Discovering Periodic Patterns using Supergraph in Dynamic Networks

Sajal Halder*, Yongkoo Han and Young-Koo Lee
Department of Computer Engineering, Kyung Hee University, Republic of Korea.
(sajal,ykhan,yklee)@khu.ac.kr

Abstract
In dynamic networks, interactions that occur periodically express especially significant meaning. However, these patterns occur infrequently, so it is difficult to detect among mass data. To identify such periodic patterns in dynamic networks, we propose single pass supergraph based periodic pattern mining technique that is polynomial unlike most graph mining problems. The proposed technique stores all entities in dynamic networks only once and calculates common sub-patterns only one time at each timestep. The performance study shows that SPBMiner method is time and memory efficient compared to others. The memory efficiency of our approach is total timesteps independent.

Keywords: Periodic patterns, Dynamic Networks, Supergraph.

1. Introduction
Dynamic networks are dreadfully powerful model for representing time varying systems that interactions among entities change time to time. It has gained current researchers attention because of its various applications such as human societies and behavior analyses, wild animal communities’ behavior matching, social networks analysis, and the mobile cell users behavior analysis. Among the analysis of these dynamic networks, finding periodic patterns is the most attractive and conveys very meaningful information yet often-infrequent pattern.

Periodic patterns mining in dynamic networks introduced by Lahiri and Berger-Wolf in [7, 6] respectively. Their proposed single pass PSEMiner algorithm finds periodic patterns in polynomial time. In this process, the pattern tree has been created that maintains all periodic patterns seen up to timestep t, and it tracks periodic or future periodic patterns. Each node indicates unique pattern. This process creates many unessential tree nodes these are time consuming and graph traversed exhaustive pattern tree. Apostolic et al. [2] have proposed another algorithm ListMiner, which solves unessential tree nodes creation problem. This method maintains a list structure. At timestep t, the graph \(G_t\) is read, the projection \(\pi_{\text{pm}}\) list is updated using \(<G_m,G_{m+p},G_{m+2p}, ...>\) among the dynamic networks, where \(p\) is period and \(m = t \mod p\). The approach is much faster than previous approach because it creates less number of list nodes and requires less number of list node traversing. However, the number of list nodes is so large and it stores same graph different times that is memory and time consuming.

Therefore, we need efficient periodic patterns mining technique that reduces exhaustive visit and stores entities only once. Supergraph is one solution which stores all common and uncommon entities (vertexes and edges) only one time and find common sub-patterns only one computation. The contribution of this paper is that we propose a supergraph based periodic patterns mining technique, which is faster than existing works and it needs less memory that is total time independent.

The remaining part of this paper is organized as follows. In section 2 the related works are described. Proposed methodology has been proposed in section 3. The effectiveness and efficiency of proposed method are shown in section 4. In section 5, we conclude the paper with a direction of future work.

2. Related Works
Periodic pattern mining in dynamic networks is very interesting research in current world. The most proposed techniques for mining periodic pattern discuss with unstructured data such as a sequence or multiple sequences data. Ozden et al. [8] introduced discovering periodic pattern association rules that show regular cyclic pattern over time, while Bettini et al. [3] presented a technique to find temporal patterns in time sequence. Partial periodic pattern mining is another very interesting research since Han et al. [5] proposed partial time series behavior mining methods. Yang et al. [9] proposed an asynchronous periodic pattern mining model that mine all patterns whose periods cover a range within...
a subsequence and whose occurrences may be shifted due to disturbance in time series data. In [10] Yang et al. introduced an efficient algorithm, InfoMiner, to mine surprising patterns and associated subsequences based on the information gain.

Lahiri and Berger-Wolf in [7, 6] introduced a new mining problem to find periodic patterns in dynamic social networks. Periodic patterns occur regularly in networks those change over time. These patterns do not exist of all time intervals, which reason they mine all periodic subgraphs that occur in a minimum number of times. They deal with the concept of closed subgraph mining that has been widely used in frequent pattern mining [4]. Occam’s Razor principle is followed for parsimony closed pattern mining. Finding periodic patterns in dynamic networks proposed PSEMiner algorithm that is polynomial unlike many related subgraph and itemset mining problem. Apostolico et al. proposed the speedup method ListMiner for periodic pattern subgraph mining in [2]. Proposed method finds patterns based on projected timesteps graph that solve unused tree node problem. Each projection π_m,p creates a list. The number of list depends on maximum period \( P_{\text{max}} \) because process mine all periodic patterns up to \( P_{\text{max}} \) and the number of list nodes in the list depends on total timesteps that is also time-consuming process.

Finally, our work is inspired by supergraph based periodic pattern mining, which is concerned with the discovery of periodic patterns that occur is dynamic networks. We propose supergraph based algorithm SPBMiner that space complexity does not depend on time. It only depends on maximum period \( P_{\text{max}} \) and average entities (vertexes and edges) number in the networks. The time complexity of SPBMiner is better than PSEMiner and ListMiner.

3. Discovering Periodic Patterns Using Supergraph

Given a dynamic network (DG) and a minimum support threshold \( \sigma \geq 2 \), we have to mine all periodic patterns (PP) that satisfy the minimum support. The set of vertexes \( V \subset N \) are uniquely defined in dynamic networks and the interactions/edges \( E \) among vertexes \( V \) are changed time to time. Interactions between vertexes may be directed or undirected and are supposed to have been recorded over a period of \( T \) isolated timestamps. The only essential is that a timestep represents a meaningful amount of real time where the periodicity of mined patterns will be set of chosen timestep.

**Definition 1.** Supergraph (SG): Supergraph is a graph stores information of all graphs with the common patterns of the graphs being stored only once.

**Definition 2.** Periodic Pattern (PP): Given a DN and an arbitrary pattern \( F \), the PP is a pair of \( < F, \text{Sp}(F) > \), if support set of \( F \) is \( |\text{Sp}(F)| \geq \sigma \) (min sup).

We now introduce our proposed algorithm for discovering all periodic patterns in dynamic network. At each timestep dynamic network is read, we maintain a supergraph update process for all entities of networks seen up to timestep \( t \). This supergraph maintains two kinds of data structure for each entity. One is the time set (TS) that stores active time of entity. When entity \( E \in F \), appears in graph \( G_t \), is active at time \( t \). Other one is a list of descriptor represents entity periodicity likes as descriptor in [7]. Once descriptors with entities cease to be periodic, they are flushed from the supergraph and insert to periodic hash table as a periodic entity. Periodic hash table is one kind of special data structure that stores entities based on periodicity phase, period and support. If a group of entities matched out same time and their period and support are same then store together and build periodic patterns.

Now we describe how to update supergraph information that is the core part of our process for periodic pattern mining. Initially the supergraph \( SG \) is empty. At timestep \( t \), graph \( G_t \) is read. The common pattern \( F = SG \cap G_t \) are updated into SG. An entity \( E \in F \) updates timestep and descriptor set including addition, deletion and modification. Those entities in \( G_t \) do not exist in SG then add with SG. We find periodic entities after checking supergraph descriptors that are flushed and satisfy minimum support. If descriptor support is greater than the minimum support, we add those entities based on period and support. For this purpose, we create periodic hash key combining starting phase, period, and support and stores entities as periodic hash values.

Algorithm 1 shows the proposed SPBMiner algorithm. The main algorithm performed from line 2 to line 9. Initially supergraph is empty. At each time step \( t \), supergraph is updated based on current graph common entities mentioned at line 4. If current graph entities do not exist in supergraph then add those entities to supergraph at line 5. In this process, current graph is compacted with supergraph. However,
there are entities in supergraph that are not appears in current graph at all timesteps. These entities update occur at line 6-7. Finally, we mine last time periodic entity by line 8.

Algorithm 1. SPBMiner({G1,G2, ..., GT},σ)

Data: G1,G2, ...,GT : Dynamic behavior graphs at timestep 1, 2, ..., T; σ : min sup;
Result: Parsimonious Periodic Behaviors Sets
1. SG = ϕ /* Behavior supergraph is empty */;
2. for t = 1 to T do
3. for all E Є Gt do
4. if E Є SG then Update Entity(SGE, t, σ);
5. else Add Entity(SG,E);
6. for all E Є SG and E Є Gt do
7. Update Entity(SGE, t, σ); /* Update entity*/;
8. Mine_Last_T imestep_Behaviors(SG, σ);

The Update Entity(SGE, t, σ) procedure tracks entities periodic set and support information that is significantly important for mining periodic patterns in dynamic networks. In this process, each entity E is updated including descriptor set and timeset updating, deleting and creating.

Discovering periodic pattern time and space complexity depend on common pattern mining between supergraph and current graph that needs $O(V+E)$ times and each entity timeset, and descriptors update time. For time set it needs $P_{max}$ time and descriptor update needs maximum $(P_{max})^2$ times according lemma 2. This yields the total time complexity is $O((V+E)T(P_{max})^2)$ and space complexity is $O((V+E)(P_{max})^2)$ that is total time independent.

4. Experimental Analysis

We construct experiments to show the efficiency of the proposed SPBMiner method is better than PSEMiner and ListMiner. All these methods were implemented in C++, and all the experiments were carried out on 3.3 GHz Intel Core i5 with 4GB memory.

4.1. Datasets

One real dynamic social network dataset and two artificial datasets are used to evaluate our SPBMiner algorithm. The Enron e-mail corpus is a publicly accessible record of e-mails commutation among employees of the Enron Corporation [1] and the artificial datasets are generated by random function. The table 1 shows the dataset set and parameter that has been used for our experiment analysis.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Timestep</th>
<th>Vertexes</th>
<th>Edges</th>
<th>Avg. Edge Density</th>
<th>$P_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>2588</td>
<td>82614</td>
<td>330452</td>
<td>0.0015</td>
<td>40</td>
</tr>
<tr>
<td>Artificial D1</td>
<td>2000</td>
<td>30</td>
<td>50</td>
<td>0.3</td>
<td>40</td>
</tr>
<tr>
<td>Artificial D2</td>
<td>2000</td>
<td>50</td>
<td>300</td>
<td>0.05</td>
<td>40</td>
</tr>
</tbody>
</table>

4.2. Time and Space Comparison

The figure 1(a) shows that SPBMiner is faster than two existing works ListMiner and PSEMiner in all the experiments except Enron dataset. The variation of network density is the main reason that explained below. In this high-density context, PSEMiner is much slower than ListMiner and ListMiner is slower than SPBMiner. The execution times comparison among three methods for artificial data D1 shows that proposed SPBMiner is around three times faster than ListMiner and around four times faster than PSEMiner and data D2 shows SPBMiner is faster than ListMiner as well as PSEMiner. Although, the theoretical time bound complexity of SPBMiner is better than these two methods, Enron dataset shows PSEMiner is faster than ListMiner and SPBMiner because of sparse dataset. The analyses of the memory requirement of three methods are presented in this section. Figure 1(b) shows the results for the comparison of the memory usage of these algorithms with $σ = 3$. The SPBMiner uses
less memory than ListMiner and PSEMiner in artificial data 1 and 2. In the dense networks SPBMiner calculate period entity very fast where PSEMiner traverse all previous graphs and ListMiner also create a large number of list node. The others dataset SPBMiner is higher than ListMiner and PSEMiner because of data sparsity.

![Graph](image)

(a) Time comparison
(b) Memory comparison

Figure 1. Time and memory comparison among SPBMiner, ListMiner and PSEMiner

5. Conclusion

The performances and the behaviors of SPBMiner were compared using real-world and artificial datasets. The experiments have been shown the performances of the algorithms depend on networks density. In the high-density networks, SPBMiner is faster than ListMiner and PSEMiner. However, periodic pattern exist a large number of redundant periodic information. To remove that redundant information parsimonious periodic pattern mining is our future works.

Acknowledgements

This work was supported a grant from the NIPA (National IT Industry Promotion Agency) in 2013 (Global IT Talents Program) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012047478).

7. References