scores are labeled as 2.

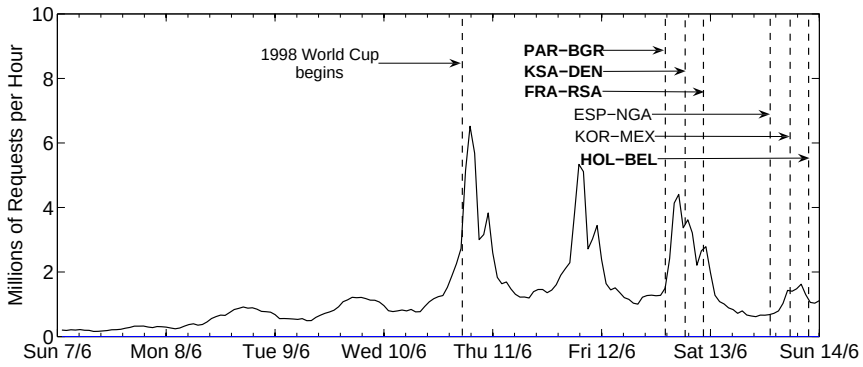


Figure 11: Hourly Traffic Volume to the 1998 World Cup Website over the period Sunday, June 7 to Saturday, June 13. Based on Figure 2 [1]).

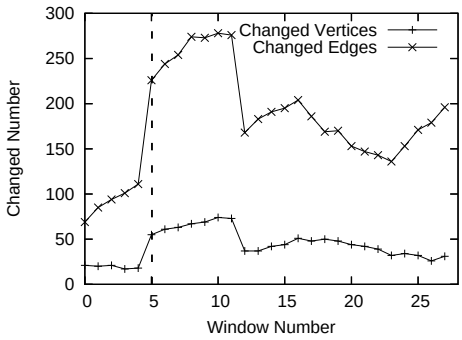


Figure 12: Number of changed edges and vertices during each time window in the US portion of the BGP graph during the period 28 August to 31 August 2005. The dotted line signifies the landfall of Hurricane Katrina.