Fast Mining of Complex Time-Stamped Events

Hanghang Tong[†] Yasushi Sakurai[‡] Tina Eliassi-Rad[§] Christos Faloutsos[†]
[†]Carnegie Mellon University, [‡]NTT Communication Science Labs, [§]Lawrence Livermore National Laboratory
[†]{htong, christos}@cs.cmu.edu, [‡]yasushi.sakurai@acm.org, [§]eliassirad1@llnl.gov

ABSTRACT

Given a collection of complex, time-stamped events, how do we find patterns and anomalies? Events could be meetings with one or more persons with one or more agenda items at zero or more locations (e.g., teleconferences), or they could be publications with authors, keywords, publishers, etc. In such settings, we want to solve the following problems: (1) find time stamps that look similar to each other and group them; (2) find anomalies; (3) provide interpretations of the clusters and anomalies by annotating them; (4) automatically find the right time-granularity in which to do analysis. Moreover, we want fast, scalable algorithms for all these problems.

We address the above challenges through two main ideas. The first (**T3**) is to turn the problem into a graph analysis problem, by carefully treating each time stamp as a node in a graph. This viewpoint brings to bear the vast machinery of graph analysis methods (PageRank, graph partitioning, proximity analysis, and CenterPiece Subgraphs, to name a few). Thus, **T3** can automatically group the time stamps into meaningful clusters and spot anomalies. Moreover, it can select representative events/persons/locations for each cluster and each anomaly, as their interpretations. The second idea (**MT3**) is to use temporal multi-resolution analysis (e.g., minutes, hours, days). We show that **MT3** can quickly derive results from finer-to-coarser resolutions, achieving up to *2 orders of magnitude* speedups. We verify the effectiveness as well as efficiency of **T3** and **MT3** on several real datasets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – Data Mining

General Terms

Algorithm, experimentation

Keywords

multi-resolution analysis, scalability, graph mining

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

In many real applications, datasets are often collected at different time stamps. At each time stamp, we might observe a set of events, where each event consists of a set of entities. Furthermore, each entity can have its own attributes. For example, in social networks, we might observe activities (events) at each day (time), where each activity involves a set of different people (entities) – each with his/her own attributes (e.g., job title). Another example is the yearly DBLP datasets, where a time stamp is 'publish year'; an event is a 'paper'; and entities are 'author,' 'conference,' etc.

How can we analyze time in such a complex context. For example, are there any two time stamps that look similar with each other? Can we find any abnormal time stamp whose behavior is very different from other time stamps? How can we interpret our findings? Furthermore, how can we do such analysis on multiple scales in an efficient way?

In this paper, we address the above challenges in multiple dimensions. First in a single scale, our method (T3) can automatically group time stamps into meaningful clusters as well as spot the abnormal stamps. For each cluster/abnormal time stamp, it also outputs the selective subsets of events/entities/attribute values as their interpretations. Here, the main idea is (1) to adopt a graph representation for the datasets at different time stamps and (2) to explore the proximity among different nodes (time/events/entities/attribute values), based on this we will find clusters and anomalies as well as their interpretations. Our experiments on several real datasets demonstrate that T3 always outputs results (i.e., clusters and anomalies as well as their interpretations) that are consistent with human intuitions. Furthermore, we propose MT3 to allow efficient analysis on multiple scales. Here, the key idea is to explore the "smoothness" (i.e., redundancy) among different scales. Our experiments show that MT3 leads to exactly the same results (i.e, *no quality loss*), but achieves significant speed-ups (up to *2 orders of magnitude*).

The main contributions of the paper are summarized as follows:

- A generic framework (T3) to mine time in complex context
- An efficient algorithm (MT3) for multiple scale analysis
- Power of our approach illustrated by extensive experiments on several real datasets

The rest of the paper is organized as follows. We begin in Section 2 with the formal problem definition. We present T3 for the single scale analysis and MT3 for the multiple scale analysis in Section 3 and Section 4, respectively. The experimental results are reported in Section 5. We review the related work in Section 6 and conclude the paper in Section 7.

Table 1: Symbols

Symbol	Definition and Description
$\mathcal{O}^1$	the 'time' object: $\mathcal{O}^1 = \{t_1, ..., t_{n_1}\}$
$\mathcal{O}^2$	the 'event' object: $\mathcal{O}^2 = \{e_1, ..., e_{n_2}\}$
$\mathcal{O}^x$	the $(x-2)^{\text{th}}$ 'entity' object: $\mathcal{O}^x = \{b_1^{(x-2)}, ..., b_{n_x}^{(x-2)}\}, (x = 3, ..., 2+p)$
$\mathcal{O}^y$	the $(y-2-p)^{\text{th}}$ 'attribute' object: $\mathcal{O}^y = \{a_1^{(y-2-p)}, ..., a_{n_y}^{(y-2-p)}\}, (y = 3+p, ..., 2+p+q)$
$\mathbf{W}^{x,y}$	the adjacency matrix ($n_x \times n_y$) from the x^{th} object to the y^{th} object ($x, y = 1, ..., 2+p+q$)
$\mathbf{D}^{x,y}$	the degree matrix: $\mathbf{D}^{x,y}(i,i) = \sum_j \mathbf{W}^{x,y}(i,j)$ and $\mathbf{D}^{x,y}(i,j) = 0 (i \neq j)$
$\mathbf{W} = [\mathbf{W}^{x,y}]$	the overall adjacency matrix ($n \times n$)
$\mathbf{0}$	a matrix with all elements equal to 0
$\mathbf{I}$	an identity matrix
p	the number of different types of entities
q	the number of different types of attributes
n_x	the number of instances for the x^{th} type of object ($x = 1, ..., 2+p+q$)
n	the number of total instances ($n = \sum_{x=1}^{2+p+q} n_x$)
s_x	the number of objects connected to the x^{th} type of object
z	the number of clusters for time stamps
$r_{i,j}$	the proximity score from node j to node i
c	$(1-c)$ is the restart probability for random walk with restart ($c = 0.95$ in this paper.)
$\mathbf{ttP} = [r_{i,j}]$	the time-to-time proximity matrix ($n_1 \times n_1$, and $i, j = 1, ..., n_1$)
$\mathbf{toP} = [r_{i,j}]$	the time-to-others proximity matrix ($(n-n_1) \times n_1$, and $i = 1, ..., n-n_1, j = 1, ..., n_1$)
$\mathbf{f}$	the aggregation function ($n_1 \times 1$ vector)
$\mathbf{g}$	the cluster membership function ($n_1 \times 1$ vector)

2. PROBLEM DEFINITION

In this section, we first introduce our notations and data representation, and then give the formal problem definitions.

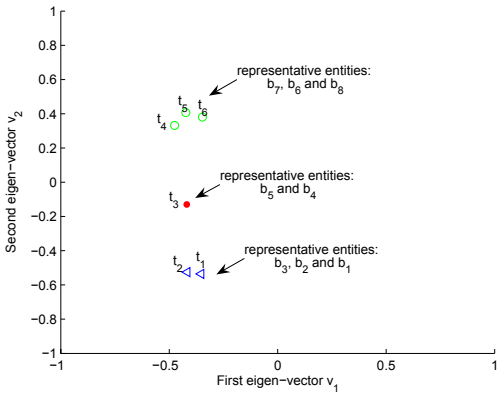
Table 1 lists the main symbols we use throughout this paper. Following standard notation, we use calligraphic letter for sets (e.g., $\mathcal{O}^1$ is the set of all time stamps), capital bolded letters for matrices (e.g., $\mathbf{W}$), and lower case bolded letters for vectors (e.g., $\mathbf{g}$). We denote the transpose with a prime (i.e., $\mathbf{W}'$ is the transpose of $\mathbf{W}$), and we use superscripts to denote the indices for object types (e.g., $\mathcal{O}^s$ is the s^{th} type of object) and the indices for block matrices (e.g., $\mathbf{W}^{x,y}$ is a block matrix of the matrix $\mathbf{W}$). For matrix/vector, we use the subscript to represent the size of the matrix/vector (e.g. $\mathbf{0}_{k \times l}$ means a matrix of size $k \times l$, whose elements are all zero). If the size of a matrix/vector is clear from the context, we omit such subscripts. Also, we represent the elements in a matrix using a convention similar to Matlab, e.g., $\mathbf{W}(i,j)$ is the element at the i^{th} row and j^{th} column of the matrix $\mathbf{W}$, and $\mathbf{W}(i,:)$ is the i^{th} row of $\mathbf{W}$, etc.

In our setting, the datasets are collected at different time stamps. At each time stamp, we observe a set of events, where each event consists of a set of entities. Furthermore, each entity may or may not have its own attributes. For example, in the running example in Table 2(a), we observe 9 events ($e_1, ..., e_9$), each of which is a social event (e.g., e_1 is a 'technical meeting', e_2 is a 'football game', etc). The events are spreaded among 6 time stamps ($t_1, ..., t_6$), each of which is a day (e.g., t_1 is 'Monday', t_2 is 'Tuesday', etc). Furthermore, each event involves 2 entities ($b_1, ..., b_8$), each of which is a person (e.g., b_1 is 'John', b_2 is 'Smith', etc) .

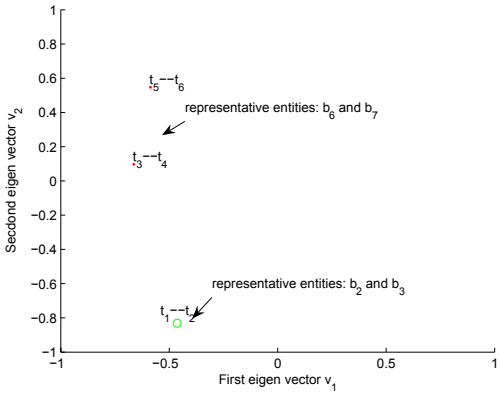
To simplify the description, we refer to 'time', 'event', each type of 'entity', and each 'attribute' as one type of object, respectively. If we have p types of entities (in the running example, $p = 1$), and q types of attributes (in the running example, $q = 0$), we define the following object set $\mathcal{O}^x (x = 1, ..., 2+p+q)$, where the first type of object is always 'time'; the second type of object is always 'event'; each of the next p objects is one type of 'entity'; and each of the next q objects is one type of 'attribute'. For the running example in Table 2(a), we have 3 types of objects in the object set $\mathcal{O}^x (x = 1, 2, 3)$. They are 'time', 'event', and 'entity', respectively. (There is no 'attribute' in this example.) Each object type has a set of instances. For example, the instances for the 'time' object ($\mathcal{O}^1$) are different time stamps (e.g., $t_1, t_2, ...$).

In this paper, we use a graph representation for the whole dataset covering all time stamps. To be specific, we treat each instance for each type of object as a node in the graph. For example, Table 2(b) gives the graph representation for the original time-stamped datasets (depicted in Table 2(a)) – where each time stamp, each event instance, and each entity instance is represented as a single node in the graph. Furthermore, the relationship between different types of objects are modeled by the adjacency matrices ($\mathbf{W}^{x,y} (x, y = 1, ..., 2+p+q)$). For example, we can use $\mathbf{W}^{1,2}$ to model the relationship between the 'time' object and 'event' object, where $\mathbf{W}^{1,2}(i,j) = 1$ iff the j^{th} event happens at the i^{th} time stamp; $\mathbf{W}^{1,2}(i,j) = 0$ otherwise. Similarly, we can use $\mathbf{W}^{2,2+x} (x = 1, .., p)$ to model the relationship between the 'event' object and the x^{th} 'entity' object, where $\mathbf{W}^{2,2+x}(i,j) = 1$ iff the i^{th} event involves the j^{th} instance of the x^{th} type of entity; $\mathbf{W}^{2,2+x}(i,j) = 0$ otherwise. We can use $\mathbf{W}^{2+x,2+p+y} (x = 1, ..., p, y = 1, ..., q)$ to model the relationship between the x^{th} type of 'entity' object the y^{th} type of 'attribute' object, where $\mathbf{W}^{2+x,2+p+y}(i,j) = 1$ iff the i^{th} instance of the x^{th} type of 'entity' has the j^{th} attribute value of the y^{th} type of 'attribute'; $\mathbf{W}^{2+x,2+p+y}(i,j) = 0$ otherwise. For the running example, two such adjacency matrices ($\mathbf{W}^{1,2}$ and $\mathbf{W}^{2,3}$) are enough to model all the relationships (see Table 2(c)).

If we always reserve the first n_1 rows/columns for the time nodes; the next n_2 rows/columns for the event nodes; followed by rows/columns for entity nodes and attribute nodes respectively; we can define $\mathbf{W} = \mathbf{W}^{x,y} (x, y = 1, ..., 2+p+q)$ as the overall adjacency matrix for the whole graph. Note that if there is no relationship between the x^{th} and the y^{th} objects, the corresponding block matrix $\mathbf{W}^{x,y} = \mathbf{0}$. Also, by this notation, we allow additional relationship



(a) The finest scale



(b) The aggregated scale (by every two time stamps)

Figure 1: The outputs for the running example in Table 2.

within the same type of object. For example, if we want to consider the continuous property of time, we can put extra links between consecutive time nodes, which will lead to a non-zero block matrix $\mathbf{W}^{1,1}$. For the running example in Table 2, its overall adjacency matrix $\mathbf{W}$ has the following format (Eq. (1)):

$$\mathbf{W} = \begin{pmatrix} \mathbf{0} & \mathbf{W}^{1,2} & \mathbf{0} \\ (\mathbf{W}^{1,2})' & \mathbf{0} & \mathbf{W}^{2,3} \\ \mathbf{0} & (\mathbf{W}^{2,3})' & \mathbf{0} \end{pmatrix} \quad (1)$$

With the above notation, our datasets can be denoted by the object set $\mathcal{O}^x (x = 1, ..., 2+p+q)$ together with the overall adjacency matrix $\mathbf{W}$. Our goal is to find (1) similar/anomalous time stamps and (2) their interpretations. In this paper, we define an anomalous time stamp as a special time cluster, which contains a single time stamp. Therefore, we define the cluster membership function $\mathbf{g}$ as an $n_1 \times 1$ vector, and each element in $\mathbf{g}$ as an integer between 1 and z (z is the cluster number for time stamps), indicating to which cluster it belongs. To provide an interpretation for each time cluster, we want to select a representative subset of instances from each type of object (except 'time' object). Thus, our problem (*The Single Scale Analysis*) can be formally defined as follows:

PROBLEM 1. The Single Scale Analysis

Given: *The datasets collected at different time stamps:* $\{\mathcal{O}^x, \mathbf{W}\}(x = 1, .., 2 + p + q)$.

Find: *(i) The cluster membership function* $\mathbf{g}$ *for time stamps (as well as the cluster number* z*); and (ii) for each time cluster, a representative subset of instances from each type of object (except 'time' object).*

For example, Fig. 1(a) shows the output of the proposed T3 (for the single scale analysis) applied to the datasets we list in Table 2, where we find 2 clusters of time stamps ($\{t_1, t_2\}$ and $\{t_4, t_5, t_6\}$) and 1 abnormal time stamp (t_3). Therefore, our cluster membership function satisfies: $\mathbf{g} = [1,\ 1,\ 3,\ 2,\ 2,\ 2]'$. For each time cluster as well as the abnormal time stamp, we also output a representative subset of the entity nodes as its interpretations.[1]

Besides the finest scale, we might also want to do the same analysis (i.e., to find the time cluster/anomaly as well as their interpretations) on some coarser scale. To this end, we introduce the aggregation function $\mathbf{f}$, which is an $n_1 \times 1$ vector. For example, if we aggregate the time by every two time stamps for the datasets in Table 2, the aggregation function $\vec{u}$ is a 6×1 vector: $\mathbf{f} = [\mathbf{1},\ \mathbf{1},\ \mathbf{2},\ \mathbf{2},\ \mathbf{3},\ \mathbf{3}]'$. Also, let $\tilde{\mathbf{g}}$ be the cluster membership function and $\tilde{z}$ be the cluster number at the aggregated scale, respectively. With this notation, our problem (*The Multiple Scale Analysis*) can be formally defined as follows:

PROBLEM 2. The Multiple Scale Analysis

Given: *(i) The datasets collected at different time stamps:* $\{\mathcal{O}^x, \mathbf{W}\}(x = 1, .., 2 + p + q)$*; and (ii) the aggregation function* $\mathbf{f}$.

Find: *(i) The cluster membership function* $\tilde{\mathbf{g}}$ *for time stamps (as well as the cluster number* $\tilde{z}$*); and (ii) for each time cluster at aggregated scale, a representative subset of instances from each type of object (except 'time' object).*

For example, Fig. 1(b) shows the output of the proposed MT3 applied to the datasets in Table 2 if we aggregate the time by every two time stamps. Notice that in this case, the abnormal time stamp (i.e., t_3 at the finest scale) disappears.

3. T3 FOR SINGLE SCALE ANALYSIS

In this section, we propose T3 to address problem 1. We first give an overview of the proposed algorithm (T3), and then introduce each component of T3 in detail.

3.1 Overview of T3

Alg. 1 gives the overview of the proposed T3 for single scale analysis. In T3, we first construct the graph representation $\mathbf{W}$ from the original raw datasets as introduced in Section 2 (step 1). Then (step 2), we will compute two proximity matrices from the adjacency matrix $\mathbf{W}$: the time-to-time proximity matrix ($\mathbf{ttP}$) and the time-to-others proximity matrix ($\mathbf{toP}$). The time-to-time proximity matrix ($\mathbf{ttP}$) will be used to find the time cluster membership function $\mathbf{g}$ (step 3); while the time-to-others proximity matrix ($\mathbf{toP}$) will be used to find the representative subset of instances as the interpretations for time cluster (step 4).

3.2 Compute the Proximity matrices

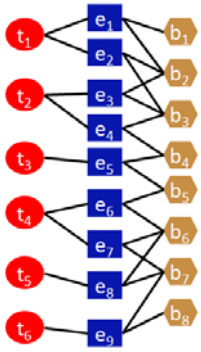
The key point in T3 is to construct two proximity matrices ($\mathbf{ttP}$ and $\mathbf{toP}$), based on which we will find the time cluster membership function $\mathbf{g}$ and its interpretations, respectively.

Alg. 2 lists detailed procedures to compute these two proximity matrices. Overall, we adopt the well-studied model of random

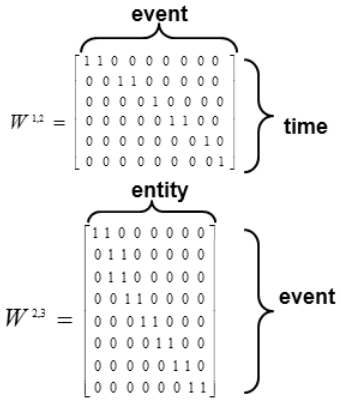
[1]For the sake of simplicity, the representative events are not shown in the figure.

Time Steps	Events	Entities
t_1	e_1	b_1, b_2
	e_2	b_2, b_3
t_2	e_3	b_2, b_3
	e_4	b_3, b_4
t_3	e_5	b_4, b_5
t_4	e_6	b_5, b_6
	e_7	b_6, b_7
t_5	e_8	b_6, b_7
t_6	e_9	b_7, b_8

(a) Original datasets



(b) Graph representation



(c) Adjacency matrices

Table 2: A running example: notations and representation illustration.

Algorithm 1 Overview of T3

1: construct the graph $\mathbf{W}$ from the raw datasets
2: compute the proximity matrices $\mathbf{ttP}$ and $\mathbf{toP}$
3: find time cluster membership function $\mathbf{g}$ based on $\mathbf{ttP}$
4: find the interpretation for each time cluster based on $\mathbf{toP}$

walk with restart [19, 27, 33] for this purpose (steps 7-12). Suppose a random particle starts from the time node j, the particle iteratively transmits to its neighborhood with the probability that is proportional to the edge weight between them; and also at each step, it has some probability $(1-c)$ to return to the starting node j. The proximity score $r_{i,j}$ is defined as the steady-state probability that the particle will finally stay at node i. A subtle point in computing the proximity matrices is how to normalize the original adjacency matrix $\mathbf{W}$. In Alg. 2, we propose to normalize it by object type (steps 1-7). That is, suppose the random particle stays at some node of type x and overall there are s_x different types of objects connected to the x^{th} type of object; then at the next step, the particle will have equal chance $(\frac{1}{s_x})$ to jump to each of s_x types of objects.

Algorithm 2 Compute the Proximity Matrices $\mathbf{ttP}$ and $\mathbf{toP}$

Input: the adjacency matrix $\mathbf{W}$ and c
Output: the proximity matrices $\mathbf{ttP}$ and $\mathbf{toP}$

1: **for** $x = 1 : 2 + p + q$ **do**
2: **for** $y = 1 : 2 + p + q$ **do**
3: normalize by object type: $\mathbf{W}^{x,y} \leftarrow \frac{1}{s_x} \cdot (\mathbf{D}^{x,y})^{-1} \cdot \mathbf{W}^{x,y}$
4: **end for**
5: **end for**
6: set $\mathbf{W} \leftarrow [\mathbf{W}^{x,y}]$
7: **for** $j = 1 : n_1$ **do**
8: let $\mathbf{e} = \mathbf{0}_{n \times 1}$; then set $\mathbf{e}(j) = 1$
9: solve $\mathbf{r}$ from the equation $\mathbf{r} = c\mathbf{W}'\mathbf{r} + (1-c)\mathbf{e}$
10: set $\mathbf{ttP}(:, j) = \mathbf{r}(1 : n_1)$
11: set $\mathbf{toP}(:, j) = \mathbf{r}(n_1 + 1 : n)$
12: **end for**

3.3 Find Time Cluster $\mathbf{g}$

Here, we want to find the cluster membership function $\mathbf{g}$ for time stamps based on the time-to-time proximity matrix $\mathbf{ttP}$. The algorithm is listed in Alg. 3. We use a spectral clustering algorithm.[2] In Alg. 3, we first use the eigen-gap [9] (step 2) to choose cluster number z. Then, we treat the first z eigen vectors as the embedding of the time nodes in the z-dimensional space (steps 3-4) and run k-means to find the final cluster membership function $\mathbf{g}$ (step 5).

Algorithm 3 Find the Time Cluster

Input: the time-to-time proximity matrix $\mathbf{ttP}$
Output: the cluster membership function $\mathbf{g}$

1: do eigen value decomposition for $\mathbf{ttP}$; let $\{\lambda_1, ..., \lambda_{n_1}\}$ be the eigen values for $\mathbf{ttP}$ (from largest to smallest) and $\{\mathbf{v}_1, ..., \mathbf{v}_{n_1}\}$ be the corresponding eigen vectors
2: find the cluster number $z = \text{argmax}_i(\lambda_{i-1} - \lambda_i)$
3: let $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_z]$
4: treat each row of $\mathbf{V}$ as a data point in z-dimensional space
5: use k-means to find z clusters on $\mathbf{V}$ and output the corresponding cluster membership function $\mathbf{g}$

As mentioned before, if we find some cluster which contains a single time stamp, we flag it as the abnormal time stamp.

One benefit of using spectral clustering method is that we can use the first few eigen vectors as the embedding of the time stamps in some low dimensional space. For example, we can visualize the time stamps by plotting its first two eigen vectors in Fig. 1 for the running example.

3.4 Find Interpretations for Time Clusters

For each time cluster, we want to select a representative subset of instance nodes from each type of object (except the 'time' object) as the interpretations for that time cluster.

Suppose we want to find the interpretations for the time cluster u $(u = 1, ..., z)$. Let $\bar{r}(j, u)$ be the average proximity score from the time cluster u to the instance node j:

$$\bar{r}(j,u) = \frac{\sum_{i=1}^{n_1} \mathrm{I}(\mathbf{g}(i) = u)\mathbf{toP}(j,i)}{\sum_{i=1}^{n_1} \mathrm{I}(\mathbf{g}(i) = u)} \quad (2)$$

[2]Notice that our framework is orthogonal to the specific clustering methods. We can plug in any clustering algorithm that takes a proximity matrix between nodes as input. For example, we could transfer the time-to-time proximity matrix $\mathbf{ttP}$ to be the normalized graph Laplacian and find its eigen-decomposition instead (step 1). Alternatively, we can normalize each row of $\mathbf{V}$ to have the unit length in step 3 as suggested in [25].

where I(.) is an indicator function, which is 1 if the condition in the parenthesis is true and 0 otherwise.

Based on $\bar{r}(j,u)$, we can define the representative score $r(j,u)$ for each instance node j w.r.t. the given time cluster u as follows:

$$r(j,u) = \bar{r}(j,u) \prod_{w=1,w\neq u}^{z} (1-\bar{r}(j,w)) \quad (3)$$

The intuition of Eq. (3) is that we want to find the node j which is close to the time cluster u (higher $\bar{r}(j,u)$ is better) and far away from other time clusters (lower $\bar{r}(j,w)(w \neq u)$ is better) on average. Finally, we can output a subset of instance nodes with high representative scores $r(j,u)$ from each type of object as the interpretations for the time cluster u.

4. MT3 FOR MULTIPLE SCALE ANALYSIS

In this section, we propose MT3 to address problem 2. Conceptually, we can apply T3 for each scale of interest independently. Here, the challenge is to make the analysis on the coarser scales as efficient as possible, given that we have already done the analysis at the finest scale.

Algorithm 4 Update the Proximity Matrices

Input: the proximity matrices $\mathbf{ttP}$ and $\mathbf{toP}$, the normalized adjacency matrix $\mathbf{W}$ at the finest scale, the aggregation function $\mathbf{f}$ and c;

Output: the proximity matrices $\mathbf{t\tilde{t}P}$ and $\mathbf{to\tilde{P}}$ at the aggregated scale.

1: set up the normalized adjacency matrix $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{x,y}]$ at the aggregated scale.
2: initialize the transformation matrices: $\mathbf{T}_1 = \mathbf{0}_{\tilde{n}_1 \times n_1}$, and $\mathbf{T}_2 = \mathbf{0}_{n_1 \times \tilde{n}_1}$
3: **for** $\tilde{i} = 1 : \tilde{n}_1$ **do**
4: find time stamps at the finest scale: $\mathcal{J} = \{i : \mathbf{g}(i) = \tilde{i}\}$
5: **for** each $i \in \mathcal{J}$ **do**
6: set $\mathbf{T}_1(\tilde{i}, i) = \mathbf{h}(i) / \sum_{i\in\mathcal{J}} \mathbf{h}(i)$
7: set $\mathbf{T}_2(i, \tilde{i}) = 1$
8: **end for**
9: **end for**
10: set $\mathbf{\Lambda} = \mathbf{I}_{n_1\times n_1} - c\mathbf{W}'_{1,1} - (1-c)(\mathbf{ttP})^{-1}$
11: update $\mathbf{t\tilde{t}P} = (1-c)(\mathbf{I}_{\tilde{n}_1\times\tilde{n}_1} - c\tilde{\mathbf{W}}'_{1,1} - \mathbf{T}'_2\mathbf{\Lambda}\mathbf{T}'_1)^{-1}$
12: update $\mathbf{to\tilde{P}} = \mathbf{toP}(\mathbf{ttP})^{-1}\mathbf{T}'_1\mathbf{t\tilde{t}P}$

In Alg. 1, the computational bottleneck lies in step 2 – i.e., to compute the two proximity matrices $\mathbf{ttP}$ and $\mathbf{toP}$. For example, our experiments show that the time for this step usually accounts for more than 95% of the overall running time of the algorithm. Therefore, our goal in *Multiple Scale Analysis* is to efficiently update these two proximity matrices ($\mathbf{t\tilde{t}P}$ and $\mathbf{to\tilde{P}}$) at the aggregated scale, given that we have already computed the proximity matrices ($\mathbf{ttP}$ and $\mathbf{toP}$) at the finest scale.

We introduce the following vector $\mathbf{h}_{\mathbf{n_1}\times\mathbf{1}}$, where $\mathbf{h}(i) :=$ number of event/entity/attribute nodes connected to the time node i at the finest scale. Suppose that we will have $\tilde{n}_1$ time stamps at the aggregated scale (i.e., $\tilde{n}_1 = \max(\mathbf{f})$). Alg. 4 gives the detailed procedure to update the proximity matrices. In Alg. 4, after we get the overall normalized adjacency matrix $\tilde{\mathbf{W}}$ at the aggregated scale (step 1), we set up two transformation matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ (steps 2-9). Then (steps 10-12), we need two matrix inversions (one $n_1 \times n_1$ in step 10 and one $\tilde{n}_1 \times \tilde{n}_1$ in step 11) to get the proximity matrices ($\mathbf{t\tilde{t}P}$ and $\mathbf{to\tilde{P}}$) at the aggregated scale. Note that in many real applications the number of time nodes at the finest scale is usually much smaller compared to the total nodes in the graph (i.e., $n_1 \ll n$). Typically, n_1 (the number of time nodes at the finest scale) is up to a few thousand whereas n (the total nodes in the graph) could be up to a few hundred thousand. For example, in the **DBLP** dataset, we only have about 49 among 988,947 time nodes at the finest scale. Therefore, we can efficiently update the proximity matrices at the aggregated scale by Alg. 4.

The correctness of Alg. 4 is guaranteed by the following theorem:

THEOREM 1. *The proximity matrices* $\mathbf{t\tilde{t}P}$ *and* $\mathbf{to\tilde{P}}$ *by Alg. 4 are correct. That is, they are exactly the same as we apply Alg. 2 to the adjacency matrix* $\tilde{\mathbf{W}}$.

PROOF. To simplify the description, we re-write the normalized adjacency matrix as the following 2×2 block form:

$$\mathbf{W} = \begin{pmatrix} \mathbf{A}^{1,1} & \mathbf{A}^{1,2} \\ \mathbf{A}^{2,1} & \mathbf{A}^{2,2} \end{pmatrix}, \quad \tilde{\mathbf{W}} = \begin{pmatrix} \tilde{\mathbf{A}}^{1,1} & \tilde{\mathbf{A}}^{1,2} \\ \tilde{\mathbf{A}}^{2,1} & \tilde{\mathbf{A}}^{2,2} \end{pmatrix} \quad (4)$$

where

$$\begin{aligned} \mathbf{A}^{1,1} &= \mathbf{W}^{1,1},\ \tilde{\mathbf{A}}^{1,1} = \tilde{\mathbf{W}}^{1,1} \\ \mathbf{A}^{1,2} &= [\mathbf{W}^{1,y}],\ \tilde{\mathbf{A}}^{1,2} = [\tilde{\mathbf{W}}^{1,y}]\ (y = 2, ..., 2+p+q) \\ \mathbf{A}^{2,1} &= [\mathbf{W}^{x,1}],\ \tilde{\mathbf{A}}^{2,1} = [\tilde{\mathbf{W}}^{x,1}]\ (x = 2, ..., 2+p+q) \\ \mathbf{A}^{2,2} &= [\mathbf{W}^{x,y}],\ \tilde{\mathbf{A}}^{2,2} = [\tilde{\mathbf{W}}^{x,y}]\ (x, y = 2, ..., 2+p+q) \end{aligned} \quad (5)$$

Notice that only time nodes change before/after the aggregation, we have,

$$\tilde{\mathbf{A}}^{2,2} = \mathbf{A}^{2,2} \quad (6)$$

Furthermore, we can verify the following equations hold for the two off-diagonal blocks in Eq. (4):

$$\begin{aligned} \tilde{\mathbf{A}}^{1,2} &= \mathbf{T}_1\mathbf{A}^{1,2} \\ \tilde{\mathbf{A}}^{2,1} &= \mathbf{A}^{2,1}\mathbf{T}_2 \end{aligned} \quad (7)$$

Define the following matrix inversion:

$$\begin{aligned} \mathbf{Q} &= (\mathbf{I} - c\mathbf{W})^{-1} \\ &= \begin{pmatrix} \mathbf{Q}^{1,1} & \mathbf{Q}^{1,2} \\ \mathbf{Q}^{2,1} & \mathbf{Q}^{2,2} \end{pmatrix} \\ \tilde{\mathbf{Q}} &= (\mathbf{I} - c\tilde{\mathbf{W}})^{-1} \\ &= \begin{pmatrix} \tilde{\mathbf{Q}}^{1,1} & \tilde{\mathbf{Q}}^{1,2} \\ \tilde{\mathbf{Q}}^{2,1} & \tilde{\mathbf{Q}}^{2,2} \end{pmatrix} \end{aligned} \quad (8)$$

By the property of random walk with restart [33], we have the following equations for the proximity matrices:

$$\begin{aligned} \mathbf{ttP} &= (1-c)(\mathbf{Q}^{1,1})',\ \mathbf{toP} = (1-c)(\mathbf{Q}^{1,2})' \\ \mathbf{t\tilde{t}P} &= (1-c)(\tilde{\mathbf{Q}}^{1,1})',\ \mathbf{to\tilde{P}} = (1-c)(\tilde{\mathbf{Q}}^{1,2})' \end{aligned} \quad (9)$$

Now, apply block matrix inversion lemma [28] to Eq. (8). Together with Eq. (4)-(9), we have

$$\begin{aligned} \frac{1}{1-c}(\mathbf{ttP})' &= (\mathbf{I} - c\mathbf{W}^{1,1} - c^2\mathbf{A}^{1,2}(I - \mathbf{A}^{2,2})^{-1}\mathbf{A}^{2,1})^{-1} \\ (\mathbf{toP})' &= c(\mathbf{ttP})'\mathbf{A}^{1,2}(I - \mathbf{A}^{2,2})^{-1} \\ \frac{1}{1-c}(\mathbf{t\tilde{t}P})' &= (\mathbf{I} - c\tilde{\mathbf{W}}^{1,1} - c^2\mathbf{T}_1\mathbf{A}^{1,2}(I - \mathbf{A}^{2,2})^{-1}\mathbf{A}^{2,1}\mathbf{T}_2)^{-1} \\ (\mathbf{to\tilde{P}})' &= c(\mathbf{t\tilde{t}P})'\mathbf{T}_1\mathbf{A}^{1,2}(I - \mathbf{A}^{2,2})^{-1} \end{aligned} \quad (10)$$

Table 3: Datasets used in our evaluations

Dataset name	p	q	n_1	n	m
NIPS	1	0	13	3,900	11,460
CIKM	2	1	15	3,299	10,228
DBLP	2	0	49	988,947	5,216,722
DeviceScan	2	0	294	114,540	684,276

In Eq. (10), we have four equations for four unknown variables ($\tilde{\mathbf{ttP}}$, $\tilde{\mathbf{toP}}$, $\mathbf{A}^{1,2}(I-\mathbf{A}^{2,2})^{-1}\mathbf{A}^{2,1}$, and $\mathbf{A}^{1,2}(I-\mathbf{A}^{2,2})^{-1}$). Solving this well-defined linear system, we have

$$\begin{aligned} \tilde{\mathbf{ttP}} &= (1-c)(\mathbf{I}-c\tilde{\mathbf{W}}'_{1,1}-\mathbf{T}'_2\mathbf{\Lambda}\mathbf{T}'_1)^{-1} \\ \tilde{\mathbf{toP}} &= \mathbf{toP}(\mathbf{ttP})^{-1}\mathbf{T}'_1\tilde{\mathbf{ttP}} \end{aligned} \quad (11)$$

where $\mathbf{\Lambda} = \mathbf{I} - c\mathbf{W}'_{1,1} - (1-c)(\mathbf{ttP})^{-1}$, which completes the proof of theorem 1. □

Based on Alg. 4, the complete algorithm for *Multiple Scale Analysis* is given in Alg. 5.

Algorithm 5 MT3 for Multiple Scale Analysis

Input: the proximity matrices $\mathbf{ttP}$ and $\mathbf{toP}$, the normalized adjacency matrix $\mathbf{W}$ at the finest scale, the aggregation function $\mathbf{f}$ and c

Output: (i) the cluster membership function $\tilde{\mathbf{g}}$ at the aggregated scale; and (ii) for each time cluster at aggregated scale, a representative subset of instances from each type of object (except 'time' object)

1: update the proximity matrices $\tilde{\mathbf{ttP}}$ and $\tilde{\mathbf{toP}}$ by Alg. 4
2: find the cluster membership function $\tilde{\mathbf{g}}$ by Alg. 3
3: for each time cluster $\tilde{u}$ in $\tilde{\mathbf{g}}$, compute the representative score $r(j,\tilde{u})$ for each instance j by $\tilde{\mathbf{toP}}$ and Eq. (3); and output a representative subset of instances from each type of object (except 'time' object) based on $r(j,\tilde{u})$

5. EXPERIMENTAL RESULTS

In this section, we introduce four real datasets and present our experimental results. All of the experiments are designed to answer the following questions:

- *effectiveness:* What is the quality of T3 and MT3 proposed in this paper?
- *efficiency:* How fast are the proposed algorithms?

5.1 Datasets

We use four real datasets, which are summarized in Table 3. For each dataset, Table 3 lists the number of different types of 'entity' objects (p), the number of different types of 'attribute' objects (q), the number of time nodes in the finest scale (n_1), the number of nodes (n) and edges (m) in the whole graph in the finest scale. We verify the effectiveness of the proposed T3 and MT3 on **NIPS**, **CIKM**, and **DeviceScan**, and measure the efficiency of our algorithms using the larger **DBLP** and **DeviceScan** datasets.

The first dataset (**NIPS**) is from the NIPS proceedings.[3] The time stamps are publication years, from 1987 to 1999. We treat paper as 'event' object and author as 'entity' object; there is no 'attribute' object in this dataset. Overall, there are 13 time nodes, 1,740 paper nodes, 2,037 author nodes, and 11,460 edges at the finest scale.

[3] http://www.cs.toronto.edu/~roweis/data.html

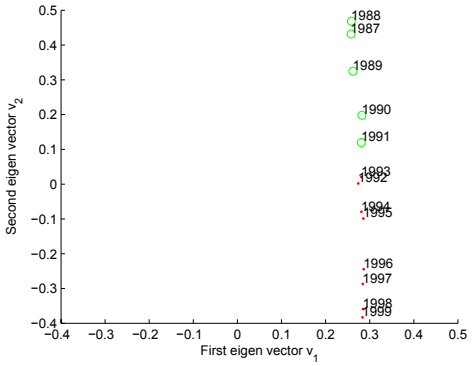


Figure 2: The embedding for the time nodes of NIPS dataset.

The **CIKM** dataset is constructed from the CIKM proceedings.[4] Again, time stamps are publication years, from 1993 to 2007. (Notice that we do not include papers from CIKM 1992 since the session information for that year is not available.) We treat paper as 'event' object. For this dataset, we have two types of 'entity' objects: the authors of the paper and the session name where the paper is presented during the conference. For the session name, we further extract 158 keywords as its attribute. Overall, there are 15 time nodes, 952 paper nodes, 1,895 author nodes, 279 session nodes, 158 keyword nodes, and 10,228 edges at the finest scale.

The **DBLP** dataset is constructed from all the papers in the DBLP.[5] Again, time stamps are publication years, from 1959 to 2007. We treat paper as 'event' object. For this dataset, we have two types of 'entity' objects: the authors of the paper and the conference where the paper is published. There is no additional 'attribute' object for this dataset. Overall, there are 49 time nodes, 567,090 paper nodes, 418,236 author nodes, 3,571 conference nodes, and 5,216,722 edges at the finest scale.

The **DeviceScan** is from MIT reality mining project.[6] Here, the 'event' object is blue tooth device scanning persons, and the time stamps are the day when such scanning events happen, from Jan. 1, 2004 to May. 5, 2005. For this dataset, we have two types of 'entity' objects: the blue tooth device and the person to be scanned; there is no additional 'attribute' object. Overall, there are 294 time nodes, 114,046 scanning nodes, 103 device nodes, 97 person nodes, and 684,276 edges at the finest scale.

5.2 Effectiveness: Case Studies

Here, we show the experimental results for the three real datasets, all of which are consistent with our intuition.

Fig. 2 gives the embedding of the time nodes for **NIPS** dataset using the first two eigen vectors ($\mathbf{v}_1$ and $\mathbf{v}_2$) of $\mathbf{toP}$, which reveal a line shape of time over publication years. Using T3, we find two time clusters (green circles vs. red dots in Fig. 2) as well as their interpretations in Table 4. From Fig. 2 and Table 4, we can see that while NIPS is a relatively stable community on the whole (e.g., the majority representative authors do not change over years), there is a topic shift from early 1990s (mainly on 'neural network' and 'neural information processing') to late 1990s (mainly on 'statistical learning').

Fig. 3 gives the embedding of the time nodes for **CIKM** dataset using the first two eigen vectors ($\mathbf{v}_1$ and $\mathbf{v}_2$) of $\mathbf{toP}$, which reveal

[4] http://www.informatik.uni-trier.de/~ley/db/conf/cikm/

[5] http://www.informatik.uni-trier.de/~ley/db/

[6] http://reality.media.mit.edu/

Time Cluster	Selected Papers	Selected Authors
1987	Presynaptic Neural Information Processing	Sejnowski_T
1988	Phasor Neural Networks	Hinton_G
1989	Phase Transitions in Neural Networks	Mozer_M
1990	The Hopfield Model with Multi-Level Neurons	Moody_J
1991	Temporal Patterns of Activity in Neural Networks	Koch_C
	Mapping Classifier Systems Into Neural Networks	Jordan_M
1992	Digital Boltzmann VLSI for Constraint Satisfaction and Learning	Sejnowski_T
1993	Sparse Code Shrinkage, Denoising by Nonlinear Maximum	Jordan_M
1994	Likelihood Estimation	Hinton_G
1995	Efficient Approaches to Gaussian Process Classification,	Koch_C
1996	Image Representations for Facial Expression Coding	Williams_C
1997	Experiences with Bayesian Learning in a Real World Application	Dayan_P
1998	Spectral Cues in Human Sound Localization,	Maass_W
1999	A PolygonalLine Algorithm for Constructing Principal Curves	Sollich_P

Table 4: The interpretations for NIPS dataset.

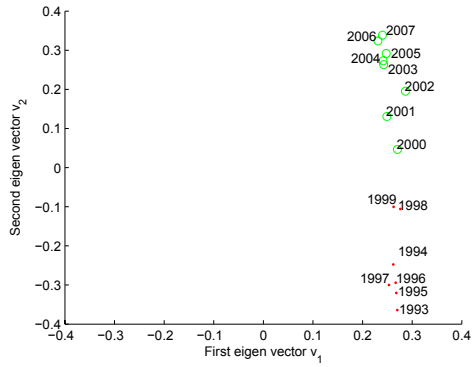
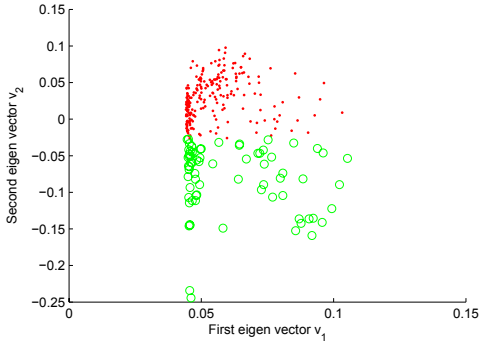
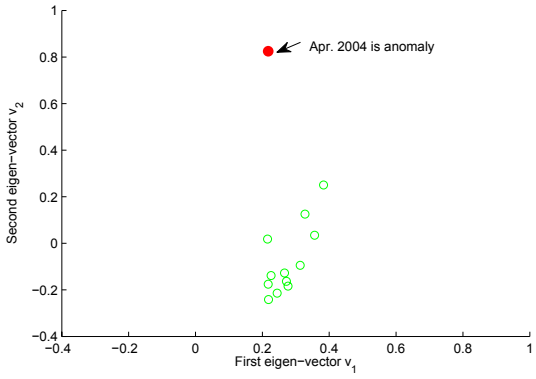


Figure 3: The embedding for the time nodes of CIKM dataset.



(a) on daily scale



(b) on monthly scale

Figure 4: The embedding for the time nodes of DeviceScan dataset.

a line shape of time over publication years as for the **NIPS** dataset. Using T3, we find two time clusters (green circles vs. red dots in Fig. 3) as well as their interpretations in Table 5. (For simplicity, we do not show the representative papers in the table.) From Fig. 3 and Table 5, we can see that while there are quite a lot of research interest in deductive databases and rule systems in the CIKM community in 1990s, attention has shifted to XML, statistical learning, language, etc since 2000.

Fig. 4 shows the results of applying the proposed MT3 to the **DeviceScan** dataset on two different scales: (a) daily scale and (b) monthly scale. From Fig. 4(a), it can be seen that, there are two time clusters on the daily scale. We found that one time cluster (green circles) corresponds to semester breaks as well as holidays; and the other cluster (red dots) corresponds to the week days during the semester. On the other hand, we found an abnormal time stamp (red dot, which is Apr. 2004) on the monthly scale (Fig. 4(b)).

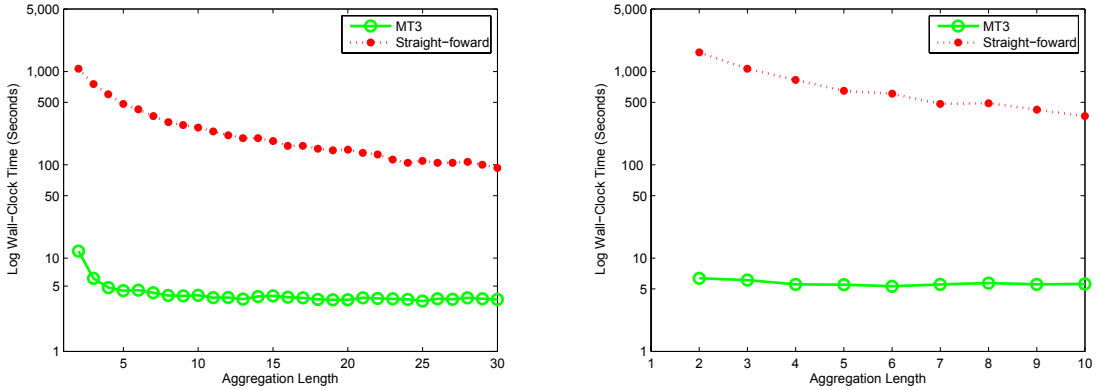
5.3 Efficiency

Here, we study the wall-clock time of the proposed MT3 using two relatively larger datasets: **DeviceScan** and **DBLP**. For these results, all of the experiments are done on the same machine with four 2.4GHz AMD CPUs and 48GB memory, running Linux (2.6 kernel). We vary the aggregation length (e.g., aggregate by every 2 time stamps, by every 3 time stamps, etc) and compare the wall-clock time by the proposed MT3 and that by applying T3 to each of the aggregated scale from scratch (referred to as the 'straight-forward' method).

Fig. 5 shows the results. Notice that time is in logarithm scale. It can be seen that the proposed MT3 is much more efficient. For example, it is 120x faster (6.1 seconds vs. 734 seconds) for **DeviceScan** dataset if we aggregate the time by every three time stamps (Fig. 5(a)); and it is 263x faster (6.0 seconds vs. 1,603 seconds) for **DBLP** dataset if we aggregate the time by every two time stamps (Fig. 5(b)). Overall, the proposed MT3 is 25x-263x faster than the straight-forward method. We would like to emphasize that such speed-ups are totally *free*, i.e., the proposed MT3 leads to exactly the same outputs as we apply T3 to each aggregated scale from scratch.

Time Cluster	Selected Sessions	Selected Authors	Selected Keywords
	deductive_&_rule_systems	elke_rundensteiner	knowledge
1993	query_processing	daniel_miranker	system
1994	knowledge_representation_&_expert_systems	andreas_henrich	unstructured
1995	object-oriented_databases	il-yeol_song	rule
1996	access_to_unstructured_information	scott_b._huffman	transaction
1997	deductive_databases	ling_liu	object-oriented
1998	document_processing	robert_j._hall	document
1999	logical_&_deductive_databases	jian_tang	deductive
	tools_for_realizing_intelligent_systems	ibrahim_kamel	ai
2000	xml_schemas:_integration_and_translation	james_p._callan	web
2001	classification	javed_a._aslam	cluster
2002	language_models	w._bruce_croft	classification
2003	corpus_linguistics	marius_pasca	language
2004	high-dimensional_indexing	james_allan	xml
2005	semistructured_data	philip_s._yu	similarity
2006	data_warehousing_and_olap	anton_leuski	stream
2007	text_classification_and_categorization	george_karypis	learn
	summarization_and_corpus_analysis	charles_clarke	mobile

Table 5: The interpretation for CIKM dataset.



(a) **DeviceScan** dataset

(b) **DBLP** dataset

Figure 5: Comparison on wall-clock time

6. RELATED WORK

In this section, we review the related work, which can be categorized into three parts: graph mining, proximity measurement on graphs and relational learning.

Graph Mining. There exists a lot of research on static graph mining, including pattern and law mining [4, 11, 13, 7, 24], frequent substructure discovery [35], influence propagation [20], community mining [14, 16, 17], etc. More recently, there has been an increasing interest in mining time-evolving graphs, such as densification laws and shrinking diameters [22], community evolution [5], dynamic communities [8], and proximity tracking [34], etc. It is worth pointing out that in these work, the focus is on utilizing the time information to better understand other nodes (event/entity/attribute) in the graphs; while in T3 and MT3 we focus on the other side of the problem, i.e., to better understand time itself based on other information (event/entity/attribute).

Measuring Proximity on Graphs. One of the most widely used proximity measurement on graphs is random walk with restart [19, 27, 33], which is the main idea behind Google's PageRank algorithm [26]. Other representative proximity measurements on static graphs include the sink-augmented delivered current [12], cycle-free effective conductance [21], survivable network [18], and direction-aware proximity [32]. Notice that the fast algorithms to compute the proximity measurements designed for querying, such as the one in [33], do not apply in our settings since the pre-computational time for these algorithms will flood the overall running time of T3 and MT3.

Also, there are a lot of applications of proximity measurements. Representative work includes connection subgraphs [12, 21, 30], content-based image retrieval [19], cross-modal correlation discovery [27], the BANKS system [1], link prediction [23], pattern matching [31], ObjectRank [6], RelationalRank [15], and NetRank [3, 2]. Among them, the most related works are [27, 6, 3, 2] in the sense that they all use a graph representation for the dataset(s). However, these approaches mainly focus on querying with or without learning; while T3 and MT3 are focusing on mining time in the context of complicated events.

Relational Learning. Sharan and Neville [29] present a two-step approach for incorporating temporal information on links (e.g., co-authorship and citation) into a relational classifier. First, they summarize the time-varying interaction as weights on links of a static summary graph. The summarization uses an exponential weighting scheme [10]. Second, they incorporate these link weights into a relational Bayes classifier. Their approach requires a sum-

mary parameter (θ), that needs to be either provided by the user or tuned by the learning algorithm. Furthermore, their approach cannot handle temporally-varying attributes. Our approach do not require a user-provided parameter and can handle time associated with any aspect of an event.

7. CONCLUSION

In this paper, we study how to find patterns in a collection of time-stamped, complex events. Our main contributions are the following:

1. We propose to treat each time-stamp as a node in a carefully constructed graph. This opens the door for the vast arsenal of graph mining algorithms (PageRank, graph partitioning, proximity analysis, CenterPiece Subgraphs, etc). We show how the proposed T3 can automatically group the time stamps into meaningful clusters, spot anomalies, and provide interpretations.

2. We propose MT3 to handle multiple scale analysis, achieving up to *2 orders of magnitude* speedups.

3. Finally, we verify the effectiveness as well as the efficiency of T3 and MT3 with experiments on several real datasets.

A promising research direction is to extend the T3 and MT3 to include additional continuous attributes, like geographical coordinates.

8. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grants No. IIS-0326322, No. IIS-0534205, and performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-CONF-405539). This work is also partially supported by the Pennsylvania Infrastructure Technology Alliance (PITA), an IBM Faculty Award, a Yahoo Research Alliance Gift, with additional funding from Intel, NTT and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

9. REFERENCES

[1] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, and S. S. Parag. Banks: Browsing and keyword searching in relational databases. In *VLDB*, pages 1083–1086, 2002.

[2] A. Agarwal and S. Chakrabarti. Learning random walks to rank nodes in graphs. In *ICML*, pages 9–16, 2007.

[3] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *KDD*, pages 14–23, 2006.

[4] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[5] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD*, pages 44–54, 2006.

[6] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, pages 564–575, 2004.

[7] A. Broder, R. Kumar, F. Maghoul1, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web: experiments and models. In *WWW Conf.*, 2000.

[8] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*, pages 153–162, 2007.

[9] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[10] C. Cortes, D. Pregibon, and C. Volinsky. Communities of interest. In *Proc. of the 4th Int'l Symp. of Intelligent Data Analysis (IDA'01)*, pages 105–114, 2001.

[11] S. Dorogovtsev and J. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1187, 2002.

[12] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD*, pages 118–127, 2004.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[14] G. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3):66–71, 2002.

[15] F. Geerts, H. Mannila, and E. Terzi. Relational link-based ranking. In *VLDB*, pages 552–563, 2004.

[16] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *9th ACM Conf. on Hypertext and Hypermedia*, pages 225–234, New York, 1998.

[17] M. Girvan and M. E. J. Newman. Community structure is social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.

[18] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In *Handbooks in Operations Research and Management Science 7: Network Models*. North Holland, 1993.

[19] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *ACM Multimedia*, pages 9–16, 2004.

[20] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[21] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. In *KDD*, pages 245–255, 2006.

[22] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, pages 177–187, 2005.

[23] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[24] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[25] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).

[27] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *KDD*, pages 653–658, 2004.

[28] W. Piegorsch and G. E. Casella. Inverting a sum of matrices. *SIAM Review*, 32(3):470, 1990.

[29] U. Sharan and J. Neville. Exploiting time-varying relationships in statistical relational models. In *1st SNA-KDD Workshop*, 2007.

[30] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *KDD*, pages 404–413, 2006.

[31] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *KDD*, pages 737–746, 2007.

[32] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *KDD*, pages 747–756, 2007.

[33] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[34] H. Tong, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SIAM-DM*, pages 704–711, 2008.

[35] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *VLDB*, pages 709–720, 2005.