Project / Thesis for the Degree of M.Sc. in CSE (Professional)  
Project/ Thesis Title  
Author Name  
Student ID: 20111...  
Department of Computer Science and Engineering  
Jagannath University  
Dhaka - 1100, Bangladesh  
December, 2016

Project / Thesis for the Degree of M.Sc. in CSE (Professional)  
Project Thesis Title  
Author Name  
Student ID: 20111...  
Department of Computer Science and Engineering  
Jagannath University  
Dhaka - 1100, Bangladesh

December, 2016

Project / Thesis Title  
by  
Author Name1  
Student ID:  
Supervised by  
Supervisor Name  
Submitted to the Department of Computer Science and  
Engineering of Jagannth University in partial fulfillment of the  
requirements for the degree of M.Sc. in CSE (Professional).  
Project / Thesis Evaluation Committee:  
Teacher Name 1 ..................................  
Teacher Name 2 .................................  
Teacher Name 3 .................................. nothing

Project / Thesis Approval  
Project/ Thesis Title  
Student’s Name:  
Student’s ID :  
&  
We the undersigned, recommend that the project / thesis completed by the  
student listed above, in partial fulfillment of M.Sc. in CSE (Professional) degree  
requirements, be accepted by the Department of Computer Science and  
Engineering, Jagannath University for deposit  
Supervisor Approval\*  
.............................  
Name of Supervisor  
Designation of Supervisor  
Departmental Approval  
Dr. Md. Abu Layek Professor Dr. Uzzal Kumar Acharjee  
Director, M.Sc. in CSE (Professional) Chairman  
Department of CSE Department of CSE  
Jagannath University  
Dhaka - 1100, Bangladesh

Project / Thesis for the B.Sc (Honors) in CSE  
Dedicated to my parents, Mr. A  
And  
Mrs. B

Abstract  
gsjhdgsjdgwdfjegu  
Key words: Dynamic Networks, Periodic Behaviors, Supergraph,  
i

Acknowledgment  
In this very special moment, first and foremost I would like to express my heartiest  
gratitude to the almighty God for allowing me to accomplish this MS study suc-  
cessfully. I am really thankful for the enormous blessings that the Almighty has  
bestowed upon me not only during my study period but also throughout my life.  
In achieving the gigantic goal, I have gone through the interactions with and  
help from other people, and would like to extend my deepest appreciation to those  
who have contributed to this dissertation itself in an essential way.  
I would like to express my heartfelt thanks to all of you for being with me with  
immense support and care and to make this work success.  
Author Name  
October, 2020  
ii

Table of Contents  
Abstract i  
Acknowledgment ii  
Chapter 1 Introduction 1  
1.1 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1  
1.2 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4  
1.3 Problem Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4  
1.4 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6  
1.5 Organization of the Thesis . . . . . . . . . . . . . . . . . . . . . . . . 6  
Chapter 2 Related Works 8  
2.1 Related Works on Unstructured Databases. . . . . . . . . . . . . . . 8  
2.2 Related Works on Structured Databases . . . . . . . . . . . . . . . . 9  
2.2.1 PSEMiner . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11  
2.2.1.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . 11  
2.2.1.2 Complexity Analysis of Periodic Subgraph Mining . 16  
2.2.1.3 Basic Description of PSEMiner Method . . . . . . . 19  
2.2.1.4 Extension to the PSEMiner Technique . . . . . . . . 22  
2.2.1.5 Space and time complexity . . . . . . . . . . . . . . 25  
2.2.2 ListMiner . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25  
2.2.2.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . 26  
2.2.2.2 Data Structures . . . . . . . . . . . . . . . . . . . . 27  
iii

Table of Contents iv  
2.2.2.3 Parameter . . . . . . . . . . . . . . . . . . . . . . . 29  
2.2.2.4 Description of the ListMiner Technique . . . . . . . 29  
2.2.2.5 Time and space complexity . . . . . . . . . . . . . . 31  
2.3 Limitation of Existing Works . . . . . . . . . . . . . . . . . . . . . . 32  
Chapter 3 Supergraph Based Periodic Behaviors Mining (SPB-  
Miner) 33  
3.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34  
3.2 Supergraph based Behaviors Mining . . . . . . . . . . . . . . . . . . 38  
3.2.1 Parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40  
3.2.2 Data Structure . . . . . . . . . . . . . . . . . . . . . . . . . . 40  
3.2.2.1 Descriptor . . . . . . . . . . . . . . . . . . . . . . . 40  
3.2.2.2 Time Set . . . . . . . . . . . . . . . . . . . . . . . . 40  
3.2.2.3 Periodic subgraph hash table . . . . . . . . . . . . . 41  
3.3 Description of the algorithm . . . . . . . . . . . . . . . . . . . . . . . 41  
3.3.1 Update Common Entities in Supergraph . . . . . . . . . . . . 42  
3.3.1.1 Timeset Update . . . . . . . . . . . . . . . . . . . . 42  
3.3.1.2 Descriptors Creation . . . . . . . . . . . . . . . . . . 42  
3.3.1.3 Descriptors Update . . . . . . . . . . . . . . . . . . 43  
3.3.1.4 Descriptors Deletion . . . . . . . . . . . . . . . . . . 43  
3.3.1.5 Entities Deletions . . . . . . . . . . . . . . . . . . . 44  
3.3.2 SPBMiner algorithm . . . . . . . . . . . . . . . . . . . . . . . 46  
3.3.2.1 Entity Updates algorithm . . . . . . . . . . . . . . . 46  
3.3.2.2 Mining Periodic Behaviors . . . . . . . . . . . . . . 47  
3.3.3 Example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50  
3.3.4 Description of the implementation . . . . . . . . . . . . . . . 57  
3.4 Time and Space Complexity . . . . . . . . . . . . . . . . . . . . . . . 57  
3.4.1 Time Complexity . . . . . . . . . . . . . . . . . . . . . . . . . 57  
3.4.2 Space Complexity . . . . . . . . . . . . . . . . . . . . . . . . 58  
3.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

Table of Contents v  
Chapter 4 Experimental Evaluation 59  
4.1 Datasets Description . . . . . . . . . . . . . . . . . . . . . . . . . . . 59  
4.1.1 Real Datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . 59  
4.1.2 Artificial Data . . . . . . . . . . . . . . . . . . . . . . . . . . 60  
4.2 Experimental Time Analysis . . . . . . . . . . . . . . . . . . . . . . . 61  
4.3 Experimental Space Analysis . . . . . . . . . . . . . . . . . . . . . . 64  
4.4 Analysis of Dense Networks Effects . . . . . . . . . . . . . . . . . . . 65  
4.5 Analysis of Parsimonious Periodic Patterns . . . . . . . . . . . . . . 66  
4.5.1 Parsimonious Periodic Pattern Division and Support Values . 66  
4.5.2 Parsimonious Periodic Pattern Division and Period Values . . 67  
4.5.3 Knowledge Discovery . . . . . . . . . . . . . . . . . . . . . . . 68  
4.5.4 Scalability Analysis . . . . . . . . . . . . . . . . . . . . . . . 68  
4.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69  
Chapter 5 Conclusions and Future Work 70  
5.1 Summary of the Dissertation . . . . . . . . . . . . . . . . . . . . . . 70  
5.2 Future Research Directions . . . . . . . . . . . . . . . . . . . . . . . 71  
Bibliography 72  
Appendix A List of Publications 77

List of Figures  
1.1 Dynamic graph structure . . . . . . . . . . . . . . . . . . . . . . . . 5  
2.1 Graph . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
2.2 The unique set representation between graphs that demonstrates the  
computation of the MCS of two graphs [1]. . . . . . . . . . . . . . . 13  
2.3 An example of a dynamic graph structure with 5 consecutive timesteps 13  
2.4 An example of a dynamic network [1]. . . . . . . . . . . . . . . . . . 22  
2.5 The pattern tree at each timestep for the dynamic network shown in  
figure 2.4, considering only edges for brevity [1]. . . . . . . . . . . . . 23  
3.1 Compact Supergraph. . . . . . . . . . . . . . . . . . . . . . . . . . . 35  
3.2 Graphrepresentationwithuniquevertexlabelandcorrespondingcon-  
nected vertex label . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36  
3.3 Maximum common subgraph between G and G . . . . . . . . . . . 36  
1 2  
3.4 SPBMiner Architecture . . . . . . . . . . . . . . . . . . . . . . . . . 39  
3.5 Dynamic Networks . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51  
4.1 Execution times comparison on real datasets. . . . . . . . . . . . . . 63  
4.2 Execution times comparison on artificial datasets.. . . . . . . . . . . 64  
4.3 Memory usage on real datasets. . . . . . . . . . . . . . . . . . . . . . 65  
4.4 Memory usage on real datasets. . . . . . . . . . . . . . . . . . . . . . 66  
4.5 Parsimonious periodic patterns vs minimum support . . . . . . . . . 67  
4.6 Parsimonious periodic patterns vs period value . . . . . . . . . . . . 68  
vi

List of Figures vii  
4.7 Finding inherent patterns from facebook and Enron dataset . . . . . 69  
4.8 Scalability test for experiment 1.4 dataset . . . . . . . . . . . . . . . 69

List of Tables  
3.1 Closed and Parsimonious Periodic subgraphs characteristics . . . . . 50  
3.2 Supergraph representation at timestep 1 . . . . . . . . . . . . . . . . 52  
3.3 Supergraph representation at timestep 2 . . . . . . . . . . . . . . . . 52  
3.4 Supergraph representation at timestep 3 . . . . . . . . . . . . . . . . 53  
3.5 Supergraph representation at timestep 4 . . . . . . . . . . . . . . . . 54  
3.6 Supergraph representation at timestep 5 . . . . . . . . . . . . . . . . 55  
3.7 Periodic Patterns . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55  
3.8 Pruning non closed and non parsimonious periodic patterns . . . . . 56  
3.9 Parsimonious Periodic Patterns . . . . . . . . . . . . . . . . . . . . 56  
4.1 Parameters of various datasets . . . . . . . . . . . . . . . . . . . . . 62  
viii

List of Algorithms  
1 SPBMiner ({BG ,BG ,...BG },σ) . . . . . . . . . . . . . . . . . . . 45  
1 2 T  
2 Update Entity(BSG ,t,σ) . . . . . . . . . . . . . . . . . . . . . . . . 48  
E  
3 Flashed(D,E,σ) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49  
4 Parsimonious Periodic Pattern (Periodic Patterns) . . . . . . . . . . 51  
ix

Chapter 1  
Introduction  
This chapter presents an overview of the periodic behavior mining in social net-  
working research including the goal, and potential applications in this area. Then,  
different existing methods are discussed briefly to figure out the advantages as well  
as disadvantages of each particular technique. Based on the discussion, motivations  
behind the proposed method are explained clearly in this thesis. The organization  
of this thesis is presented at the end of this chapter.  
1.1 Overview  
The goal of periodic behavior miming is to identify regularly occurring behavior in  
dynamic networks. In general, a dynamic network is a powerful model for repre-  
senting time varying systems where interactions among entities change from time to  
time. Dynamic network analysis has gained the attention of the computer science  
community because of its different applications such as human societies and behav-  
ioranalysis, wildanimalcommunities’behaviormatching, sensornetworksactivities  
prediction and the study of uses of mobile cell.  
Dynamic network dataset size increases very rapidly. Not all the information in  
networks conveys interesting meaning. The definition of interesting depends on the  
context. Data mining is the studies of extracting interesting information from data.  
The items those are frequent in database must hold some valuable meaning such  
as market basket analysis [2], which indicates items purchase behavior of consumer.  
Structured data mining has attracted researchers attention in the last few years.  
Graphs are probably one of the best representations of structured data and relevant  
1

1.1 OVERVIEW 2  
research field is extremely vast. Here an overview of the most interesting problems  
and their solution techniques are discussed briefly.  
In the middle of 1990s, the first studies on graph mining methods [3,4] proposed  
discovering graph representation technique for structures data. In [5], an induc-  
tive logic programming based technique has been proposed that extracts frequent  
subgraph from graph data. An efficient method for frequent subgraph mining is  
proposed by Nijssen and Kok in [6].  
Since the 2000s, the data mining research community showed great effort in pe-  
riodic pattern mining. Their research interest is distributed in several domains [7–9]  
like as transactional datasets, daily traffic patterns, stock data, meteorological data  
and web logs data. Periodic patterns in graph database is also incredibly interesting  
and information has been shown in [1,10] for human periodic behavior analysis in  
social network and celebrity selection. Therefore, periodic patterns mining is one of  
the important tasks in data mining.  
Ozden et al. [11] introduced discovering periodic pattern association rule that  
show regular cyclic pattern over time, while Bettini et al. [12] presented a tech-  
nique to find temporal patterns in time sequence. Partial periodic pattern mining  
is another very interesting research since Han et al. [13] proposed partial time se-  
ries behavior mining methods. Yang et al. [14] proposed a 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. Huang and Chang proposed another asynchronous method in [15]. In this  
their method, minimum number of repetitions required for valid segment and valid  
sequence.  
The introduction of a probabilistic model in periodic pattern mining is another  
topic of research. In this technique, the value of a pattern is continuous and mono-  
tonical that probability decreases with occurrence. In [16] Yang et al. introduced  
an efficient algorithm, InfoMiner, to mine surprising patterns and associated subse-  
quences based on the information gain.

1.1 OVERVIEW 3  
Graph mining is incredibly interesting research in current world. Dynamic net-  
works represent a sequence of graphs over time. Changing of graphs from time to  
time mention entities behaviors change in dynamic networks. Graph vertexes repre-  
sent the members of population and interaction between two members at particular  
time is represented by establishing an edge between two vertexes.  
The population in dynamic networks can be diverse in nature: humans [17,18],  
animals[19],networkedcomputers[20]. Socialnetwork[6]analysisisthebest-known  
example in dynamic networks analysis. Among the analysis of dynamic networks,  
findingperiodicpatternismostinterestingandconveysverymeaningfulinformation  
yet often-infrequent pattern.  
In this perspective, Lahiri and Berger-Wolf in [1,10] introduced a new mining  
problem to find periodic patterns in dynamic social networks. Periodic patterns  
occur regularly in networks that change over time. These patterns do not exist in all  
timeintervals, that’swhytheymineallperiodicsubgraphsthatoccurinaminimum  
number of times. They deal with the concept of closed subgraph mining that has  
beenwidelyusedinfrequentpatternmining[21]. Occam’sRazorprincipleisfollowed  
for parsimony closed pattern mining. PSEMiner algorithm is proposed for finding  
periodic subgraph in dynamic networks which is polynomial unlike many related  
subgraphanditemsetminingproblem. Here,theycreatepatterntreethatmaintains  
all parsimonious subgraphs shown up to timestep t, and it tracks subgraphs that is  
periodic or might be periodic at some point in future. In timestep t, the graph G  
t  
is read, and the pattern tree is traversed and common subgraph between G and  
t  
tree node is updated with the new information including modifying, adding and  
deleting tree node. Each node indicates unique subgraph. This process creates a lot  
of unessential tree node which is time consuming.  
Apostolico et al. proposed the speedup method ListMiner for periodic pattern  
subgraph mining in [22]. Proposed method identified patterns based on projected  
timesteps graph that solve unused tree node problem. Exact number of tree nodes  
is created that must be essential for pattern mining. This method maintains a list

1.3 MOTIVATION 4  
of structure. At timestep t, the graph G is read, and the projection π list is  
t p,m  
updated adding new list nodes, where p is period and m = t mod p. This approach  
is much faster than previous approach because it creates less number of list nodes.  
However, thenumberoflistnodesisnotasmallnumberandmanynodesstoresome  
common information redundantly that is memory consuming. The number of list  
depends on maximum period P because the process mine all periodic patterns  
max  
up to P and the number of list nodes in the list depends on entities set and total  
max  
timesteps that is also a time-consuming process.  
1.2 Motivation  
From the above analysis, it can be seen that periodic pattern mining in social net-  
works plays a significant role. Although, PSEMiner and ListMiner mine periodic  
patterns correctly, these process store every timesteps graph < G ,G ,...,G >  
1 2 t  
in memory and finding periodic pattern from G it computes common subgraph  
t  
patterns among < G ,G ,...,G > those patterns mining is time consuming. Dy-  
1 2 t−1  
namic networks is a continuous process and it varies time to time. If some graph G  
t  
occurs one time and will never happen in future, existing processes maintain that  
graph also. Therefore, our motivation of this work is to propose a method for peri-  
odicbehaviorsminingthatovercomesthelimitationsofexistingworks. Myproposed  
technique helps to mine periodic behaviors that need storing of dynamic networks  
entities (vertexes and edges) only one time and need common entities computation  
once for graph G . This process is time and memory efficient.  
t  
1.3 Problem Statement  
Dynamic network is a model of structure that changes over time. This structure is  
generally modeledas real-world phenomena where aset of uniquelyidentifiable enti-  
ties are interacting with each other over time. In global organization the interaction  
between entities can be a complex network system over time [23]. In this thesis we

1.4 PROBLEM STATEMENT 5  
deal with the detection of a specific types of interaction behavior, called periodically  
reoccurring behavior in networks that has been changed change dynamically. Our  
main goal is to mine parsimonious periodic behavior even if it persists only for a  
short period and infrequent. More clearly it can say, given a dynamic network (DG)  
and a minimum support threshold (σ ≥ 2), the periodic subgraph mining problem  
is to mine all parsimonious periodic subgraph embeddings in DG that satisfy the  
minimum support. The periodic subgraph embedding (PSE) for a subgraph F is  
S(t,p,s) that means subgraph F occurs at first t times, continues until s times  
where occurs interval is p. We define S(t,p,s) = {t,t+p,...,t+p(s−1)}, where  
t ≥ 0 and p,s ≥ 1. A subgraph F may have several periodic supports set. Not all  
of those periodic supports are parsimonious. Let subgraph F occurred two periodic  
set P = S(t ,p ,s ) and P = S(t ,p ,s ). We say that P subsumes/contains P  
1 1 1 1 2 2 2 2 1 2  
if and only if S(t ,p ,s ) ̸⊆ S(t ,p ,s ). That result P is parsimonious periodic  
2 2 2 1 1 1 1  
pattern for subgraph F but P is not parsimonious.  
2  
A B A B A B A B A B  
D C D C D C D C  
G G G G G  
1 2 3 4 5  
Figure 1.1: Dynamic graph structure  
For example, iteration between A and B occur < 1,2,4,5 > timesteps graph.  
Supposeminimumσ ≥ 2thenwefindperiodicpatternsP = {1,1,2},P = {1,3,2},  
1 2  
P = {1,4,2}, P = {2,2,2}, P = {2,3,4} and P = {4,1,2}. On the other hand,  
3 4 5 5  
interaction between C and D occur < 1,2,3 > timesteps. We find periodic patterns  
P = {1,1,2}, P = {1,2,2}, P = {2,1,2} and P = {1,1,3}. Periodic patterns P ,  
6 7 8 9 6  
P , P subsumed in P . This purpose C and D interaction contains only P periodic  
7 8 9 9  
patters that is parsimonious periodic pattern. Our goal is finding all parsimonious  
periodic patterns in dynamic networks.

1.5 CONTRIBUTIONS 6  
1.4 Contributions  
The main contributions of this dissertation are described as follows: We introduce  
the problem of finding regularly occurred periodic patterns in dynamic networks.  
The design and development of SPBMiner, an online algorithm that improves the  
worst case time performance of the algorithms [1,10] by at least σ factor propor-  
tional to the total number of timestamps in dynamic networks. It shows σ times  
better performance than algorithm [22]. Proposed miner method space complexity  
is independent on dynamic networks timestamps. It’s space complexity is (σ2ln(T))  
times less than [1,10] and σ times less than [22].  
We propose a supergraph based technique that stores all dynamic networks in-  
teraction entities (vertexes and edges) only one times. Each entity maintains a data  
structure, which stores an occurred time set and a list of periodic descriptor sets.  
When one entity holds interactions in networks, its descriptor and timestep update  
are performed with modifying, deleting and adding descriptors. Each timestep su-  
pergraph conveys individual periodic entities. We combine those periodic entities  
based on starting position, period, and support and find periodic patterns. Then we  
mineparsimoniousperiodicpatterns. Extensiveperformanceanalysesshowthatour  
proposed method is significantly efficient and effective for periodic behaviors mining  
in dynamic networks when networks density is medium or high.  
1.5 Organization of the Thesis  
The dissertation is organized as follows:  
• • Chapter 1 Introduction. In this chapter an introduction to the periodic  
patterns mining researches is presented. The definition, importance and exist-  
ing approaches are clearly introduced. After that, the dissertation focuses the  
contribution.  
• • Chapter 2 Related Work. This chapter first shows the state of the  
art methods of the periodic patterns mining research. Then describe two

1.5 ORGANIZATION OF THE THESIS 7  
existingperiodicpatternminingworksPSEMinerandListMinerindynamic  
networks. The limitations of these methods are clearly addressed, as these are  
the focuses of this dissertation.  
• • Chapter 3 SPBMiner. We present our proposed technique for mining  
periodic behaviors in dynamic networks.  
• • Chapter 4 Experiments Analysis. In this chapter, it has been shown  
the effectiveness and efficiency of our proposed method.  
• •Chapter5ConclusionandFutureWork. Finally,thischapterconcludes  
the dissertation indicating the limitations and future works.

Chapter 2  
Related Works  
In data mining research, Periodic patterns mining appears in different perspective.  
In this chapter, periodic patterns mining relevant literatures in unstructured and  
structured databases have been reviewed. The main focus of this thesis is mining  
periodic patterns in dynamic networks these are the structured data representation  
model. At first, some periodic patterns mining research in unstructured database  
are described briefly. Then we discuss structured periodic patterns mining prob-  
lem related works those are most relevant our work. The two proposed algorithms  
PSEMiner andListMiner areexplainedindetails. Finally, weconcludethischap-  
ter mentioning some limitations of existing works and make clear our motivation.  
2.1 Related Works on Unstructured Databases  
Most of the proposed periodic patterns mining techniques deal with unstructured  
datasuchsequentialandtransactionaldatabases. Thisresearchareaisnotourmain  
goal; asaresult, weexplainsomeproposedtechniquesinbriefly. Inthemostgeneral  
model, given a sequence of symbol set S = {x ,x ,...,x }, where each symbol x  
1 2 T i  
represents a universal set L. A pattern sequence P = {y ,y ,...,y }, which period  
1 2 p  
is p and each y ⊆ L (cid:83) {∗}. The ′∗′ character indicates matching any symbol. This  
i  
pattern-mining problem mine all such patterns from the input sequence data those  
satisfy minimum support.  
Han et al. first introduced partial periodic pattern mining algorithm in time-  
series databases [13]. Mining interesting properties of partial patterns such as Apri-  
ori property [24] and the max-subpattern hit set property have been maintained in  
8

2.2 RELATED WORKS ON STRUCTURED DATABASES 9  
this algorithm. Ma and Hellerstein [25] proposed a similar, Apriori inspired method  
containing two level wise algorithms in unknown periods value. They also proposed  
novel approach based on chi-square test for defining periodicity. Yang et al. [14] in-  
troduced a new asynchronous type of periodic pattern mining algorithm that mine  
all patterns within coverage rage of data sequence and maximum number of dis-  
ruption allowed. Huang et al. [15] proposed another asynchronous method, which  
valid segment and valid sequence is measured by minimum number of repetitions of  
patterns.  
In periodic pattern mining problem, the introduction of a probabilistic model is  
anotherinterestingtopicofresearch. Inthistechnique, apatternvalueiscontinuous  
and monotonically, which probability decreases with number of occurrences. In [16]  
Yang et al. established an efficient algorithm named InfoMiner that mine surprising  
patterns and associated subsequences based on the information gain. Yin et al. [26]  
proposedprobabilitybasedthelatentperiodictopicanalysisintextdatabase. Latent  
periodictopicanalysisfindalatenttopicspacetofitthedatacorpusaswellasdetect  
whether a topic in periodic or not.  
Mining frequent patterns [24,27–29] from transactional database has been ac-  
tively and widely studied in data mining. Tanbeer et al. [30] introduced a novel  
concept of mining periodic frequent patterns in transactional database. Those pat-  
terns are frequent and appear at a regular interval by given user in the database  
is periodic frequent pattern. Periodic pattern mining is interesting in others data  
mining research area. Sumithi and Sathiyabama [31] proposed an efficient method  
that discovers the hidden periodic patterns from a spatio-temporal database. They  
show that if there are any periodic pattern could unveil important information to  
data analyst as well as facilitate data management substantially.  
2.2 Related Works on Structured Databases  
Periodic structured data mining is especially interesting research in current world.  
A dynamic network is an extraordinarily powerful mathematical representation for

2.2 RELATED WORKS ON STRUCTURED DATABASES 10  
time-varyingsystemsthoseconsistofmanyinteractionsamongentitiesinstructured  
datamodel. Itismostusefulwhentheenvironmentextremelycomplexandbehavior  
is continuously changing over time. It is supplementary influential representation  
than canonical social network that allows one to map explicitly system structure  
changes [23].  
Dynamic networks represent a sequence of graphs over time. Graph vertexes set  
characterize the members of population and interactions among members at some  
particular time are represented by establishing edges among vertexes. Dynamic net-  
works population can be diverse nature: humans [17,18], animals [19], networked  
computers [20]. Social network [6] analysis is the best-known example in dynamic  
networks analysis. Monthly e-mail, yearly family reunions, monthly banking in-  
formation reports and familiar face of stranger at the morning coffee shop are all  
periodic a significant that are easily missing to the study in the collection of public  
interactions data [10]. Among the analyses of dynamic networks, finding periodic  
patterns is most interesting and conveys very meaningful information yet often-  
infrequent pattern. There are two potential applications. First one, these periodic  
patterns represent stable interaction patterns that can be of qualitative interest in  
and of themselves. For example, ecologists study animals movements and social  
patterns are found by tracking devices [32]. Periodic subgraphs are correspond-  
ing seasonal association or mating patterns that are hidden in mass quantities of  
arbitrary animals movements [19]. The second important application is periodic be-  
haviorcanbepredictbehaviorbyvirtueofrepeatingregularly. Forexample, mining  
predictable interactions from sensors logs can be used in different types of mobile  
and ubiquitous applications [33]. Yan et al. [34] proposed regular behavior miner  
algorithm that mine maximal frequent subgraph in dynamic networks.  
In this perspective mining periodic patterns in dynamic networks, Lahiri and  
Berger-Wolf in [1,10] proposed PSEMiner algorithm. This algorithm mine peri-  
odic patterns in networks those change over time and occur in a minimum number  
of times. They deal with the concept of closed subgraph mining that has been

2.2 RELATED WORKS ON STRUCTURED DATABASES 11  
widely used in frequent pattern mining [21]. Occams Razor principle is followed for  
parsimony closed pattern mining. Finding periodic subgraph in dynamic networks  
is polynomial unlike many related subgraph and itemset mining problem. In this  
process, pattern tree structure is created that maintains all periodic subgraphs seen  
up to timestep t, and it tracks subgraphs that is periodic or might be periodic at  
some point in future. At timestep t, the graph G is read, the entire pattern tree is  
t  
traversed and common subgraph between G and tree node is updated with the new  
t  
information including modifying, adding and deleting tree nodes. In this process,  
a number of common subgraphs are created that is useless. Each node indicates  
unique subgraph. There is another method ListMiner [22] that speedup periodic  
patterns mining. This method finds patterns based on projected timesteps graph  
that solve unused tree node problem. Exact number of tree nodes are created those  
must be essential for pattern mining. This method maintains a list structure. At  
timestep t, the graph G is read, and the projection π list is updated adding  
t p,m  
new list nodes, where p is period and m = t mod p. This approach is T faster than  
previousapproachbecauseitcreateslessnumberoflistnodes. Ourmainmotivation  
is above two methods PSEMiner and ListMiner. The following subsections we  
describe both techniques in detail.  
2.2.1 PSEMiner  
2.2.1.1 Preliminaries  
The concept of periodic subgraph mining in dynamic networks has been firstly pro-  
posed by this algorithm in [1,10]. The algorithm makes a single pass over the data  
and capable of accommodating perfect closed subgraph patterns periodicity. Dy-  
namic networks are a representation of interaction between a set of unique entities,  
which change time to time. Let V ∈ N represent the set of entities. Interactions be-  
tweenentitiesmaybedirectedorundirectedandaresupposedtohavebeenrecorded  
over a period of T isolated timesteps. We use natural quantizations specific to each  
of our dataset such as one day per timesteps. The only essential is that a timestep

2.2 RELATED WORKS ON STRUCTURED DATABASES 12  
representameaningfulamountofrealtimewheretheperiodicityofminedsubgraphs  
will be set of chosen timesteps. The purpose of this study is known the periodic  
interactions between elements, whose belong to dynamic network.  
1. Definition. Graph: A graph G = (V,E) is a simple interactions set E among  
entities set V. The interactions E and entities V in G represent with unique label  
set R ∈ N.  
A  
B C  
Figure 2.1: Graph  
Figure 2.1 shows the graph that can be represent by integer set R =  
{1,2,3,4,5,6} where vertex set {A,B,C} labeled by {1,2,3} and edge set  
{A−B,A−C,B−C} labeled by {4,5,6} respectively. These unique labeled rep-  
resentations show that two graphs similarity measurement is very easy. Suppose  
given two graph G and G with unique vertexes and edges labels, testing whether  
1 2  
G is subgraph of G or vice versa is defined by the corresponding R and R rep-  
1 2 1 2  
resentation are subset of each other.  
The maximum common subgraph (MCS) between two graphs find by unique  
label vertex and edge set intersection between two corresponding representation  
set. Figure ?? shows the maximum common subgraph (MCS) calculation process  
between two graphs using unique labeled vertex and edge set.  
2. Definition. Dynamic Networks: A dynamic network DN =< G ,G ,...,G >  
1 2 T  
is a time series graphs, where graph G = (V ,E ) represent interactions set E  
t t t t  
among unique vertex set V ⊆ V at timestep t.  
t

2.2 RELATED WORKS ON STRUCTURED DATABASES 13  
Figure 2.2: The unique set representation between graphs that demonstrates the  
computation of the MCS of two graphs [1].  
Figure2.3showstheexampleofadynamicnetworkwithfivetimestep. Definition  
?? reduces the high computational complexity of many algorithmic tasks on graphs  
database. Sincevertexesandedgesarerepresentedbyuniqueentity,eachtimestept,  
vertexsetV ingraphG areuniquethatreducecomputationalcomplexityforcertain  
t t  
hard graph mining problems, such as maximum common subgraph and subgraph  
isomorphism testing [35,36].  
A B A B A B A B A B  
D C D C D C D C  
G G G G G  
1 2 3 4 5  
Figure 2.3: An example of a dynamic graph structure with 5 consecutive timesteps  
3. Definition. Periodic Graph: A graph G = (V ⊆ V ,E ⊆ V ×V) in dynamic  
g  
networkDN =< G ,G ,...,G >isaperiodicgraphwithperiodp, ifGisasubgraph  
1 2 T  
of < G ,G ,...,G >, where 0 ≤ x ≤ p and n ≥ σ (min support).  
x x+p x+np  
In figure 2.3 the graph with vertex {A,B,D} and their connected edge  
{A−B,A−D,B−D} is periodic with period 2 because it occurs at timestep 2  
and 4. Vertex set {C,D} and their connected edge is periodic with period 1, it

2.2 RELATED WORKS ON STRUCTURED DATABASES 14  
appears in timestep 1, 2 and 3. If minimum support is 2 then it is period also at  
time {1,2} and {2,3}. In large dynamic network, the number of periodic patterns  
could be very large. Reducing these kinds of redundant, closed periodic subgraphs  
are mined from networks. The periodic graph does not appear each timestep, if it  
appears minimum times σ in dynamic network with period p it is periodic. The  
number of times graph occurs periodically in networks called periodic support set.  
4. Definition. PeriodicSupportSet: GivenadynamicnetworksDNofTtimesteps  
and any subgraph F = (V, E). The periodic support set S (F) of F in DN is the set  
p  
of all timesteps t, which start at t timestep and repeating every p period interval  
i  
where F is a subgraph of DN, which denote F subgraph of G . The representation  
t  
of support set S (F) = (t ,p,s) = {t ,t +p,...,t−i+p(s−1)}, such that ∀t (  
P i i i i  
t ∈ S (F) ↔ F ⊆ G ) and neither G nor G contains F as subgraph.  
i p ti ti−p ti+ps  
F is frequent periodic pattern if its support exceeds a user defined minimum  
support threshold value σ ≤ T.  
Definition4isthebasicformulationofwellknownfrequentpatternminingprob-  
lemthatsatisfythedownwardclosureproperty. Indownwardclosureproperty,every  
sub pattern of a frequent periodic pattern F is also frequent.  
5. Definition. Closed Subgraph : For any subgraph F = (V, E) in a dynamic  
networks DN of T timesteps is closed if it is maximal for its support set.  
There are a difference between frequent closed subgraph support set and peri-  
odic closed subgraph support set. A single subgraph F can have multiple periodic  
pattern support set to allow dis-join and overlapping periodic behavior. Thus, we  
requireextractionofallperiodicsubgraphembeddings, ratherthanjusttheperiodic  
subgraphs. According [1], the definition of periodic subgraph embedding is follows.  
6. Definition. Periodic Subgraph Embedding (PSE): Given a dynamic network DN  
and a arbitrary subgraph F. The periodic subgraph embedding (PSE) is a pair of  
< F,S (F) >, where F is closed subgraph over a periodic support set S (F) with  
p p  
|S (F)| > σ and S (F) is temporally maximum for F.  
p p

2.2 RELATED WORKS ON STRUCTURED DATABASES 15  
Mining frequent periodic closed behavior patterns is a well-designed solution to  
the redundancy of general frequent pattern mining problem, since it captures all the  
information of the more general formulation produce small number of output results  
except any loss of information. However, there are periodic closed patterns that  
periodicity contains by other periodicity, This kinds of patterns also redundant. To  
avoid this kinds of redundancy parsimonious pattern is proposed.  
7. Definition. Parsimonious PSE: A PSE that is not subsumed by any other PSE  
is Parsimonious PSE.  
1. Property. Subsumption : Two periodic subgraphs F and F , their support set  
1 2  
S (F ) = (t ,p ,s )andS (F ) = (t ,p .s ). F supportset< F ,S (F ) >contains  
p 1 1 1 1 p 2 2 2 2 1 1 p 1  
orsubsumedF supportset< F ,S (F ) >ifandonlyifthefollowingconditionhold.  
2 2 p 2  
i. F ⊆ F  
2 1  
ii. t ≥ t  
2 1  
iii. p mod p = 0 and p < p  
2 1 1 2  
iv. t +p (s −1) ≤ t +p (s −1)  
2 2 2 1 1 1  
v. (t −t ) = p.k for some integer k > 0.  
2 1  
Forexample,asubgraphFofperiod1withadequatesupport10,thenitalsoperiodic  
at period 2 and 3 if minimum threshold is 3. In this case, PSE that period 2 and 3  
are not Parsimonious PSE.  
8. Definition. Periodic Subgraph Mining Problem: Given a dynamic network DN  
and a minimum support threshold σ ≥ 2, the Periodic Subgraph Mining Problem  
mine all parsimonious periodic subgraph embeddings in DN those satisfy the mini-  
mum support.  
Since real world networks, there are some event that are not exactly periodic,  
present a definition of what constitutes near periodicity.  
9. Definition. Noisy Subgraph: A noisy subgraph exhibits jitter in its period if its  
period is near-constant rather than constant. If a jitter value of J ≥ 0, the extend  
support of subgraph F as follows: S(F) =< t : F ⊆ G > and ∀i : |t −t | ≤ p±J.  
t i+1 i

2.2 RELATED WORKS ON STRUCTURED DATABASES 16  
2.2.1.2 Complexity Analysis of Periodic Subgraph Mining  
This section discusses the proof (taken from [1,10]) time complexity of periodic  
subgraph mining is polynomial unlike many related subgraph mining problems.  
To this purpose method, the unique labeling of vertexes and edges plays significant  
role. TheproofshowsthePSEminingproblemissolvableinpolynomialtimethatis  
contrasttothemoregeneralfrequentsubgraph-miningproblem,whichisNP−hard  
for enumeration and #P −complete for counting periodic subgraph, even though  
unique vertex labels are considered [37,38]. The concept of projection of a discrete  
time series data in [39] is used to find the maximum number of PSEs in dynamic  
networks.  
10. Definition. Projection: Given a dynamic network DN, a projection π =<  
p,m  
G ,G ,G ,...,G > is a subsequence of graphs among DN, where p is  
m m+p m+2p m+sp  
the period of projection and 0 ≤ m < p is the phase offset.  
Itshouldbeclearfromthedefinitionofperiodicityandprojectionthatthesubgraph  
of every graph in the projection is a periodic subgraph if s is greater than or equal  
minimum support σ that means periodic support set is ≥ σ.  
1. Proposition. Let F be the maximal common subgraph (MCS) of any s ≥ σ  
consecutive positions of any projection π . If F ̸= ϕ, then it is a periodic subgraph  
p,m  
and the s consecutive timesteps form projection π are part of a PSE for F.  
p,m  
Proof: Suppose the MCS of F of any s ≥ σ consecutive positions is not empty,  
that implies F is maximal over a support set of at least σ periodic timesteps. Sub-  
graphF mightormightnotbetemporallymaximal. However,ontheothercase,the  
s timesteps are part of some valid periodic support set of size at least σ. According  
to definition 6 it satisfy the condition of periodic subgraph and thus F is a periodic  
subgraph.  
1. Corollary. In the worst case, computational complexity of mining periodic sub-  
graph embeddings in a dynamic network, the MCS of every s ≥ σ consecutive posi-

2.2 RELATED WORKS ON STRUCTURED DATABASES 17  
tions is not empty then contains a unique PSE.  
Proof: If every periodic subgraph subset is s ≥ σ timesteps in the dynamic network  
contains a unique maximal common subgraph, then they all need to be enumerated  
by any mining algorithm. It has been showen that it is possible using an explicit  
edifice. At different edge in each s ≥ σ consecutive positions of every projection to  
ensure each edge is part of a unique periodic subgraph embedding. Let edge e be  
created in this way with support set (SP) in some π . Considering only SP , we  
p,m  
know that it is temporally maximal for the edge e because e does not exist in any  
othertimesteps. Moreover, theMCSofSP isnon-emptybecauseitcontainsatleast  
the edge e. Thus, each edge is part of a unique PSE whose support set is SP . Since  
a different edge was placed in every s ≥ σ consecutive positions of every projection,  
the number of PSEs is equal to the number of edges created. No additional PSEs  
can be created since every permissible support set, i.e. with support greater than σ  
is already part of a unique PSE. Therefore, the described structure is a worst case  
instance for its size.  
The next step unambiguously computes the upper bound on the total number of  
PSEs in the worst-case network instances. According to corollary 1, we only need  
to count the number s ≥ σ consecutive positions of every projection to derive this  
bound. In order to do this, we first state the bounds on several other parameters.  
2. Proposition. In a dynamic network with T timesteps, the maximum period of  
any periodic subgraph with support at least is P = ⌊(T −1)/(σ−1)⌋.  
Proof: For a given period p, the subgraph is F ⊆ G . In the other T −1 timesteps,  
1  
for every periodic embedding subgraph F ⊆ G in σ − 1 consecutive timesteps  
j  
< T ,T ,...,T >. Thelastindex1+p(σ−1) ≤ T,becauseT istheindex  
1+p 1+2p 1+p(σ−1)  
ofthelasttimestep. Fromthisinequalityithasbeenderivedthatp ≤ (T−1)/(σ−1).  
3. Proposition. In a dynamic network with T timesteps, the length of any projec-  
tion is |Π | = ⌈(T −m)/p⌉.  
p,m

2.2 RELATED WORKS ON STRUCTURED DATABASES 18  
Proof: Since π =< G ,G ,G ,... >, the projection starts after m  
p,m m m+p m+2p  
timesteps,andsothereareT−mtimestepsremaining. Sinceindexesoftwofollowing  
timesteps differ by p positions that the length of profection π is ⌈(T −m)/p|.  
p,m  
Given the above expressions, the exact bound for mining number of closed PSE  
can be obtained by construction [1].  
1. Theorem. In a dynamic network with T timesteps, there are at most  
O(T2ln(T)) closed PSEs at minimum support σ.  
σ  
Proof: From Corollary 1, the maximum number of PSEs possible in a dynamic  
network at minimum support σ is equal to the number of s ≥ σ length windows over  
all possible projections of the network. For a given projection π and value of s,  
p,m  
it is clear that the number of length-s windows over the projection is |π |−s+1.  
p,m  
Thus, for a given value of s, the number of length-s windows over all projections  
can be obtained by substituting the expressions from propositions ?? and ??:  
⌊T−1⌋ (cid:16)(cid:108) (cid:109) (cid:17)  
(cid:80) s−1 (cid:80)p−1 T−m −s+1  
p=1 m=0 p  
The expression mention the maximum period of a pattern from proposition 2,  
where σ was replaced by s since we only want projections which contain at least  
one length-s window for any s. The outer summation is over all possible periods  
that find from 2. The inner summation is over all possible phase offset values  
m for a given period p. Finally, the term inside the summation is the number  
of length − s windows in any projection, where |π | has been substituted from  
p,m  
Propositionrefprojectionlength. Wenowsumthisexpressionoverallpossiblevalues  
of s, which run from σ to T, and relax the floor and ceiling expressions for an  
asymptotic closed form approximation.  
⌊T−1⌋  
(cid:88) T (cid:88) s−1 (cid:88) p−1 (cid:18)(cid:24) T −m (cid:25) (cid:19)  
−s+1 (2.1)  
p  
s=σ p=1 m=0  
⌊T−1⌋  
(cid:88) T (cid:88) s−1 (cid:88) p−1 (cid:18)(cid:24) T −m+p (cid:25) (cid:19)  
≈ −s+1 (2.2)  
p  
s=σ p=1 m=0

2.2 RELATED WORKS ON STRUCTURED DATABASES 19  
From the above formula 2.2, we find the simplifies expression O(T2.H(T−1)),  
σ−1  
where H(n) = (cid:80)n 1 is approximated by ln(n). Thus, the number of closed PSEs  
k=1 k  
at minimum support σ is O(T2ln(T)).  
σ  
2. Theorem. Periodic Subgraph Mining in dynamic networks is in P.  
Proof: Suppose an algorithm that outputs the maximum common subgraph of  
everyσ lengthwindowofeveryprojection. Sincethevertexesandedgesareuniquely  
labeled that the common can be found in time O(V +E) [35]. In the worst case,  
the O(T2ln(T/σ)) periodic patterns computing time Θ((V + E)T2ln(T/σ)) that  
guaranteed to every closed periodic subgraph is mined. Thus, the mining problem  
is in P.  
2.2.1.3 Basic Description of PSEMiner Method  
The main algorithm has been used a special data structure called pattern tree. This  
structure maintains information about embedding subgraphs seen up to timestep  
t and tracks subgraphs that is periodic or might become periodic at some point  
in future. At timestep t, the graph G is read and the pattern tree is updated  
t  
with modification, addition and deletion tree nodes information. Each tree node  
represents one unique subgraph. The most important parameter of this algorithm is  
the maximum period P . When the P is restricted, the algorithm perform as  
max max  
an online algorithm, retaining only the parts of dataset in memory that periodicity  
be calculated. However, in many applications, this information is irrelevant such as  
sensor data streaming. In this case, unrestricted maximum period value must be set  
and requires large computational burden and the entire dataset hold on in memory.  
Data Structures: The algorithm maintains five primary data structures to track  
PSEs.  
Pattern tree: Thetreestructurecharacterizesasubgraphrelationshipamong  
periodic subgraphs. It maintains all PSEs up to timestep t and tracks all periodic

2.2 RELATED WORKS ON STRUCTURED DATABASES 20  
subgraphs those are periodic or might become periodic in future.  
Tree node: The node of pattern tree is tree node. Each tree node contains  
a different subgraphs and a list of descriptors. Descriptors modification, addition  
and deletion are the primary operations on the tree node. Except root node, every  
tree node must observe; all descendants of a tree node are associated with proper  
subgraphs of F, but not all subgraphs of F are necessarily its descendants in the  
tree.  
Descriptor: A descriptor D is one kind of data structure that represents peri-  
odic support set (t ,p,s) for tree node, where starting time t , period p and support  
i i  
s. It is unique for subgraphs F. The last time of descriptor define t = t +p(s−1)  
j i  
and the expected time t = t +p. At time step t, a descriptor is alive if t ≥ t  
e j e  
otherwisedescriptorisnotaliveandmustbeflashedoutfromthepatterntreeandif  
it satisfy minimum support than write subgraph as PSE. A descriptor where t = t  
i j  
is a special case called anchor descriptor it does not represent periodic support set  
and it always live unless P is defined and t−t > P that means it in no longer  
max i max  
needed.  
Subgraph hash map: The random access of tree node that is associating  
with subgraphs is required finding subgraphs position directly. For this purpose,  
subgraph hash map function exists for graphs since the set representation R has a  
global ordering by R ⊂ N.  
Timeline list: Thetimelinelistisanoptionalcomponentthatlinkstreenodes  
to the future timesteps at which they are expected to appear.  
Description of Pattern Tree Update: The update process of pattern tree is  
the core part of this periodic subgraphs mining methods. Initially tree node of the  
pattern tree is empty. At each timestep t, graph G traverses the pattern tree in a  
t  
breadth-first search (BFS) to update tree node with the new information contained

2.2 RELATED WORKS ON STRUCTURED DATABASES 21  
in G . This leave out every tree node N with subgraph F which MCS(F,G ) = Φ.  
t t  
The algorithm first search MCS(F,G ) is in hash table if the node exists in pattern  
t  
tree then update it otherwise a new node is created as a child of N. Moreover,  
entire graph G exists in pattern tree ensure by adding a new child of root as an  
t  
anchor node. At each time t, graph G traverse the tree, one of the following three  
t  
conditions holds at each treenode N with subgraph F. Let C = MCS(F,G ) be the  
t  
maximal common subgraph of G and F.  
t  
Update descriptors: If F subset of G , that means F has appeared entirely  
t  
at timestep t. Suppose D is a descriptor in N and t = t +p is the next expected  
e j  
time.  
(a) If t = t, then time step t is added to D to ensure temporal maximality.  
e  
(b) If t < t, then D is no longer live. It is written to the output stream if its  
e  
support is greater than or equal to σ , and removed from the tree.  
(c) If t > t, then the expected time has not been processed yet, so nothing is  
e  
happened.  
(d) If D is an anchor descriptor then timestep t might be second occurrence of  
F, a new descriptor D′ is created with period p′ = t − t and phase offset  
i  
m′ = (t −1) mod p′. If N does not contain a descriptor with the same period  
i  
and phase offset of D′ then add D′ as a descriptor at N.  
ForeverychildnodeN′ withsubgraphF′ ofNthatF′ ⊂ F ⊆ G . Sotheprocess  
t  
updates all descriptor of the N′ without calculating MCS, that save computation  
time.  
Propagate descriptors: Let C is not empty the above condition does not  
hold then C ⊂ F is present at timestep t. If a treenode for C does not already exist  
in the tree, determined using the subgraph hash map, it is created as a child of N  
with subgraph F. If D be any descriptor at N and t = t, then D represents a PSE  
e

2.2 RELATED WORKS ON STRUCTURED DATABASES 22  
which subgraph C must inherit and continue. The treenode for C receives a copy  
of D, if a live descriptor of the same period and phase offset does not already exist.  
The subgraph F with descriptor D is written to the output stream if the support of  
D is greater than or equal to σ, and then D is removed from treenode N.  
Dead Subtree: If C is empty, then there are no common subgraph between  
G and F and no descriptor at N are directly affected by the observation of G . The  
t t  
common subgraph calculations for all descendant of N are avoided.  
Figure 2.4: An example of a dynamic network [1].  
Figure2.5showsthestructureofpatterntreeduringtheexecutionofthemethod  
at each timestep on the dynamic network from Figure 2.4. For simplicity, we have  
describedaespeciallybasicversionofthetechnique. Themainaspectofthismethod  
isthatitoutputsallPSEs,whichsupersetareallPPSEs. Non-parsimoniousPSEs  
can be post-processed out of the output.  
2.2.1.4 Extension to the PSEMiner Technique  
Mining Parsimonious PSEs: The most significant improvement of this tech-  
nique is to mine only parsimonious PSEs in dynamic networks. Mining parsimo-  
nious PSEs from PSEMiner an indicator bit is set to each descriptor to indicate  
subsumption. The indication bit is cleared when the descriptor is created. When  
any descriptor D from tree node N with subgraph F is flushed, its subsumed bit is  
checked if it is cleared, the D is compared to all other live descriptors at N. If D  
is subsumed by another descriptor, it is not written to the output. On the other

2.2 RELATED WORKS ON STRUCTURED DATABASES 23  
Figure 2.5: The pattern tree at each timestep for the dynamic network shown in  
figure 2.4, considering only edges for brevity [1].  
case, if D subsumes some other descriptors D′, the subsumed bit for D′ is set. If  
the support of D′ increases in the future, its subsumed bit is cleared. However, if  
the indicator bit is set then the descriptor is not written in output as parsimonious  
PSE.  
Including Smoothing: In view of the fact that real-world dynamic networks  
are unlikely to contain perfectly periodic subgraphs, Lahiri and Berger-Wolf [1,10]  
usedsmoothingasamechanismforaccommodatingimperfectperiodicitysubgrpahs.  
Given a user-defined smoothing parameter S ≥ 1, the dynamic network DN =<  
G ,G ,...,G > is mapped in a new network G′, in which each element G =  
1 2 T i′  
(cid:83) (cid:83)  
G ... G +S. However, the following two circumstances handle the elimination  
i i  
of artifacts introduced by the smoothing process.  
1. The minimum period P is set to S.  
min  
2. PSEs of the same subgraph those share the same period, and those differ in  
their starting positions by at most S −1 timesteps, are merged together. In other  
words, only the PSE with the highest support is retained.  
By introducing this smoothing mechanism, they allow a window of timesteps  
within which the order of events does not matter. No smoothing is performed at  
S = 1.

2.2 RELATED WORKS ON STRUCTURED DATABASES 24  
Sorted descriptor list : Descriptor list at each node can be stored sorted by  
the next expected timestep. So, for a given time step, only those descriptors are  
read that expected time lass than or equal to current time. This process reduces  
the number of descriptors that need to be examined during each tree update, at the  
computational cost of having to sort the list of descriptors after each update. If the  
number of descriptors per tree node is not very large, the computational overhead  
is minimal in practice.  
Lazy tree updates: Generally, thealgorithmspendsmostoftherunningtimefor  
calculating intersections of integer sets. However, the MCS of two graphs is calcu-  
latedintimelinearinthenumberofverticesandedges, thesizeofthegraphsresults  
in a relatively expensive intersection computation. The sparsity of the network gen-  
erally results in a relatively small number of tree nodes, which means many such  
intersections between large sets must be performed. Thus, to improve the practical  
efficiency of the algorithm, we can delay calculating intersections until it is essential.  
Using a timeline to trim the tree: The timeline is a technique that associates  
each future timestep with a list of tree nodes those have at least one descriptor  
expected at that timestep. It can be dynamically updated unimportant cost once  
per tree node and stored in space linear in the number of tree nodes. After the  
tree update at timestep t, all tree nodes are still associated with timestep t are  
guaranteed not to have been visited during the tree update, and have at least one  
descriptor that is no longer periodic. These tree nodes can be visited and the invalid  
descriptors removed. Thus, at the end of each tree updates operation, the tree node  
onlycontainsdescriptorsthosearelivedatthenexttimestep. Thistechniqueensures  
that the pattern tree contains a minimal number of descriptors and tree nodes at  
any given timestep.

2.2 RELATED WORKS ON STRUCTURED DATABASES 25  
2.2.1.5 Space and time complexity  
Theorem 1 shows that the maximum number of PSEs are O(T2ln(P )) where T  
max  
is the total timestep and P is the maximum period. For every time step t, the  
max  
tree is completely traversed. Thus, the worst-case time complexity of the algorithm  
involves traversing each descriptor in the tree once for each timestep and calculating  
the MCS at each treenode. The MCS of two graphs can be calculated in time  
O(V +E). ThisyieldsatotaltimecomplexityofO((V +E)T3ln(P )). Theworst-  
max  
casespacecomplexityofouralgorithmisO((V +E+P2 )T2ln(P )whenP is  
max max max  
specified. If P is unrestricted then the time complexity is O((V +E)T3ln(T/σ))  
max  
and space complexity is O((V +E))T2ln(T/σ)+T3lnT.  
2.2.2 ListMiner  
The ListMimer algorithm in [22,40] mine periodic pattern in dynamic networks  
that improves the worst-case time complexity of PSEMiner by a factor T. In the  
PSEMiner, it can be observed that at each time step t, the graph G , traverses  
t  
every node of pattern tree. At each node N the following options occur: (i) the  
maximal common subgraph C = MCS(N,G ) is computed (ii) the corresponding  
t  
node N with subgraph C is searched in the pattern tree, and (iii) each descriptor at  
1  
node N is then checked for consistency of periodicity in t. However, the descriptor  
1  
is updated, otherwise either it is deleted or no action takes place. If no action is  
taken, the time consuming MCS computation is useless.  
ThemainideaoftheListMiner algorithmisreducemaximalcommonsubgraph  
computation. Forthisinspiration,onlytimestepsinwhichthegraphsareintersected  
has been considered that contain a periodic subgraph. Consider a fixed period p,  
everytimesteptbelongstoasingleprojectionπ ,wherem=(tmodp). Thus,every  
p,m  
projection is considered separately, because graphs belong to different projections  
cannot be periodic with the same period p.  
More correctly, for a fixed period p, the T timesteps are partitioned into p  
projections. For example, if p = 1, there is a single projection contains all the

2.2 RELATED WORKS ON STRUCTURED DATABASES 26  
timesteps graphs. If p = 2, all graphs are partitioned two projections such as  
π =< G ,G ,G ,G ,... > and π =< G ,G ,G ,G ,... >, etc. However, a list  
2,1 1 3 5 7 2,0 2 4 6 8  
iscreatedforeveryprojection. Thislistcontainstheintersectionbetweeneverypos-  
sible sequence of consecutive graphs. For example if the projection is π the list is  
2,1  
(cid:84) (cid:84) (cid:84) (cid:84)  
composed by G ,G ,...,G G ,G G ,...,G G G and so on. In this way  
1 3 1 3 3 5 1 3 5  
every possible PSE is generated. By iterating the process for all possible choices  
of period p, all PSE will be found. In the following subsections, these concepts are  
formalized and the discussion of the technique is given.  
2.2.2.1 Preliminaries  
11. Definition. Projection π =< G′,G′,...,G′ > is given, where G′ = G  
p,m 1 2 x j pj+m  
and x = ⌈(T −m)/p⌉, we call run S every subsequence of consecutive graphs from  
i,j  
π . For two fixed indexes i and j, 1 ≤ i ≤ j ≤ x we have S =< G′,...,G′ >.  
p,m i,j i j  
4. Proposition. Every timestep t, graph G for period p belongs to a single projec-  
t  
tion π where m=t mod p.  
p,m  
Proof: From definition, G ∈ π if t=qp+m for some q. It is known that for  
t p,m  
every t ∈ Z there is a unique remainder m ∈ N such that t=qp+m where p,q ∈ Z  
and p > 0. Since m is unique, G belongs only to π .  
t p,m  
5. Proposition. For a given period p, there are exactly p projections.  
Proof: The previous proposition the number of all possible values of remainders  
m is p.  
In the projection π , the maximal common subgraph for consecutive positions  
p,m  
s ≥ σ is not empty, is a periodic subgraph, and the s consecutive timesteps are part  
of a PSE. Therefore the method considered every period p, every projection π  
p,m  
and computing the MCS among all graphs of every S of length at least σ, and  
i,j  
saving only subgraphs which are temporally maximal. Using the following property  
the MCS of every run can be calculated in time V +E.

2.2 RELATED WORKS ON STRUCTURED DATABASES 27  
2. Property. Given a dynamic graph stream < G ,G ,...,G > where 1 ≤ i ≤  
i i+1 x  
x ≤ t, the maximal common subgraph (MCS) of this run is:  
MCS of < G ,G ,...,G > = MCS of < MCS of < G ,G ,...,G >,G >.  
i i+1 x i i+1 x−1 x  
Proof: Using the associative property of intersection between sets this property  
can be proved. Since from definition 11 every graph can be considered as a set of  
natural numbers, the intersection of < G ,G ,...,G > is equal to the intersection  
i i+1 x  
between < (G ,G ,...,G ),G >. Using this theorem, the MCS of a given run  
i i+1 x−1 x  
S , can be obtained calculating the MCS between S and the j-graph of run  
i,j i,j−1  
S . The time needed for such intersection is V +E.  
i,j  
2.2.2.2 Data Structures  
Mining periodic subgraphs in dynamic networks ListMiner algorithm uses three  
primary data structures: lists, listnodes and a bidimensional array that contains  
every list. To mine only parsimonious subgraphs another data structure hash map  
is necessary.  
Everylist containssome listnodes thatisassociated toaspecificprojectionπ .  
p,m  
Every listnode describes a run S of the projection and used to describe a single  
i,j  
temporally maximal PSE.  
List: Everyprojectionπ , isassociatedwithaspecificlist, where1 ≤ p ≤ P ,  
p,m max  
0 ≤ m < p. Every node of the list contains the MCS among all graphs of projection  
π . Specifically, each time step t, every list L contains listnodes in reverse order  
p,m  
where S is the first node and S is the last node. The MCS of all runs S  
t,t 1,t x,t  
is temporally maximal, where 1 ≤ x ≤ t. Therefore, each node N in the list has  
a graph that is properly contained in the graph of its predecessor. This property  
allows efficient traversal of every list by the mining algorithm, and also allows the  
list to be built and manipulated quickly.  
Listnode: Asingleprojectionπ isrepresentbyalistL,everylistnodedescribes  
p,m  
a single PSE that contains:

2.2 RELATED WORKS ON STRUCTURED DATABASES 28  
Start index: This is the first timestep index of the PSE;  
End index: This is the last timestep index of the PSE  
Graph G: It is the graph, which represent the PSE. The MCS subgraph between  
all graphs from timestep T to timestep T is mentioned by graph G which  
start end  
indexes differ by period p.  
Support: It is the number of elements of the support set. It is calculating by  
following way: Support = (T −T )/p. This data structure is equivalent to the  
end start  
descriptor used in Berger Wolf ’s algorithm [1].  
Bidimensional array: The bidimensional array A is used to store all lists. The  
list associated to the projection π is stored at position A[m][p]. Using A allows  
p,m  
to perform list lookup in constant time.  
Subgraph hash map: For mining parsimonious PSE this data structure is used.  
ThekeyofhashmapisagraphidthatiscalculatedbyMCSamonggraphs. Forevery  
key, the associated object is a list of descriptors. Descriptor is a triple < p,s,e >  
where p is the period, s is the first timestep and e is the last timestep of the PSE.  
This information is added to the hash map when a listnode is flushed out from the  
list.  
Since two PSE could have the same graph, the object associated to every key  
is a list of descriptors. Before writing in output a PSE P with graph G, G is used  
as a key to access to the hash map. If it exists in the hash map, the corresponding  
list is traversed, and for every descriptor D the algorithm controls if D subsumes P.  
Otherwise it is printed in output and its descriptor is added to the list. If it does  
not exist in the hash map then P is the first PSE with graph equal to G. This  
means that all other PSEs with graph equal to G will have a period greater than  
the period of P. Therefore, P can be safely printed in output because it cannot be  
subsumed.

2.2 RELATED WORKS ON STRUCTURED DATABASES 29  
2.2.2.3 Parameter  
The algorithm is like as PSEMiner algorithm that is a single-pass, polynomial  
time and space algorithm for mining all closed PSE in a dynamic network. It does  
not require any parameters, but it optionally accepts the following: (i) Minimum  
support threshold σ ≥ 2 and (ii) Maximum period P (default: unrestricted).  
max  
2.2.2.4 Description of the ListMiner Technique  
ThealgorithmstartscreatinganemptybidimensionalarrayA. Attimestept, graph  
G is read and stored the list in the bidimensional array in position A[p][m] where  
t  
m = tmodp. Beginwith, anewlistnodeN = (G ,t,t,1)isaddedattheheadofthe  
t  
list because it could be the first element of a future PSE. Thereafter, the function  
update is called that calculates the MCS between the graph in each listnode and  
G . Whenever a PSE of a subgraph node is detected; it is checked for subsumption,  
t  
and eventually printed in output. When all timesteps have been elaborated, some  
listnodes could remain in the lists because the next expected time of those graphs  
could be equal or greater than T. However, the algorithm must further check others  
PSE subsume the PSE. If it is not subsumed and the support of every PSE is  
greater than, or equal to σ then it must report it in output. For more clearness, the  
description of update stage and subsumed stage are given below.  
Update Procedure: The update procedure is the core part of this ListMiner  
algorithm. For every timestep t all the lists associated to projections π , where  
p,m  
p ≤ min(t,P )andm = tmodp,areupdatedwiththenewinformationcontained  
max  
in G . The update process of list L is started by adding a head listnode for G . This  
t t  
listnode is built as follows: the graph is set to G , start and end indexes are set to  
t  
t because t is the first and the last index of the run, and the support is set equal  
to 1. During the traversal of the list, one of the following three conditions holds at  
(cid:84)  
each listnode N with graph F. Let C = F G be the MCS of G and F.  
t t  
1. If F ⊆ G that means subgraph C = F entirety appeared in G . Therefore  
t t

2.2 RELATED WORKS ON STRUCTURED DATABASES 30  
the MCS is F, and the listnode is updated in the following way:  
o Graph and start index are unaffected  
o End index is set to t because the last timestep where the MCS(C) occurs is t.  
o Supportisincrementedbyoneunitbecausethereisanothertimestep(t)where  
C appears.  
Given that all successors of a node N with subgraph F′ ⊂ F ⊆ G , then F′ is  
t  
a also subgraph of G . However, the algorithm updates all successors of node N in  
t  
the same way without calculating the MCS, thus saving computational time.  
2. If C is empty, that means MCS(G ,F) = ϕ have no common subgraph. List  
t  
node N with subgraph F and all its successors are eliminated from the rest of the  
list and, if their supports satisfy the minimum support, flushed out from the list and  
store in output as PSE.  
3. If C is not empty and F ̸⊆ G , a subgraph C of F is present at timestep t.  
t  
In this case the algorithm first check if the listnode parameters describe a subgraph  
that is frequent and not subsumed. If it is so, it is printed in output. Then the  
algorithm updates the listnode N in the following way:  
o Graph is set to C.  
o Start index is unaffected.  
o End index is set to t because C appears at time t.  
o Support is equal to support(N)+1.  
The next listnode in the list is then considered.  
Moreover, whenever the update involves not just the start/end indexes, but also  
the graph variable. If they are equal, the previous node is deleted, since it would  
represent the same graph within a smaller periodic interval, therefore it would not  
respect the condition of temporal maximality.

2.2 RELATED WORKS ON STRUCTURED DATABASES 31  
Subsumed Procedure: This procedure mine parsimonious PSE. In order to do  
this procedure uses a subgraph hash map H. For a given PSE, P with graph F the  
procedurechecksifalistassociatedtoF existsinH. Ifnot, thenP isnotsubsumed  
becauseitisthefirstPSE withgraphF andprintedinoutputandstoredinthehash  
map. Otherwise, for every descriptor of the list the algorithm verifies its existence  
in other descriptors that respects parsimonious conditions If there is periodicity P  
that is subsumed and it is not printed in output, otherwise P is memorized in the  
hash map and flushed in output.  
2.2.2.5 Time and space complexity  
In the above discussion it has been observed that there are exactly p projections for  
a given period p and the length of every projection is |π | = ⌈(T −m)/p⌉.  
p,m  
Since the algorithm creates a new listnode for every element of the projection  
the maximum number of listnode is the length of the projection. For every timestep  
t and for every list, in the worst case the algorithm calculates the MCS for every  
node of the list and Gt.  
Therefore the number of MCS is:  
P (cid:88)max (cid:88) p ⌈(T− (cid:88) m)/p⌉  
(j) (2.3)  
p=σ m=o j=0  
The summations are: for every period p, for every projection with period p the  
algorithm creates a list. The number of elements in the list is increased by one at  
every step. Therefore, also the number of MCS to calculate is increased by one at  
every step, from 0 to the maximum length of the projection.  
Since the MCS can be computed in O(V + E) time, the total complexity of  
the basic algorithm (without subsumption) is O((V +E)T2ln(P ). Since P  
max max  
is unrestricted in the worst case, its maximum value is O(T/σ). Therefore the  
complexity time in the worst case is O((V +E)T2ln(T/σ) that is smaller by a factor  
T than PSEMiner [1,10].

2.3 LIMITATION OF EXISTING WORKS 32  
For every period p there are p projections with O(T/p) elements. Therefore the  
number of listnodes for every period is O(T). Every listnode contains an associated  
graph. Therefore the total space complexity is O(P (V +E)T). In the worst case  
max  
P is unrestricted (P = O(T/σ)), so the space complexity is O((V +E)T2/σ).  
max max  
2.3 Limitation of Existing Works  
From the previous sections, it has been shown for mining periodic patterns each  
graph G needs a large number of MCS computation. If dynamic networks density  
t  
were medium or high, its computation cost would be very high. However, one effi-  
cient method that reduces number of MCS computation and mine periodic patterns  
efficiently in dynamic networks is very essential.

Chapter 3  
Supergraph Based Periodic Behaviors Mining  
(SPBMiner)  
This chapter presents the main contribution of the thesis: the design and develop-  
ment of SPBMiner, a supergraph based periodic behaviors mining technique that  
improves the worst-case time and space complexity of PSEMiner and ListMiner  
algorithms. From the previous chapter, it has been observed that in PSEMiner  
that at each time step t, graph G must traverses every node of the pattern tree that  
t  
build by graphs < G ,G ,...G >. The exhaustive visit of pattern tree performed  
1 2 t−1  
ateachtimestepisinefficient. Italsogeneratesuselesscommonsubgraphswhentree  
node expected time is less than current time t. An alternative approach ListMiner  
has been proposed where the graphs in dynamic networks are partitioned based on  
period p. This process creates unique list for all possible period p, and phase m  
= t mod p. It traverses only list nodes rather than the trees from the graph and  
the common subgraph is created when it is necessary. It is shown in chapter 2 that  
at timestep t, ListMiner stores same graph G at p times for each period p. The  
t  
approach is extremely memory consuming because dynamic networks are generally  
large. Both of these two techniques store entire subgraphs differently though the  
numbers of interactions are same. If one interaction is changed, the whole graph  
should be restored. This redundant information storing is exceedingly memory con-  
suming.  
The key idea of the method proposed in this thesis is to reduce the num-  
ber of common subgraph computation. It needs only one MCS calculation at  
each timestep graph G . On the other hand, all common and uncommon behav-  
t  
33

3.1 PRELIMINARIES 34  
iors/subgraphs entities among dynamic networks are stored only once that requires  
lessmemoryandcapableofavoidingredundantinformationstorage. Moreprecisely,  
we propose supergraph based period behavior mining algorithm called SPBMiner.  
Supergraph is a graph stores information of all graphs with the common patterns  
of the graphs being stored only once. In the next sections we will describe the  
methodology of SPBMiner. At each time t, find periodicity of all entities periodic-  
ity individually and identify periodic entities. After selecting periodic entities, those  
entities are combined and periodic behaviors/subgraphs are recognized. The follow-  
ing sections formalize these concepts and detailed descriptions of the algorithms are  
presented.  
3.1 Preliminaries  
Dynamic networks are a representation of interactions among a set of unique popu-  
lations that change from time to time. Let V ∈ N represent the set of populations.  
Interactions among populations may be directed or undirected and are supposed to  
have been recorded over a period of T isolated timestamps. We use natural quanti-  
zation, specific to each of our dataset such as one day/ one hour per timesteps. The  
only essential is that a timestep represents a meaningful amount of real time where  
the periodicities of mined behavior subgraphs will be set of chosen timestep.  
12. Definition. DynamicNetworks: AdynamicnetworkDN =< G ,G ,...,G >  
1 2 T  
is time series graphs, where G = (V ,E ) is a simple interactions E among popu-  
t t t t  
lations V ∈ V at timestep t. Interaction and populations denoted as follows. These  
t  
interactions and populations are called entities in the rest of this thesis.  
(i) l(v ) is the unique label of population v ∈ V .  
i i t  
(ii) Interaction between v and v represent as (v ,v ) ∈ E where l(v ) < l(v ).  
i j i j t i j  
Figure 2.3 shows the example of a dynamic network with five timesteps. Definition  
12 reduces the high computational complexity of many algorithmic tasks on graphs  
database. Since a vertex represents a unique population, each timestep t, vertex

3.1 PRELIMINARIES 35  
set V in graph G are unique that reduce computational complexity for certain  
t t  
hard graph mining problems, such as maximum common subgraph and subgraph  
isomorphism testing [35,36].  
13. Definition. Supergraph: Supergraph is a graph database that stores all the  
graphs into one graph with the common subgraphs of the graphs being stored only  
once.  
Figure 3.1 shows the supergraph SG that compacts two graph G and G where  
1 2  
common subgraph is stored only once.  
1 2 1 2 1 2  
4 3 4 3 4 3  
G G SG  
1 2  
Figure 3.1: Compact Supergraph.  
3. Property. Graph Representation: For a graph G = (V, E) with unique vertex  
labels, the set representation ℜ for G is formed by vertex and edge represent as two  
unique vertex labels in ℜ where ℜ is natural number.  
Sinceeachvertexisuniquelyexpressedbyitslabel,eachedgeisalsoexpressedby  
the interaction between two unique vertexes. This allows each vertex to be labeled  
asauniqueinteger,evenacrossdifferentgraphoverthesamevertexset. Twographs  
will be same if their vertex label sets are same and its corresponding edge means  
connected vertexes label sets are same. Figure 3.2 shows two graphs G and G  
1 2  
where vertexes sets are same but edges sets are different that’s why they are not  
same. Connectivity information is remain unchanged in this representation. Each  
vertex is connected with other vertexes as connected edges.  
4. Property. Subgraph Testing: The measurement of subgraph testing whether G  
1  
is a subgraph of G or vice versa can be done by checking unique vertex label repre-  
2

3.1 PRELIMINARIES 36  
Figure 3.2: Graph representation with unique vertex label and corresponding con-  
nected vertex label  
sentation sets and their corresponding connected edge edge label of G is subset of  
1  
G or vice versa.  
2  
5. Property. Maximum Common Subgraph (MCS): The MCS between two graphs  
is defined by common vertex label and their corresponding common connected vertex  
label. It may be connected or disconnected subgraph. We use intersection operation  
(cid:84)  
to represent the common subgraph of two graphs.  
From figure 3.2 we find the maximum common subgraph contains all unique ver-  
tex labels < 1,2,3,4 > and two common connected edge (1,2) and (3,4). Following  
figure 3.3 shows the maximum common subgraph representation and its structure.  
1 2  
4 3  
MCS  
Figure 3.3: Maximum common subgraph between G and G  
1 2  
6. Property. Hashing: Since the unique vertex labels set represent by ℜ has a  
global ordering by feature of ℜ ∈ N, A graph can be hashed like a vertex set. Each  
vertex connected with other vertexes that also represent by ℜ ∈ N, another hashed  
can be used for denote connected edges within graph hashed.

3.1 PRELIMINARIES 37  
Forperiodicpatterns, timeindicationisparticularlyimportant, becauseperiodic  
patterns depend on period and starting time. A pattern occurring number is also  
essential for counting support.  
14. Definition. Periodic Support Set: Given a dynamic network DN of T  
timesteps and any subgraph F = (V, E). The periodic support set S (F) of F in  
p  
DN is the set of all timesteps t, which start at t timestep and repeating every p  
i  
timesteps where F is a subgraph of DN, which denote F subgraph of G . The rep-  
ti  
resentation of support set S (F) = (t ,p,s) = {t ,t +p,...,t +p(s−1)}, such that  
P i i i i  
∀t (t ∈ S (F) ↔ F ⊆ G ) and neither G nor G contains F as subgraph.  
i i p ti ti−p ti+ps  
F isafrequentperiodicpatternifitssupportexceedsauserdefinedminimum  
support threshold value σ.  
Definition 14 is the formulation that is derived from the well known frequent  
pattern mining problem that satisfy the downward closure property. In downward  
closure property, every sub pattern of a frequent periodic pattern F is also frequent.  
15. Definition. Closed Subgraph: For any subgraph F = (V, E) in a dynamic  
network DN of T timesteps is closed if it is maximal for its support set. In other  
words, while F support is maintaining, no vertex or edge can be added of F.  
There are a difference between frequent closed subgraph support set and peri-  
odic closed subgraph support set. A single subgraph F can have multiple periodic  
pattern support set to allow disjoin and overlapping periodic behavior. Thus, we  
requireextractionofallperiodicsubgraphembeddings, ratherthanjusttheperiodic  
subgraphs. The basic definition of periodic subgraph embeddings (PSEs) is given  
at definition 6 there is the possibility that a periodic subgraph embedding carries  
information in other periodic subgraph embeddings. Suppose a periodic graph’s  
period is p, it is also periodic at 2p and for every multiple of p, which depends on  
only the threshold value. If they are frequent, they will be the output as periodic  
embedding and redundancy problem is occurred.

3.2 SUPERGRAPH BASED BEHAVIORS MINING 38  
Mining parsimonious periodic subgraphs embedding is a well-designed solution  
to the redundancy of general periodic patterns. Since it captures all the information  
and produces small number of output results that is defined in the previous chapter  
at definition 7. It maintains subsumption property in property1. For example, a  
subgraph F of period 1 with adequate support 10, then it is also periodic at period  
2 and 3 if minimum threshold is 3. In this case, PSE at period 2 and 3 are not  
Parsimonious PSE.  
3.2 Supergraph based Behaviors Mining  
The existing works PSEMiner and ListMiner find all PSEs based on tree and  
list structure. In these cases, tree node and list node represent periodic pattern  
that stores common entities redundantly. These two processes need to convey all  
timesteps graph < G ,G ,...,G > for mining periodic patterns at T timesteps  
1 2 T−1  
graph G . Dynamic networks are online process, total timestamps increase time to  
T  
time that decrease the mining performance because it needs to check all previous  
graph information or their common subgraphs. To solve this kind of problem we  
propose supergraph-based method that stores common entities of all graphs only  
onceandstorestheperiodicinformationoftheentitiescalleddescriptors. Formining  
periodic patterns, it needs only one comparison between current graph and previous  
supergraph.  
We now introduce SPBMiner, our proposed algorithm for mining all periodic  
subgraphs in dynamic networks. At first, we start by describing the most basic form  
of the algorithm, which mines periodic subgraphs. Then prune non-closed and non-  
parsimonious periodic subgraphs and mine parsimonious periodic subgraphs. After  
that, we mention some simple optimization of the basic algorithm that improves the  
performance. The basis architecture of SPBMiner has been shown in figure ??.  
SPBMiner works on following idea: at each timestep dynamic network is read,  
we maintain a supergraph embeddings all networks entities seen up to timestep t.  
This supergraph maintains two kinds of data structure for each entity. One is time-

3.2 SUPERGRAPH BASED BEHAVIORS MINING 39  
BehaviorSupergraphModel  
Behavior Supergraph Updates  
roivaheB  
tnerruC  
hparG  
Supergraph based periodic behaviors mining architecture  
Dynamic Networks  
Enron, Facebook  
Descriptor set Time set  
Applications ParsimoniousBehaviors Periodic Behaviors  
Corporate Hierarchy,  
Parsimonious Closed  
Groups Analysis, Similar Periodicity  
Periodic Periodic  
Recommendation System, Behavior Matching  
Behaviors Behaviors  
Behaviors Analysis  
Figure 3.4: SPBMiner Architecture  
set, which stores active time of entity. Other one is descriptor list, which represents  
entity periodicity. Once entities cease to be periodic, they are flashed from the su-  
pergraph and insert to periodic hash table as a periodic entity. Periodic hash table  
is one kind of data structure that stores entities base on period and starting time. If  
agroupofentitiesflashedoutsametimeandtheirperiodandsupportaresamethen  
store together and build periodic subgraph. After mining periodic subgraphs we use  
another kind of hash that stores subgraph as key and corresponding hash table keys  
represent descriptors information save as value. Each descriptor is checked when it  
is saved in value list. If its supergraph with the same descriptors does not exist, it is  
closed. If it is not subsumed by other descriptors, it is considered as parsimonious  
descriptor and subgraph is the parsimonious subgraph for the descriptor. The al-  
gorithm parameters, data structures and proof of correctness are described in the  
following sections.

3.2 SUPERGRAPH BASED BEHAVIORS MINING 40  
3.2.1 Parameters  
Our algorithm is online periodic subgraph mining single pass and space algorithm  
for mining all parsimonious periodic subgraph embeddings in a dynamic network.  
It needs two parameters as follows.  
i. Minimum support threshold σ ≥ 2 (default 3)  
ii. Maximum period P (default 40).  
max  
The online algorithm bound the maximum period of supergraph entities. When  
entitiescreatenewdescriptor, itgeneratesmaximumP numberofdescriptorand  
max  
the minimum support threshold is used to indicate flushed entity is periodic or not.  
3.2.2 Data Structure  
As the algorithm looks into the stream graphs, it maintains three kinds of data  
structures: timeset, descriptor set and periodic subgraph hash table.  
3.2.2.1 Descriptor  
16. Definition. Descriptor: A descriptor D is one kind of data structure that rep-  
resents periodic support set (t ,p,s) for any entities, where starting time t , period p  
i i  
andsupports. DescriptorDforanentityEisliveifit’sexpectedtime(t = t +ps) >  
e i  
current time t or if t = t and entity E is present at G . A descriptor that is not  
c e c t  
live considered as a periodic entity if it support satisfies minimum threshold value.  
In supergraph, each entity contains a set of descriptor that is periodic or be  
periodic in future timesteps. Generating future periodic descriptors, each entity  
needs to store occurred time in previous timesteps.  
3.2.2.2 Time Set  
17. Definition. Time Set (TS): Time set is one kind of data structure that stores  
active time of the entity.

3.3 DESCRIPTION OF THE ALGORITHM 41  
The length of entity time set depends on maximum periodic length. Our proposed  
methodisonlineperiodicpatternsmininginthatcaseweconsiderP asmaximum  
max  
period. The timeset maintains the following lemma.  
1. Lemma. The maximum size of TS for any entity is P .  
max  
Proof: Periodic entity repeats every p period time intervals. Descriptor indicates  
entity periodicity information and support. If one entity appears in current graph,  
it updates descriptors of common supergraph entity. If descriptor expected time  
is equal to current time, then update these descriptors information. On the other  
hand, current entity generates a set of descriptors that would be periodic next time.  
In this time previous appeared time of entity is needed and these time is stored in  
timeset (TS). For new descriptor, maximum period is P because if it appeared in  
max  
previous and live it already exists in descriptors. However, entity TS is maximum  
P stores all periodic time exclusive of missing any information.  
max  
3.2.2.3 Periodic subgraph hash table  
Supergraphflushedoutperiodicsupportsetforeachentity. Ourmaingoalisfinding  
periodic subgraphs. In support of that principle, we need to add flushed entities  
descriptor to generate subgraphs. Hash table is especially efficient for this variety  
of structure. We have used starting position, period and support combined as a key  
value of hash table and have stored corresponding entities into patterns.  
3.3 Description of the algorithm  
Now we describe the update of supergraph information that is the core part of our  
process for periodic patterns mining. Initially the supergraph SG is empty. At  
timestep t, graph G is read. The common entities between SG and G are updated  
t t  
into SG. Entity updates timeset and descriptor set including addition, deletion  
and modification. Descriptors are flushed at deletion step. If descriptors support  
are greater than minimum threshold value, the entity is periodic. The process at

3.3 DESCRIPTION OF THE ALGORITHM 42  
time t is completed ensuring all uncommon entities in current graph G includes in  
t  
supergraph.  
The entire process partitions three parts. We describe each part in the following  
subsections in details.  
3.3.1 Update Common Entities in Supergraph  
The supergraph entity information: timeset and descriptor sets are updated when  
entity is active. An entity appears in graph G is active otherwise inactive. Each  
t  
graph G , the supergraph should be updated with common and uncommon entities.  
t  
(cid:84)  
Let F = SG G be the maximal common subgraph of supergraph SG and G .  
t t  
3.3.1.1 Timeset Update  
Entity timeset presents active time steps of the entity. Each graph G , time set  
t  
presents only single current timestep t. However, supergraph SG compact a set of  
timeseriesgraphsthatresultitshouldbestoredentityactivetimesteps. Thosetime  
stepsareneededtogenerateentityperiodicitythosearenotperiodicatthismoment  
but would be periodic in future. SPBMiner stores only maximum periodic number  
of timesteps instead of all time because entity appears reiteration at each periodic  
time interval. If timeset size is maximum, which is defined in Lemma 1, then delete  
timesteps using first in first out (FIFO) method.  
3.3.1.2 Descriptors Creation  
When entity E ∈ F, is active at current time t, it can creates some new periodic  
descriptors these are not currently periodic. Using entity timeset, it creates future  
periodic descriptors. Descriptor D period is the difference between timeset time and  
current time.  
2. Lemma. Each entity E, the new created descriptor at time t is maximum P .  
max  
Proof: New descriptor indicates entity periodicity that would be periodic at  
some point in future. In our method, we define maximum period P . Thus,  
max

3.3 DESCRIPTION OF THE ALGORITHM 43  
entity can generate descriptor that period is 1 to P . So, the maximum number  
max  
of descriptor at any time step is maximum P .  
max  
Suppose an entity timeset TS [1, 2, 3, 4] and current time is 5. It can create  
four descriptors. These descriptors periodic support sets are like as (1,4,2), (2,3,2),  
(2,2,2) and (4,1,2) where first element is starting position, second one is period and  
last one is support value.  
3.3.1.3 Descriptors Update  
IfentityE ∈ F, i.e. entityE isactiveattimestept, LetD isadescriptorofE ∈ SG  
and te = t +ps be the next expected time for D. If te = t, then D has appeared  
i  
where it was expected and created a new descriptor D′ = D. Time t is added to D′  
support to ensure temporal maximality. If the support of D is greater than or equal  
to σ then remove from supergraph entity E and stores in the periodic hash table.  
The following property is maintained at updates stage.  
3. Lemma. For entity E, the maximum number of updated descriptors at time t is  
min(t,P ).  
max  
Proof: Descriptor D update occurs when expected time is equivalent to current  
time. Each period p, there is exacts p number of descriptors that should be updated  
at time t. These descriptors are defined by period and phase such as p mod m where  
0 ≤ m < p. Ifperiodp = 4thendescriptorsperiodandphasepairs(1,0), (2,0), (3,1)  
and (4,0) should be updated. The maximum value of period p is P . Therefore,  
max  
the number of descriptors at any time t will be min(t,P ).  
max  
3.3.1.4 Descriptors Deletion  
Suppose entity E has descriptor D that has expected time te < t, then D has not  
appeared when expected and is no longer live. If it’s support ≥ σ, it will be stored  
in periodic hash table as a periodic entity and will remove D from E ∈ SG.

3.3 DESCRIPTION OF THE ALGORITHM 44  
4. Lemma. At time step t, average 2∗P number of descriptors have been deleted  
max  
from entity E.  
Proof: Lemma 2 and Lemma 3 show that at time t, maximum P number of  
max  
descriptorshavebeencreatedandupdated. Thesemeansmaximum2∗P number  
max  
of descriptors are added at each timestep. The number of deleted descriptors will  
be same as added descriptors in entire timesteps. However, the numbers of deleted  
descriptors are not fixed and there is no upper bound because at last step a large  
number of descriptors be alive. In this issue, we can define the average number of  
descriptors deletion at each time step and it will be same as creating and updating  
number 2∗P .  
max  
Each time step t, entity E creates new descriptors, updates descriptors and  
finally deletes descriptors that may be periodic or not. Therefore, the number of  
descriptors at entity E maintain following property define in lemma 5.  
5. Lemma. The maximum number of descriptor of any entity at time t is P2 .  
max  
Proof: The number of descriptors of entity depends on size of entity time set.  
Lemma2provedthateverytimestepentitycreatesP futurebehaviordescriptors  
max  
including period 1....P . If same periodic descriptors exist, then update those  
max  
descriptors by creating new descriptors and delete old descriptors. Each periodic  
descriptor can be started at any position of phase m, where 0 ≤ m < P . Thus,  
max  
the maximum number of descriptors of entity is P2 .  
max  
3.3.1.5 Entities Deletions  
An entity appears in G is active otherwise inactive. Sorting of long time inactive  
t  
entity in graph is inefficient because it needs memory and more computation time  
for mining common subgraph between supergraph and G .  
t  
6. Lemma. Inactive entity survive maximum P time in supergraph.  
max

3.3 DESCRIPTION OF THE ALGORITHM 45  
Algorithm 1: SPBMiner ({BG ,BG ,...BG },σ)  
1 2 T  
Data: BG ,BG ,...,BG : Dynamic behavior graphs at timestep  
1 2 T  
1,2,...,T; σ : min sup;  
Result: Parsimonious Periodic Behaviors Sets  
1 BSG ← ϕ /\* Behavior supergraph is empty \*/;  
2 for t = 1 to T do  
3 for ∀ E ∈ BSG do  
4 if E ∈ BG t then  
5 Update Entity(BSG E ,t,σ,com); /\* Update common entity  
\*/;  
6 else  
7 Update Entity(BSG E ,t,σ,uncom); /\* Update uncommon  
entity \*/;  
8 if t−E|ST|−1 > P max then  
9 Remove(BSG,E); /\* Remove dead entity \*/;  
10 end  
11 end  
12 end  
13 for ∀ E ∈/ BSG and E ∈ BG t do  
14 Add Entity(BSG,E); /\* Add uncommon entity \*/;  
15 end  
16 end  
17 MineLastTimestep Behaviors(BSG,σ); /\* Mining periodic behaviors  
at last timesteps \*/;

3.3 DESCRIPTION OF THE ALGORITHM 46  
Proof: For each entity, maximum P periodic descriptors are contained.  
max  
When one periodic expected time is passed, those periodic descriptors are removed.  
After P time there is no descriptor in entity if entity does not become active  
max  
within this period. When entity descriptor set is empty upto P time, we said  
max  
this entity is dead. Then we can delete those entities from supergraph because there  
is no chance to be periodic within bounded period if it appears in future it will  
create new periodic descriptors.  
Suppose entity E, appears in 1 and 6 timesteps and P = 3 then we can delete  
max  
entity at time step 5 because it already reached dead entity. Next time if it appears,  
it will be considered as new entity and stores newly.  
3.3.2 SPBMiner algorithm  
Algorithm 1 shows the proposed SPBMiner algorithm. The main algorithm per-  
forms from line 2 to line 17. Initially supergraph is empty. At each time step t,  
supergraph is updated based on current graph entities that are mentioned at line 5.  
However, there are entities in supergraph that are not appeared in current graph at  
all timesteps. These entities update occur at line 7. If any entity remains inactive  
in consecutive P times, we can remove it by line 9. There are some entities in  
max  
currentgraphthosedonotexistinsupergraph. Thoseentitiesbecomecompactwith  
supergraph at line 14. Finally, we mine periodic entity by line 17 those are live at  
last time step.  
3.3.2.1 Entity Updates algorithm  
The core part of the SPBMiner algorithm is Update Entity() procedure. This  
procedure tracks entities periodicity and support information that is significantly  
important for mining periodic behaviors in dynamic networks. Algorithm 2 shows  
theentityupdatesalgorithm. EntityEisupdatedincludingdescriptorsetandtime-  
setupdate, deletionandcreation. Descriptoroperationsareperformedbyline1-14.  
When descriptor expected time is equal to current time, it creates new descriptors,

3.3 DESCRIPTION OF THE ALGORITHM 47  
and updates its information and flashed out old descriptors with corresponding en-  
tity by line 3-8. If descriptors expected time are less than current time then flashed  
out descriptors and corresponding entity. These flashed entity would be periodic  
entity if it support is greater than or equal to σ (minimum support).  
The other data structure of an entity is timeset (TS) that stores last P times  
max  
when the entity was active. At timestep t, entity may be periodic or will be periodic  
at some point in future. For generating future periodic descriptor set, the algorithm  
creates new descriptors at line 20. If descriptors do not exists in entity then add  
by line 22 and finally add current time t in timeset TS at line 26. This algorithm  
returns update entity to main SPBMiner algorithm.  
Algorithm 3 shows flashed procedure for descriptor D with entity E. If the sup-  
port of D is greater than minimum support σ then consider as a periodic entity. All  
those entities based on it start timestep, period and support have been combined.  
Forthispurpose,Icreateperiodichashtablekeycombiningstartingposition,period  
and support in line 1. If entity first creates periodic descriptor set into hash table,  
otherwise add to corresponding hash key value by line 6. If hash key contains then  
find out hash value and add current entity and store again based on same hash key  
mentioned by line 4.  
3.3.2.2 Mining Periodic Behaviors  
Periodic hash table values indicate periodic behaviors. However, these kinds of be-  
haviors are neither closed nor parsimonious. When mining closed and parsimonious  
periodic behaviors, we have to maintain two basic lemmas those are defined by  
lemma 7 and lemma 8.  
7. Lemma. Let periodic behaviors F support set S(F) = (m, p, s) then F is closed  
behaviors if there are no S(F′) = (m,p,s′) where s′ > s and F ⊆ F.  
Proof: According to definition 5, closed periodic behaviors are those subgraphs  
which have no superset with same support or same graph with larger support. Then  
straightforward we can find out closed periodic subgraphs from periodic hash table.

3.3 DESCRIPTION OF THE ALGORITHM 48  
Algorithm 2: Update Entity(BSG ,t,σ)  
E  
Data: BSG : Behavior supergraph entity E; t: current timestep; σ :  
E  
min sup;  
Result: Update Behavior Supergraph Entity BSG  
E  
1 for ∀ D ∈ BSGD do  
E  
2 /\* Descriptors update in entity E \*/;  
3 if D.te == t then  
4 D′ = D;  
5 D′.sup = D.sup+1;  
6 D′.te = D′.last+D′.period;  
7 Insert Descriptor(BSGD,D′); /\* Insert Update Descriptor \*/;  
E  
8 Flashed(D,E,σ); /\* Delete Descriptor \*/;  
9 else  
10 if D.te < t then  
11 Flashed(D,E,σ); /\* Delete Descriptor \*/;  
12 end  
13 end  
14 end  
15 for ∀ t′ ∈ BSGTS do  
E  
16 /\* Create new descriptors \*/;  
17 if t−t′ > P max then  
18 Remove(BSGTS,t′); /\* Delete time t′ \*/;  
E  
19 else  
20 D = new Descriptor(t′,t,t−t′,2);  
21 if D ∈/ BSGD then  
E  
22 Add Descriptor(BSGD,D); /\* Add new descriptor \*/;  
E  
23 end  
24 end  
25 end  
26 Add Time(BSGTS,t); /\* Add time t to Time Set \*/;  
E  
27 return BSG E /\* Return update supergraph entity \*/;

3.3 DESCRIPTION OF THE ALGORITHM 49  
Algorithm 3: Flashed(D,E,σ)  
Data: D: Descriptor; E: graph entity; σ:min sup  
Result: Insert into Periodic Hash  
1 hashkey ← CreateHashKey(D.period,D.phase,D.support); /\* Create  
hashkey \*/;  
2 if D.support ≥ σ then  
3 if Periodic Table contains Hashkey then  
4 Periodic Subgraph ←  
(cid:83)  
Find Value(Priodic Patterns,hashkey) E;  
Set Value(Priodic Patterns,hashkey,Periodc Subgrpah,D.start);  
5 else  
6 Set Value(Priodic Patterns,hashkey,E,D.start);  
7 end  
8 end  
Suppose subgraphF = (A,B),(C,D), S(F) = (1,1,3) is closed because position  
(1,1,6) subgraph F′ ̸⊇ F. On the other hand F = (A,B),(B,C), S(F) = (2,2,2)  
is not closed because position (2,2,3) subgraph F′ ⊆ F.  
Using this lemma, I can prune non-closed periodic subgraphs that reduce redun-  
dantinformation. Anotherkindofredundantperiodicitysubgraphsexistinperiodic  
hash table subsumed by others is defined in property 1. Mining parsimonious peri-  
odic patterns by the following lemma puts strongly effective influence.  
8. Lemma. Let periodic subgraph F support set S(F) = (m, p, s) then F is subsumed  
by S(F′) = (β ∗⌊m/p⌋,β,s′) if F ⊆ F′, p mod β == 0 and s′ ≥ p∗s.  
Proof: I can limit myselves to the discovery of all periodic subgraphs of period  
1. If subgraph F is periodic at (m,p.s) support set. It also periodic at (⌊m/p⌋,1,s′)  
of projection of graph, for all 1 < p ≤ P and 0 ≤ m < p. Then it also is periodic  
max  
at divisor of p. If β mod p == 0 then β is divisor and (β ∗⌊m/p⌋,β,s′) is periodic  
support with graph F′. If F′ > F and s′ ≥ p∗s then F is subsumed by F′ .

3.3 DESCRIPTION OF THE ALGORITHM 50  
Suppose subgraph F = (A,B),(C,D), S(F) = (2,2,2) that means it occurs  
2,4,6 timesteps. It may be subsumed by (1,1,s′) where s′ ≥ 2 ∗ 2. That reason  
F′ = (A,B),(C,D) = F S(F′) = (1,1,6) subsumed F which shows in table 3.1. F′  
also subsumed subgraph at position (2,2,3). So F is subsumes by F′ and F is not  
parsimonious.  
Table 3.1: Closed and Parsimonious Periodic subgraphs characteristics  
Hash Key Pattern Closed Parsimonious  
1 1 6 (A,B),(C,D) Yes Yes  
1 1 3 (A,B),(B,C) Yes Yes  
2 2 2 (A,B),(C,D) No No  
2 2 3 (A,B),(C,D) Yes No  
Algorithm 4 shows the parsimonious periodic patterns mining procedure. Each  
hash key mentions the periodicity of pattern F that contains periodic descriptor  
as a hash value. The pattern F checks if a pattern F’ is associated with large  
support and contains periodicity information with F’ periodicity. It checks closed  
and parsimonious patterns. Finally, from the algorithm we can mine parsimonious  
periodic patterns.  
3.3.3 Example  
Suppose the dynamic network in figure 3.5 is the input. We explain our SPBMiner  
algorithm in details systematically. It maintains supergraph update including de-  
scriptorandTSoperationsateachtime. Newperiodicpatternsarefoundandinsert  
into hash table then mine parsimonious periodic patterns from periodic patterns  
hash table. In this example we consider σ = 2.  
At time step 1, graph G is considered supergraph because initially supergraph  
1  
is empty and the representation of supergraph is shown in table 3.2. In this time

3.3 DESCRIPTION OF THE ALGORITHM 51  
Algorithm 4: Parsimonious Periodic Pattern (Periodic Patterns)  
Data: BSG : Periodic Patterns: All periodic patterns;  
E  
Result: Parsimonious Periodic Patterns  
1 Iterator ← Periodic Patterns.begin();  
2 while Itegator < Periodic Patterns.end() do  
3 Hash key ← Iterator.Value();  
4 Descriptor D ← Iterator.Value ();  
5 Find(phase,period,support) ← D;  
6 Hash newKey = new Key(phase,period,sup′);where(sup′ ≥ support)  
7 if Periodic Patterns(newKey)&(F′ =  
Value.Periodic Pattenrs(newkey)) ⊇ F then  
8 Delete.Periodic Patterns(newKey); /\* Delete nonclosed patterns  
\*/;  
9 end  
10 Hash newKey = new Key(β∗⌊m/p⌋,β,sup′); where (sup′ ≥ support)&p  
mod β == 0  
11 if Periodic Patterns(newKey)&(F′ =  
Value.Periodic Pattenrs(newkey)) ⊇ F then  
12 Delete.Periodic Patterns(newKey); /\* Delete non parsimonious  
patterns \*/;  
13 end  
14 end  
15 return Periodic Patterns ;  
A B A B A B A B A B  
D C D C C D C C  
G G G G G  
1 2 3 4 5  
Figure 3.5: Dynamic Networks

3.3 DESCRIPTION OF THE ALGORITHM 52  
periodic patterns hash table is empty.  
Table 3.2: Supergraph representation at timestep 1  
Super Graph Periodic Patterns  
V V Timeset Descriptors Keys Patterns  
1 2  
B A 1 (1,1,1)  
A 1 (1,1,1)  
C  
B 1 (1,1,1)  
B 1 (1,1,1)  
D  
C 1 (1,1,1)  
At time step 2, graph G is read and updates the supergraph. It updates time  
2  
set and descriptors. Except interaction (B −D) all interactions are appeared. The  
common interactions with supergraph creates new descriptors (1,1,2) based on ex-  
isting descriptors (1,1,1). Existing descriptors (1,1,1) are flashed from supergraph.  
It should not be added in periodic patterns because its support is less than σ. No  
new interaction appear so no need to add interactions to supergraph. The current  
supergraph structure representation is reported in table 3.3.  
Table 3.3: Supergraph representation at timestep 2  
Super Graph Periodic Patterns  
V V Timeset Descriptors Keys Patterns  
1 2  
B A 1,2 (1,1,2)  
A 1,2 (1,1,2)  
C  
B 1,2 (1,1,2)  
B 1  
D  
C 1,2 (1,1,2)  
GraphG isreadandupdatesthesupergraphattimestep3. Italsoupdatestime  
3  
set and descriptors like as same way those are described at time step 2. Interactions  
(A−B),(A−C) and (B−C) appear at graph G . The existing descriptors (1,1,2)  
3

3.3 DESCRIPTION OF THE ALGORITHM 53  
expected time is current time. So it creates duplicate descriptor and updates it  
as (1,1,3). It also creates (1,2,2) and (2,1,2) descriptors for future periodicity.  
All descriptors, which expected time is equal to current time or less than current  
time flashed out from supergraph with interactions duplicate entity. If it satisfies  
minimum support threshold value then inserts into periodic patterns hash table  
where hash key is defined as starting position, period and support. That’s why  
descriptors(1,1,2)isflashedoutandcorrespondinginteractionsisstoredinperiodic  
hash table, which hash key is 1−1−2. Table 3.4 shows the current supergraph  
presentation and periodic pattern hash table.  
Table 3.4: Supergraph representation at timestep 3  
Super Graph Periodic Patterns  
V V Timeset Descriptors Keys Patterns  
1 2  
B A 1,2,3 (1,1,3)(1,2,2),(2,1,2) 1-1-2 (A-B),(A-C),(B-C)(C-D)  
A 1,2,3 (1,1,3)(1,2,2),(2,1,2)  
C  
B 1,2,3 (1,1,2)(1,2,3),(2,1,2)  
B 1  
D  
C 1,2  
At time step 4, graph G is read and updates the supergraph. All interactions  
4  
in supergraph are active that why it updates or creates new descriptors based on  
timeset times. Interactions (A−B),(A−C) and (B −C) update and create new  
descriptors for future time. On the other hand (B−D) and (C−D) interactions are  
empty descriptors those way they create new descriptors. The descriptors, which  
expected time is equal or less then current time should be flashed out and create  
periodic patterns. Because of this flashed out descriptors (1,1,3) and (2,1,2) create  
twoperiodicpatterns. Thestructureofsupergraphandperiodicpatternshashtable  
after time 4 are shown in table 3.5.  
Graph G is read and updates the supergraph at time step 5. Interactions  
5  
(A−B),(A−C) and (B −C) are active that why they update descriptors based

3.3 DESCRIPTION OF THE ALGORITHM 54  
Table 3.5: Supergraph representation at timestep 4  
Super Graph Periodic Patterns  
V V Timeset Descriptors Keys Patterns  
1 2  
B A 1,2,3,4 (1,1,4)(1,2,2),(2,1,3), 1-1-2 (A-B),(A-C),(B-C)(C-D)  
(1,3,2)(3,2,2),(3,1,2) 1-1-3 (A-B),(A-C),(B-C)  
A 1,2,3,4 (1,1,4)(1,2,2),(2,1,3), 2-1-2 (A-B),(A-C),(B-C)  
(1,3,2)(3,2,2),(3,1,2)  
C  
B 1,2,3,4 (1,1,4)(1,2,2),(2,1,3),  
(1,3,2)(3,2,2),(3,1,2)  
B 1,4 (1,3,2)  
D  
C 1,2,4 (1,3,2),(2,2,2)  
on existing descriptors and also create new descriptors, which would be periodic in  
future and flashed out those descriptors which are not live. The table 3.6 shows the  
current supergraph and periodic patterns.  
However, graphG isthelastgraphinourdynamicnetwork. Supergraphshould  
5  
be traversed and flushed out interactions and store those periodic patterns, which  
satisfy minimum support. Then finally, we find all periodic patterns like table 3.7.  
The periodic patterns in periodic hash table all patterns are neither closed nor  
parsimonious. Mining parsimonious periodic patterns, we should check two kinds  
of property. First, one is closed pattern mining and second one is parsimonious  
pattern mining that means others do not subsume its periodicity. If any pattern  
satisfies two properties, we said that it is parsimonious periodic pattern. At first,  
we prune non-closed patterns. After removing non-closed patterns, those patterns  
may be parsimonious or not. There are a number of patterns that are subsumed  
by other patterns mention in table 3.8. After pruning non-parsimonious patterns  
we find parsimonious patterns these are the output of our proposed SPBMiner  
algorithm shows in table 3.9.

3.3 DESCRIPTION OF THE ALGORITHM 55  
Table 3.6: Supergraph representation at timestep 5  
Super Graph Periodic Patterns  
V V Timeset Descriptors Keys Patterns  
1 2  
B A 1,2,3,4,5 (1,1,5)(1,2,3),(2,1,4), 1-1-2 (A-B),(A-C),(B-C)(C-D)  
(1,3,2)(2,2,2),(3,1,2), 1-1-3 (A-B),(A-C),(B-C)  
(1,4,2)(2,3,2),(3,2,2),(4,1,2) 1-1-3 (A-B),(A-C),(B-C)  
A 1,2,3,4,5 (1,1,5)(1,2,3),(2,1,4), 2-1-2 (A-B),(A-C),(B-C)  
(1,3,2)(2,2,2),(3,1,2), 1-1-4 (A-B),(A-C),(B-C)  
(1,4,2)(2,3,2),(3,2,2),(4,1,2) 1-2-2 (A-B),(A-C),(B-C)  
C  
B 1,2,3,4 (1,1,5)(1,2,3),(2,1,4), 2-1-3 (A-B),(A-C),(B-C)  
(1,3,2)(2,2,2),(3,1,2), 3-1-2 (A-B),(A-C),(B-C)  
(1,4,2)(2,3,2),(3,2,2),(4,1,2)  
B 1,4 (1,3,2)  
D  
C 1,2,4 (1,3,2),(2,2,2)  
Table 3.7: Periodic Patterns  
Keys Patterns Keys Patterns  
1-1-2 (A-B),(A-C),(B-C),(C,D) 2-1-4 (A-B),(A-C),(B-C)  
1-1-3 (A-B),(A-C),(B-C) 1-3-2 (A-B),(A-C),(B-C),(B-D),(C-D)  
2-1-2 (A-B),(A-C),(B-C) 2-2-2 (A-B),(A-C),(B-C),(C-D)  
1-1-4 (A-B),(A-C),(B-C) 3-1-3 (A-B),(A-C),(B-C)  
1-2-2 (A-B),(A-C),(B-C) 1-4-2 (A-B),(A-C),(B-C)  
2-1-3 (A-B),(A-C),(B-C) 2-3-2 (A-B),(A-C),(B-C)  
3-1-2 (A-B),(A-C),(B-C) 3-2-2 (A-B),(A-C),(B-C)  
1-1-5 (A-B),(A-C),(B-C) 4-1-2 (A-B),(A-C),(B-C)  
1-2-3 (A-B),(A-C),(B-C)

3.3 DESCRIPTION OF THE ALGORITHM 56  
Table 3.8: Pruning non closed and non parsimonious periodic patterns  
Patterns Keys Closed Reason Parsimonious Reason  
1-1-2 Yes Yes  
(A-B),(A-C),(B-C),(C,D)  
2-2-2 Yes Yes  
1-1-3 No 1-1-4  
2-1-2 No 2-1-3  
1-1-4 No 1-1-5  
1-2-2 No 1-2-3  
2-1-3 No 2-1-4  
3-1-2 No 3-1-3  
1-1-5 Yes Yes  
(A-B),(A-C),(B-C)  
1-2-3 Yes No 1-1-5  
2-1-4 Yes No 1-1-5  
3-1-3 Yes No 1-1-5  
1-4-2 Yes No 1-1-5  
2-3-2 Yes No 1-1-5  
3-2-2 Yes No 1-1-5  
4-1-2 Yes No 1-1-5  
(A-B),(A-C),(B-C),(B-D),(C-D) 1-3-2 Yes Yes  
Table 3.9: Parsimonious Periodic Patterns  
Patterns Keys Start Period Phase Support  
1-1-2 1 1 0 2  
(A-B),(A-C),(B-C),(C,D)  
2-2-2 2 2 0 2  
(A-B),(A-C),(B-C) 1-1-5 1 1 0 5  
(A-B),(A-C),(B-C),(B-D),(C-D) 1-3-2 1 3 1 2

3.4 TIME AND SPACE COMPLEXITY 57  
3.3.4 Description of the implementation  
ThealgorithmisimplementedinC++. Inthenextsection,mostimportantelements  
of the program are presented.  
Graph: The graph is represented by integer vector where interactions present by  
integer value and it is unique.  
Super Graph: Super graph like as graph. Each interaction has descriptor set an  
timeset which indicate entity periodicity information and active time respectively.  
Timeset (TS): It is integer array that size is maximum period and value is time.  
Descriptor Set: It is the descriptor object vector in entity. It describes the entity  
periodicity information like as period, phase, support, start and last occurred times.  
Periodic Hash Table: The periodic has table is implemented by google dense  
hash map [41]. The key of hash table is the combination of starting position, period  
and support (key = start-period-support).  
Maximalcommonsubgraph: ForeverytimestepG wehavetocalculatetheMCS  
t  
between supergraph SG and G . For MCS computation common interactions  
t−1 t  
are checked. Parsimonious Periodic Patterns: In this step pattern is the key and  
corresponding descriptors are value of the hash map.  
3.4 Time and Space Complexity  
3.4.1 Time Complexity  
Supergraph maintaining is the main part of our proposed algorithm. Suppose V  
is the vertex number of our supergraph. Then total entities are V ∗ (V − 1)/2 if  
supergraphisstronglyconnected. However,worstcasesupergraphentitiesisO(V2).  
Lemma 5 proved that every entity has maximum (P )2 descriptors. Updating  
max  
these descriptors, need (P )2 time and updating timeset (TS) needs P time.  
max max  
So each timestep requires O(V2)((P )2 + P ) = O(V2(P )2) times. This  
max max max  
yieldsthetotaltimecomplexityisO((V2)T(P )2)whenP isspecified. IfP  
max max max  
is not specified than the time complexity is O(V2T(T/σ)2) . In practice, however,

3.5 SUMMARY 58  
the vertex number is smaller and descriptor set is sorted. Then the computational  
cost is usually smaller then the worst case, as we will demonstrate.  
3.4.2 Space Complexity  
For every timestep t, supergraph has maximum V2 entity and each entities con-  
tains TS and descriptor sets. According to lemma 1 and 5, the size of time set and  
descriptor set is P and (P )2 respectively. Therefore the total space complex-  
max max  
ity is O((V2)(P )2) that is total time independent. In the worst case P is  
max max  
unrestricted P = T/σ, so the space complexity is O(V2(T/σ)2).  
max  
3.5 Summary  
Inthischapter, wehaveintroducedanattractiveperiodicpatternsminingtechnique  
SPBMiner, which mine periodic patterns as well as closed and parsimonious pe-  
riodic patterns from dynamic networks over the user given minimum support. The  
efficiency of the finding such patterns is shown in the complexity analysis section  
which is better than the existing PSEMiner and ListMiner methods.

Chapter 4  
Experimental Evaluation  
In this chapter, the performances and the characteristics of SPBMienr are com-  
pared with two existing algorithms PSEMiner and ListMiner. I used three real  
worlddynamicsocialnetworksintheexperimentalresultanalysis. Artificialdatasets  
are also created to better understanding of every algorithm. I clearly mention the  
differences, weak and strong points in my algorithm.  
For this comparison, these three algorithms are implemented in C ++. I im-  
plemented SPBMienr and ListMiner and used PSEMiner source code available  
in [42]. The experiments are run on 3.3 GHz Intel Core i5 with 4GB RAM, in  
windows 7. These algorithms use google dense/sparse hash library [41]. In all ex-  
periments, the reported computation time is the sum of the user and CPU time.  
Memory usage is the maximum resident set size reported by C ++ memory uses  
function.  
4.1 Datasets Description  
Three real dynamic social networks datasets and six artificial datasets are used to  
evaluate our SPBMiner algorithm.  
4.1.1 Real Datasets  
Dynamic networks are collected from different sources and covering a range of in-  
teraction dynamics. These networks are described below.  
59

4.1 DATASETS DESCRIPTION 60  
Enron e-mails: The Enron e-mail corpus is a publicly accessible record of e-mails  
commutation among employees of the Enron Corporation [43]. Senders and list of  
recipients timesteps were extracted from message headers for each e-mail on file. It  
has been considered a day as the quantization timestep that means if at least one  
e-mail was sent between two individuals employees on a particular day we can said  
that interaction is presented.  
Facebook Wall Post: This facebook dataset [44] gathered wall post information  
about 90,269 users September 26, 2006 to January 22, 2009. In total, there are  
observed 838,092 wall posts, for an average of 13.9 wall posts per user. This covers  
communication between 188,892 distinct pairs of users. One-day quantization is  
measured as unique timestep.  
Reality Mining: Cell phones with nearness tracking technology were distributed  
to 100 students at the Massachusetts Institute of Technology over the course of an  
academic year [33]. The timestep quantization is chosen as 1 day.  
4.1.2 Artificial Data  
Artificial datasets are used to better understand the performances of these algo-  
rithms. The intention of these set of experiments is to illuminate why and when  
our algorithm outperforms the others. The following datasets are created using  
GraphGen a synthetic graph generator [45].  
Experiment1.1: Thisdatasetcreatesrandomgraphwith50differentinteractions  
among 10000 populations at each timestep and the total timestep is 1000. The  
interactions represent a sequence of 50 different random numbers among 1 to 10000.  
Experiment 1.2: This dataset creates random graph with 100 different inter-  
actions among 10000 populations at each timestep and the total timestep is 1000.

4.2 EXPERIMENTAL TIME ANALYSIS 61  
The interactions represent a sequence of 100 different random numbers among 1 to  
10000.  
Experiment 1.3: This dataset creates random graph with 200 different inter-  
actions among 10000 populations at each timestep and the total timestep is 1000.  
The interactions represent a sequence of 200 different random numbers among 1 to  
10000.  
Experiment 1.4: This dataset creates random graph with 400 different interac-  
tions among 10000 populations at each timestep and the total timestep is 1000. The  
interactions represent a sequence of 400 different random numbers between 1 and  
10000.  
Experiment 1.5: This dataset generates graph with 800 various interactions  
among 10000 populations at each timestep and the total timestep is 1000. The  
interactions represent a sequence of 800 different random numbers 1 to 10000.  
Experiment 1.6: This dataset builds random graph with 1000 different interac-  
tions among 10000 populations at each timestep and the total timestep is 1000. The  
interactions represent a sequence of 1000 different random numbers 1 to 10000. The  
table ?? shows the datasets in details.  
The variation of artificial data parameters for the artificial networks expresses  
diversity characteristic of networks including low density (Experiment 1.1 and 1.2),  
medium density (Experiment 1.3 and 1.4) and high density (Experiment 1.5 and  
1.6). The experiments analyses will show the density of networks is a parameter  
that has a significant influence on our SPBMiner algorithm.  
4.2 Experimental Time Analysis  
This section shows the execution times comparison analysis among our proposed  
method and other two existing works. The execution times of three techniques are

4.2 EXPERIMENTAL TIME ANALYSIS 62  
Table 4.1: Parameters of various datasets  
Dataset Timestep Vertexes Edges Avg. Active Edge Density P  
max  
Enron 2588 82614 330452 0.0015 40  
Reality mining 544 100 4900 0.025 40  
Facebook 1563 46951 193337 0.002 40  
Experiment 1.1 1000 150 10000 0.005 40  
Experiment 1.2 1000 150 10000 0.01 40  
Experiment 1.3 1000 150 10000 0.02 40  
Experiment 1.4 1000 150 10000 0.04 40  
Experiment 1.5 1000 150 10000 0.08 40  
Experiment 1.6 1000 150 10000 0.10 40  
performed where minimum support σ = 3 and mining patterns are parsimonious.  
The reality mining dataset is high density network. The number of vertexes is  
low (100) and the number of timesteps is medium (544). The SPBMiner algorithm  
generates periodic descriptors for each entity (interactions between two vertexes)  
that result low number of vertexes interactions mining needs short time and less  
memory. Similarly, experiment 1.5 creates a sequence of 400 casual numbers among  
1 to 10000 and experiment 1.5 creates 800 numbers among 1 to 10000. These  
networks are also dense. Therefore, these dense networks are changed by time to  
time defined by different graph structures. The probability of common subgraph  
computation between two graphs are high that reason ListMiner and PSEMiner  
needs more MCS computation.  
The figure ?? shows that SPBMiner is faster than two existing works in reality  
mining datasets. It’s vertex number is low and it is dense that way it creates  
more common sub-behaviors’ set in ListMiner and PSEMiner. On the other  
hand, Facebook dataset is medium density and its time is slightly high. Although  
in this case my algorithm performs better than PSEMiner but it is not good  
as ListMiner because it mine common patterns between 387 entities where our

4.2 EXPERIMENTAL TIME ANALYSIS 63  
proposed process traverse large than 387 entities at each timestamps that why need  
little bit large computation time. In Enron dataset due to sparse data my propose  
method SPBMiner shows low performance than PSEMiner and ListMiner. The  
variation of network density is the main cause that explained below using artificial  
datasets.  
1000  
800  
600  
400  
200  
0  
Enron Reality Facebook  
)s(  
emiT  
noitucexE  
SPBMiner ListMiner PSEMiner  
Figure 4.1: Execution times comparison on real datasets.  
In the high-density context, PSEMiner is much slower than SPBMiner. Be-  
cause PSEMiner builds periodic pattern tree using graphs and current graph tra-  
verse the entire tree node and find common subgraph. Since the dense graph, it  
need more time to compute MCS between graph and tree nodes. The execution  
times comparison among three methods in the figure 4.2 shows that at experiment  
1.1, PSEMiner is faster than ListMiner and SPBMiner. The SPBMiner exe-  
cution time analyses have been shown in experiment 1.3 ListMiner is minimum 1.5  
times slower and PSEMiner is 2 times slower than SPBMiner. Another density  
dataset experiment 1.4 shows our proposed SPBMiner is around 1.5 times faster  
than ListMimer and around four times faster than PSEMiner. In the medium  
density context, analysis of experiment 1.3 reports that all these three methods ex-  
ecution times are almost same. Although, the theoretical time bound complexity of  
SPBMiner is better than ListMiner and PSEMiner, experiment 1.1 and 1.2 reports  
PSEMiner is faster than ListMiner and SPBMiner because of sparse dataset.  
In this case, MCS computation is less and faster. According to above analyses, we

4.3 EXPERIMENTAL SPACE ANALYSIS 64  
can say that SPBMiner is time efficient when network density is medium or high.  
3000  
2500  
2000  
1500  
1000  
500  
0  
Ex-1.1 Ex-1.2 Ex-1.3 Ex-1.4 Ex-1.4 Ex-1.5  
)s(  
emiT  
noitucexE  
SPBMiner ListMiner PSEMiner  
Figure 4.2: Execution times comparison on artificial datasets.  
4.3 Experimental Space Analysis  
TheanalysisofthememoryrequirementofSPBMiner,ListMinerandPSEMiner  
arepresentedinthissection. Figure4.3showstheresultscomparisonofthememory  
usage of these algorithms with σ =3.  
SPBMiner use less memory in facebook dataset because it density is not so  
high and it create less number of periodic behaviors than Enron datasets and pe-  
riodic behaviors length is small that way it needs less memory. In Reality dataset  
requires large memory because for each entity descriptors set generate P2 descrip-  
max  
tors maximum that why it needs large memory. Enron dataset has large number of  
entities that way it requires little bit large memory. In conclusion we said that our  
SPBMiner methods is memory efficient in medium density networks.  
SPBMiner uses less memory than ListMiner and PSEMiner in experiment  
1.1 , 1.2 and 1.3 in figure 4.4. The others dataset SPBMiner is slightly higher than  
ListMiner.  
This behavior can be justified by theoretical analysis of the space complexity.  
The space complexity of PSEMiner is ((V + E)N + P2 + G) where N is the  
max  
numberofnodesinthetree, Gisthenumberofdescriptor, andV,Earethenumber  
of vertexes and edges, respectively. The space complexity of ListMiner is always

4.4 ANALYSIS OF DENSE NETWORKS EFFECTS 65  
20  
16  
12  
8  
4  
0  
Enorn Reality Facebook  
)BM(egasU  
yromeM  
SPBMiner ListMiner PSEMiner  
Figure 4.3: Memory usage on real datasets.  
((V +E)TP ). Since the most part of the memory is used to store graphs, the  
max  
dominant term of the space complexity expression in PSEMiner is (V+E)N. The  
space complexity of SPBMiner is ((V +E)P2 . In dataset experiment 1.1, 1.2,  
max  
1.3, 1.4 numbers of interactions (vertexes and edges) is 50, 100,200 and 400. P  
max  
is 40 so memory complexity of our process is less than existing methods ListMiner  
and PSEMiner. In experiment 1.5, entities are 800 and P = 40 that way  
max  
ListMiner requires 800\*1000\*40 that is less then requires theoretical memory in  
SPBMiner, which is 10000∗(40)2. Thus ListMiner is memory efficient. If time  
stepwouldbetoolargethatcaseSPBMiner wouldbememoryefficient. Therefore,  
whenthenumberofentitiesdensityismoreandtimestepsarelowordensityishigh  
and total timesteps are also high than the space complexity of SPBMiner is better  
then PSEMiner and ListMiner.  
4.4 Analysis of Dense Networks Effects  
In previous section, we report the SPBMiner algorithm efficiency depend on net-  
work density. In section we will explain how density is affected our approach. An-  
alyzing density effects, from figure 4.2 experiment 1.3 network density is 2% that  
time all these algorithms execution time is almost same. But experiment 1.4 net-  
work density is 4%, total number of interactions 10000 and 400 is active at each  
timesteps. In this time our SPBMiner is 23% faster than ListMiner and 49%

4.5 ANALYSIS OF PARSIMONIOUS PERIODIC PATTERNS 66  
16  
14  
12  
10  
8  
6  
4  
2  
0  
Ex-1.1 Ex-1.2 Ex-1.3 Ex-1.4 Ex-1.4 Ex-1.5  
)BM(  
yromeM  
egasU  
SPBMiner ListMiner PSEMiner  
Figure 4.4: Memory usage on real datasets.  
faster than PSEMiner. Experiment 1.5 is dense data that case SPBMiner is 30%  
faster than ListMiner and 75% faster than PSEMiner. All these experiment net-  
works continue 1000 timesteps and considered minimum threshold sigma= 3 and  
P = 40. Finally, we say that our approach is exceptionally faster than existing  
max  
works in medium and dense datasets.  
4.5 Analysis of Parsimonious Periodic Patterns  
In this section, the analysis of parsimonious periodic patterns mined by the  
SPBMimer algorithm is reported. The analysis is performed in two tracks. First,  
one is number of periodic patterns vs support and second one is number of periodic  
patterns vs period.  
Experiments analyses showed that the highest number of parsimonious periodic  
patterns and the highest values of support occur on high-density networks.  
4.5.1 Parsimonious Periodic Pattern Division and Support Values  
Figure 4.5 shows the number of parsimonious periodic patterns (y axis) for each  
real dataset based on support value (x-axis). It has been shown that dense dataset  
reality mining produce a large number of parsimonious periodic patterns more than  
10000thoughitstotaltimestepisonly544. Withtheincreasesofsupportitspatterns

4.5 ANALYSIS OF PARSIMONIOUS PERIODIC PATTERNS 67  
numberdecreaserapidly. Facebookdatasetproducelargenumberofdataatsupport  
3,4,5thatrepresentmostofthepatternsarewithinsupport3to5andthenitreduce  
very rapidly and at the end support 19 it produces only 7 patterns. On the other  
hand, Enron dataset produce large dataset also and reduces the periodic pattern in  
a sequentially after support 6. All these experiment run based on period P that  
max  
has been mentioned in table 4.1.  
100000  
10000  
1000  
100  
10  
1  
3 4 5 6 7 8 9 10111213141516171819  
snrettaP  
fo  
rebmuN  
Facebook Enron Reality Mining  
minimum support (σ)  
Figure 4.5: Parsimonious periodic patterns vs minimum support  
4.5.2 Parsimonious Periodic Pattern Division and Period Values  
Figure4.6showsthenumberofparsimoniousperiodicpatterns(yaxis)foreacharti-  
ficial dataset based on different period value (x-axis). It has been shown that Enron  
dataset produces a large number of parsimonious periodic patterns and generating  
patterns are decreases with the increases of period. It clearly shows that at period  
7, 14, 21, 28and35produceslargenumberofperiodicpatternsthatindicateateach  
week they have a particular day that day their email communication be high. It  
may be their weekly report of or performance analysis email. On the other facebook  
dataset number of patterns are not so very period to period though there are some  
differences between them . All these patterns are mined based on minimum support  
σ = 3.

4.5 ANALYSIS OF PARSIMONIOUS PERIODIC PATTERNS 68  
1600  
800  
400  
200  
1 3 5 7 9 111315171921232527293133353739  
snrettaP  
fo  
rebmuN  
Facebook Enron Reality Mining  
Period  
Figure 4.6: Parsimonious periodic patterns vs period value  
4.5.3 Knowledge Discovery  
In the figure 4.7 (a) and (b) have been shown that facebook dataset contains pair  
to pair communication that are weekly and continue up to 7 weeks and other one  
is daily and 2 days interval wall post communication continue up to 10 times. Fig-  
ure 4.7(c) represents interesting communications where one email users send emails  
continuously 84 days. From those kinds of relationship, we said that they are very  
strongly connected in the Enron Corporation. In figure 4.7(d) shows one email users  
send email each week interval and it continue around 4 month. From this kinds  
or relationship, I suppose that they are works one projects for 4 month and their  
communication is strong for these period.  
4.5.4 Scalability Analysis  
Execution time depends on minimum support and maximum period value. With  
the increases of maximum period, the execution time increase. On the other hand,  
with increase of minimum support the execution time reduces. Figure 4.8(a) shows  
that the minimum support scalability where P = 50. And figure 4.8(b) shows  
max  
that the scalability analysis of our proposed SPBMiner as well as ListMiner and  
PSEMiner. OurproposeSPBMinerismorescalablethantwoexistingworksbecause  
when P increase it create a large number of nodes and it more comparison that  
max  
result it needs large time. In this experiment I use artificial data experiment 1.4 and

4.6 SUMMARY 69  
1417 12012 al.frienwire.com  
juan.padronr  
kevin.dine  
1418 24113 ryab.williams seung-taek.oh eric.saibi  
(a) Facebook: period 7 support 7 (c) Enron: period 1 support 84  
10919 Kenny.soignet  
chris.foster  
matthew.lenhart  
jeffrey.shankma  
liz.toylor n  
kimperly.hillis keith.holst  
30404 30405  
marc.horowitz scott.neal  
(b) Facebook: period 1 & 2 support 10 (d) Enron: period 7 support 31  
Figure 4.7: Finding inherent patterns from facebook and Enron dataset  
minimum support = 3.  
1200  
1000  
800  
600  
400  
200  
0  
30 40 50 60 70  
)s(  
emiT  
noitucexE  
1000  
900 SPBMiner ListMiner PSEMiner  
800  
700  
600  
500  
400  
300  
200  
100  
0  
σ = 3 σ = 5 σ = 7 σ = 9 σ = 11  
(a) Execution time vs minimum support (b) Execution time vs maximum period  
)s(  
emiT  
noitucexE  
SPBMiner ListMiner PSEMiner  
Figure 4.8: Scalability test for experiment 1.4 dataset  
4.6 Summary  
In this chapter, we have shown the effectiveness and efficiency of our proposed  
SPBMiner. Experimental space and time analysis show that our method are sig-  
nificantly efficient for medium and dense dynamic networks and outperform the  
existing algorithm in both execution time and memory usage. We also mine some  
interesting periodic patterns form real datasets that represent very informative in-  
formation.

Chapter 5  
Conclusions and Future Work  
In this chapter we summarize the research works presented in this dissertation and  
make final concluding remarks with few directions for future works.  
5.1 Summary of the Dissertation  
The main contribution of this dissertation is the design and development of  
SPBMiner, an efficient algorithm for solving the periodic behaviors mining prob-  
lem in dynamic networks. The time complexity of SPBMiner is ((V +E)TP2 ),  
max  
where V is the size of the population of networks, E is the set of interactions among  
populations, T is the number of observations (timesteps) and P is the maxi-  
max  
mum period. Our proposed SPBMiner improves the worst case time complexity  
of ListMiner by /sigma and PSEMiner by (/sigma2ln(T)) times .  
The theoretical analyses of our proposed method support experimental analyses.  
The performances and the behaviors of SPBMiner, ListMiner and PSEMiner  
were compared using two real-world dynamic social networks and several 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. Contrarily in a low-density context PSEMiner  
and ListMiner is faster than SPBMiner. However, in the worst case dataset  
analysis confirms that SPBMiner is actually more efficient than ListMiner and  
PSEMiner. Moreover, in real scenarios, where the maximum period P is re-  
max  
stricted, SPBMiner took few seconds to execute and uses less than 15 MB of  
memory.  
70

5.2 FUTURE RESEARCH DIRECTIONS 71  
Finally, qualitative analyses of parsimonious patterns show the periodicities in-  
teractionsamongstudentsandcorporateexecutives. Thedailyandweeklybehaviors  
of interactions among people are shown in the experiments. Additionally, the mined  
parsimonious patterns revealed the hidden characteristics of the interactions among  
the population. In particular, the patterns characteristic of college students are  
shown peer-to-peer and corporate relationship are mostly hierarchical.  
5.2 Future Research Directions  
With the capabilities of the proposed method, we plan to investigate and explore  
the following related problems, extensions:  
• We would like to investigate whether other efficient algorithms would lead to  
better discovery of periodic patterns because our model does not maintain effective-  
ness and efficiency in offline networks when period is unrestricted.  
• Designing sequential periodic behaviors mining technique is significantly inter-  
esting in data mining and knowledge discovery. We would like to develop efficient  
method for mining sequential periodic patterns in dynamic networks.

Bibliography  
[1] M. Lahiri and T. Y. Berger-Wolf, “Periodic subgraph mining in dynamic net-  
works,” Knowledge and information systems, vol. 24, no. 3, pp. 467–497, 2010.  
[2] R. Agrawal, R. Srikant et al., “Fast algorithms for mining association rules,”  
in Proc. 20th Int. Conf. Very Large Data Bases, VLDB, vol. 1215, 1994, pp.  
487–499.  
[3] D. J. Cook and L. B. Holder, “Substructure discovery using minimum descrip-  
tion length and background knowledge,” arXiv preprint cs/9402102, 1994.  
[4] K.Yoshida, H.Motoda, andN.Indurkhya, “Graph-basedinductionasaunified  
learning framework,” Applied Intelligence, vol. 4, no. 3, pp. 297–316, 1994.  
[5] L. Dehaspe and H. Toivonen, “Discovery of frequent datalog patterns,” Data  
Mining and Knowledge Discovery, vol. 3, no. 1, pp. 7–36, 1999.  
[6] S.WassermanandK.Faust,Social network analysis: Methods and applications.  
Cambridge university press, 1994, vol. 8.  
[7] K.-Y. Huang and C.-H. Chang, “Mining periodic patterns in sequence data,”  
Data Warehousing and Knowledge Discovery, pp. 401–410, 2004.  
[8] M. Zhang, B. Kao, D. W. Cheung, and K. Y. Yip, “Mining periodic patterns  
with gap requirement from sequences,” ACM Transactions on Knowledge Dis-  
covery from Data (TKDD), vol. 1, no. 2, p. 7, 2007.  
72

BIBLIOGRAPHY 73  
[9] D. Lo, S.-C. Khoo, and C. Liu, “Efficient mining of iterative patterns for soft-  
ware specification discovery,” in Proceedings of the 13th ACM SIGKDD inter-  
national conference on Knowledge discovery and data mining. ACM, 2007, pp.  
460–469.  
[10] M. Lahiri and T. Y. Berger-Wolf, “Mining periodic behavior in dynamic so-  
cial networks,” in Data Mining, 2008. ICDM’08. Eighth IEEE International  
Conference on. IEEE, 2008, pp. 373–382.  
[11] B. Ozden, S. Ramaswamy, and A. Silberschatz, “Cyclic association rules,”  
in Data Engineering, 1998. Proceedings., 14th International Conference on.  
IEEE, 1998, pp. 412–421.  
[12] C. Bettini, X. S. Wang, S. Jajodia, and J.-L. Lin, “Discovering frequent event  
patterns with multiple granularities in time sequences,” Knowledge and Data  
Engineering, IEEE Transactions on, vol. 10, no. 2, pp. 222–237, 1998.  
[13] J. Han, G. Dong, and Y. Yin, “Efficient mining of partial periodic patterns in  
time series database,” in Data Engineering, 1999. Proceedings., 15th Interna-  
tional Conference on. IEEE, 1999, pp. 106–115.  
[14] J. Yang, W. Wang, and P. S. Yu, “Mining asynchronous periodic patterns in  
time series data,” Knowledge and Data Engineering, IEEE Transactions on,  
vol. 15, no. 3, pp. 613–628, 2003.  
[15] K.-Y. Huang and C.-H. Chang, “Asynchronous periodic patterns mining in  
temporal databases,” in Proc. of the IASTED International Conference on  
Databases and Applications (DBA), 2004, pp. 43–48.  
[16] J. Yang, W. Wang, and P. S. Yu, “Mining surprising periodic patterns,” Data  
Mining and Knowledge Discovery, vol. 9, no. 2, pp. 189–216, 2004.  
[17] A. Chapanond, M. S. Krishnamoorthy, and B. Yener, “Graph theoretic and  
spectral analysis of enron email data,” Computational & Mathematical Organi-  
zation Theory, vol. 11, no. 3, pp. 265–281, 2005.

BIBLIOGRAPHY 74  
[18] J. Diesner and K. M. Carley, “Exploration of communication networks from  
the enron email corpus,” in SIAM International Conference on Data Mining:  
Workshop on Link Analysis, Counterterrorism and Security, Newport Beach,  
CA. Citeseer, 2005.  
[19] 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,¡ i¿ equus burchellii¡/i¿,” Animal  
Behaviour, vol. 73, no. 5, pp. 825–831, 2007.  
[20] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On power-law relationships of  
the internet topology,” in ACM SIGCOMM Computer Communication Review,  
vol. 29, no. 4. ACM, 1999, pp. 251–262.  
[21] J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent pattern mining: current  
status and future directions,” Data Mining and Knowledge Discovery, vol. 15,  
no. 1, pp. 55–86, 2007.  
[22] A. Apostolico, M. Barbares, and C. Pizzi, “Speedup for a periodic subgraph  
miner,” Information Processing Letters, vol. 111, no. 11, pp. 521–523, 2011.  
[23] M.Lahiri, “Measuringandminingdynamicnetworks,” Ph.D.dissertation, Uni-  
versity of Illinois, 2011.  
[24] R. Agrawal, T. Imielin´ski, and A. Swami, “Mining association rules between  
sets of items in large databases,” in ACM SIGMOD Record, vol. 22, no. 2.  
ACM, 1993, pp. 207–216.  
[25] S. Ma and J. L. Hellerstein, “Mining partially periodic event patterns with  
unknown periods,” in Data Engineering, 2001. Proceedings. 17th International  
Conference on. IEEE, 2001, pp. 205–214.  
[26] Z. Yin, L. Cao, J. Han, C. Zhai, and T. Huang, “Lpta: A probabilistic model  
for latent periodic topic analysis,” in Data Mining (ICDM), 2011 IEEE 11th  
International Conference on. IEEE, 2011, pp. 904–913.

BIBLIOGRAPHY 75  
[27] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate gen-  
eration,” in ACM SIGMOD Record, vol. 29, no. 2. ACM, 2000, pp. 1–12.  
[28] X.Zhi-jun,C.Hong,andC.Li,“Anefficientalgorithmforfrequentitemsetmin-  
ing on data streams,” in Advances in Data Mining. Applications in Medicine,  
Web Mining, Marketing, Image and Signal Mining. Springer, 2006, pp. 474–  
491.  
[29] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, “Cp-tree: a tree  
structure for single-pass frequent pattern mining,” in Advances in Knowledge  
Discovery and Data Mining. Springer, 2008, pp. 1022–1027.  
[30] ——, “Discovering periodic-frequent patterns in transactional databases,” in  
Advances in Knowledge Discovery and Data Mining. Springer, 2009, pp. 242–  
253.  
[31] N. Sumathi and S. Sathiyabama, “Periodic-pattern tree miner: An efficient  
algorithm to mine the periodic patterns from the spatio-temporal database,”  
European Journal of Scientific Research, vol. 81, no. 2, pp. 246–262, 2012.  
[32] 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,” in ACM Sigplan Notices, vol. 37, no. 10. ACM,  
2002, pp. 96–107.  
[33] N. Eagle and A. Pentland, “Reality mining: sensing complex social systems,”  
Personal and ubiquitous computing, vol. 10, no. 4, pp. 255–268, 2006.  
[34] L. Yan and J. Wang, “Extracting regular behaviors from social media net-  
works,” in Multimedia Information Networking and Security (MINES), 2011  
Third International Conference on. IEEE, 2011, pp. 613–617.  
[35] P. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl, “On graphs with unique  
nodelabels,”GraphBasedRepresentationsinPatternRecognition,pp.409–437,  
2003.

BIBLIOGRAPHY 76  
[36] M. Lahiri and T. Y. Berger-Wolf, “Structure prediction in temporal networks  
using frequent subgraphs,” in Computational Intelligence and Data Mining,  
2007. CIDM 2007. IEEE Symposium on. IEEE, 2007, pp. 35–42.  
[37] G. Yang, “The complexity of mining maximal frequent itemsets and maximal  
frequent patterns,” in Proceedings of the tenth ACM SIGKDD international  
conference on Knowledge discovery and data mining. ACM, 2004, pp. 344–  
353.  
[38] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino, “On the complexity of  
generating maximal frequent and minimal infrequent sets,” in STACS 2002.  
Springer, 2002, pp. 133–141.  
[39] M. G. Elfeky, W. G. Aref, and A. K. Elmagarmid, “Periodicity detection in  
time series databases,” Knowledge and Data Engineering, IEEE Transactions  
on, vol. 17, no. 7, pp. 875–887, 2005.  
[40] M. Barbares, “Periodic subgraph mining in dynamic networks,” 2010.  
[41] Google hash library: http://code.google.com/p/google-sparsehash/, version  
2.2.  
[42] http://compbio.cs.uic.edu/software/periodic/.  
[43] Enron dataset:http://www.cs.cmu.edu/ enron/.  
[44] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, “On the evolution  
of user interaction in facebook,” in Proceedings of the 2nd ACM workshop on  
Online social networks. ACM, 2009, pp. 37–42.  
[45] http://www.cse.ust.hk/graphgen/.

Appendix A  
List of Publications  
International Journal Papers  
1. Sajal Halder, Yongkoo Han, A. M. Jehad Sarkar and Young-Koo Lee. An  
Entertainment Recommendation System using the Dynamics of User Behavior  
over Time. Decision in process in the Journal of Systems and Software.  
2. Md. Rezaul Karim, Sajal Halder , Byeong-Soo Jeong, and Ho-Jin Choi.  
Efficient Mining Frequently Correlated, Associated-correlated and Independent  
PatternsSynchronouslybyRemovingNullTransactions. HumanCentricTech-  
nology and Service in Smart Space, pages 93-103, 2012.  
3. Sajal Halder, A. M. Jehad Sarkar and Young-Koo Lee. A synthetic  
trajectory-based moving objects generator. Under review in International Jour-  
nal of Artificial Intelligence Tools.  
4. Sajal Halder, Md. Mostofa Kamal Rasel, Yongkoo Han, and Young-Koo  
Lee. Mining Spatiotemporal Moving Objects Swarm. Under review in Kyung  
Hee University Journal..  
77

LIST OF PUBLICATIONS 78  
International Conference Papers  
5. Sajal Halder, Yongkoo Han and Young-Koo Lee. Discovering Periodic Pat-  
terns using Supergraph in Dynamic Networks. Accepted in 5th International  
Conference on Data Mining and Intelligent Information Technology Applica-  
tions (ICMIA),Jun 18-20, South Korea, 2013.  
6. SajalHalder,A.M.JehadSarkarandYoung-KooLee. MovieRecommendation  
System Based on Movie Swarm. Second International Conference on Cloud  
and Green Computing (CGC), China, Nov 1-3, 2012.  
7. Sajal Halder, Md. Samiullah, A. M. Jehad Sarkar and Young-Koo Lee.  
MovieSwarm: Information Mining technique for Movie Recommendation Sys-  
tem. In the 7th International Conference on Electrical and Computer Engi-  
neering (ICECE), Bangladesh, Dec 20-22, 2012.  
Thesis/Project Works  
8. Sajal Halder, Uzzal Kumar Dutta, Uttam Kumer Biswas and Asish Kumar  
Biswas “Classification of Multiple Protein Sequences by means of Irredundant  
Patterns”, B.Sc. Final Year Project, Department of Computer Science and  
Engineering (CSE), University of Dhaka (DU), Bangladesh, February, 2011.