Mining Periodic Interaction Patterns in Social Networks

S.RAVI KISHAN[1] **& P.DEVI**[2]

[1]Department of CSE, VR Siddhartha Engineering College,
[2]Pursuing Master Degree in VR Siddhartha Engineering College.
Email:suraki@vrsiddhartha.ac.in, deviposani@gmail.com

Abstract

Social interactions that occur regularly typically correspond to significant yet often infrequent and hard to detect interaction patterns. To identify such regular behavior, we propose a new mining problem of finding periodic or near periodic subgraphs in dynamic social networks. We analyze the computational complexity of the problem, showing that, unlike any of the related subgraph mining problems, it is polynomial. We propose a practical, efficient and scalable algorithm to find such subgraphs that takes imperfect periodicity into account. We demonstrate the applicability of our approach on several real-world networks and extract meaningful and interesting periodic interaction patterns.

1. Introduction

Due largely to recent technological advances, one can now monitor large populations of humans, animals and networked computers, and know precisely when one entity in the population interacts with another. Whether it is an exchange of e-mail or a phone call between humans, connections made between networked computers or a GPS-tagged wild Zebra stallion associating with a specific harem [13], real-world patterns of interactions and associations can be studied and recorded at a finer time scale than ever before. In this paper, we propose a formal and practical method for identifying periodically recurring patterns from streams of interaction data, and show that the technique can be used to explore the inherent periodicity of interactions in a population.

Social networks are graphs that are used to model and analyze the structure of relationships between a set of entities. The nature and scope of social network data, however, have grown well beyond traditional applications in sociology [28] and contemporary study in physics [2, 26]. Conventional social networks are now superseded by continuous streams of dynamic interaction data, or dynamic networks, which have opened the way to new techniques for analyzing the underlying populations [2–4, 9, 22, 23].

We propose a novel data mining problem for dynamic networks: *periodic subgraph mining*, or the discovery of all interaction patterns that occur at regular time intervals. Consider two potential applications of this technique: the first, based on the premise that periodic patterns represent stable interaction patterns, is that periodic interaction patterns can be of qualitative interest in and of themselves. Ecologists, for example, tag wild herds of animals with tracking devices in order to study their movements and social patterns [19]. Periodic subgraphs in this case correspond to seasonal association or mating patterns, which are of biological interest, especially if these patterns are hidden in large quantities of effectively random animal movements and associations [13]. A second application stems from the fact that, by virtue of repeating regularly, periodic behavior can be *predictable* behavior. Dynamically mining predictable interactions from sensor logs can be used, for example, in various types of ubiquitous and mobile computing [11].

We therefore center our work around two fundamental claims necessary for both applications: first, that many social systems inherently *do* contain patterns that recur with a natural periodicity. Second, that all such patterns can be extracted for analysis from a dynamic network in an efficient and tractable manner.

Our contributions in this paper are as follows:

- We formally define periodic subgraph mining for dynamic networks in Section 2. Our definition combines concepts from frequent pattern mining [14] and earlier formulations of periodic pattern mining in other types of data (e.g. sequences [30] and eventsets [15, 17]).
- We theoretically analyze the problem in Section 3 and show that it lies in the complexity class P (polynomial), unlike the more general *frequent subgraph mining* problem. We also obtain asymptotic and exact upper bounds on the time complexity of enumerating all periodic sub-

graphs in a dynamic network.

- In Section 4, we propose a novel measure for ranking a mined periodic subgraph by how close it is to being perfectly periodic, rather than by the number of it occurs.
- In Section 5, we describe a polynomial time, parameter-free, single-pass algorithm for mining all periodic subgraphs from a dynamic network. Our algorithm also accommodates the mining of noisy periodic subgraphs in which the period is 'almost' constant.
- In Section 6, we evaluate our algorithm and the general characteristics of periodic subgraphs on four diverse, real-world dynamic networks spanning corporate executives, college students with Bluetooth cellphones, GPS-tagged wild animals and Hollywood celebrities.

2. Problem Definition and Related Work

Let $V \in \mathrm{N}$ represent a set of unique entities whose interactions are recorded over a period of time and divided into T discrete timesteps of equal duration. This type of data constitutes a *dynamic network* [22], where the objects of interest are interaction patterns between vertices and how they change over time. The analysis of dynamic networks is a relatively new field and recent work has focused on random graph models for the generation of dynamic networks [2, 22, 25], mining for frequently occurring patterns [4, 9], detecting communities and analyzing graph theoretic properties [3, 20].

In our case, we are interested in finding interaction patterns in dynamic networks that occur periodically. The approach we synthesize is based on two different problems in data mining – frequent pattern mining in transactional and graph databases [14, 18, 21], and periodic pattern mining in unidimensional and multidimensional sequences [12, 15, 17, 24, 30, 31]. By combining aspects of both approaches, we obtain a theoretically sound characterization of periodic behavior in dynamic networks, that we will show in Section 6 is also empirically plausible.

Definition 1. A *dynamic network* $\mathrm{G} = *G_1, ..., G_T$ is a time-series of graphs, where $G_t = (V_t, E_t)$ is the graph of interactions E_t observed at timestep t, among the set of uniquely labeled entities $V_t \subseteq V$.

A key characteristic of each graph $G_t \in \mathrm{G}$ is that vertices are uniquely labeled. From a computational point of view, this class of graphs has more in common with sets, and various hard graph problems such as maximum common subgraph and subgraph isomorphism are reduced to quadratic complexity in the number of nodes [10]. Interactions can be either directed or undirected, and the exact definition of what constitutes an 'interaction' depends on the application. Generally, a continuous stream of interaction data is quantized into temporal segments of equal duration; the methods of such quantization are beyond the scope of this paper. The types of data that we deal with, however, have natural quantizations such as hours or days.

Before we formally define the problem of mining periodic subgraphs, we recall some concepts from the field of frequent pattern mining and extend them to dynamic networks.

Definition 2. For an arbitrary graph $F = (V_f \subseteq V, E_f \subseteq V_f \times V_f)$, its *support set* in G is the set of timesteps where F is a subgraph of G_t, denoted $F \pm G_t$.

$$S(F) = \{t : F \pm G_t\}$$

F is a *frequent subgraph* of G if $|S(F)| \geq \sigma$ where $1 \leq \sigma \leq T$ is a user-defined *minimum support* threshold.

Let $F^{(\sigma)}$ be the set of all frequent subgraphs of G at minimum support σ. Definition 2 implies that for any frequent subgraph $F \in F^{(\sigma)}$, all its subgraphs are also frequent and thus are also in $F^{(\sigma)}$. This is the *downward closure* property and is the basis of early pattern mining algorithms like the *a priori* algorithm [1]. Naturally, there is a large amount of redundant information in such a set. The concepts of maximal and closed frequent subgraphs aim to reduce the size of $F^{(\sigma)}$ without losing much (or any) information.

Definition 3. A frequent subgraph $F \in F^{(\sigma)}$ is *maximal* if there is no other frequent subgraph $F \in F^{(\sigma)}$ where $F \prec F$. $F^{(\sigma)}_{max} \subseteq F^{(\sigma)}$ is the set of maximal frequent subgraphs.

Definition 4. A frequent subgraph $F \in F^{(\sigma)}$ is *closed* if it is maximal at some support $\sigma \geq \sigma$ [27]. $F^{(\sigma)}_{closed} \subseteq F^{(\sigma)}$ is the set of closed frequent subgraphs.

In Figure 1, $F_1 = \{(1, 2), (1, 3), (1, 4), (1, 5)\}$ is the only frequent maximal subgraph at $\sigma = 2$. At the same minimum support, there are three closed frequent subgraphs: $F_1, F_2 = \{(1, 2), (1, 3)\}$ and $F_3 = \{(1, 4), (1, 5)\}$, the latter two of which are maximal at $\sigma = 3$. Note that $F^{(\sigma)}_{max} \subseteq F^{(\sigma)}_{closed} \subseteq F^{(\sigma)}$, and that $F^{(\sigma)}_{closed}$ can be exponentially smaller than $F^{(\sigma)}$. Furthermore, both $F^{(\sigma)}_{max}$ and $F^{(\sigma)}$ can be recovered from $F^{(\sigma)}_{closed}$ (albeit not in polynomial time). We thus restrict our attention to closed subgraphs only, as they represent in a certain sense coherent and complete interaction patterns.

Definition 5. A *periodic support set* or *periodic subgraph embedding* (PSE) of an arbitrary graph $F = (V_f, E_f)$ in G, where $V_f \subseteq V$, is a maximal, ordered set of timesteps where F is a subgraph of G_t, such that the difference between consecutive timesteps in the set is constant.

$$S_P(F) = *t : F \pm G_t, \quad \text{and} \quad \forall i : t_{i+1} - t_i = p$$

the term 'purity' to refer to average purity for the remainder of this paper.

Definition 10. The *average purity* of a subgraph $F = (V, E)$ is the average purity of all of its edges.

$$avgPurity(F) = \frac{1}{|E|} \sum_{e \in E} purity(e)$$

3. The Algorithm

We now present an efficient, single-pass, polynomial time and space algorithm for mining all closed periodic subgraphs in a dynamic network. Our algorithm does not require any parameters to be set but *optionally* accepts the following:

1. Minimum support threshold $\sigma \geq 2$ (default: 2).
2. Minimum and maximum period P_{min} and P_{max} (default: unrestricted).
3. Maximum jitter in period $J \geq 0$ (default: 0).

Although there is a natural bound on the maximum period if the number of time steps T is finite and known *a priori*, our algorithm is designed for cases when this information is not available, such as in streaming sensor data. Unrestricted period size in such cases places a very large computational burden on the algorithm as the entire dataset has to be retained. The optional user-defined P_{max} parameter limits the maximum period of mined patterns and results in a truly online algorithm.

3.1. Mining PSEs using a Pattern Tree

The foundation of the algorithm is a *pattern tree* that maintains information about all patterns that are either currently periodic or could become periodic at a future timestep. As each new timestep is read, the pattern tree is traversed and updated with the information. Any patterns that are no longer periodic are flushed, and new periodic patterns are possibly created.

The algorithm maintains two data structures: the pattern tree and a *subgraph hash map*. Each node in the pattern tree contains a subgraph and a *descriptor* for each closed periodic embedding of the subgraph encountered. Formally, a descriptor for a subgraph F is an ordered pair $D = *S = S_P(F), p$, where S is the periodic support set of the embedding of F and p the period. Let `next(D)` represent the timestep at which F is next expected, i.e. the last element of the support set plus the period.

The structure of the pattern tree is subject to a single constraint: all descendants of a node F represent proper subgraphs of F, but not all subgraphs of F are necessarily its descendants. This property allows the tree to be built and manipulated quickly and represented using very little space. Since the pattern tree is traversed for each new observation, once a new observation is known not to have any common subgraph with a particular node, all descendants of that node can be eliminated from the tree traversal. Figure 3 shows an example of a pattern tree.

The second component of the algorithm, the subgraph hash map, associates an arbitrary subgraph with its node in the tree, if one exists, in amortized constant lookup time. This is used by the update algorithm, and due to the unique node labels of each graph in the dynamic network, hashing subgraphs is efficient.

Proposition 5. The time complexity of hashing a periodic subgraph $F = (V, E)$ of a dynamic network is at most the complexity of hashing a string of length $|V| + |E|$.

Proof. Since $V \in \mathbb{N}$, any given subgraph contains at most one node $v = n$ for any $n \in \mathbb{N}$. Thus, each edge is uniquely identified by an arbitrary mapping $\tau : v_1 \times v_2 \to \mathbb{N}$. Since no self-loops are allowed in the input, a singleton vertex v can be represented as $\tau(v, v)$. Applying τ to every edge and singleton vertex in a subgraph maps it to a set of natural numbers, which can be hashed as a string. □

Redundancy in mined patterns can be reduced by introducing the notion of pattern *subsumption*. For example, a periodic embedding of subgraph F at period 2 with support 5 also contains an embedding at period 4 with support 3, but the latter contains no new information given then former. Thus, the period 2 pattern subsumes the period 4 pattern. Note that subsumption does not strictly fall within Definition 6.

Definition 11. Given two descriptors, $D_1 = *S_1, p_1$ and $D_2 = *S_2, p_2$, for the same periodic closed subgraph, D_1 *subsumes* D_2 if $S_2 \subseteq S_1$ and $p_2 = k \cdot p_1$ for some integer $k > 0$.

Although the algorithm is designed to operate in a single pass of the data, if the P_{max} parameter is kept unrestricted, the entire dataset will be retained in memory as a subgraph of any timestep *might* become periodic in the future. With a restricted P_{max}, only the relevant portion of the data will be retained.

3.2. Update Algorithm

We now describe the update algorithm for the pattern tree. Starting with an initially empty pattern tree, at timestep t the algorithm reads the next graph G_t from the input stream and traverses the pattern tree to update nodes with the new information. At any point in the execution of the algorithm, a complete list of periodic subgraphs seen so far can be obtained from the tree.

For each G_t, we selectively traverse the tree in a breadth-first manner. We end each tree update by ensuring that a node for G_t in its entirety exists in the tree with an anchor descriptor for timestep t. This accounts for the possibility that G_t is the first occurrence of a periodic subgraph. During the breadth-first traversal of the tree, one of the following three conditions hold at each tree node N [1]:

Update descriptors If $N \pm G_t$, then N has appeared in its entirety at timestep t. For all descriptors D, if `next(D)` $= t$, then t is added to the support set. If `next(D)` $< t$, then the descriptor is removed from the tree and flushed to the output stream if its support is greater than σ. Finally, any anchor descriptors at N are used to generate new periodic descriptors if the resultant period $p \leq P_{max}$.

Propagate descriptors Let $C = N \cap G_t = \emptyset$ be the non-empty maximal common subgraph of N and G_t. A subgraph C of N is present at timestep t, and if N has any descriptors D such that `next(D)` $= t$, then the node for C receives a copy of D if it is not subsumed by an existing descriptor at C. If a node for C does not already exist in the tree (determined using the hash map), it is created as a child of N. Figure 3 illustrates this case. At timestep 5, there is no node for the subgraph $F_3 = \{(1, 4), (1, 5)\}$, although one exists for the larger $F_1 = \{(1, 2), (1, 3), (1, 4), (1, 5)\}$. However, F_3 has been present at timesteps 1 and 3 as well, so it needs to receive a copy of the descriptor $D = *S = \{1, 3\}, p = 2$.

Dead subtree If C is empty, then G_t and N have no common subgraph. Furthermore, no child of N will have any common subgraph with G_t either, since they are all subgraphs of N. The subtree rooted at N is therefore eliminated from the rest of the tree traversal.

Figure 3 shows the pattern tree at each timestep during the execution of the algorithm on the network from Figure 1. For brevity, we have described a very basic version of the full algorithm. Clearly, nodes and descriptors expire after a certain amount of time and can be deleted from the tree, improving efficiency. Furthermore, subsumption of redundant descriptors can be performed efficiently while the algorithm is running. The basic outline described here, however, is efficient even without these optimizations.

To output all periodic patterns at any point during the execution of the algorithm, we traverse the tree and print all descriptors $D = *S, p$ and their associated subgraphs where $|S| \geq \sigma$.

3.3. Complexity Analysis

There are two types of nodes in the tree: those with only an anchor descriptor, and those with valid periodic descriptors. The latter type represent periodic closed subgraphs with support greater than or equal to 2. Since each descriptor corresponds to a unique periodic embedding, the number of non-anchor descriptors (and hence the number of such nodes) is bounded by the number of periodic closed subgraphs at $\sigma = 2$, or $O(T^2 \ln T)$. At most one anchor descriptor is added per timestep, so the asymptotic bound on the total number of nodes does not change. Since the tree is traversed exactly once per timestep, the overall worst-case time complexity for the algorithm is $O((V + E)T^3 \ln T)$. The maximum period parameter (if set) greatly reduces the time and space complexity of the algorithm.

[1] When referring to a pattern tree node, we generally refer to the periodic subgraph which it contains.

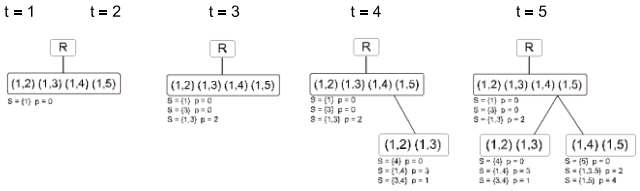


Figure 3. Pattern tree for Figure 1.

Algorithm 1 UPDATETREE(G_t)
Require: G_t is the graph of timestep t

```
1:  Q ← new queue
2:  push(Q, root.children)
3:  while N ← pop front(Q) do
4:      C ← G_t ∩ N
5:      if C is not empty then
6:          if N ± G_t then
7:              UPDATEDESCRIPTORS(N)
8:          else
9:              W ← FINDNODE(N) or NEWNODE(N, C)
10:             PROPAGATEDESCRIPTORS(N, W)
11:         end if
12:         push(Q, children(N))
13:     end if
14: end while
15: W ← FINDNODE(G_t) or NEWNODE(root, G_t)
16: Add anchor descriptor for G_t to W.
```

3.4. Jitter Heuristic

The jitter heuristic allows the detection of patterns with periods that are 'almost' constant. For a subgraph with period p, we allow a distance of $p \pm J$ timesteps between consecutive occurrences instead of exactly p. In cases where more than one occurrence of a subgraph satisfies this criteria, the occurrence that minimizes the time difference from the expected position is chosen. We note that the jitter heuristic does not preserve the theoretic complexity bounds of the algorithm, but is nonetheless efficient in practical

cases as we will demonstrate in Section 6.

4. Experimental Evaluation

We use four real-world dynamic social networks to evaluate our algorithm and the general characteristics of periodic subgraphs.

41. Datasets

The tested dynamic social networks were collected using a variety of sources and cover a range of social dynamics.

Enron E-mails The Enron e-mail corpus is a publicly available database of e-mails sent by and to employees of the now defunct Enron corporation.[2] Timestamps, senders and lists of recipients were extracted from message headers for each e-mail on file. We chose a day as the quantization timestep, with a directed interaction present if at least one e-mail was sent between two individuals on a particular day.

Plains Zebra Social interactions of Plains zebra (*Equus burchelli*) in Kenya were recorded by direct observations made by behavioral ecologists from Princeton University [13]. The data is made from visual scans of the populations, typically once a day over periods of several months. Each entity is a Plains zebra and the interactions represent association as determined by GPS spatial proximity and the domain knowledge of ecologists.

Reality Mining Cellphones with proximity tracking technology were distributed to 100 students at the Massachusetts Institute of Technology over the course of an academic year [11]. The timestep quantization was chosen as 4 hours [8].

IMDB Celebrities The Internet Movie Database (IMDB)[3] maintains a large archive of tagged and dated photographs of individuals associated with the production of commercial entertainment, including actors, directors and musicians. One might reasonably assert that a degree of social association exists between people photographed together by the popular press. Thus, similar to the methodology of the Plains Zebra sightings, we collected metadata on 45,477 photos with two or more people, which collectively represents a partial structure of the social network of people associated with the entertainment industry. The quantization period was one day.

4.2. Experimental Setup

We implemented our algorithm in C++ and ran it on a dual-core AMD Athlon 64 system with 2 GB of RAM and Linux kernel 2.6.24. The subgraph hash map was implemented using the Google `dense_hash_map` library[4], which is optimized for speed over memory usage. For frequent closed subgraph mining, we converted the dynamic networks to transactional itemsets using Proposition 5 and used the open-source MAFIA algorithm [7] running on an Intel Xeon quad-core server with 24 GB of RAM and Linux kernel 2.6.22. Since mining frequent subgraphs at low minimum support is generally intractable, we started at a very high minimum support value and progressively reduced it until either the size of the mined pattern file exceeded 512 MB or the algorithm did not terminate after 5 days.[5]

Dataset	Vertices	Timesteps	Avg. density
Enron	82,614	2,588	0.028 ± 0.064
IMDB	15,011	13,967	0.22 ± 0.23
Plains Zebra	313	1,276	0.31 ± 0.27
Reality Mining	100	2,940	0.23 ± 0.17

Table 1. Dataset characteristics

We first ran a series of experiments on our algorithm with $\sigma = 3$ and no jitter, *i.e.* mining only perfectly repeating patterns. We then ran a second set of experiments with $\sigma = 3$ and variable amounts of jitter. For Enron and IMDB, we chose a jitter of $\pm$ 2 days so that the resulting 5-day slack window would capture monthly and annual patterns, *e.g.* those that occur on the first Monday of every month. Since Reality Mining is a relatively dense and heavily periodic dataset, we chose a minimal jitter value of $\pm$ 1 timestep, or $\pm$ 4 hours. For the Plains Zebra dataset, we chose the jitter to be the average time between consecutive observations.

Table 2 shows the performance of our algorithm and the number of periodic closed subgraphs mined from each dataset. The frequent subgraph mining algorithm took on the order of days to complete even at much higher minimum support values than 3. However, since it is inherently an exponential time algorithm, as compared to our polynomial time periodic subgraph miner, we do not compare their relative performances. Note that the P_{max} parameter, not the minimum support, is the determining factor in the scalability of the algorithm. We left P_{max} unrestricted, so Table 2 represents worst-case performance in this regard.

4.3. Results

We return to our two initial claims: that dynamic social networks have inherent periodicity, and that these periodic interaction patterns can be extracted in an efficient and tractable manner. We also comment on the usefulness

[2] Available at http://www.cs.cmu.edu/ ~enron/

[3] http://www.imdb.com

[4] http://code.google.com/p/google-sparsehash/

[5] The time and space requirements of the MAFIA algorithm grew exponentially with decreasing σ, as is theoretically expected.

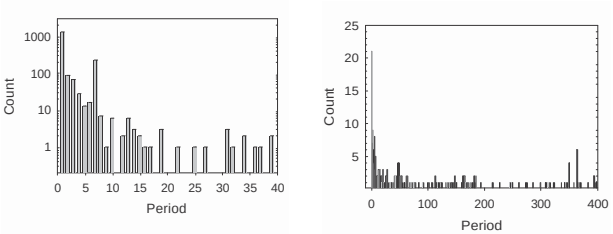
Dataset	No jitter			Variable jitter			
	Time (s)	*Memory (MB)*	*# Patterns*	*Jitter*	*Time (s)*	*Memory (MB)*	*# Patterns*
Enron	53.5	157	84,017	$\pm$ 2	150	180	863,112
IMDB	1.76	29	68	$\pm$ 2	1.89	29	276
Plains Zebra	3.56	27	2,241	$\pm$ 6	5.14	32	34,887
Reality Mining	156	110	98,258	$\pm$ 1	342	120	227,441

Table 2. Performance of the periodic subgraph mining algorithm at $\sigma = 3$.

of our proposed measure and on the qualitative properties of some interesting mined subgraphs.

4.3.1 Inherent Periodicity

Figure 4 shows the distribution of periods of patterns mined from the Enron and IMDB datasets. For Enron, we restrict our attention to patterns with a high average purity, i.e. patterns which are likely to capture truly periodic behavior. We note that daily interaction patterns are the most prevalent periodic patterns, followed by weekly patterns as manifested by the clear peak at $p = 7$. For the IMDB dataset, we notice a similar peak at about $p = 364$. This can be explained by celebrity sightings at annual events – awards shows, for example. Thus, our algorithm captures the inherent periodicity in the datasets with no prior knowledge and shows promise as a tool for exploratory analysis. The Plains Zebra dataset showed a wide range of periodicities, as one might expect of animal behavior.



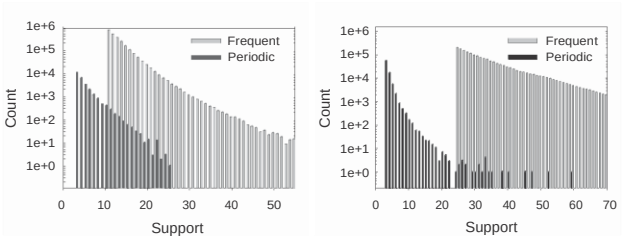
(a) Enron: $J = 0$, $avgPurity > 0.8$ (b) IMDB: $J = 2$

Figure 4. Number of patterns at each period

4.3.2 Tractability

Table 2 demonstrates that our algorithm is eminently tractable in terms of execution time and space usage. However, periodic subgraph mining is also tractable in the sense that, unlike frequent subgraph mining, it does not generate an overwhelming number of patterns. Figure 5 shows characteristic examples of the support distribution of frequent subgraphs compared to periodic subgraphs. Ignoring the fact that we were unable to mine frequent subgraphs at supports below 25 for Enron and 11 for Plains (which manifests in the left-truncated histograms for frequent subgraphs in Figure 5), we can see that periodic subgraphs are much fewer and exist at lower support levels than frequent subgraphs. Thus, under practical circumstances, the majority of this important class of patterns would be out of reach of frequent subgraph mining algorithms.

Figure 6 shows the size of the pattern tree at each timestep for the Enron and Plains datasets on a logarithmic scale. It can be seen that the actual tree size is a small fraction of the theoretical upper bound, even when the jitter heuristic is used. Furthermore, limiting the maximum period of mined patterns has a large impact on reducing the tree size, as expected.

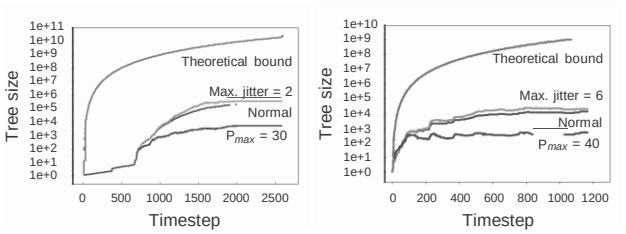


(a) Plains: $J = 6$ (b) Enron: $J = 0$

Figure 5. Partial view of support distribution of frequent (gray) and periodic (black) patterns. The frequent pattern distribution is left-truncated due to intractability.

4.3.3 Qualitative Analysis

We now turn our attention to some qualitatively interesting periodic subgraphs discovered by our algorithm illustrating a range of periodic behavior. Figure 7(a) illustrates a somewhat complex pattern from the IMDB photo database that repeated approximately every week. Although the support is relatively low, what is interesting about this subgraph is the repeated non-trivial grouping of people, all of whom turned out to be contestants on a weekly 'reality television' show. Figure 7(b) is also from the IMDB database and is an approximately annually repeating pattern. The three indi-



(a) Enron (b) Plains

Figure 6. Number of pattern tree descriptors compared to theoretical bound.

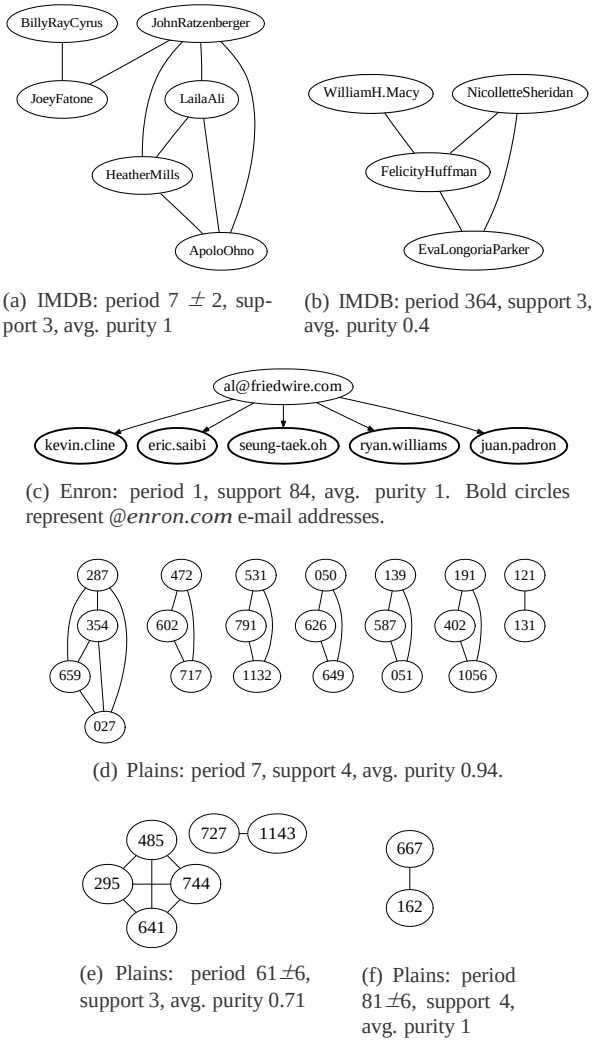
viduals in the clique are actresses in a popular (circa 2004) television show, while the fourth node is the spouse (as of 2008) of one of the actresses. Given this context, the low average purity of the pattern is to be expected as the three actresses are very likely to have appeared together in between what are likely to be award shows. The nontrivial links in such patterns are particularly interesting and are indicative of the show's progression or relationships other than co-starring.

The subgraph shown in Figure 7(c) has the highest periodic support in the Enron dataset, repeating every day for 84 consecutive days, including weekends. This is representative of a large number of similar periodic patterns in Enron, in which one person emails a group of people with periods ranging from 1 to 14 days. As shown earlier in Figure 4, weekly emails seem to be particularly popular in a corporate setting such as this, and could be used to infer functional communities within the corporation.

Finally, we turn to the Plains Zebra dataset and to one of the most intriguing patterns shown in Figure 7(d). Although it is quite likely that the period of 7 days is an artifact of the manner in which the population was sampled, the high purity of the pattern indicates that this is a relatively stable grouping. It is also by far the largest and most repetitive such pattern, parts of which are periodic at other times as well. In contrast, the subgraphs that repeat over multiple months are shown in Figures 7(e) and 7(f). Although the support of the latter two patterns is relatively low, the high purity of Figure 7(f) stands out and is representative of a large number of small but highly periodic patterns. Moreover, all the patterns are of interactions of stallion male Zebras and correspond to their harems grouping for a period of time. Such groupings are indeed considered more stable for short periods of time than bachelor associations [13].

5. Conclusion

We have proposed and formalized the *periodic subgraph mining* problem for dynamic networks and analyzed the



(a) IMDB: period 7 $\pm$ 2, support 3, avg. purity 1

(b) IMDB: period 364, support 3, avg. purity 0.4

(c) Enron: period 1, support 84, avg. purity 1. Bold circles represent *@enron.com* e-mail addresses.

(d) Plains: period 7, support 4, avg. purity 0.94.

(e) Plains: period 61$\pm$6, support 3, avg. purity 0.71

(f) Plains: period 81$\pm$6, support 4, avg. purity 1

Figure 7. Examples interesting periodic subgraphs.

computational complexity of enumerating all periodic subgraphs. We have shown that there are at most $O(T^2 \ln \frac{T}{\sigma})$ closed periodic subgraphs at minimum support σ in a dynamic network of T timesteps. Furthermore, we have described a polynomial time, one-pass algorithm to mine all periodic subgraphs, including a 'jitter' heuristic for mining subgraphs that are not perfectly periodic. We have also proposed a new measure, *purity*, for ranking mined subgraphs according to how perfectly periodic a subgraph is. We have demonstrated our algorithm on four real-world dynamic social networks, spanning interactions between corporate executives, college students, wild Zebra and Hollywood celebrities. Our algorithm efficiently mines all periodic patterns, and is a provably tractable and meaningful alternative to using frequent subgraph mining to look for

interesting patterns in dynamic networks. We have shown that dynamic social networks contain inherently periodic patterns, and our technique shows promise for exploratory analysis of natural periodicities.

.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, pg. 487–499, 1994.

[2] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311(3–4):590–614, 2002.

[3] T. Y. Berger-Wolf and J. Saia. A framework for analysis of dynamic social networks. In *Proc. of the 12th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pg. 523–528, 2006.

[4] K. M. Borgwardt, H.-P. Kriegel, and P. Wackersreuther. Pattern Mining in Frequent Dynamic Subgraphs. In *Proc. of the 6th IEEE Intl. Conf. on Data Mining*, pg. 818–822, 2006.

[5] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In *Proc. of the 19th Annual Symp. on Theoretical Aspects of Computer Science*, pg. 133–141, 2002.

[6] B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *Proc. of the 7th IEEE Intl. Conf. on Data Mining*, pg. 63–72, 2007.

[7] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. *Proc. of the 17th Intl. Conf. on Data Eng.*, pg. 443–452, 2001.

[8] A. Clauset and N. Eagle. Persistence and Periodicity in a Dynamic Proximity Network. DIMACS/DyDAn Wkshp. on Comput. Methods for Dynamic Interaction Networks, 2007.

[9] P. Desikan and J. Srivastava. Mining Temporally Evolving Graphs. In *Proc. of WebKDD 2004*, pg. 22–25, 2004.

[10] P. J. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl. *On Graphs with Unique Node Labels*, *LNCS* vol. 2726, pg. 409–437. Springer Berlin, 2003.

[11] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[12] M. G. Elfeky, W. Aref, and A. Elmagarmid. Periodicity detection in time series databases. *IEEE Trans. on Knowledge and Data Engineering*, 17(7):875–887, 2005.

[13] I. R. Fischhoff, S. R. Sundaresan, J. Cordingley, H. M. Larkin, M.-J. Sellier, and D. I. Rubenstein. Social relationships and reproductive state influence leadership roles in movements of plains zebra, *Equus burchellii*. *Animal Behaviour*, 73(5):825–831, May 2007.

[14] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent Pattern Mining: Current Status and Future Directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

[15] J. Han, Y. Yin, and G. Dong. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *Proc. of the 15th Intl. Conf. on Data Engineering*, pg. 106–115, Los Alamitos, CA, 1999. IEEE Computer Society.

[16] H. He and A. Singh. Graphrank: Statistical modeling and mining of significant subgraphs in the feature space. In *Proc. of the 6th Intl. Conf. on Data Mining*, pg. 885–890, 2006.

[17] K.-Y. Huang and C.-H. Chang. SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporal Databases. *IEEE Trans. on Knowl.and Data Eng.*, 17(6):774–785, 2005.

[18] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proc. of the 4th Eur. Conf. on Principles of Data Mining and Knowl. Disc.*, pg. 13–23, 2000.

[19] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. *ACM SIGPLAN Notices*, 37(10):96–107, 2002.

[20] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proc. of the 32nd annual ACM Symp. on Theory of Comput.*, pg. 504–513, 2000.

[21] M. Kuramochi and G. Karypis. Frequent subgraph discovery. *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, pg. 313–320, 2001.

[22] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. of the 11th ACM SIGKDD Intl. Conf. on Knowl. Disc. in Data Mining*, pg. 177–187, 2005.

[23] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. of the 12th Intl. Conf. on Inf. and Knowl. Management*, pg. 556–559, 2003.

[24] S. Ma and J. L. Hellerstein. Mining partially periodic event patterns with unknown periods. In *Proc. of the 17th Intl. Conf. on Data Eng.*, pg. 205–214, 2001.

[25] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):25102, 2001.

[26] M. E. J. Newman. The structure of scientific collaboration networks. *PNAS*, 98:404–409, 2001.

[27] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

[28] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[29] G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proc. of the 10th ACM SIGKDD Intl. Conf. on Knowl. Disc. and Data Mining*, pg. 344–353, 2004.

[30] J. Yang, W. Wang, and P. S. Yu. Infominer+: Mining partial periodic patterns with gap penalties. In *Proc. of the 2002 IEEE Intl. Conf. on Data Mining*, page 725, 2002.

[31] J. Yang, W. Wang, and P. S. Yu. Mining asynchronous periodic patterns in time series data. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):613–628, 2003.

