

A Query Based Approach for Mining Evolving Graphs

Andrey Kan¹ Jeffrey Chan^{1,2} James Bailey¹ Christopher Leckie¹

¹ NICTA Victoria Research Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Australia
Email: {akan, jkcchan, jbailey, caleckie}@csse.unimelb.edu.au

² Digital Enterprise Research Institute
National University of Ireland, Galway
Ireland
Email: jkc.chan@deri.org

Abstract

An evolving graph is a graph that can change over time. Such graphs can be applied in modelling a wide range of real-world phenomena, like computer networks, social networks and protein interaction networks. This paper addresses the novel problem of querying evolving graphs using spatio-temporal patterns. In particular, we focus on answering selection queries, which can discover evolving subgraphs that satisfy both a temporal and a spatial predicate. We investigate the efficient implementation of such queries and experimentally evaluate our techniques using real-world evolving graph datasets — Internet connectivity logs and the Enron email corpus. We show that it is possible to use queries to discover meaningful events hidden in this data and demonstrate that our implementation is scalable for very large evolving graphs.

Keywords: spatio-temporal data mining, evolving graphs, dynamic graph analysis, spatio-temporal analysis, spatio-temporal query, querying evolving graphs, event discovery

1 Introduction

An evolving graph represents a graph that changes over time, which can be characterised by a series of temporal snapshots. Figure 1 shows an example evolving graph over three snapshots (time points).

Evolving graphs are very useful for modelling real world phenomena, such as communication networks, social networks, protein interactions and businesses collaboration. Some specific examples are:

- In the analysis of computer networks, the changing topology of a network can be naturally modelled as an evolving graph. A vertex may represent a workstation and an edge may indicate the existence of a communication path at a given time point. The following questions may then arise: “is there any spatially adjacent group of connections that fail synchronously?”, “which groups of connection paths are not persistent over the specified period?”. Answering such questions can facilitate fault localisation and the identification of unstable network regions.

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Eighth Australasian Data Mining Conference (AusDM 2009), Melbourne, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 101, Paul J. Kennedy, Kok-Leong Ong and Peter Christen, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

- A corpus recording the email activity over a period of time for an organization can be analysed as an evolving graph. Vertices may represent employees and an edge indicates that an email was sent between employees during a certain time period. Questions that may arise might be “are there groups of people having their communication intensified over some window of time?”, “are there people having collaboration that lasted for approximately a month?”.
- An evolving graph approach may be used for criminal investigations (Krebs 2002, Xu and Chen 2004). Vertices represent individuals and an edge corresponds to suspected relations between two individuals. Such relationships may be inferred from different sources: phone calls, bank transfers or people being seen together. In this scenario, a criminal investigator may wish to explore the following query “what relations appear or disappear synchronously between certain groups of people and when does it happen?”.

These examples all share a common feature: there is interest in discovering matches for a specific spatio-temporal pattern. This motivates us to investigate a query based approach to mining evolving graphs.

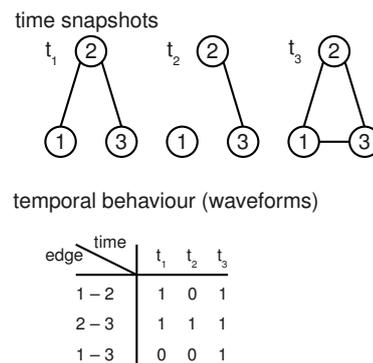


Figure 1: Sample evolving graph consisting of three snapshots. The set of vertices remains unchanged, whereas the set of edges evolves over time. Temporal behaviour of each edge can be represented by a string (waveform), where each symbol encodes the presence (“1”) or absence (“0”) of the edge at a particular time point.

To the best of our knowledge, we are the first to investigate the querying of evolving graphs by spatio-temporal patterns.

In contrast to standard database querying, querying for evolving graphs is more exploratory in nature. When undertaking analysis of an evolving graph, the user may have only a general idea of what patterns to search for. So a typical search strategy is likely to begin with the user posing an initial set of queries. The user will then iteratively analyse the obtained results and refine the queries. Since evolving graphs may be very large, it is very important that such queries can be evaluated efficiently.

There are several existing works on mining evolving graphs. Work by Chan et al. (2008) defines regions of correlated spatio-temporal changes within an evolving graph and presents an algorithm for finding all such regions. Work in Jin et al. (2007) considers a graph in which vertex labels are changing over time and presents an approach to finding frequent trend motifs, i.e., frequent subgraphs sharing the same trends in label changes. Work in Borgwardt et al. (2006) presents a method for mining frequent dynamic subgraphs, which are subgraphs that are connected, share temporal behaviour and occur at least a predefined number of times. Work in Lahiri and Berger-Wolf (2008) studies the detection of periodic behaviour in subgraphs.

These works each enumerate all occurrences (patterns) of some specified kind: inter-correlated regions, trends, frequent subgraphs or periodic behaviour. In contrast, a query based approach allows the user to specify the desired result in a specific and flexible way. For example, one might be interested only in inter-correlated regions with a particular temporal behaviour or only in frequent subgraphs that disappear at a certain time point. Since it is more specific, another advantage of a query based approach is that it is significantly faster compared to enumeration style methods.

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

- We propose a model for querying evolving graphs by spatio-temporal patterns (Section 2).
- We present two algorithms for implementing select style queries (Sections 3.1 and 3.2). We analyse the correctness, completeness, worst-case complexity and relative advantages of each algorithm.
- We run experiments on two real-world datasets (Sections 4.1 and 4.2). Experiments show that using our query model, it is possible to find subgraphs that relate to real-world events.
- We evaluate our implementations on synthetic evolving graphs (Section 4.3). Evaluation shows that querying can be implemented in an efficient and scalable manner. We found that our query implementation requires less than half a second for a graph with 10,000 changing edges.

2 Problem Statement

In this section, we formally define the problem in terms of the evolving graphs (input), the correlated subgraphs (output), and the query model that specifies the type of correlated subgraphs in which we are interested. We begin in Section 2.1 by defining the concept of an evolving graph. In Section 2.2, we define the concept of a correlated evolving graph with particular focus on how its temporal and spatial behaviour can be specified in a query. Finally, in Section 2.3 we define our query formulation and the corresponding query satisfaction problem.

2.1 Evolving Graphs

One of the main inputs for a query is an evolving graph. We begin by defining an evolving graph as well as the related concept of an evolving subgraph, based on the formulations presented by Borgwardt et al. (2006).

Definition 1. (Evolving Graph) *An evolving graph is a sequence of consecutive graph snapshots $G_{ts} \dots G_{te}$ that have the same set of vertices V , but possibly different sets of edges E_t , where $t = ts, \dots, te$.*

Let $E = \bigcup_{t=ts}^{te} E_t$ be a union of edges of all graphs in the sequence. We denote an evolving graph as $eg = (V, E, ts, te, \mathcal{E})$.

\mathcal{E} is a set of strings specifying the temporal behaviour of edges. For each edge e in E , there is a string $\mathcal{E}(e)$ with symbols numbered from ts to te . If an edge e is included in snapshot G_t , then the corresponding symbol $\mathcal{E}(e)[t] = "1"$, otherwise (edge e is deleted from G_t) $\mathcal{E}(e)[t] = "0"$.

In the scope of the present work we consider only undirected, unweighted graphs and discrete time points, i.e., a sequence of graph snapshots. We focus on edge changes, assuming the set of vertices remains the same. Note that a changing set of vertices $V_t, t = ts, \dots, te$, can be replaced by a union set $V = \bigcup_{t=ts}^{te} V_t$.

Definition 2. (Evolving Subgraph) *Given two evolving graphs $eg_1 = (V_1, E_1, ts_1, te_1, \mathcal{E}_1)$ and $eg_2 = (V_2, E_2, ts_2, te_2, \mathcal{E}_2)$, eg_1 is an evolving subgraph of eg_2 if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $[ts_1, te_1] \subseteq [ts_2, te_2]$ and \mathcal{E}_1 contains substrings from \mathcal{E}_2 , taken in the interval $[ts_1, te_1]$ for all e in E_1 . Saying that eg_1 is an evolving subgraph of eg_2 is equivalent to saying that eg_1 is included in eg_2 . This inclusion relation is denoted as $eg_1 \subseteq eg_2$.*

In the substring definition above, a substring must match a consecutive subsequence of characters, i.e., no gaps are allowed. For example, given the string "abcde", "abc" and "cd" are its substrings, whereas "ace" and "cba" are not substrings.

An evolving subgraph can be considered as a spatial subgraph of the evolving graph, extracted over some sub-period of time (see Figure 2). Note that an evolving subgraph is an evolving graph itself.

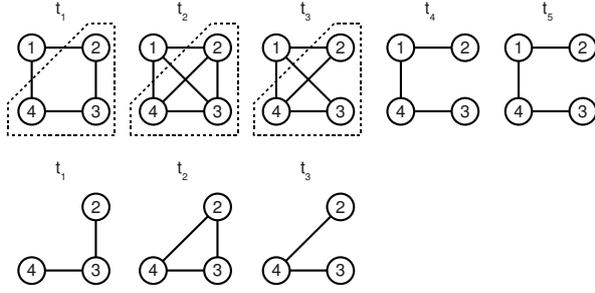
2.2 Correlated Evolving Graphs

We now consider the specific type of evolving subgraph that we seek in our query satisfaction problem, i.e., correlated evolving graphs that satisfy certain temporal and spatial characteristics. Our aim is to find evolving subgraphs whose edges exhibit a common *temporal* behaviour (in terms of insertion and deletion), as well as sharing specific *spatial* properties, such as their topological proximity.

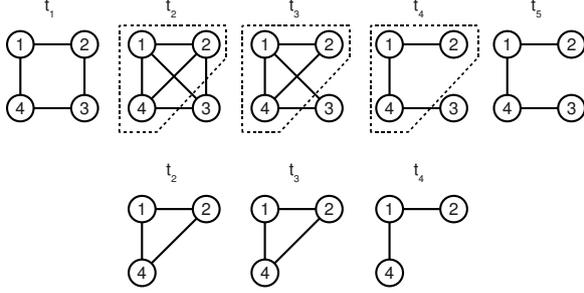
Work by Chan et al. (2008) defined this task as an unsupervised learning problem, where the aim is to find all correlated subgraphs in a given evolving graph. In contrast, our aim is to formulate the search for correlated subgraphs as a query answering problem, where we search for correlated subgraphs that match a given query, which specifies the temporal evolution of interest.

The temporal behaviour of a changing edge can be represented by the following strings (as proposed by Chan et al. 2008). A **waveform** (denoted as W) is a string consisting of "1" and "0" symbols, reflecting the insertion or deletion of an edge from the snapshots of a graph. Symbols in the waveform are numbered starting from 1.

Another string, called the **transition sequence**, represents the overall shape of the corresponding



a) evolving graph and a sample evolving subgraph



b) the same evolving graph and another sample evolving subgraph

Figure 2: The figure illustrates the concept of evolving subgraph. Two sample subgraphs extracted from the same evolving graph are shown. The dotted line highlights the particular subgraph extracted.

waveform. Each occurrence of the “10” pattern in the waveform (i.e., deletion) is represented by the “-” symbol in the transition sequence, and an occurrence of the “01” pattern (i.e., addition) is represented by the “+” symbol. All other parts of the waveform are ignored. See Figure 3 for an example. We denote the transition sequence for a waveform W as $tran(W)$.

waveform	000111	110011	1011001
graphical representation			
transition sequence	+	--	+-++

Figure 3: Sample waveforms and their transition sequences. The symbols of a waveform encode the presence (“1”) or absence (“0”) of an edge in an evolving graph at a particular time point. The symbols of a transition sequence encode a changing state of the edge: appearance (“+”) or disappearance (“-”).

In our model, the result of a query is a set of subgraphs in which all edges follow the waveform provided in the query. However, the edges in the resulting subgraphs do not need to have exactly the same waveform as in the query. It is sufficient to have similar waveforms with respect to some distance metric. We call this metric **temporal distance**. Chan et al. (2008) introduce a distance measure called “modified Euclidean distance” and argued that this measure fits well in the context of their problem statement. Since we use the same definition of waveform, we adopt the “modified Euclidean distance” in our work. Given two waveforms W_1, W_2 ,

where $length(W_1) = length(W_2) = L$

$$d_{m_euc}(W_1, W_2) = \begin{cases} 1, & \text{if } trans(W_1) \neq trans(W_2); \\ \frac{1}{L} \sum_{k=1}^L W_1[k] \oplus W_2[k], & \text{otherwise.} \end{cases}$$

Here, the operator $x_1 \oplus x_2$ represents an exclusive OR: it equals 0 if the symbols x_1, x_2 are the same, and is 1 otherwise.

This distance lies in the range $[0, 1]$, with smaller values corresponding to a better match between the waveforms. If the transition sequences differ, the highest value (complete mismatch) is returned. Otherwise the number of mismatched symbols is taken into account.

Note that strings specifying the temporal behaviour of edges in an evolving graph are essentially waveforms. The difference is that the symbols in $\mathcal{E}(e)$ are numbered starting from ts . For any edge we can obtain its waveform W_e by re-numbering the symbols in the string $\mathcal{E}(e)$.

Definition 3. (Correlated Edge) Consider an edge e of an evolving graph and the waveform W_e of the edge. Given an arbitrary waveform W and a user-defined threshold θ , the edge is correlated with W if the lengths of W and W_e are the same and these waveforms are sufficiently similar: $d(W, W_e) \leq \theta$.

Now we extend the concept of correlation to an evolving graph.

Definition 4. (Correlated Evolving Graph) An evolving graph $eg = (V, E, ts, te, \mathcal{E})$ is correlated with a waveform W if all of its edges are correlated with W and the graph $G(V, E)$ is connected. We denote the correlation to a waveform as $e \sim W$ for edges and $eg \sim W$ for evolving graphs.

Note the spatial constraint in the definition above: we require connectedness. This requirement reflects the localization property of real-world events. Another way to impose a spatial constraint is a predicate. In the context of our query model, we define the predicate as follows.

A **predicate** on an evolving graph $P(eg) = \{true|false\}$ is a function that takes an evolving graph as input and produces a Boolean value (true or false) as output. Our only requirement for a valid predicate is that it must be implementable in linear time complexity with respect to the number of vertices and edges in the input graph. We require this constraint to guarantee the efficiency of our querying algorithm (complexity analysis is presented in Section 3.3).

Predicates can be used to impose both spatial and temporal constraints. Consider an evolving graph $eg = (V, E, ts, te, \mathcal{E})$. We denote the number of vertices and edges in the graph as n_v and n_e respectively. The following are examples of possible predicates.

“Timing predicate” returns true if the evolving graph satisfy certain timing constraints:

$$P_{time}(eg) = (ts > 3 \text{ and } (te - ts) > 5).$$

“Size predicate” returns true if the evolving graph has more than N edges:

$$P_{size}(eg) = (n_e \geq N).$$

“Clique predicate” (assuming a simple undirected graph) returns true if a graph $G(V, E)$ is a clique (each vertex is connected to all other vertices):

$$P_{clique}(eg) = (n_e == n_v * (n_v - 1) / 2).$$

Note that a finite Boolean formula over the predicates is a predicate itself. For example in $P = ((P_1 \text{ and } P_2) \text{ or } P_3)$, all of P_1, P_2, P_3 and P are predicates.

We are now ready to define our main operator for performing queries.

2.3 Query Model

In our query model, the data is represented by evolving graphs, the required spatio-temporal properties are specified by waveforms and predicates, and the queries are implemented in the form of a selection operator.

Our model should be considered as a stepping stone towards a general evolving graph query language with a more sophisticated algebra. By analogy with the relational algebra, the model might be extended with operators like join and aggregation.

At present, selection is the only operator in our model. This operator is essential and our experiments show that it is sufficiently powerful to produce practically useful results.

Definition 5. (Query) A (spatio-temporal) query Q is a pair $\{W, P\}$, where W is a waveform and P is a predicate.

Definition 6. (Selection Operator) A selection operator takes as input an evolving graph eg , and a query $Q = \{W, P\}$. The output is a set of evolving subgraphs correlated with W , which satisfy P :

$$\sigma(eg, Q) = \{sg \subseteq eg : sg \sim W, P(sg) = true\}$$

We require the evolving subgraphs in the output to be maximal, i.e., there is no evolving subgraph that is included in another evolving subgraph from the same output set.

Figure 4 illustrates sample queries and their results. Recall that for a correlated evolving subgraph we require the corresponding graph $G(V, E)$ to be connected. Here V is the set of vertices, which is the same for all snapshots of an evolving subgraph, $E = \cup E_t$ is the union of edges in all snapshots.

The problem that we address in this paper is how to design an algorithm that can answer queries in a scalable manner on large evolving graphs. In the next section, we present our algorithms for implementing the queries.

3 Algorithms for Query Satisfaction

In this section, we present an efficient algorithm to implement our query selection operator. We call the algorithm “Select Basic” and describe it, along with a correctness and completeness analysis, in Section 3.1. We then present a modified version of the algorithm, which runs faster, but requires extra memory and prior indexing of an evolving graph. We call the modified version “Select Indexed” and describe it in Section 3.2. Finally we analyse the worst-case time complexity for both algorithms in Section 3.3.

A naive approach to implement a query $\sigma(eg, W, P)$ is to find all evolving subgraphs correlated with W , filter out those subgraphs that do not satisfy P , and finally select the maximal evolving subgraphs. The drawback of this approach is the computational cost of checking the inclusion relation for all pairs of evolving subgraphs in the intermediate set of results. Instead we propose the following algorithm.

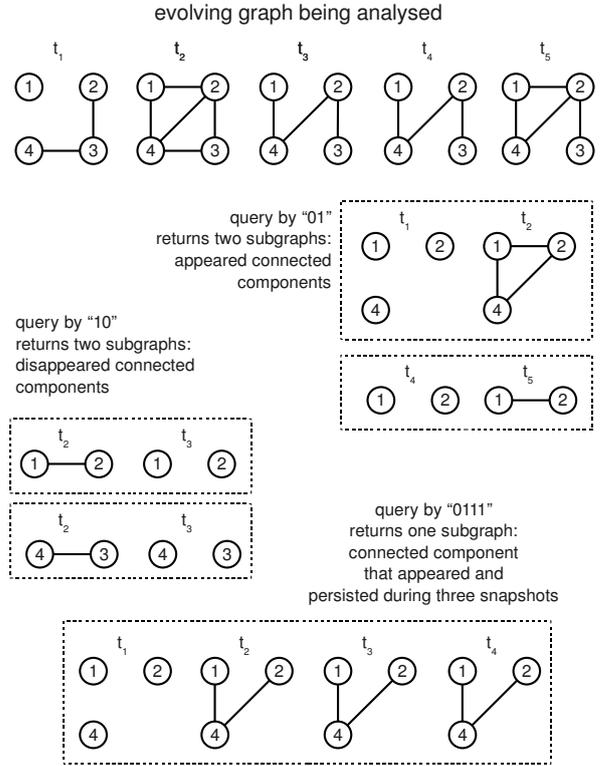


Figure 4: Results produced by selection operator

3.1 Select Basic

To implement a query we propose an algorithm called “Select Basic”. Inputs to this algorithm are an evolving graph eg , a waveform pattern W (with the length l_W) and a predicate P . The output is the result of selection $\sigma(eg, W, P)$.

The key principle of this algorithm is to select evolving subgraphs at each time point $t \in [t, te]$ independently of all other time points. The algorithm iterates over each time point and finds $sg_t \subseteq eg$, an evolving subgraph, defined on $[t, t + l_W - 1]$. sg_t includes all edges that are correlated with W over $[t, t + l_W - 1]$, and sg_t includes only such edges. After being found, sg_t is then partitioned into evolving subgraphs, such that each subgraph is correlated with W (and connected). Finally, any partitions of sg_t that do not satisfy P are filtered out.

In the “Select Basic” algorithm, a function *SubgraphWithCorrEdges()* is implemented in a direct way: it starts with empty sg_t , iterates over all edges e in E and calculates the correlation with waveform W over $[t, t + l_W - 1]$. If the correlation is sufficiently strong, e is added to sg_t .

Lemma 7. Algorithm “Select Basic” is a correct and complete implementation of the selection operator (see proof in Appendix A.1).

Considering the exploratory nature of querying, a common scenario consists of running a large number of queries over the same evolving graph. We found that it is possible to optimize this common scenario. We present a modified version of the query implementation in the following section.

3.2 Select Indexed

While algorithm “Select Basic” is computationally efficient (as shown in the next section), there are

Algorithm 1 Select Basic

Input: evolving graph $eg = (V, E, ts, te, \mathcal{E})$; waveform W ; predicate P

Output: set of matching evolving subgraphs $R = \sigma(eg, W, P)$

```
1:  $R = \emptyset$ 
2: foreach  $t$  in  $[ts, te]$ 
3:   // get subgraph containing all edges
4:   // correlated to  $W$  during period  $[t, t + l_W - 1]$ 
5:    $sg_t \leftarrow \text{SubgraphWithCorrEdges}(eg, t, W)$ 
6:   //  $P_t$  is a set of connected subgraphs
7:    $P_t \leftarrow \text{Partition}(sg_t)$ 
8:   foreach  $sg$  in  $P_t$ 
9:     if  $(P(sg))$ 
10:       add  $sg$  to  $R$ 
11:   endfor
12: endfor
```

specific application scenarios where greater efficiency can be achieved. We propose a modification of “Select Basic”, which is more efficient when multiple queries are made from the same evolving graph. The modified algorithm is called “Select Indexed”, because it uses indexing, as we discuss in detail in this section.

We found that the implementation of function $\text{SubgraphWithCorrEdges}()$ is a main contributor to running time of a query. Thus we focused on optimizing this function and algorithm “Select Indexed” is a modification of “Select Basic”, using an alternative implementation of $\text{SubgraphWithCorrEdges}()$.

There are two ideas behind the optimisation. The first idea is to prune away edges that do not change over the entire evolving graph. We need to store a list of such edges, but we do not need to store and analyse their waveforms.

The second idea is to index all possible evolving subgraphs in eg in advance, and then use this precomputed data when satisfying queries. Such an indexing procedure requires additional execution time, but indexing is needed only once. Furthermore, it can be performed as a background process without requiring users to wait.

While indexing can be applied to make querying faster, the storage for indexing results requires additional memory. The amount of memory overhead is estimated in the next section.

Indexing results are stored in two hash tables called “Inner” and “Outer”. The “Inner” table is a hash table, with a waveform W as a key and the set of edges, having exactly this waveform over some period $[t, t + l_W - 1]$, as a value. The “Outer” table has a tuple $\{t, l_W; \text{tran}(W)\}$ as a key and a pointer to an “Inner” hash table as a value. A key $\{t, l_W; \text{tran}(W)\}$ points to the “Inner” table in which all keys are waveforms starting from t with length l_W and transition sequences $\text{tran}(W)$.

Consider algorithms “Prune” and “Indexing”. At first, all edges that do not change across the entire evolving graph eg are saved in a separate list and pruned from eg . Subsequent indexing is performed only on changing edges.

3.3 Complexity Analysis

The inputs to the “Select” algorithms are an evolving graph eg , a waveform W and a predicate P . The

Algorithm 2 Function $\text{SubgraphWithCorrEdges}()$ in “Select Indexed”

Input: evolving graph without constant edges eg' ; list of constant edges E_C ; current time point t ; waveform W ; hash tables $Inner, Outer$

Output: $sg_t \subseteq eg$ is an evolving subgraph, containing all edges correlated with W over $[t, t + l_W - 1]$, and containing only such edges

```
1: // set an empty evolving graph
2: // which is defined over  $[t, t + l_W - 1]$ 
3:  $sg_t \leftarrow (\emptyset, \emptyset, t, t + l_W - 1, \mathcal{E} = \emptyset)$ 
4: if (all symbols in  $W$  equal to 1)
5:   add  $E_C$  to edges set of  $sg_t$ 
6: endif
7:  $trSeq \leftarrow$  transition sequence of  $W$ 
8:  $keyOuter \leftarrow \{t, l_W, trSeq\}$ 
9:  $Inner \leftarrow \text{Find}(Outer, keyOuter)$ 
10: if ( $Inner$  table found)
11:   foreach  $\{key; value\}$  in  $Inner$ 
12:      $W_e \leftarrow key$  // a waveform
13:      $eSet \leftarrow value$  // a set of edges
14:     if ( $W$  and  $W_e$  are correlated enough)
15:       add  $eSet$  to the edges set of  $sg_t$ 
16:     endif
17:   endfor
18: endif
```

evolving graph consists of $T = te - ts + 1$ snapshots. We focus on changing edges and ignore vertices that have no adjacent edges, thus we can put $n = |V| = |E|$ (here, $E = \bigcup_{t=ts}^{te} E_t$ is a union of edges in all snapshots). A waveform has length l_W . A predicate runs in linear time with respect to the number of vertices and edges in the input graph, as we required in Section 2.2.

We now consider the worst-case complexity of the “Select Basic” algorithm. The algorithm consists of T iterations. At each iteration we apply the function $sg_t = \text{SubgraphWithCorrEdges}()$, partitioning and filtering. sg is an evolving subgraph of eg , thus the subgraph sg can have at most n vertices and n edges. Partitioning is essentially splitting into connected components and can be performed in $O(n + n)$ steps. Since we required that the predicate can be implemented in linear time (Section 2.2), each call of predicate $P(sg)$ runs in $O(n_{sg} + n_{sg})$, where n_{sg} is the number of edges in sg . After partitioning the sum of vertices and edges in all partitions is at most $(n + n)$. Therefore all calls of $P(sg)$ in one iteration run in $O(n + n)$ and overall complexity for “Select Basic” can be written as $O(T \times (n + O(\text{SubgraphWithCorrEdges})))$.

In “Select Basic” $\text{SubgraphWithCorrEdges}()$ iterates over all edges in E and calculate the correlation with waveform W . Recall that we use a linear metric “modified Euclidean distance” for correlation. Thus the complexity of this function is $O(n \times l_W)$.

The total worst-case time complexity for algorithm “Select Basic” is $O(T \times n \times l_W)$, i.e., a query can be implemented in linear time with respect to its inputs, namely the parameters of the evolving graph being queried and the length of the required waveform.

“Select Indexed” differs from “Select Basic” only in the implementation of $\text{SubgraphWithCorrEdges}()$.

Algorithm 3 Prune (Constant Edges)

Input: evolving graph $eg = (V, E, ts, te, \mathcal{E}t)$ **Output:** $eg' = eg$ without constant edges; set of constant edges E_C

```
1:  $eg' = eg$ 
2:  $E_C = \emptyset$ 
3: foreach  $e$  in  $E$ 
4:   if  $(\mathcal{E}(e)[t] = "1"$  for each  $t$  in  $[ts, te])$ 
5:     // edge is "constant"
6:     add  $e$  to  $E_C$ 
7:     remove  $e$  from  $eg'$ 
8:   endif
9: endfor
```

Algorithm 4 Indexing

Input: $eg' = eg$ without constant edges**Output:** outer and inner hash tables $Outer, Inner$

```
1:  $Outer = \emptyset$ 
2: foreach  $e$  in  $E'$ 
3:   foreach  $l$  in  $[1, T]$ 
4:     foreach  $t_1$  in  $[ts, te]$ 
5:        $t_2 \leftarrow t_1 + l - 1$ 
6:        $trSeq \leftarrow$  tran. sequence of  $e$  over  $[t_1, t_2]$ 
7:        $keyOuter \leftarrow \{t, l, trSeq\}$ 
8:        $Inner \leftarrow FindOrCreate(Outer, keyOuter)$ 
9:        $W_e \leftarrow$  waveform of  $e$  over  $[t_1, t_2]$ 
10:       $keyInner \leftarrow \{W_e\}$ 
11:      // set of edges that have
12:      // the same waveform over  $[t_1, t_2]$ 
13:       $eSet \leftarrow FindOrCreate(Inner, keyInner)$ 
14:      add  $e$  to  $eSet$ 
15:    endfor
16:  endfor
17: endfor
```

In the optimised implementation of this function in “Select Indexed” the time complexity is determined by the number of insertions of edges into sg_t (lines 5 and 15) and the number of correlation computations (line 14). In the worst case, the evolving subgraph sg_t can have as many edges as eg and correlations may need to be calculated for every single edge. In this case, function *SubgraphWithCorrEdges()* has time complexity $O(n \times l_W)$ and algorithm “Select Indexed” has the same time complexity $O(T \times n \times l_W)$ as the “Select Basic”.

However, in practice the number of edges in sg_t tends to be much less than n , because we expect different groups of edges to follow different behaviours at some time point. Furthermore, the computation of correlation occurs only once per group of edges that have the same transition sequence. Therefore we expected that the complexity of the optimised *SubgraphWithCorrEdges()* function would be on average $O(k \times l_W)$, where $k \ll n$ is some constant. Under this assumption, the complexity of “Select Indexed” is $O(T \times k \times l_W)$.

“Select Indexed” requires prior preprocessing (“Prune” and “Indexing”) of an evolving graph. The worst-case time complexity of the algorithm “Prune”

is $O(T \times n)$.

Now consider one iteration of the inner loop in the algorithm “Indexing”. Computing a transition sequence can be performed incrementally as time progresses. Searching a hash table is considered as a pseudo-constant operation. Therefore the overall worst-case time complexity of algorithm “Indexing” is $O(T \times T \times n)$.

Storing the indexing data (hash tables) requires additional space. For each waveform W that occurred at a certain time point and has a certain length, there can be no more than n' matching edges, where n' is the number of edges that experienced at least one change in the evolving graph eg . The total number of these unique waveforms is $(T \times T)$, i.e., a waveform can occur at any time point in $t \in [ts, te]$ and have a length $l \in [1, T]$. Therefore the overall memory complexity is $O(T \times T \times n')$.

In the next section, we compare the relative advantages of each of our two algorithms in terms of their execution time on various types of datasets.

4 Experimental Evaluation

In this section we evaluate the effectiveness and efficiency of our proposed algorithms on a range of real-world and synthetic datasets. The aims of our experiments have been: i) to evaluate the practical feasibility of querying evolving graphs and gauge the interestingness of the results, and ii) to measure the computational scalability of our algorithms for querying large graphs.

To evaluate the practical feasibility of querying, we have analysed evolving graphs built from two real-world datasets (Sections 4.1 and 4.2). The first dataset consists of snapshots of the routing topology of the backbone of the Internet. This dataset was collected by Chan et al. (2008). The second dataset is the Enron email corpus as presented by Klimt and Yang (2004). These datasets are particularly interesting because there are known real-world events that we expect to be reflected in the data. Furthermore, there have been other studies that have analyzed the same datasets from different perspectives. Thus we can compare and contrast our findings with these related works.

To measure the computational scalability, we have evaluated our algorithms on synthetic evolving graphs (Section 4.3). We generated random evolving graphs with different sizes and measured the running times required for querying each graph.

We also compare our results with findings on the same datasets from related works (Section 4.4).

4.1 Evaluation on Internet BGP Routing Topology Graph

Our first experiment was the analysis of a part of the Internet backbone routing topology. At the backbone level, the Internet comprises a set of Autonomous Systems (ASs), each of which is a network under a single controlling authority. The Border Gateway Protocol (BGP) is responsible for establishing routes between these ASs. From a high level perspective, an AS can be considered as a node in the Internet connectivity graph, where each AS has a unique number. Therefore the Internet can be represented as an evolving graph, where a vertex corresponds to an AS and is labeled by its AS number. An edge in such a graph corresponds to an existing routing path (connectivity) between two ASs. The topology of the Internet may change over time, thus the graph is evolving.

Chan et al. (2008) have constructed an evolving graph from the AS connectivity logs. In their work they present the details of building the graph and argue for the difficulty and importance of analysing changing edges of the graph. We have used a copy of their graph in our experiment.

The evolving graph represents only the US part of the Internet, i.e., each vertex corresponds to an AS registered in the USA. The graph consists of around 10,000 vertices and 18,000 edges. Only about 700 edges are changing, while others persist in the graph during the whole time period. The graph spans over the 41 time snapshots, which we number starting from one. Snapshots are taken in two hour intervals. The first snapshot is taken on the 28 August 2005 at 1 pm and the last on the 31 August 2005 at 10 pm (times are in UTC format).

In August 2005 there were Hurricane Katrina landfalls in some southern US states. Hurricane Katrina was a major event, which is known to severely affect the Internet infrastructure in the region of the landfalls (Cowie et al. 2005). The second landfall of Katrina occurred in Louisiana approximately between snapshots 12 and 13 (August 29, between 10 and 11 am UTC).

We were interested in the major effects of Katrina landfalls, thus we filtered out small results using the predicate of our query. For all queries we used a predicate that returns true for evolving graphs containing more than 3 edges.

Two hour snapshots provide quite a fine temporal granularity. Therefore the consequences of the same event may appear in neighbouring snapshots. To address this, for all queries we set the correlation threshold as 0.2. This means, we allowed 80% to 100% correlation between found subgraphs and a waveform in a query, rather than requiring exact match.

We used three different temporal patterns. The first pattern is “111000”. It was designed to discover failure regions, more specifically, connections that were stable for several snapshots and then disappear for an extended period. We did not use the “10” pattern, because this query might result in additional subgraphs corresponding to random short term network failures. (Of course one can use “10” or any other pattern, depending on the graph being analysed and the particular purpose of the analysis. For example, we later use the “01” pattern in our experiment on the Enron dataset.)

The second pattern is “000111”. Subgraphs with such temporal behaviour can be considered as recovery regions.

The last pattern is “110011” and it corresponds to a failure, followed by a prompt recovery. For each query we obtained a set of subgraphs and manually reviewed the largest subgraphs in each set.

Recall that each vertex represents an AS with a unique numerical identifier. Chan et al. (2008) show that by using the AS number it is possible to retrieve two types of information: the US state where the AS is registered (and most likely physically located), and the organization to whom the AS belongs. We used this information when presenting our results in Figure 5. For selected vertices either the US state or organizational affiliation is shown. Waveforms in the figure show the temporal behaviour of the majority of the edges in the corresponding subgraph.

Evolving graph A most likely corresponds to a network failure due to the second landfall of Hurricane Katrina: all edges of the graph are connected to ASs in Louisiana and the timing of failure matches the timing of the landfall. Cowie et al. (2005) reported that a large percentage of destroyed networks around 2 pm UTC (snapshot 13) were in Louisiana.

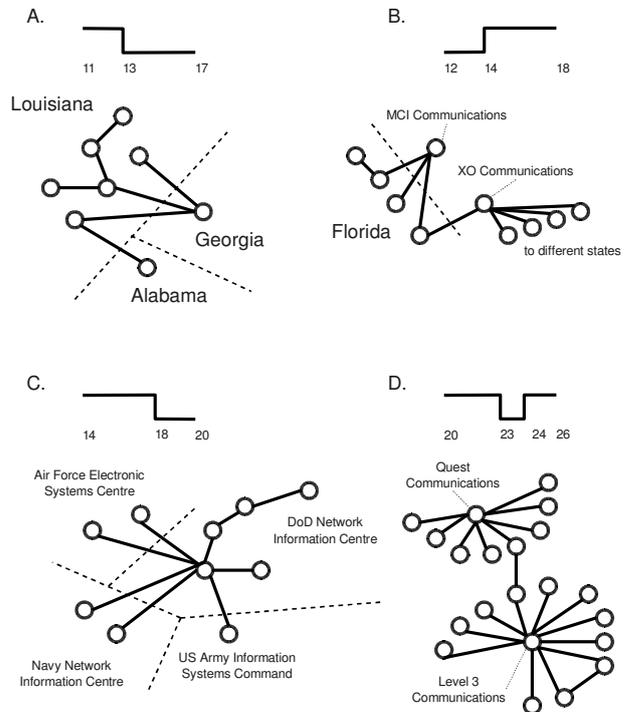


Figure 5: Some of the results from querying the AS connectivity graph. The results are interpretable, i.e., each found subgraph can be related to a real world event. For selected vertices either the US state or organizational affiliation is shown. Waveforms represent the behaviour of the majority of the edges in a subgraph. Time stamps are snapshot numbers.

Evolving graph B represents a recovery region. There was an earlier landfall of Katrina in Florida. Although the timing of this landfall is outside of the period we analyse, we note the connections with Florida and suggest that graph B represents the recovery of major communication networks from this earlier strike.

Other interesting events are represented by evolving graphs C and D. Evolving graph C represents a failure of a segment of a military network: all its ASs are registered within the Department of Defence, Navy and Air Forces. Graph D corresponds to a failure and a quick recovery of two major Internet Service Providers.

We can clearly associate four distinct events to each of the graphs A — D. Note that graphs A and B were not merged together, despite being spatially adjacent and graphs A and C were not merged, despite having similar temporal behaviour. This demonstrates the importance of both spatial and temporal components of queries in order to separate the events properly.

In this section, we have described experiments on the AS connectivity evolving graph. Our work shows that using several queries, we are able to find subgraphs that can be related to known real-world events. In the next section we describe an experiment on another real-world dataset.

4.2 Evaluation on Enron Email Corpus

The Enron email corpus is a large set of emails collected from the Enron corporation over 3.5 years (Klimt and Yang 2004). The dataset was used in a number of studies and has been analysed from different perspectives (Diesner and Carley 2005,

Berry and Browne 2005, Rowe et al. 2007, Borgwardt et al. 2006). We built an evolving graph from the Enron corpus and used querying to analyse this dataset.

Each email in the dataset contains sender and recipients addresses and a time stamp. Borgwardt et al. (2006) turned the Enron dataset into an evolving graph. They divided the total period in which emails were collected into a number of intervals. They represented employees as vertices and put an edge in a particular snapshot if there was an email between the corresponding employees in this time interval.

The graph of Borgwardt et al. contains only 15 snapshots, which results in a large time interval per snapshot. We were not satisfied with this granularity and we built an evolving graph that is different in several ways.

First, snapshots in our graphs are taken in one week intervals.

Second, we decided to restrict the analysis to the year 2001. Shetty and Adibi (2004) reported that the number of emails is not equally distributed over time, and there is a much larger number of emails in 2001 compared to other years. In addition, it is known that this year was particularly rich in events for the Enron organization. Examples of the events are the California power crisis and the bankruptcy of Enron.

Third, we used a threshold for the number of emails. We expected an event in an organization to be reflected by a “higher than usual” email traffic between employees related to the event. Therefore we calculated the average number of emails sent within one week between two employees and set a threshold above this number. In the i^{th} snapshot of our evolving graph we put an edge if there were more than 3 emails sent between the corresponding employees within the i^{th} week.

Since there are duplicate emails in the dataset, we first pruned any duplicates. The evolving graph we built consists of 140 vertices and 244 edges. All edges are changing, i.e., there is no edge that exists in all snapshots. The graph has in total 52 snapshots (52 weeks of the year 2001).

In order to demonstrate our flexibility in constructing queries, we used three temporal patterns that were not used for the AS connectivity graph. We filtered out small results with a predicate returning true, if a graph contains more than 3 edges. We expected email discussions of the same event to have mostly synchronous starting times, i.e., within the same week. Thus we required results to be 100% synchronised within a pattern, by setting the correlation threshold to zero.

The result of each query is a set of evolving subgraphs. We manually reviewed the largest subgraphs. Each evolving subgraph may be related to event(s) in the organization, and vertices represent employees related to the event(s). Each edge corresponds to a number of emails. We analysed the contents of these emails in order to understand what event(s) they might correspond to. We present the evolving subgraphs found by different temporal patterns in Table 1. The first pattern we used is “01”. It can be interpreted as an “event occurrence”. One of the evolving subgraphs found corresponds to emails sent within week 4 (late January 2001). There was an electricity crisis in California in 2000 — 2001. In mid January 2001 a blackout affected hundreds of thousands of citizens. This blackout was followed by a declaration of a state of emergency by Governor Davis. Enron was one of the largest energy companies in the USA at that time. Therefore it is not surprising to find this event in the emails communication of

Enron employees. Subjects or bodies of almost all emails of the evolving subgraph are related to the political and other consequences of the electricity crisis in California.

Another waveform we used for querying is “0110”. This kind of behaviour can be interpreted as short term cooperation: average email traffic, followed by intensification for two weeks and decreasing afterwards. One of the graphs found indicates that such a pattern occurred in mid August 2001. A review of the emails corresponding to the subgraph showed that there were two main discussion topics. The first topic relates to a conference call with Portland. This can be concluded from some email subjects and the following text found within the emails: “... conference call with Portland”, “I have organized ... groups to attend Portland for one week ...”. The second relates to some activities related to “Cash/Prompt”. Several emails have “Cash/Prompt” keyword and one of the emails says: “I have decided to move Frank Ermis from his role as Prompt/Season ...”.

Lastly, in a “mid term cooperation” query (“011110”), one of the found evolving subgraphs is related to the daily “TRV” reports which were distributed during several weeks.

Note the variety of temporal behaviours for which we can search. We can encode different situations of interest (e.g., “short term cooperation”) with different waveforms. Note also the comprehensiveness of the discovered information: for each subgraph we can infer the main discussion topics (event), the time of pattern occurrence and the list of participants in the discussion.

In summary, our experiment on the Enron corpus shows that, as in the case with the AS connectivity graph, we are able to find evolving subgraphs that can be related to known real-world events. The following section presents our evaluations of running time for querying synthetic graphs.

4.3 Evaluation on Synthetic Graphs

The purpose of our evaluation on synthetic graphs has been to analyse how the query execution time varies under different conditions.

There are a number of factors that can affect the execution time for a query. In Section 3 we introduced two algorithms for implementing a query. We report the results of experiments on synthetic graphs for both algorithms, where appropriate.

Another factor affecting the execution time is the size of the evolving graph being analyzed. The size can be expressed as the number of vertices, edges and time snapshots. In Section 3.3 we showed that the number of snapshots and the number of edges contribute to the time complexity in a similar manner, and the number of vertices is bounded by the number of edges. Therefore in synthetic graphs, we vary only the number of edges.

The last factor that we considered is the length of a waveform in a query. The worst-case time complexity for the algorithms is $O(T \times n \times l_S)$, thus we expected a linear increase in execution time for longer waveforms. In our evaluation we used several waveforms with lengths varying from 2 to 32.

We generated random evolving graphs with the number of edges varying from 50 to 100,000. All edges are changing. Each graph has $T = 100$ snapshots. After generating the graphs we modified some edges in order to guarantee that for the queries that we try, the result is non-empty.

We implemented both algorithms in C++ and ran our queries on a PC with Intel(R) Core(TM)2 Duo E8400 @ 3GHz CPU and 3GB RAM. We repeated

Table 1: Some of the results from querying the evolving email graph. The results can be related to real world events.

Query pattern and its interpretation	Found evolving subgraphs (week numbers shown)	Selected subjects of emails, corresponding to the edges	Related event(s)
01 — "event occurrence"	late January  james.steffes kay.mann jeffrey.hodge david.delainey elizabeth.sager john.lavorato keith.holst mike.grigsby phillip.allen richard.sanders jeff.dasovich	Cheney: White House Mtg Mon On Calif. Pwr Prob...; Edison's Filing in District Court in L.A.; Governor Davis; Governor Davis names advisors; New Bill Introduced in CA Legislature.	California power crisis
0110 — "short term cooperation"	mid August  mike.grigsby frank.ermis keith.holst jason.wolfe matt.smith	West Power ...; FW: West Power Rotation; FW: Western Strategy Briefing; Frank Ermis and Matt Lenhart; Cash/Prompt and Prompt/ Season Traders.	"West Power" conference call with Portland; activities within "Prompt/Season"
011110 — "mid term cooperation"	late September — October  errol.mclaughlin larry.may dutch.quigley mike.maggi john.arnold	TRV Notification: (NG - PROPT P/L - 09/28/2001); TRV Notification: (NG - PROPT P/L - 10/01/2001); TRV Notification: (NG - PROPT P/L - 10/03/2001); TRV Notification: (NG - PROPT P/L - 10/04/2001); TRV Notification: (NG - PROPT P/L - 10/11/2001).	daily "TRV" reports

each execution time measurement several times and report averaged values.

Recall that the "Select Indexed" requires prior indexing of an evolving graph. Therefore, at first, we analysed the execution times and memory consumption required for indexing evolving graphs of different sizes. Results are presented in Table 2.

The results show that there is a roughly linear dependency of both execution time and memory consumption on the number of edges. Execution time values remain within reasonable limits. Furthermore the indexing can be performed as a background process and thus does not require an operator to wait.

In contrast, memory consumption is a critical issue. We found that when an evolving graph has more than 5,000 changing edges, indexing information requires more than 1GB of storage. We did not apply the "Select Indexed" for larger graphs, because it required more memory than the maximum file size allowed in our operating system.

Note that in the "Indexing" algorithm, only the number of changing edges contribute to the running time and memory consumption, because constant edges are pruned away before indexing. In our synthetic evaluation all edges are changing, i.e., we found that the memory consumption is an issue for graphs with more than 5,000 *changing edges*. This is quite a large number. For reference, the AS connectivity graph that we analysed contains 700 changing edges. The evolving graph we built from the Enron corpus has 244 changing edges. Furthermore, for larger graphs we still can use the "Select Basic", which does not require indexing.

The next experiment aimed to analyse the dependency of query execution time on the size of the evolving graph. Results are presented in Figure 6. We measured execution times for both algorithms on the synthetic graphs with 50 — 5,000 edges (Figure 6.a)

Table 2: Execution times and memory consumption for indexing evolving graphs with different sizes (algorithm "Indexing"). For each graph $T = 100$

Number of edges	Time (sec)	Memory (MB)
50	4	11
100	7	22
500	37	111
1,000	70	220
5,000	360	1,125

and for the "Select Basic" only on graphs with 10,000 — 100,000 edges (Figure 6.b). In all queries we used the same waveform "01010101".

The results show that "Select Indexed", when applicable, significantly outperforms "Select Basic". Another conclusion that can be drawn is that both algorithms demonstrate good scalability: roughly linear dependency of the running time on the number of edges (note the quasi-logarithmic scale of x axes).

In the last experiment we evaluated the running times for different query waveforms on the same evolving graph. We used a synthetic graph with $|E| = 1,000$ edges and $T = 100$ snapshots. Waveform lengths vary from 2 to 32. The results are presented in Figure 7.

According to the worst-case time complexity analysis in Section 3.3, running times should be longer for longer waveform lengths. In contrast, experimental results show that measured running times decrease for both algorithms. We explain this phenomenon as follows. In practice the running time is less than it would be in the worst case. This time depends on the number of the results found by the query. On average, for longer patterns there are fewer results. Therefore for longer waveforms we observe

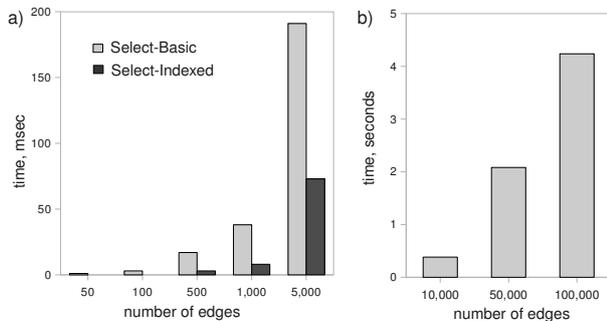


Figure 6: Execution times for querying graphs with different numbers of edges and the same number of snapshots $T = 100$. In all cases the same waveform “01010101” was used for querying. In a) two algorithms are compared. In b) only “Select Basic” is used. Note the quasi-logarithmic scale of the x axes.

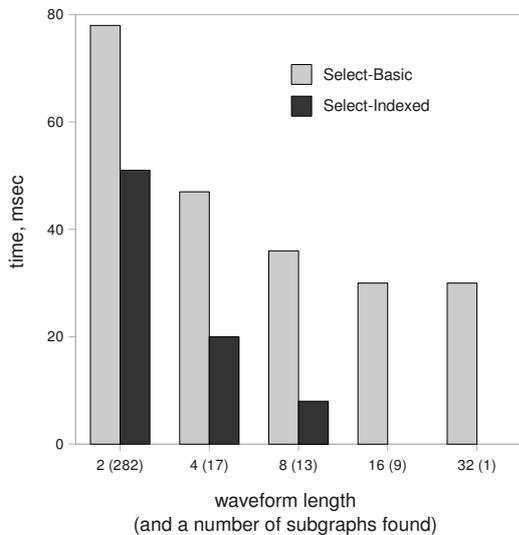


Figure 7: Execution times for different query waveforms. Two algorithms are tested. Queries are performed from the same evolving graphs with $|E| = 1,000$ edges and $T = 100$ snapshots. Note the logarithmic scale of the x axis.

decreased running times.

In summary, our evaluation on synthetic graphs demonstrates several properties of our algorithms. First, “Select Indexed” can significantly outperform “Select Basic”. On graphs with more than 5,000 changing edges, memory consumption is a critical issue for indexing. For such graphs “Select Basic” can still be used.

Second, both algorithms demonstrate good scalability: the execution time increases roughly linearly with the size of the graph.

Last, in contrast with the theoretical worst-case analysis, longer waveforms in queries result in faster query execution. This happens because for longer patterns there are fewer matching results and, in practice, running time depends more on the number of results found by a query, then on the waveform length.

The following section presents comparing and contrasting our findings on two real-world datasets with the results of previous studies.

4.4 Results Discussion

Some of the related studies report evaluation results on the same real-world datasets as we used in our experiments. This allows us to compare and contrast our findings with the work of others.

Our first experiment was performed using the Internet AS connectivity evolving graph. Chan et al. (2008) report their findings on this evolving graph. Comparing our findings, we note that by using three queries we are able to find subgraphs corresponding to the all the regions that they reported.

Consider also the performance evaluation on synthetic datasets in the work of Chan et al. (2008) and in our work. A query can be executed ten thousand times faster than enumerating all inter-correlated regions from the same graph. This is because finding inter-correlated regions essentially results in a large set of all possible query results. If we are interested in regions with a particular temporal behaviour, we need to search this set itself. In contrast, we can directly search for a required waveform by using querying.

The second dataset that we used is the Enron email corpus. There have been many works that analysed this dataset. Borgwardt et al. (2006) constructed an evolving graph from the dataset. They searched for frequent dynamic subgraphs. However they report only quantitative results, without relating the found subgraphs to events or persons.

The results of Diesner and Carley (2005) provide a deeper insight into the organizational structure of the Enron corporation. In particular they report the list of “key players” in the organization. In our results “key players” can be inferred from the subgraph topology (see Table 1). For example, from our results, we note that James Steffes and Mike Grigsby can be considered as key persons. Diesner et al. also included these employees in their list of “key players”. Note that in our study we are also able to elaborate in what particular situations these people played an important role. Diesner et al. do not present such an analysis. They also do not provide references to concrete events or discussion topics.

Berry and Browne (2005) analysed the contents of emails using non-negative matrix factorization. They report several discovered topics of discussion. Some of the topics match our findings (e.g., the California power crisis), while others differ. This can happen due to the following reasons. First, the Enron corpus consists of emails systematically collected for a subset of employees. A number of emails were sent to or received from people outside of this subset. Our purpose has been a demonstration of the capabilities of querying, rather than thorough investigation of the Enron case. Thus, in our analysis, we omitted emails that have either the sender or the recipient outside of the set of employees, for which the dataset was collected.

Second, we were searching for topics developing according to a temporal pattern. In contrast, the method of Berry et al. returns all topics, from which they reviewed the largest. For example, one of the topics we have found is the “TRV” reports. Berry et al. do not report this topic, apparently because it was not large enough. However “TRV” reports might be of particular interest as an example of mid-term cooperation patterns in Enron.

Regarding the computational efficiency, note that Berry et al. presented a method in which the input length is determined by the number of symbols in all emails. In contrast, the input length for querying is bounded by the number of emails.

In summary, our experiments on real-world and synthetic datasets demonstrate that:

- querying evolving graphs can discover real-world events, reflected in the datasets;
- querying can be performed within a reasonable amount of time even on large graphs (hundreds of milliseconds for a graph with 10,000 edges);
- performance degrades roughly linearly as the size of the evolving graph grows;
- a single query runs roughly ten thousand times faster than an approach that enumerates all inter-correlated regions (which was the approach used by Chan et al. 2008);
- querying is capable of identifying discussion topics, without the need for analysis of the contents of all emails in the Enron corpus.

5 Related Work

We categorize related work by the field of study. Our work focuses on mining evolving graphs by querying them with spatio-temporal patterns. Therefore, the main fields related to our study are: “mining evolving graphs”, “spatio-temporal patterns” and “querying”.

Mining evolving graphs by different spatio-temporal patterns has been addressed by a number of works. The pattern proposed by Chan et al. (2008) is a region of correlated spatio-temporal changes. This region is an evolving subgraph in which all edges experience similar temporal behaviour. Lahiri and Berger-Wolf (2008) “propose a new mining problem of finding periodic or near periodic subgraphs in dynamic social networks”. Jin et al. (2007) focus on evolving graphs with changing weights. They propose a pattern called “trend motif occurrence”. This is essentially a connected subgraph, in which all vertices have decreasing or increasing weights. Borgwardt et al. (2006) aim to search for frequent dynamic subgraphs, i.e., multiple occurrences of identical topological subgraphs with the same temporal behaviour.

Spatio-temporal patterns are used not only for mining evolving graphs, but also for other datasets. For example, Celik et al. (2006) introduce a new co-occurrence pattern as frequent spatio-temporal collocation of objects with different types. They propose a method for finding such patterns in a general spatio-temporal dataset. This method does not address graphs specifically. Hadjieleftheriou et al. (2005) propose a more flexible pattern: an ordered list of spatial predicates. The order (exact or relative) in the list is a temporal predicate. They describe a framework for querying such patterns from a set of trajectories. They do not consider particular issues of querying evolving graphs.

An evolving graph can be thought of as a database of strings of “0” and “1”. Querying substrings over a string database is a much studied field. Many in-memory and disk based algorithms were proposed (e.g., Kahveci and Singh, 2001, Meek et al., 2003). Usually a string edit distance and its variations are used to measure similarity between strings. However in the context of spatio-temporal queries, similarity is measured by temporal distance, using waveforms and transition sequences of compared strings. Thus it is not a trivial task to apply algorithms based on the string edit distance to querying evolving graphs. Furthermore, the spatial dimension (graph topology) is not considered in these algorithms.

Querying static graphs has been studied by a number of researchers. Zhang et al. (2009) focus on efficient searching of required topological pattern occurrences in very large graphs. He and Singh

(2008) propose a formal language for querying and manipulating graphs. They presented a graph algebra as an extension of relational algebra. They also address some challenges of pattern matching. Trissl and Leser (2006) introduce an index structure to facilitate reachability and distance queries in a graph.

The studies that address querying graphs disregard the fact that graphs may change over time. In other words, these studies do not consider evolving graphs.

The works related to mining evolving graphs do directly address the challenges and opportunities provided by temporal change. These works aim to search for different spatio-temporal patterns: inter-correlated region, periodic behaviour, trend motif occurrence, and frequent dynamic subgraphs. Each of these patterns has potentially useful practical applications. However, given the large variety of application domains and research questions one might have when mining evolving graphs, a more flexible way of defining patterns is of great interest. Therefore we introduced a framework that allows a user to define a spatio-temporal constraint according to the needs of a particular analysis.

6 Conclusions and Future Work

We have presented a novel approach for mining evolving graphs: queries for user-defined spatio-temporal patterns. Users can specify the required temporal and spatial constraints and search for matching evolving subgraphs. We formally posed the problem of querying by spatio-temporal patterns. This problem was addressed by two algorithms that implement the query: the first one is a straightforward implementation and the second uses indexing. Using indexing allows us to execute queries faster, but this optimization is achieved with the cost of increased memory consumption.

We evaluated our approach on two real-world datasets: the Internet AS connectivity graph and the Enron email corpus. We also ran experiments on synthetic evolving graphs. We discussed the obtained results and related them to other studies that used the same datasets.

We summarize the **importance of our findings** as follows:

- we evaluated querying on the datasets that relate to very common real life phenomena: Internet topology and emails in an organization;
- the datasets studied are large (up to 700 changing edges), which makes manual analysis of changes infeasible; thus there is a need for automated analysis tools;
- we are able to discover real-world events, reflected in the data, using querying;
- our implementation is fast and scalable: a query runs for hundreds of milliseconds for a graph with 10,000 edges and the running time increases roughly linearly as the size of a graph grows.

These promising results inspire us to continue our work on querying along the following directions for **future work**:

- consider operators for manipulating query results and extend our query model to a richer query language for evolving graphs;
- consider evolving graphs with weighted and directed edges; for example, for the email corpus, it would be possible to use directed edges and weight them according to the email traffic volume.

Acknowledgement

This work is partially supported by National ICT Australia and Science Foundation Ireland (SFI) under CLIQUE Strategic Cluster, grant number 08/SRC/II407. National ICT Australia is founded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

- Berry, M. and Browne, M. (2005), 'Email surveillance using non-negative matrix factorization', *Computational & Mathematical Organization Theory* **11**(3), 249–264.
- Borgwardt, K., Kriegel, H., Wackersreuther, P. and Munich, G. (2006), Pattern mining in frequent dynamic subgraphs, in 'Proceedings of the 6th International Conference on Data Mining', pp. 818–822.
- Celik, M., Shekhar, S., Rogers, J., Shine, J. and Yoo, J. (2006), Mixed-Drove Spatio-Temporal Co-occurrence Pattern Mining: A Summary of Results, in 'Proceedings of the 6th International Conference on Data Mining', pp. 119–128.
- Chan, J., Bailey, J. and Leckie, C. (2008), 'Discovering correlated spatio-temporal changes in evolving graphs', *Knowledge and Information Systems* **16**(1), 53–96.
- Cowie, J., Popescu, A. and Underwood, T. (2005), Impact of Hurricane Katrina on Internet Infrastructure, Technical report, Renesys Corporation.
- Diesner, J. and Carley, K. (2005), Exploration of communication networks from the enron email corpus, in 'Proceedings of Workshop on Link Analysis, Counterterrorism and Security, SIAM International Conference on Data Mining', pp. 21–23.
- Hadjieleftheriou, M., Kollios, G., Bakalov, P. and Tsotras, V. (2005), Complex spatio-temporal pattern queries, in 'Proceedings of the 31st International Conference on Very Large Data Bases', VLDB Endowment, pp. 877–888.
- He, H. and Singh, A. (2008), Graphs-at-a-time: query language and access methods for graph databases, in 'Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data', ACM New York, NY, USA, pp. 405–418.
- Jin, R., McCallen, S. and Almaas, E. (2007), Trend motif: A graph mining approach for analysis of dynamic complex networks, in 'Proceedings of the 7th IEEE International Conference on Data Mining', IEEE Computer Society Washington, DC, USA, pp. 541–546.
- Kahveci, T. and Singh, A. (2001), An efficient index structure for string databases, in 'Proceedings of the International Conference on Very Large Data Bases', Citeseer, pp. 351–360.
- Klimt, B. and Yang, Y. (2004), 'The enron corpus: A new dataset for email classification research', *Lecture Notes in Computer Science* **3201**, 217–226.
- Krebs, V. (2002), 'Mapping networks of terrorist cells', *Connections* **24**(3), 43–52.
- Lahiri, M. and Berger-Wolf, T. (2008), Mining Periodic Behavior in Dynamic Social Networks, in 'Proceedings of the 8th IEEE International Conference on Data Mining', IEEE Computer Society Washington, DC, USA, pp. 373–382.
- Meek, C., Patel, J. and Kasetty, S. (2003), Oasis: An online and accurate technique for local-alignment searches on biological sequences, in 'Proceedings of the 29th international conference on Very large data bases-Volume 29', VLDB Endowment, pp. 910–921.
- Rowe, R., Creamer, G., Hershkop, S. and Stolfo, S. (2007), Automated social hierarchy detection through email network analysis, in 'Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web Mining and Social Network Analysis', ACM New York, NY, USA, pp. 109–117.
- Shetty, J. and Adibi, J. (2004), The Enron email dataset database schema and brief statistical report, Technical report, University of Southern California.
- Trissl, S. and Leser, U. (2006), GRIPP Indexing and Querying Graphs Based on Pre- and Postorder Numbering, Technical Report 207, Humboldt-Universitaet zu Berlin.
- Xu, J. and Chen, H. (2004), 'Fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks', *Decision Support Systems* **38**(3), 473–487.
- Zhang, S., Li, S. and Yang, J. (2009), GADDI: distance index based subgraph matching in biological networks, in 'Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology', ACM New York, NY, USA, pp. 192–203.

Appendix A

A.1 Proof of Lemma: Correctness and Completeness of Algorithm "Select Basic"

Consider two time points $i, j : i \neq j$ and two evolving subgraphs sg_i and sg_j , containing all edges correlated with W over $[i, i + l_W - 1]$ and $[j, j + l_W - 1]$ respectively, and containing only such edges. None of the evolving subgraphs of sg_i is included in any of the evolving subgraphs of sg_j , because their temporal intervals are not included in one another. Furthermore, partitioning (which is essentially splitting into connected components) produces a set of non-overlapping and connected evolving graphs. Therefore, the evolving graphs in the output are maximal. The partitioning does not affect the temporal characteristics, thus the result of partitioning is a set of evolving subgraphs that are correlated with W . After the filtering step, the evolving subgraphs in the results set satisfy P . Therefore all evolving subgraphs in R satisfy the selection constraints and algorithm "Select Basic" is correct.

We refer to an evolving subgraph $sg \subseteq eg$, that is correlated with W over $[t, t + l_W - 1]$ and satisfies P as an *eligible subgraph*. Consider an eligible subgraph sg , which is not included in any other eligible subgraph. In algorithm "Select Basic" sg is included in sg_t (line 5). Then, after partitioning sg_t (line 7), the evolving subgraph sg becomes one of the partitions and this partition is not filtered out (line 9). Therefore all maximal eligible evolving subgraphs are included in the set of results and algorithm "Select Basic" is complete. \square