Année 2018

Ecole Doctorale du Pacifique

Thèse

présenté devant
l'Université de la Nouvelle-Calédonie

par
Zhi Cheng

pour obtenir le diplôme de
docteur spécialité Informatique

MINING RECURRENT PATTERNS IN A DYNAMIC ATTRIBUTED GRAPH.
Application to aquaculture pond monitoring by satellite images

Soutenue publiquement le 16 Octobre 2018, devant le jury composé de :

Président

Christophe Menkes, Directeur de Recherche .... ENTROPIE / IRD

Rapporteurs

Sandra Bringay, Professeur .... UMR LIRMM / Université de Montpellier 3 LIRMM

Philippe Fournier-Viger, Professeur .... Harbin Institute of Technology (Shenzhen)/ Chine

Examinateurs

Hugues Lemonnier, Chercheur .... LEAD / IFREMER

Frédéric Flouvat, Maître de Conférences .... ISEA / UNC

Directrice

Nazha Selmaoui-folcher, Maître de Conférences, HDR .... ISEA / UNC

Acknowledgements

First and foremost, I would like to express my profound gratitude to my supervisor, Mrs. Nazha Selmaoui-Folcher, for her tireless help. Although it was difficult for me to make quick progress at the beginning, with her patient guidance and wise advices, I learned a lot. The door to her office was always open whenever I ran into a trouble spot or had a question about my research or writing.

I would like to express my deepest thanks to Mr. Frédéric Flouvat, for his patient supervision and insightful comments. His guidance helped me in all the time of research and writing of this thesis.

My sincere thanks go to the members of the team LEAD/IFREMER, Mr. Hugues Lemonnier and Mr. Benoît Soulard, for their time and interesting comments. I would like to thank especially Mrs. Niken Financia GUSMAWATI, a PHD in biology and ecology whom I worked with during my thesis, for her tremendous field surveys and her support in specialist fields.

It would also like to express my sincere gratitude to the two reviewers of my thesis, Mrs. Sandra BRINGAY and Mr. Philippe Fournier-Viger for their time and effort to invariably improve my manuscript. Their insightful comments and instructive questions help me to widen my research from various perspectives and inspire more ideas. I would also thank the member of my thesis committee Mr. Christophe Menkes, for his advices and suggestions.

It is a pleasure to thank my friends and my colleagues in institute of exact and applied sciences (ISEA): Cyril Francois, Jannai Tokotoko, Nicolas Folcher, Camille Pasquet, Charlotte Carré, Yves Le Guevel, Aurélie Boula, Monika Le Mestre, Jordan Prévot, and Pauline Fey, for the wonderful times we shared. They gave me the necessary distractions from my research and made my stay in New Caledonia memorable.

Last but not the least, I would like to thank my family, especially my wife, for their continuous love, support and help throughout my thesis.

Résumé

Dans cette thèse, nous nous sommes intéressés à l'analyse de données spatio-temporelles. Plusieurs algorithmes de fouille de données ont été développés pour extraire des modèles locaux (appelés aussi "motifs") tels que les motifs séquentiels ou les sous-graphes dynamiques. Cependant, ces approches souffrent de plusieurs limitations lorsqu'on traite des phénomènes spatio-temporels complexes. Ces domaines de motifs ne prennent pas en compte toutes les interactions spatio-temporelles possibles ou ne considèrent que des informations limitées sur les objets étudiés. Par exemple, les motifs séquentiels se concentrent sur les évolutions temporelles sans tenir compte des évolutions spatiales. En outre, la plupart des algorithmes d'extraction de sous-graphes étudie des graphes dynamiques labélisés. Cependant, ils ne considèrent qu'un seul attribut par noeud et ignorent les autres caractéristiques des objets étudiés.

Dans ce manuscrit, nous proposons d'étudier un graphe dynamique attribué pour fournir une représentation plus riche des phénomènes spatio-temporels. L'extraction de motifs dans des graphes dynamiques attribués est une tâche particulièrement complexe car la structure du graphe, les noeuds et les attributs associés à chaque noeud peuvent changer dans le temps. Pour cela, nous avons défini un nouveau domaine de motifs appelé motifs récurrents. Ces motifs, qui sont des séquences de sous-graphes connexes, représentent des évolutions récurrentes des sous-ensembles d'attributs associés à des sous-ensembles de noeuds. Pour extraire ces motifs, nous avons développé un nouvel algorithme, appelé **RPMiner**, utilisant une stratégie originale basée sur des intersections successives de composantes connexes apparaissant dans la séquence. Nous avons utilisé plusieurs contraintes pour réduire l'espace de recherche et rendre le calcul possible. Une étude expérimentale sur des jeux de données synthétiques et réels (réseau de co-auteurs DBLP et données de trafic aérien US Flight) montre la généricité de notre approche, l'intérêt des motifs extraits et l'efficacité de notre algorithme.

Nous avons effectué également une évaluation poussée de notre approche sur les données du projet INDESO (suivi de bassins aquacoles en Indonésie par imagerie satellitaire). Pour cela, un processus d'extraction de connaissances (KDD) complet a été développé : du prétraitement des données à la visualisation et à l'interprétation des résultats. Il vise à mieux comprendre les pratiques des fermiers pour un développement durable de ces ressources côtières en Indonésie. Ce processus s'appuie tout d'abord sur une méthode automatique et robuste pour extraire les bassins d'aquacoles à partir d'images satellitaires à faible contraste. Ensuite, il utilise des méthodes d'extraction de motifs fréquents afin de mettre en avant certaines pratiques des fermiers. Pour cela, nous avons appliqué dans un premier temps un algorithme d'extraction de motifs séquentiels pour analyser l'évolution des bassins dans le temps et comprendre les pratiques des fermiers. En parallèle, nous avons aussi appliqué notre algorithme **RPMiner**, qui prend en compte à la fois les dimensions spatiales et temporelles. Les motifs extraits ont été interprétés par des experts en aquaculture. Les résultats obtenus ont permis de confirmer certaines pratiques et d'en mettre en avant d'autres.

Abstract

In this thesis, we are interested in analyzing spatio-temporal data. Numerous algorithms have been developed to extract local models (also called "patterns") such as sequential patterns or dynamic subgraphs. However, these approaches suffer from several limitations when dealing with complex spatio-temporal phenomena. These pattern domains do not consider all possible spatio-temporal interactions or only consider limited information about studied objects. For example, sequential pattern mining methods focus on temporal evolutions without considering spatial ones. Besides, most of graph mining algorithms study labeled graphs. They only consider one attribute per vertex instead of all objects characteristics. In our work, we propose to study dynamic attributed graph, because they provide a richer representation of spatio-temporal phenomena. Extraction of patterns in dynamic attributed graph is a particularly complex task because graph structure, vertices and attributes associated with each vertex can change over time. For this purpose, we define a new pattern domain called recurrent patterns. These patterns, which are sequences of connected subgraphs, represent recurrent evolutions of subsets of attributes associated to vertices. To extract these patterns, we develop a new algorithm, **RPMiner**, using an original strategy based on successive intersections of connected components. We use several constraints to reduce the search space and make the computation feasible. Experimental study on both syndetic and two real-world datasets (DBLP dataset and Domestic US Flight dataset) show the genericity of our approach, the interest of extracted patterns and the efficiency of our algorithm. We also do an in-depth experimental evaluation of our approach on the INDESO project data (aquaculture pond monitoring in Indonesia by satellite images). A complete KDD process has been developed: from pre-processing of data to visualization and interpretation of results. It aims to better understand farming practices for sustainable development of these coastal resources in Indonesia. This process is firstly based on an automatic and robust method to extract aquaculture ponds from low contrast satellite images. Next, this process extracts frequent patterns to highlight some farming practices. For this, we have firstly applied a sequential pattern mining to analyze temporal evolutions of aquaculture ponds and to understand farming practices. In parallel, we also apply our algorithm, **RPMiner**, which considers both spatial and temporal aspects. Extracted patterns were interpreted by aquaculture experts. Results confirm several practices and highlight others.

Contents

List of figures

List of algorithms

Chapter 1

Introduction

Contents

1. Context

The development of sensor technologies and social media have greatly improved information collection and made huge amounts of data available. Faced with increasing data, new technologies are needed to help humans in transforming and resuming automatically these data to useful knowledge. (Fayyad *et al.*, 1996) presented a process for Knowledge Discovery in Databases (KDD). This process could be very complicated and steps may vary according to different nature of data and objective of applications. As shown in Fig. 1.1, the KDD process is an interactive and iterative multi-steps process. This process is composed of the following steps: data selection, pre-processing, transformation, data mining, post-processing and interpretation. The data selection step consists in selecting the sources of information. These sources of information could be structured (e.g. transaction, sequential or graph database) or unstructured data (e.g. books, images or videos). Then, data is preprocessed and transformed into appropriate data structure for mining. Data mining algorithms are chosen according to data types and applications. Finally, domain experts interpret solutions. This iterative process is repeated as long as required. We could note that data mining is just one step of the KDD process. However, it attracts the most attention. So far, many research efforts have been largely dedicated to define more relevant pattern domains and to develop scalable algorithms.

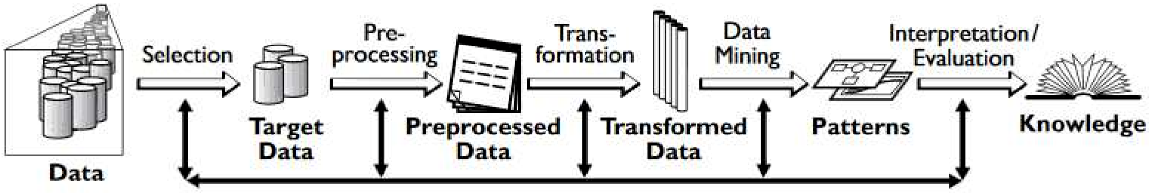


Figure 1.1 – Process of the Knowledge Discovery in Databases (KDD) (Fayyad *et al.*, 1996)

This work was inspired by the issue of sustainable aquaculture development in Indonesia, which was carried out as part of the "INDESO"[1] project (Infrastructure Development of Space Oceanography), in which the laboratory participated through collaboration with the LEAD/IFREMER team. This collaboration resulted in a co-supervision of N. Gusmawati's thesis defended in July 2017. This project aims to better manage marine and coastal resources in Indonesia. For this purpose, domain experts have conducted many field surveys and manual satellite images analysis (Gusmawati *et al.*, 2017) to study this spatio-temporal data. However, manual analysis and interpretations are very time consuming and can not scale to large data. Moreover, the vast majority of data cannot be processed by humans in a manual way, because of structure complexity of data (sequences, trees and graphs etc.)

[1] We would like to thank the INDESO project for providing the data

and large amount of information (e.g. objects, characteristics of objects and relationships between vertices) that they carried. To this aim, we are interested in developing new data mining algorithm permitting to study such complex spatio-temporal data.

2. Challenges

2.1 Representations of complex spatio-temporal data

Modeling and understanding spatio-temporal data is a major issue for a wide range of applications (e.g. understanding and managing aquaculture ponds, soil erosion monitoring, epidemic monitoring etc.). For example, dengue epidemic is characterized by a set of factors causing the propagation of the disease across time and space. When the epidemic is declared in a quarter, the question is to understand how and according to which factors, it will spread in other quarters. Even if this propagation depends on the direct environment of a zone (water, mangrove etc.) and a set of circumstances evolving over time (humidity, heat, precipitation etc), the dynamics of overall propagation is far from being under control if we consider all the possible interactions between those factors. To deal with such problems, various methods have been developed to help experts to discover interesting knowledge from spatio-temporal data. The objective of these methods is to find spatio-temporal relations between variables and events without a priori hypothesis. Recently, two data representations are largely used to study spatio-temporal data. The first is sequential data, where each sequence represents the evolution of an individual object. The second is dynamic labeled graph, where vertices represent objects, edges describe relationships between vertices. Each vertex is labeled by only one attribute. However, these data representations are limited to study complex spatio-temporal data, as mentioned by (Moser *et al.*, 2009):

“often vertex attributes and edges contain complementary information, i.e. neither the relationships can be derived from the vertex attributes nor vice versa. In such scenarios the simultaneous use of both data types promises more meaningful and accurate results“

Hence, a dynamic attributed graph has been proposed to describe many complicated data (e.g. spatio-temporal data, biological data or social data) (Desmier *et al.*, 2012; Desmier *et al.*, 2013; Kaytoue *et al.*, 2014). This graph permits to provide a richer representation of real-world phenomena where vertices represent objects, edges represent spatial relations or other types of interactions between vertices and attributes describe the characteristics of vertices. In this work, edges and attributes may evolve over time while vertices are fixed. As a consequence, this representation is still limited because in many real-world phenomena (s.t. soil erosion monitoring or aquaculture monitoring), vertices (i.e. studied objects) can also change (divide, merge, appear, or disappear) over time. A more general data representation is needed.

2.2 Mining complex spatio-temporal data

Many sequential pattern mining algorithms are developed to study spatio-temporal data. For example, (Celik *et al.*, 2006; Celik *et al.*, 2008; Celik, 2015) developed an algorithm based on a generate-test strategy to mine spatio-temporal co-occurrence pattern, i.e. subsets of two or more different event-types whose instances are often located in spatial and temporal proximity. It extracts size k spatio-temporal patterns and then uses them to generate size $k+1$ candidates. (Aydin and Angryk, 2016) proposed two Apriori-based algorithms to mine spatio-temporal event sequences. (Huang *et al.*, 2007) developed a temporal slicing-based algorithm to mine sequential patterns from spatio-temporal event data sets. It slices firstly the database according to time and hashes events into slices. Then, they mine sequential patterns from each slice. (Alatrista-Salas *et al.*, 2012) proposed an algorithm based on pattern-growth approach to discover spatio-sequential patterns. It permits to analyze the evolution of areas considering their features and their direct neighboring environment.

Besides, a large amount of graph mining algorithms have been proposed in the literature and could be used to discover interesting patterns from spatio-temporal data. For example, (Inokuchi and Washio, 2010b) developed a method to mine frequent patterns called FRISSs (frequent relevant induced subgraph subsequence) efficiently from labeled graph sequences. They first construct union graphs for all graph sequences, then all frequent, induced subgraphs are extracted from these union graphs by using a graph mining algorithm based on depth-first strategy. In (Ozaki and Ohkawa, 2009), authors developed an algorithm to discover correlated sequential subgraphs from a sequence of labeled graphs. They developed a level-wise algorithm CorSSS (based on the Apriori algorithm) based on a tree shaped data structure that uses the generality ordering among patterns.

However, classical sequence mining algorithms and graph mining algorithms can hardly be adapted to study a dynamic attributed graph, because it is much more complex compared with sequential data (do not consider relations between objects) or dynamic labeled graph (only consider one attribute per vertex).

To the best of our knowledge, few methods have been proposed to mine a dynamic attributed graph. Mining such graphs is a complex task because every vertex is associated to a set of attributes (instead of a single label). (Desmier *et al.*, 2012) extracted cohesive co-evolutions in a dynamic attributed graph. These patterns represent a set of vertices with the same attributes and a similar neighborhood over a set of timestamps (vertices and attributes were fixed). The authors extended their work in (Desmier *et al.*, 2013) to integrate constraints on the graph topology and on the attribute values. However, in these works, all vertex attributes have to follow the same trend over time. Moreover, they don't take temporal evolutions of vertices into account. In (Kaytoue *et al.*, 2014), the authors define the triggering pattern problem which allows to find temporal relationships between the vertex attributes and their topological properties (degree, betweeness, number of cliques etc.). The authors design an algorithm TRIGAT to mine triggering patterns using a pattern-growth approach. However, it doesn't consider the pattern neighbors neither their evolutions over time.

3. Contributions

In this thesis, my contributions are of two types:

3.1 Methodological Contributions

In our work, we define a new pattern domain in a dynamic attributed graph, more precisely, a sequence of attributed graph called recurrent pattern. This kind of pattern extracted form a dynamic attributed graph describes temporal evolutions of connected vertices appearing in the same way over time. They are in some ways the sequences of connected subgraphs verifying several constraints. These constraints aim to reduce the search space and extract meaningful patterns. We propose to use two constraints considering the graph structure, i.e. the connectivity and the cohesiveness. Temporal continuity enables to target patterns which describe evolutions around a common core of individuals. Gap allows to study evolutions in a short term and in a long term. Moreover, frequency is used to calculate the number of occurrences of patterns.

We also propose a novel algorithm **RPMiner** to extract recurrent patterns in a dynamic attributed graph. Different from other strategies based on depth-first search, breadth-first search, successive projections of data or generate-test, our algorithm is based on graph intersections at different times. Size-i patterns generated by intersecting graphs at times T_i containing t_i are extended by the patterns generated at times T_{i+1}. Thus, times are processed incrementally and patterns of different sizes can be generated at each iteration. The advantage of this approach it to avoid the generation of a large number of patterns which don't verify the constraints and to explore the dynamic attributed graph in an incremental manner. Our algorithm permits to extract recurrent patterns.

To study the performances of our approach, several experimentations are conducted on both syndetic and real-world datasets. They demonstrate that our generic algorithm enables to discover relevant patterns efficiently.

3.2 Contributions to aquaculture monitoring

To study evolutions of aquaculture ponds, we develop a complete KDD process: from pre-processing to visualization and interpretation of results. Firstly, we propose an automatic and accurate method to extract aquaculture ponds from very complicated and low contrast satellite images. Secondly, we develop several methods to identify ponds' attributes. Thirdly, two automatic processes are developed to transform images of aquaculture ponds to sequential data and a dynamic attributed graph. Fourthly, we apply a sequence mining algorithm to study temporal evolutions of aquaculture ponds. In parallel, we also apply our algorithm, **RPMiner**, which considers both spatial and temporal aspects. Finally, the extracted patterns are visualized on original satellite images and validated by domain experts. Results show that recurrent patterns extracted by our **RPMiner** algorithm enable domain experts to identify farms, understand various managements of farmers and study the spread of disease and mangrove over adjacent ponds. It could give experts a new insight to study such spatio-temporal phenomena.

4. Organisation of the manuscript

In chapter 2, we review existing approaches to mine patterns in sequence data in section 2.1. Section 2.2 presents a survey about dynamic labeled graph mining. Finally, we present related works on dynamic attributed graphs (section 2.3).

Chapter 3 details our contributions from a methodological point of view. In section 3.1, we define a new pattern domain and some interesting constraints. Then, in section 3.2, an algorithm is presented and its performances are evaluated using artificial datasets. On the other hand, experiments on two real world datasets confirm the relevance of extracted patterns.

Chapter 4 presents a detailed application dealing with aquaculture monitoring in Indonesia. Section 4.1 introduces the problematic. In section 4.2, we describe the aquaculture farming data. Section 4.3 presents the EDB method to segment satellite images and compares it with two existing segmentation methods. Then, in Section 4.4, we present automatic methods to identify pond attributes. Section 4.5 details our methods that transform this time series of satellite images into sequential data and a dynamic attributed graph. Then, we visualize and interpret the results obtained with sequence mining algorithm (section 4.6) as well as the results extracted by our algorithm **RPMiner** (section 4.7). Finally, we conclude this part and compare these two different data mining methods.

In chapter 5, we conclude this thesis by giving a summary of the thesis and some perspectives for future work.

Chapter 2

State Of The Art

Contents

1. Sequential pattern mining

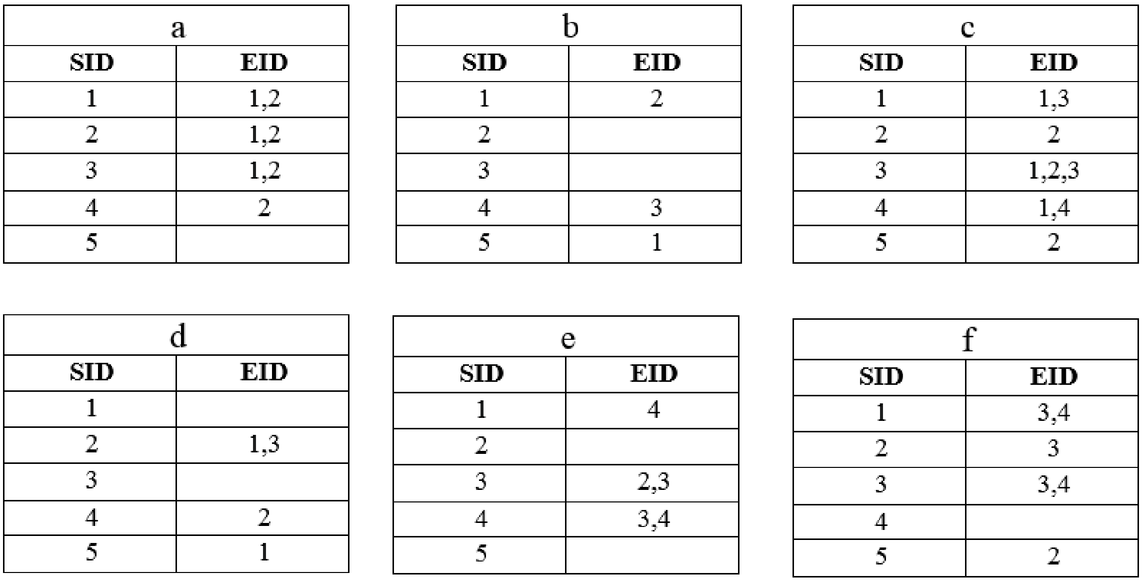
1.1 Theoretical framework

Sequential pattern mining methods are widely applied in various areas such as market basket analysis (Srikant and Agrawal, 1996), bioinformatics (Wang *et al.*, 2007), web usage analysis (Iváncsy and Vajk, 2006), text analysis (Pokou *et al.*, 2016) and time series of satellite images (Sanhes *et al.*, 2013; Sanhe, 2014). Discovering frequent subsequences in a set of sequences has been an important research topic. Let $I = \{i_1, i_2, ..., i_n\}$ be the set of all items, an itemset is a subset of I. Without loss of generality, we assume that the items are listed in alphabetical order. A sequence is an ordered list of itemsets $s =< I_1, I_2, ..., I_n >$ such that $I_k \subseteq I$ $(1 \leq k \leq n)$. A sequence is said to be of length k or k-sequence if it contains k items. A sequence database SDB is a set of sequences $SDB =< s_1, s_2, ..., s_m >$. Each sequence has an identifier $(SIDs) \in \{1, 2, ..., m\}$. For example, Table 2.1 shows a sequence database composed of five sequences with $SID \in \{1, 2, 3, 4, 5\}$. A sequence $s_b =< B_1, B_2, ..., B_m >$ is said to contain another sequence $s_a =< A_1, A_2, ..., A_n >$ if and only if there exists integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $A_1 \subseteq B_{i_1}$, $A_2 \subseteq B_{i_2}$,..., $A_n \subseteq B_{i_m}$. For example, in Table 2.1, the sequence $< \{a, c\}, \{a, b\}, \{c, f\}, \{e, f\} >$ contains the sequence $< \{c\}, \{f\}, \{e\} >$ but not the sequence $< \{c\}, \{f\}, \{d\} >$. The support of a sequence s_a in a sequence database is defined as the number of sequences containing s_a, i.e., $sup(s_a) = \{s|s_a \subseteq S \wedge s \in SDB\}$. In Table 2.1, the support of the sequence $< \{a, c\}, \{c, f\}, \{f\} >$ in the database is two because this sequence is contained in two sequences (sequences 1 and 3).

Table 2.1 – A sequence database

SID	Sequence
1	$< \{a, c\}, \{a, b\}, \{c, f\}, \{e, f\} >$
2	$< \{a, d\}, \{a, c\}, \{d, f\} >$
3	$< \{a, c\}, \{a, c, e\}, \{c, e, f\}, \{f\} >$
4	$< \{c\}, \{a, d\}, \{b, e\}, \{e\} >$
5	$< \{b, d\}, \{c, f\} >$

Vertical databases are also widely used to represent a sequence database. In a vertical database, every item is associated with an ID-list. For every item in the database, its ID-list is a set of $< SID$ $(SequenceID), EID$ $(EventID) >$ which indicates where each item appears in the sequence database. For example, Fig. 2.1 shows the vertical database of the sequence database displayed in Table 2.1. The ID-list $< SID, EID >$ of item d indicates that d appears in the first and the third itemset of sequence 2 and in the second itemset of sequence 4.



a	
SID	EID
1	1,2
2	1,2
3	1,2
4	2
5	

b	
SID	EID
1	2
2	
3	
4	3
5	1

c	
SID	EID
1	1,3
2	2
3	1,2,3
4	1,4
5	2

d	
SID	EID
1	
2	1,3
3	
4	2
5	1

e	
SID	EID
1	4
2	
3	2,3
4	3,4
5	

f	
SID	EID
1	3,4
2	3
3	3,4
4	
5	2

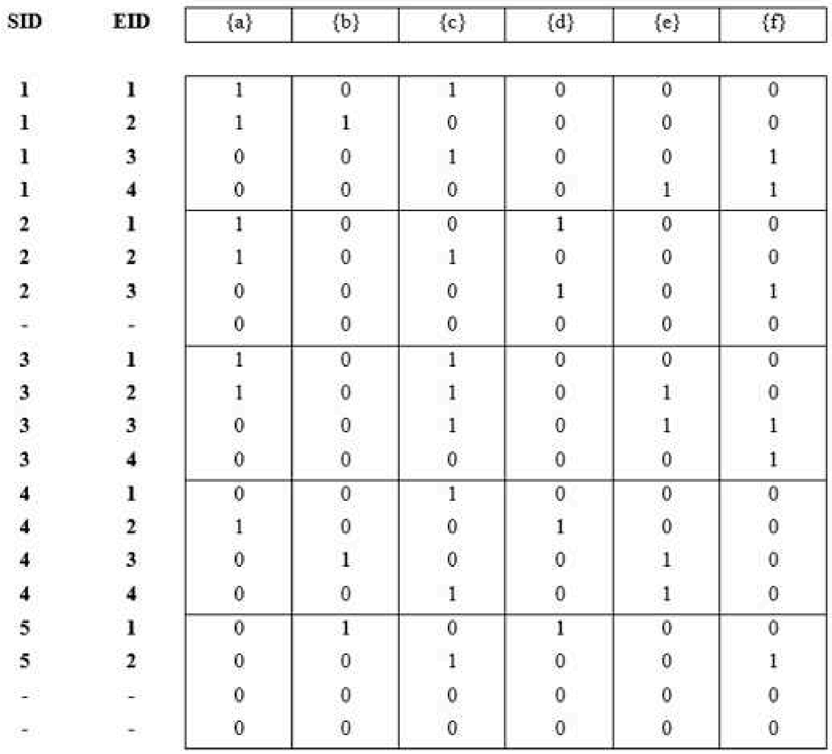
Figure 2.1 – The vertical representation of the sequence database shown in Table 2.1

1.2 Mining strategies

Three main approaches were proposed to mine sequential patterns. Some of them use a breadth-first search strategy such as GSP (Srikant and Agrawal, 1996) and AprioriALL (Agrawal and Srikant, 1995). For example, algorithm GSP first scans the database to generate frequent 1-sequences and keep them in memory. Then it uses the k-sequences to generate sequences of length $k+1$. This recursive process stops until no patterns could be generated. However, this strategy has several drawbacks. Firstly, a large number of database scans is needed to calculate the support of each candidate pattern. Secondly, GSP could generate nonexistent patterns in the database because candidates are generated just by combining smaller pattern without accessing the database. Finally, it requires a large amount of memory because it keeps all frequent k-sequences in memory to generate $(k+1)$-length patterns.

Some algorithms use another strategy. They traverse the search space using a depth-first search strategy such as SPADE (Zaki, 2001), SPAM (Ayres *et al.*, 2002), CM-Spam (Fournier-Viger *et al.*, 2014) and CM-Spade (Fournier-Viger *et al.*, 2014). In (Zaki, 2001), authors develop a depth-first search algorithm SPADE to discover frequent sequential patterns from a vertical database. The advantage of using vertical database is that the IDList of any pattern allows to directly calculate its support without accessing database. Moreover, any $(k+1)$-sequence generated by extending a k-sequence with an item i, can be created without scanning the database (join the ID-list of k-sequence with the ID-list of item i). Thus, only the intermediate id-lists for two consecutive levels (k-sequences and $(k+1)$-sequences) are maintained in memory. Consequently, Spade is much more efficient than previous breadth-first search algorithms.

However, one of the main limit of Spade is that it is not efficient to mine a sequence database containing long sequences, because IDLists could be very large and as a consequence, the join operation of IDLists will be very costly. For this purpose, two algorithms bitSPADE (Aseervatham *et al.*, 2006) and SPAM (Ayres *et al.*, 2002) were proposed based on a bitmap representation. Each bitmap associated with an item has a bit which indicates the item position in the dataset. If item i appears in the $t-th$ itemset of sequence j, then the $t-th$ bit of sequence j for item i is set to one and otherwise to zero. Table 2.2 shows an example of bitmap representation of the sequence dataset in Table 2.1. The vertical bitmap of each item is composed of five sections where each section corresponds to a sequence. For example, $<1,2>$, i.e., the second itemset of the first sequence contains items a and b, so the second bits of the first sequence for items a and b are set to 1. SPAM and bitSPADE are much more efficient than SPADE in terms of runtime and memory usage particularly on dense or long sequences (many bits of items of this data are set to 1). However, both SPAM and bitSPADE are based on a generate-test strategy, so they inevitably generate a huge amount of infrequent candidates. To solve this problem, in (Fournier-Viger *et al.*, 2014), authors develop CM-Spam and CM-Spade algorithms based on the concept of co-occurrence pruning. Firstly, they scan the whole database to generate a structure called the Co-occurrence MAP (CMAP) composed of all frequent 2-sequences. Then in the search process, for each considered pattern s_a, if its two last items are not frequent 2-sequences, the pattern s_a is infrequent and there is no need to perform the join operation. Thus in practice, CM-Spam and CM-Spade avoid testing lots of infrequent candidates. CM-Spam (Fournier-Viger *et al.*, 2014) and CM-Spade (Fournier-Viger *et al.*, 2014) are the most efficient algorithms, they outperform all current sequence mining algorithms by more than one order of magnitude.



SID	EID	{a}	{b}	{c}	{d}	{e}	{f}
1	1	1	0	1	0	0	0
1	2	1	1	0	0	0	0
1	3	0	0	1	0	0	1
1	4	0	0	0	0	1	1
2	1	1	0	0	1	0	0
2	2	1	0	1	0	0	0
2	3	0	0	0	1	0	1
-	-	0	0	0	0	0	0
3	1	1	0	1	0	0	0
3	2	1	0	1	0	1	0
3	3	0	0	1	0	1	1
3	4	0	0	0	0	0	1
4	1	0	0	1	0	0	0
4	2	1	0	0	1	0	0
4	3	0	1	0	0	1	0
4	4	0	0	1	0	1	0
5	1	0	1	0	1	0	0
5	2	0	0	1	0	0	1
-	-	0	0	0	0	0	0
-	-	0	0	0	0	0	0

Table 2.2 – Bitmap representation of the sequence database shown in Table 2.1

Pattern-growth strategies such as PrefixSpan (Pei *et al.*, 2004) and FreeSpan (Han *et al.*, 2000) were also proposed. Their approach is based on depth-first exploitation and database projections. In (Pei *et al.*, 2004), authors propose a pattern-growth algorithm named PrefixSpan to mine frequent sequential patterns from in an horizontal database. PrefixSpan scans first the database to generate all 1-sequence patterns. Then PrefixSpan constructs projected database based on the prefix of the pattern being extended. PrefixSpan extends the prefix with the items which are frequent in the projected database to form longer sequential patterns. This process continues until no more patterns could be extended. The advantage of PrefixSpan is that it generates only patterns occurring in the database. However, a serious problem of PrefixSpan and all pattern-growth algorithms is that constructing projected databases is very time-consuming and could take a huge amount of space in memory.

1.3 Other constraints

These algorithms may find a huge amount of patterns which is time-consuming and difficult to analyze. To reduce the number of patterns, three condensed representations have been studied: closed patterns, maximal patterns and generator sequential patterns. A closed sequential pattern is a sequential pattern that has no supersequence with the same support (Huang *et al.*, 2006; Wang *et al.*, 2007; Yan *et al.*, 2003). A maximal sequential pattern is a sequential pattern such that it is not contained in any other sequential patterns (Lu and Li, 2004; Luo and Chung, 2005; Lin *et al.*, 2007). A generator sequential pattern is a sequential pattern that have no subsequence having the same support (Lo *et al.*, 2008; Gao *et al.*, 2008; Yi *et al.*, 2011).

Other constraints have been integrated into the mining process to reduce the search space, reduce the number of patterns and extract more interesting patterns. For example, in (Fournier-Viger *et al.*, 2008), authors extend the algorithms BIDE by integrating gap constraints (minimum and maximum time interval between two consecutive itemsets in sequential pattern) and duration constraint (maximum time interval between the first itemset and the last itemset of a sequential pattern). Gap constraint can be very useful because it permits to follow consecutive evolutions or study seasonal evolutions according to different applications. In (Pei *et al.*, 2007), authors propose several constraints in pattern-growth algorithms such as item constraint (items that must appear or not in every extracted pattern) and length constraints (minimum/maximum number of items per pattern). This constraint helps study the evolution having at least one special item (special items) which is (are) more important than other items. For example, for a time series of aquaculture ponds satellite images, experts are especially interested in patterns describing ponds with activity and without activity because activity is the key factor to study the evolution. Several extensions of sequential pattern mining problems have also received much attention such as multi-dimensional sequential pattern mining (Pinto *et al.*, 2001; Songram and Boonjing, 2008), top-k sequential pattern mining (Fournier-Viger *et al.*, 2013), weighted sequential pattern mining (Yun and Leggett, 2006; Ren *et al.*, 2008), high-utility sequential pattern mining (Yin *et al.*, 2012; Lan *et al.*, 2014), uncertain sequential pattern mining (Muzammal and Raman, 2010; Muzammal and Raman, 2011; Zhao *et al.*, 2014) and periodic pattern mining (Tanbeer *et al.*, 2009; Kiran and Reddy, 2009; Kiran *et al.*, 2016). These extensions permit to model sequential database though different ways and discover more pertinent and meaningful patterns according to different characteristics of data and for special needs.

2. Pattern mining in dynamic graphs

Graphs are more and more playing a prominent role in modeling complex structures. A large number of graph mining algorithms have been developed (Aggarwal and Wang, 2010; Cook and Holder, 2006) for various application domains such as remote sensing, social networks, epidemiology and bioinformatics (Berlingerio *et al.*, 2011; Prakash *et al.*, 2014; Sanhes *et al.*, 2013). Various types of graphs have been studied in the literature such as static graphs (Inokuchi *et al.*, 2000; Huan *et al.*, 2003; Kuramochi and Karypis, 2004; Deshpande *et al.*, 2005), multidimensional graphs (Berlingerio *et al.*, 2011) and attributed graphs (Moser *et al.*, 2009; Khan *et al.*, 2010; Pasquier *et al.*, 2013). Recently, dynamic labeled graphs have received much attention (Ahmed and Karypis, 2015a; Berlingerio *et al.*, 2009; Borgwardt *et al.*, 2006; Lahiri and Berger-Wolf, 2010; Bogdanov *et al.*, 2011; Inokuchi and Washio, 2010b; Ozaki and Ohkawa, 2009). However, to our best knowledge, few methods have been proposed to mine a dynamic attributed graph which provides a richer information.

In this chapter, we present related works on dynamic graph. It can be categorized into two models, i.e., a single dynamic graph (network) and a graph sequence database (a collection of graph sequences).

A single dynamic graph (network) is a sequence of labeled graphs in which vertices represent entities, edges denote the relationships or connections between entities (Ozaki and Ohkawa, 2009). Vertices and edges could appear and disappear over time. Fig. 2.2 shows an example of labeled dynamic graph. It is a sequence of graphs $\mathcal{G} = \langle G_{t_1}, G_{t_2}$, ..., $G_{t_{max}} \rangle$ which represents the evolution of a graph over a set of time $\mathcal{T} = \{t_1, \ldots, t_{max}\}$. For each time $t \in T$, $G_t = (V_t, E_t, \lambda_t)$ is a graph where $V_t \subseteq \mathcal{V}$ is the set of vertices at time t, $E_t \subseteq V_t \times V_t$ is the set of edges at time t, with a labeling function λ: $V \cup E \mapsto \sum$, assigning to vertices and edges labels from an alphabet $\sum$. These labels represent properties, and for simplicity we assume that they do not change with time. A dynamic graph is used as a general representation for many real world applications. For example, Fig. 2.2 could represent a social network, where each vertex is a person, edges describe interactions between humans. Vertices increase or decrease when a person joins or leaves the community in the social network, edges evolve when relationships are created/deleted.

A graph sequence database is composed of a collection of graph sequences (Inokuchi and Washio, 2008). Fig. 2.3 depicts a database of two graph sequences d_1 and d_2 where $d_1 =< g_1^{(1)}$ $g_1^{(2)}$ $g_1^{(3)}$ $g^{(4)} >$ and $d_1 =< g_2^{(1)}$, $g_2^{(2)}$, $g_2^{(3)}$, $g_2^{(4)} >$. Each graph sequence is an ordered list of labeled graphs i.e. $d =< g^{(1)}$, $g^{(2)}$, ... , $g^{(t_{max})} >$. The total number of vertices in the graph sequence $\sum_{j=1}^{l} |V(g^{(j)})|$ is the size of the graph sequence and each vertex v in $g^{(j)}$ has an unique ID. Many real world applications can generate such database. For example, an email

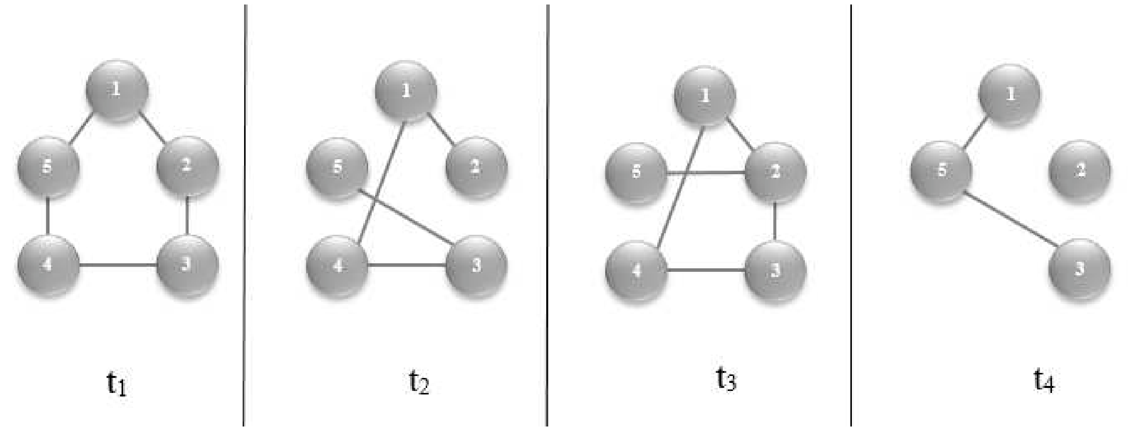


Figure 2.2 – Example of a labeled dynamic graph

communication network can be represented by daily/weekly graph sequence data, where vertices are persons assigned by a unique ID and edge represents personal communication by email. The total number of days/weeks is the number of sequences.

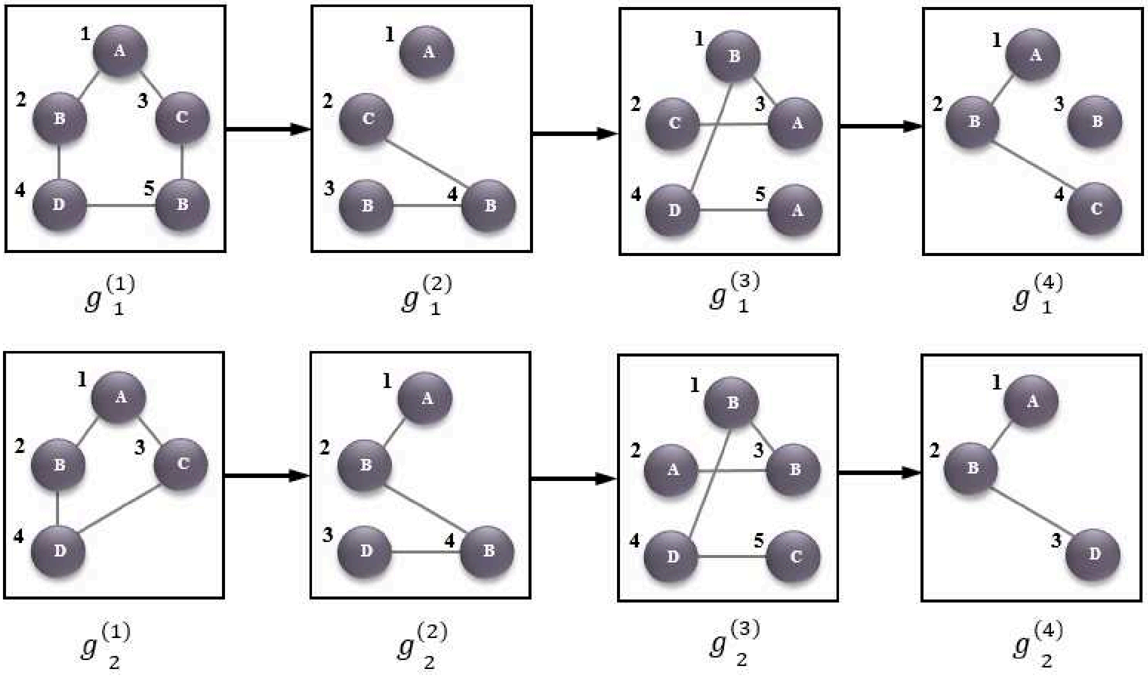


Figure 2.3 – Example of graph sequence database

Mining a single dynamic graph and mining a collection of graph sequences have been studied separately. For a single dynamic graph, the frequency of a pattern is the number of its occurrences (i.e., embeddings) in this graph. While in a graph sequence database, the frequency of a pattern is defined as the number of graph sequences that the pattern occurs in. Algorithms developed to mine a graph sequence database is not adapted to mine a single dynamic graph, while the latter algorithms can be used to study a graph sequence database.

Many different methods have been designed to study dynamic labeled graph. Some studies focus on extracting high-scoring subgraphs, i.e. highest-scoring (based on edge weights) connected subgraph over a time sub-interval (Bogdanov *et al.*, 2011). Some studies focus on mining frequent subgraphs (Jiang *et al.*, 2010), i.e., subgraphs whose support are greater than a user defined threshold. Some studies aim at discovering the evolution of subgraphs through time (Holder and Cook, 2009), i.e. how the subgraphs structure change between consecutive timestamps. Some other studies discover δ-contiguous closed 3-cliques patterns (Cerf *et al.*, 2009b), i.e. maximal sets of densely connected vertices that run along some nearly contiguous timestamped graphs.

2.1 High-scoring subgraphs

In (Bogdanov *et al.*, 2011), authors define the problem of extracting high-scoring connected temporal subgraphs (HDS). The score is the sum of edge weights and edge weights evolve through time, i.e., it is a dynamic graph with numeric edge labels. Approaches for finding high-scoring subgraphs have various applications. For example, ENRON is a dataset (Diesner *et al.*, 2005) where vertices are employees. An edge exists if there is at least one message between two personal accounts and edge weight is the number of messages. High-scoring subgraphs could represent communication workflow. Given an edge-weighted evolving network $\mathcal{G} = \langle G_{t_1}, G_{t_2}\ , ..., G_{t_{max}} \rangle$ with $\lambda = (\lambda^1, ..., \lambda^{t_{max}})$ s.t. $\lambda^i : E \to \{-1, 1\}$. A temporal subgraph is a pair $(G, [i, j])$ where $G(V, E, \lambda)$ is a connected graph and $[i, j]$ is a sub-interval of $[1, t_{max}]$.

The HDS problem is then to extract the complete set of temporal sub-graphs that maximizes the score, i.e., the sum of all edge-weights of the pattern. For example, as shown in Fig. 2.4 (a), an edge-evolving network could represent a traffic flow on a freeway network over time (Bickel *et al.*, 2003). Solid lines (edges with score 1) represent low occupancy, i.e., traffic flows freely, and dashed lines (edges with score -1) signify that flow and velocity decreases. At the beginning, freeways between districts (A, B), (A, C) and (B, E) are at low occupancy, then freeway (A, C) remains at low occupancy and terminates at t_5. The heaviest dynamic subgraph over the sub-interval $[1, 3]$ is the set of edges $\{(A, B)$, (A, C), $(B, E)\}$ with a score of 5 and the heaviest dynamic subgraph over the sub-interval $[1, 5]$ is the set of edges $\{(A, B)$, (A, C), (B, E), $(D, E)\}$ with a score of 8.

The authors develop an algorithm based on filter-and-verify strategy to extract heaviest dynamic subgraph that scales to large networks with long evolution extent. It is very efficient by using tight upper bounds to prune irrelevant time intervals instead of enumerating all possible intervals.

2.2 Weighted frequent sub-graphs

In (Jiang *et al.*, 2010), authors aim to extract weighted frequent subgraphs from a single sequence of edge weighted graphs. For many applications, a frequent subgraph with a higher edge weight value is more meaningful than others with the same support threshold. A sequence of edge weighted graphs $\mathcal{G}$ is defined as $\mathcal{G} = \langle G_{t_1}, G_{t_2}\ , ..., G_{t_{max}} \rangle$. $G_t = (V_t, E_t, \lambda_t)$ is

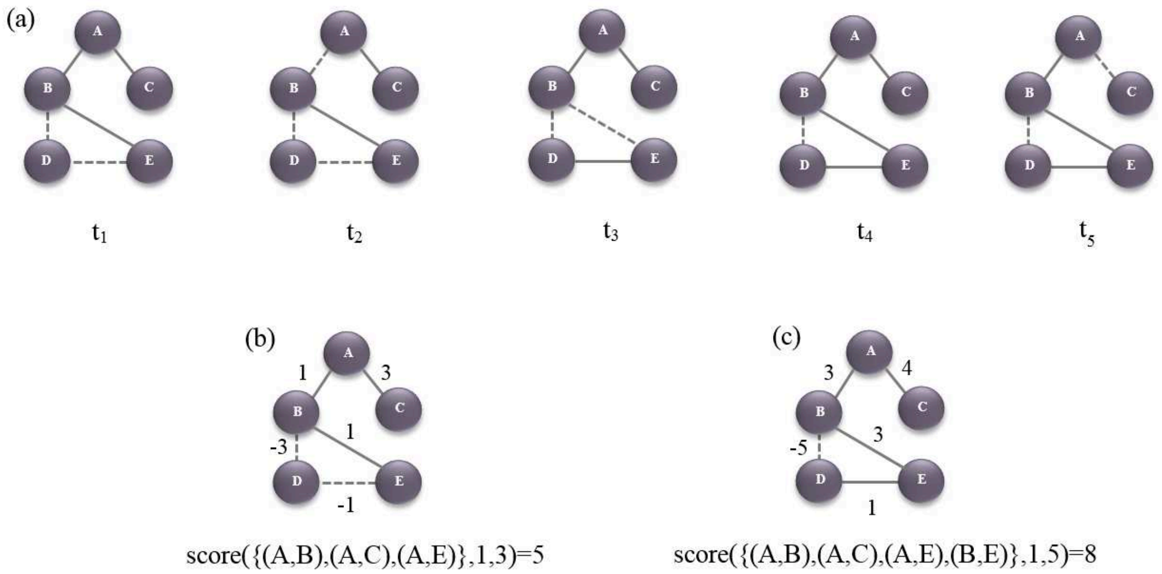
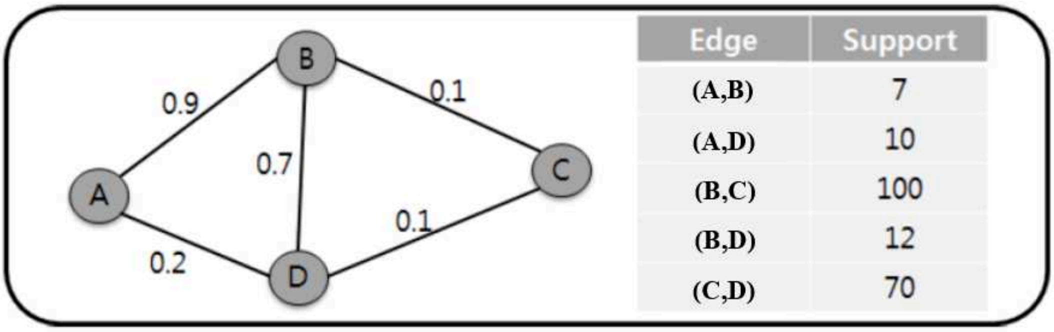

(a)
t1
t2
t3
t4
t5
(b)
1
3
1
-3
-1
score({(A,B),(A,C),(A,E)},1,3)=5
(c)
3
4
3
-5
1
score({(A,B),(A,C),(A,E),(B,E)},1,5)=8

Figure 2.4 – (a) example of edge-evolving network where each edge is scored either 1 or -1 (solid or dashed, respectively). (b) the heaviest subgraph composed of $\{(A, B)$, (A, C), $(B, E)\}$ over the sub-interval [1, 3]. (c) the heaviest subgraph composed of $\{(A, B)$, (A, C), (B, E), $(D, E)\}$ over the sub-interval [1, 5].

a labeled graph where V_t is a set of vertices, E_t is a set of edges and λ_t is a function that associates edges with a set of numeric labels. The authors propose three edge weighting schemes (Average Total Weighting, Affinity Weighting and Utility Based Weighting) and incorporate three strategies into three weighted variations of the gSpan algorithm (ATW-gSpan, AW-gSpan, and UBW-gSpan) to mine weighted frequent subgraphs. Instead of searching graphs and testing isomorphism, gSpan (Yan and Han, 2002) constructs canonical DFS (depth first search) code for each graph. Based on these codes, gSpan adopts the depth-first search strategy to extract frequent subgraphs efficiently. Compared with the gSpan algorithm, these three approaches are mainly designed to reduce search space by discovering the most relevant sub-graphs. The proposed algorithms are more efficient than gSpan in terms of runtime and memory use.

The frequent subgraphs extracted by previous methods consider only supports or weights. However, such patterns can be meaningless if it has a low affinity value, i.e. internal elements (edges) of the pattern are scarcely correlated even though it satisfies minimum support and weight thresholds at the same time. In other word, previous algorithms (Günnemann and Seidl, 2010; Jiang *et al.*, 2010; Ozaki and Etoh, 2011) could consume much time and memory in conducting graph isomorphic test to find useless sub-graphs (a NP-hard problem).

As an example, Fig. 2.5 illustrates an email communication database, the table on the right shows edge supports (i.e. the number of email communications) and edge attributes represent corresponding weights. If we do not apply affinity measures (support affinity and weight affinity), all the super graphs extracted from the graph are valid. However, if we consider affinity conditions, some needless patterns will not be generated. As shown in the



Edge	Support
(A,B)	7
(A,D)	10
(B,C)	100
(B,D)	12
(C,D)	70

Figure 2.5 – An email communication database (Lee and Yun, 2012)

table of Fig. 2.5, the edges (A,B), (A,D) and (B,D) have lower support value than (B,C) and (C,D). Thus, the former edges have a different tendency compared with the latter edges. For example, A-B-D (cyclic) often represents personal communications while B-C-D (path) could be spam attacks or advertisements because the supports (number of emails) of edges (B,C) and (C,D) are quite high. Thus, only A-B-D and B-C-D are valid patterns. But if we consider weight affinity in the graph, A-B-D (cyclic) becomes invalid because the weight value of (A,D) is much lower than (A,B) and (B,D) and thus this pattern has a low weight affinity value. As a consequence, to avoid such patterns, (Lee and Yun, 2012) define two affinity measures (support affinity and weight affinity). Given a sub-graph G, the support affinity of G is the ratio of the support of G in the database to the highest support of edges of G. It's weight affinity is the ratio of the minimum edge weight in G to the maximum one.

Given an edge weighted graph, the mining problem consists in extracting all sub-graphs whose support affinities and weight affinities are both greater or equal to two minimum affinities thresholds. The authors propose an efficient depth-first search algorithm MWSA to extract valid sub-graphs by pruning useless patterns. By using the affinity measures and their anti-monotone properties, MWSA permits to prune patterns efficiently.

2.3 Frequent pattern mining from a collection of graph sequences and a single dynamic graph

A graph sequence database is composed of a collection of labeled graph sequences $< sid, d >$, where sid is the ID of a graph sequence and d is a graph sequence. Each graph sequence is an ordered list of labeled graphs i.e., $d =< g^{(1)}, g^{(2)}, \dots , g^{(t_{max})} >$. Many real world applications can generate such databases. For example, an email communication network can be represented by daily/weekly graph sequence data, where vertices are persons identified by a unique ID and edges represent personal communications by email. The total number of days/weeks is the number of sequences.

In (Inokuchi and Washio, 2008), the authors propose a method GTRACE (Graph Transformation sequence mining) based on a depth-first strategy, to mine frequent sequences of graphs from a graph sequence database. They define transformation rules that represent graphs under the assumption that the change over sequential graphs is gradual. However, GTRACE becomes intractable for Enron graph sequences containing more than 7 graphs and

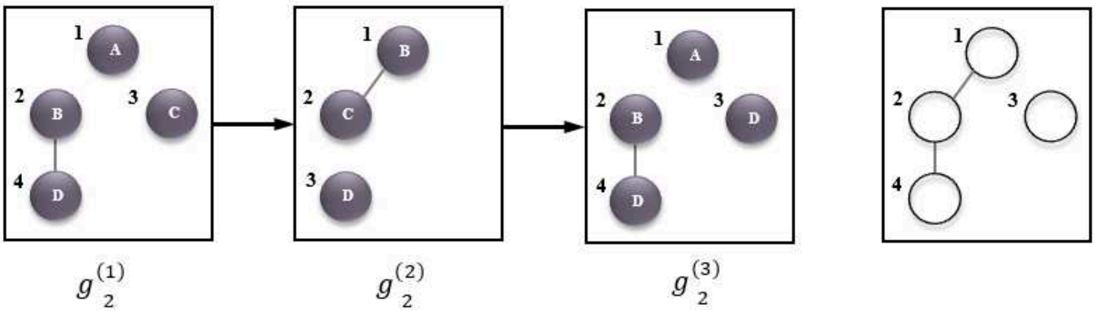
100 unique vertices. To solve this issue, the same authors propose a more efficient method (Inokuchi and Washio, 2010a) to find frequent patterns called FTSs (Frequent Transformation Subsequences) from longer (more graphs) and larger (more vertices) graph sequences. Given a graph sequence $d =< g^{(1)}\ g^{(2)}\ ...\ g^{(t_{max})} >$, the difference between two consecutive graphs is a sequence of small changes $s^{(j)} = < g^{(j,1)}, ..., g^{(j,m_i)} >$ called intrastate. It is a set of transformations between g_i and g_{i+1}. Each transformation represents insertion, deletion or relabeling of a vertex or an edge. Then, frequent patterns are extracted based on transformation rules.

However, change between two consecutive graphs g_i and g_{i+1} has to be gradual. To overcome this limit, the authors (Inokuchi and Washio, 2010b) improve their method to mine frequent patterns relevant induced subgraph subsequences from graph sequences containing long sequences and large graphs efficiently. They first construct a union graph (a graph composed of all vertices and edges of these graphs) for each graph sequence. Then all frequent induced subgraphs are extracted from these union graphs by using a conventional graph mining algorithm. A subgraph g' of g is an induced graph if and only if two vertices in $V(g')$ are adjacent in both g' and g. A subgraph sequence $d =< g^{(1)}, g^{(2)}, ... , g^{(l)} >$ is called relevant if the union graph $g_u(d)$ of d is a connected graph, where $g_u(d) = (V(g_u(d)), E(g_u(d)))$ is defined as follows:

$V(g_u(d)) = \bigcup_{j=1,...,l}\{id(v)|v \in V(g^{(j)})\}$

$E(g_u(d) = \bigcup_{j=1,...,l}\{(id(v), id(v'))|(v, v') \in E(g^{(j)})\}$

For example, as shown in Fig. 2.6 (a), the vertices with unique IDs 1 and 4 are not connected directly in any of the three graphs. However, they are connected to the vertex 2 in the second graph and the first graph respectively. So vertices 1 and 4 are relevant via the vertex 3. Given a graph sequence database and a minimum support threshold, the aim is thus to extract all frequent patterns whose union graphs are connected.



(a) A graph sequence d

(b) A union graph of d

Figure 2.6 – Graph sequence d and its union graph $g_u(d)$

The authors (Inokuchi and Washio, 2010b) propose FRISSMiner algorithm based on depth-first search strategy to mine Frequent Relevant, and Induced Subgraph Subsequences. FRISSMiner is more efficient than GTRACE in terms of runtime and memory usage because it only generates patterns whose union graphs are connected and do not verify the connectivity of the vertices in each graph. It allows to study graph sequences containing longer sequences (20 graphs) and larger graphs (5000 unique vertices) compared with (Inokuchi and Washio, 2008). However, relevant patterns are not actually "relevant" for many applications because extracted patterns are not necessarily connected graphs. For example, Fig. 2.7 (a) could represent a spatio-temporal database dealing with aquaculture monitoring. It is composed of two sequences where each vertex is assigned to an aquaculture pond and each edge represents the spatial relationship between ponds. One label of the vertex-labels set {Active, Semiactive, Abandoned} is assigned to each pond. It is interesting to understand how a set of adjacent ponds evolve over time, because adjacent ponds always interact with each other. However, a relevant subgraph subsequence pattern may not be interesting if only union graphs are connected while subgraphs are not connected. Fig. 2.7 (b) shows a frequent relevant and induced subgraph subsequence extracted from the database. Although the union graphs of d_1 and d_2 are both connected, the edge (1,2) is not connected in $g_1^{(2)}$, $g_1^{(3)}$ and $g_2^{(2)}$ and the edge (2,4) is not connected in $g_1^{(1)}$, $g_1^{(3)}$ and $g_2^{(1)}$.

In (Ozaki and Ohkawa, 2009), the authors develop an algorithm based on a levelwise strategy to discover correlated sequential subgraphs from a single sequence of labeled graphs i.e. frequent sequences of subgraphs whose components are correlated with each other. They propose a correlation criterion named (m, θ, k). m represents the size of subsequences, θ is the minimum threshold for the correlation (Tan *et al.*, 2002) which aims to verify a strong relationship between two components of a pattern and k is a positive integer accessing the maximum number of exceptional subgraphs (uncorrelated subgraph in the pattern with respect to the minimum correlation threshold θ).

Given a graph sequence $\mathcal{G}$, a minimum support threshold σ $(1/|\mathcal{G}| \leq \sigma \leq 1)$, a minimum correlation threshold θ $(0 \leq \theta \leq 1)$, a size of subgraphs of pattern $m \geq 1$ and a maximum exceptional subgraphs allowed $k \geq 1$, then the problem is to enumerate the complete set of successive correlated sequential subgraphs. The authors develop a levelwise algorithm CorSSS using hash tables to store sets of prefix and postfix trees.

In order to discover more correlated patterns, several measures are proposed in (Yun, 2007), such as sequential support-confidence (s-confidence) and sequential weight-confidence (w-confidence). S-confidence is the ratio of the minimum support of items within a given pattern to the maximum support of items within this pattern. It describes the s-affinity among items within pattern. A sequential pattern is a sequential s-affinity pattern if its s-confidence is no less than a minimum s-confidence. W-confidence is the ratio of the minimum weight of items within a given pattern to the maximum weight of items within this pattern. It reflects the w-affinity among items within sequence. A sequential pattern is a sequential w-affinity pattern if its w-confidence is no less than the minimum weight confidence. The authors (Yun, 2007) develop a new algorithm based on successive projections to extract weighted sequential patterns with similar support and/or weight. These measures give a balance between support measure and weight measure.

2.4 Dynamic plane subgraphs

In (Prado *et al.*, 2013), the authors are interested in studying dynamic plane graphs where each graph is a single sequence of labeled ordered graph constructed by defining the circular

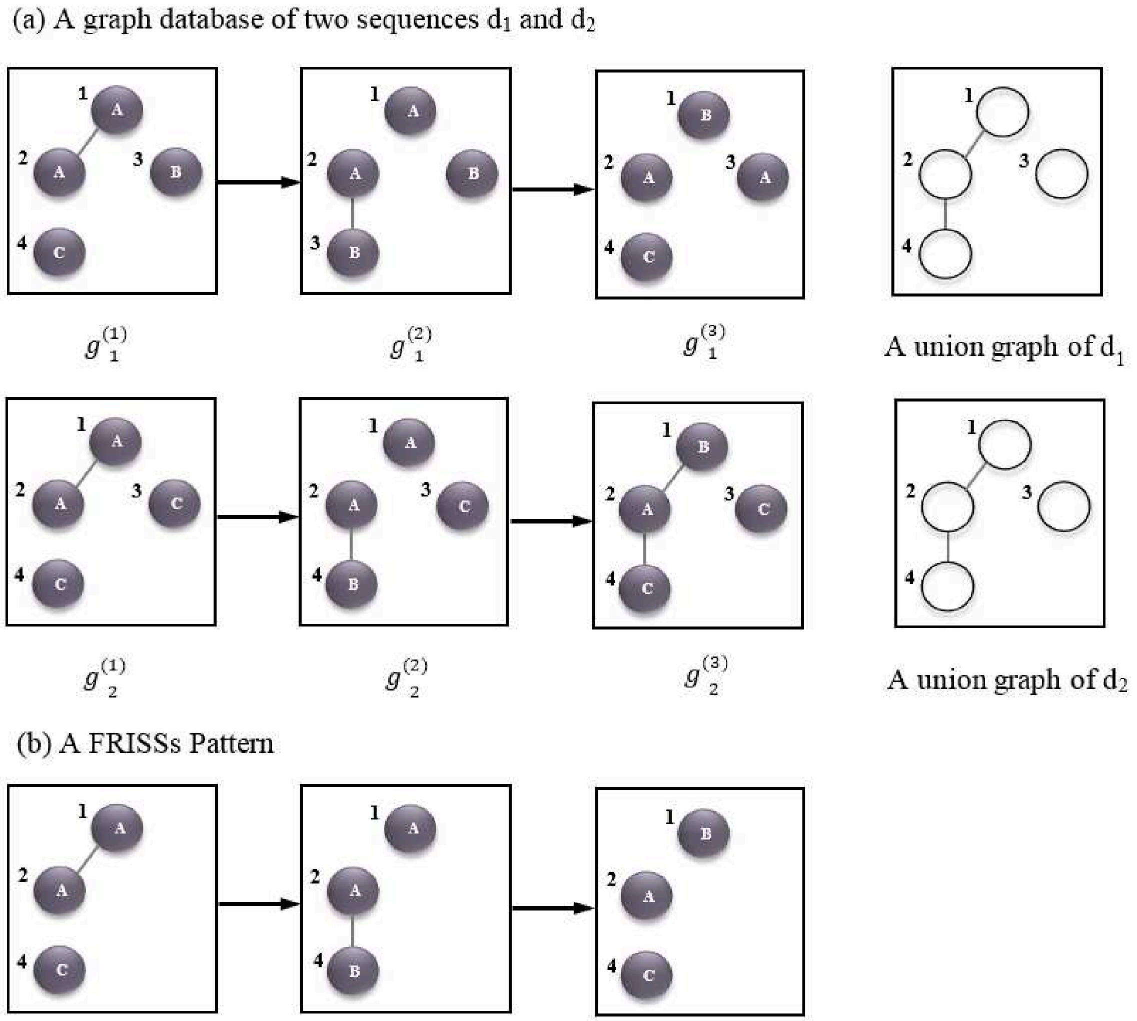


Figure 2.7 – (a) A graph database DB (b) a frequent relevant induced subgraph subsequence of DB

ordered list of neighbours of a vertex v. They aim to discover spatio-temporal pattern in such dynamic plane graphs. Given a dynamic graph $\mathcal{G} = \langle G_1, ..., G_{tmax} \rangle$ in which each vertex is related to spatial coordinates (x, y) and a plane graph P, the set of occurrences of P in $\mathcal{G}$ is defined as $Occ(P) = \{(i, f) | f\, is\, an\, occurrence\, of\, P\, in\, G_i\}$, where P is a plane subgraph isomorphism to G_i. Two occurrences are close if their spatial distance is lower than a threshold ϵ and their temporal distance is lower than a threshold τ. The connected component of occurrences of P is then defined as a spatiotemporal pattern based on P, where two occurrences are connected if they are close. As an example, in video applications, each video frame can be regarded as a plane graph in which each vertex is an object (a segmented region of the frame) associated with spatial information such as the barycenter of the object, and edges represent spatial relationships between vertices. Finding spatio-temporal patterns permit to track a given object in a video over time. Fig. 2.8 shows a video of four frames. As we can see, occurrences of 1, 2 and 2, 3 are spatially and temporally close, i.e. they appear in consecutive frames at similar positions. So a spatio-temporal pattern $\{1, 2, 3\}$ is extracted. Similarity, $\{5, 6, 7\}$ and $\{4\}$ are also spatio-temporal patterns. However, $\{1, 5\}$ is not a spatio-temporal pattern as the occurrence of 1 and 5 are quite far. The problem is then defined as extracting a complete set of spatio-temporal patterns with a frequency greater than a user threshold. For this purpose, the authors develop the algorithm DyPlagram based on a depth-first search strategy.

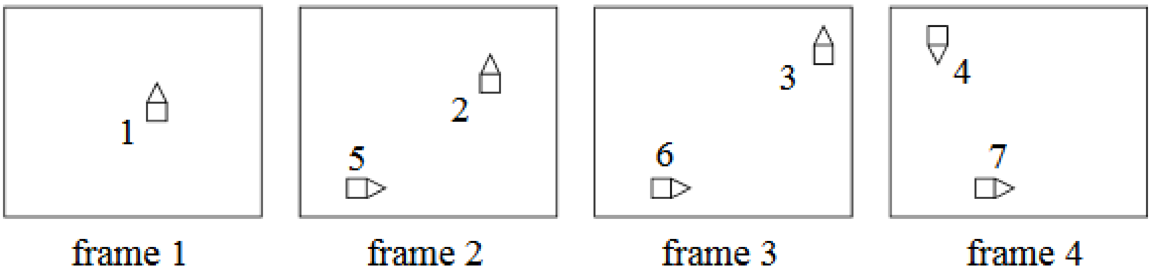


Figure 2.8 – (a) A graph database DB (b) a frequent relevant induced subgraph subsequence of DB

Compared with other mining approaches aimed at finding subclasses of graphs (Yan and Han, 2002; Inokuchi *et al.*, 2000; Nijssen and Kok, 2004), DyPlagram is more efficient in terms of time and memory for the following reasons. Firstly, finding subgraphs in a dynamic graph is a NP-complete problem while subgraph isomorphism tests for plane graphs is polynomial (Damiand *et al.*, 2009). Secondly, other algorithms extend a pattern by a single edge at a time, which generates huge amounts of extensions, while proposed patterns can only be extended with faces (a connected region of the plane which is bounded by a circuit of edges). In (Diot *et al.*, 2012), the authors extend their work by introducing additional spatio-temporal constraints and propose the algorithm DyPlagram_ST that takes in to account new constraints.

2.5 Periodic subgraphs

Many interesting patterns occurring regularly are often infrequent. To extract such regular behaviors, (Lahiri and Berger-Wolf, 2008) consider the problem of mining periodic subgraphs in dynamic labeled graph. Mining periodic subgraph patterns is interesting in many domains. For instance, in a dynamic graph which represents movements and interactions of animals, a pattern could describe their periodic behaviors such as seasonal association.

Given a sequence of labeled graphs $\mathcal{G} = \langle G_{t_1},\ G_{t_2}\ , ...,\ G_{t_{max}} \rangle$ over a set of time $\mathcal{T} = \{t_1, \ldots, t_{max}\}$. A periodic subgraph embedding (PSE) of a subgraph $C = (V, E, \lambda)$ in $\mathcal{G}$ is a maximal, ordered set of timesteps T where C is a subgraph of G_t and the gap between every two consecutive elements t_i and t_{i+1} in T is constant. Each periodic subgraph embedding (PSE) $C = (V, E, \lambda)$ is associated with a triplet (b, p, s) where b is the first timestep, p is the period and s is the support of PSE. The aim is then to mine the complete set of closed periodic subgraphs whose support is greater than a user defined threshold. The authors propose a single pass, polynomial time and space algorithm called PSEMiner to find periodic patterns. However, this algorithm is not efficient because the pattern tree is browsed in a breadth-first manner at every timestep. Thus, many unessential tree nodes are generated.

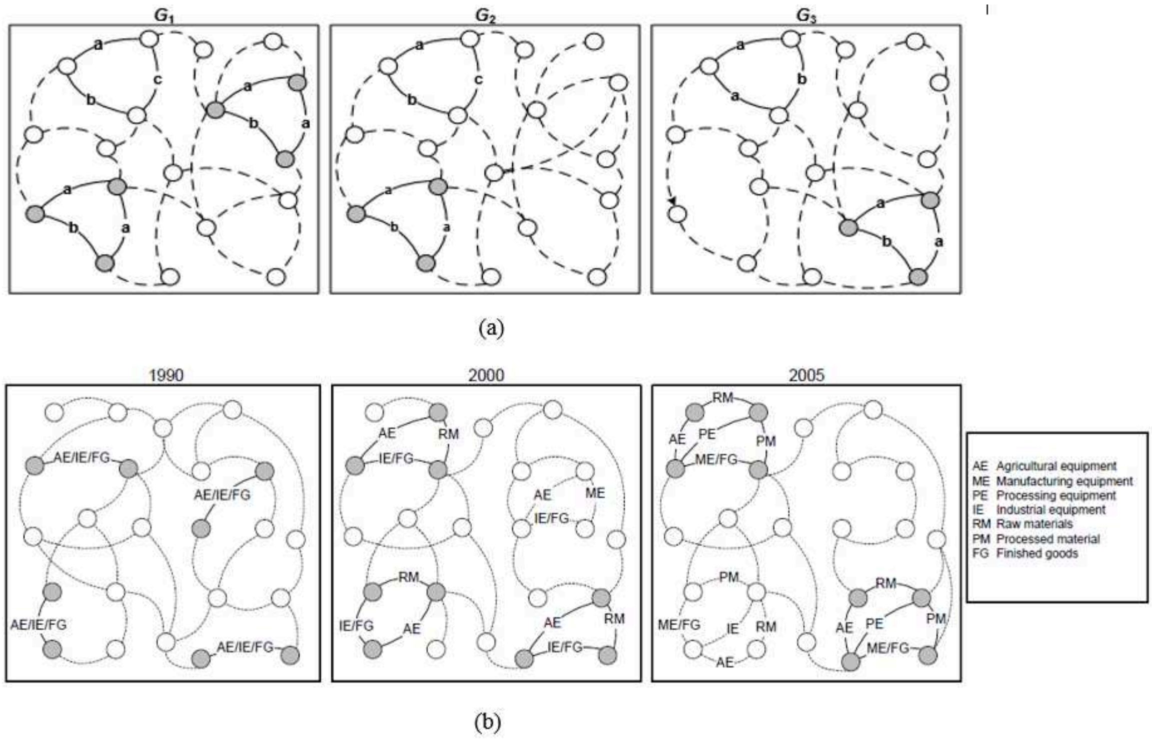
To solve this problem, (Apostolico *et al.*, 2011) propose a more efficient algorithm ListMiner, to mine periodic subgraph patterns. To avoid browsing the pattern tree, input graphs are partitioned by periodic value p of the form $\mathcal{G}_x^p = \{G_x, G_{x+p}, ..., G_{x+np}\}$, $x = 1, 2, ..., p-1$. They create p lists where each list node represents a unique periodic pattern. It allows to use previously computed intersections of graphs to compute following ones. This algorithm is more efficient in terms of execution time. However, it saves many redundant common interactions because new nodes are generated whenever interaction is changed within the graph over time. In summary, these two methods PSEMiner and ListMiner store separate graphs as long as one entity of a large graph is modified over time.

In (Halder *et al.*, 2017), the authors propose a super-graph (common vertices and edges of graphs) based periodic patterns mining algorithm, named SPPMiner, which improves the existing technics in both execution time and memory usage. Firstly, at each time interval t, an empty super-graph is initialized. Then it is updated with current graph entities (vertices and edges). Once entities stop to be periodic, they are removed from the super-graph. This super-graph becomes a periodic entity if it satisfies the minimum support threshold. Compared with previous algorithms, the main advantage of this approach is that only one maximal common pattern calculation is needed for each time interval p. Moreover, all common and uncommon pattern entities (vertices and edges) in dynamic graphs are stored only once.

2.6 Coevolving patterns in dynamic graph

In (Ahmed and Karypis, 2015a), authors define a new class of dynamic labeled graph patterns named coevolving relational motifs (CRMs). A relational motif is a subgraph that appears frequently in a single graph or several graphs. As shown in Fig. 2.9 (a), the frequent subgraph composed of the three shaded vertices, connected by labeled edges a, a and b, is a relational motif occurring four times (two times in G_1 and once in G_2 and G_3 respectively). Coevolving relational motifs (CRMs) are relational patterns that change in a consistent way (have at least one edge in common) through time. Fig. 2.9 (b) depicts an example of CRM. The first relational motif (M_1) composed of two shaded vertices and a labeled edge AE/IE/FG occurs four times in the first graph for 1990. Then three of these four motifs evolve in the same way by joining an additional vertex in the second relational motif M_2 in 2000. Finally, two of these three motifs evolve to M_3 composed of four shaded vertices and five labeled edges AE, RM, PE, PM and ME/FG in 2005. The sequence $< M_1$, M_2, $M_3 >$ is a CRM. The authors develop a depth-first search algorithm, named CRMminer, which allows to extract recurring sets of vertices whose relations (edges) change in a consistent way over time from a single dynamic labeled graph.

In (Ahmed and Karypis, 2015b), authors improved CRMminer by defining a new class of patterns, referred as coevolving induced relational motifs (CIRMs). More formally, a CIRM of length m is a tuple $\{N, < M_1, ..., M_m >\}$, where N is a set of vertices and each $M_j = (V_j, E_j)$ is an induced relational motif. A CIRM has to satisfy the following constraints:



(a)

(b)

Figure 2.9 – (a) A relational pattern (b) A coevolving relational motif, (Ahmed and Karypis, 2015a)

(1) it occurs at least ϕ times, (2) each occurrence uses a non-identical set of nodes, (3) $M_j \neq M_{j+1}$, and (4) $|N_j| \geq \beta|N|$ where $0 < \beta \leq 1$.

Given a dynamic network $\mathcal{G}$ containing T timesteps, a minimum support ϕ ($1 \leq \phi$), a minimum number of vertices k_{min} per CIRM and a minimum number of motifs m_{min} per CIRM, the problem is to enumerate all frequent coevolving induced relational motifs. To this aim, the authors develop the algorithm, CIRMiner, to extract all frequent coevolving induced relational motifs. CIRMiner is based on a depth-first search strategy. It is much more efficient than CRMminer because it generates only a small group (i.e., induced patterns) of all frequent CRMs. It results in a noticeable reduction in number of patterns and execution time.

2.7 Dynamic graphs as Boolean Tensors

Discovering closed patterns in ternary relations from a collection of graph sequences has received much attention. It provides insight into many real world applications. For instance, in a bicycle rental system, n-ary relation patterns permit to understand rider behaviors (riders' popular and active stations, popular cycling routes over different periods etc.). Three approaches were proposed to mine this type of patterns, namely CubeMiner (Ji *et al.*, 2006), Trias (Jaschke *et al.*, 2006), and Data-Peeler (Cerf *et al.*, 2008) (Cerf *et al.*, 2009a). Data-Peeler follows a depth-first exploration approach. It is more general than the two former algorithms because only Data-Peeler can deal with n-ary relations and mine patterns that satisfy a large class of piecewise anti-monotonic constraints. In (Cerf *et al.*, 2009b), the authors define Data-Peeler algorithm to discover δ-contiguous closed 3-cliques patterns, i.e. maximal sets of vertices densely connected that run along some nearly contiguous timestamped graphs. Such a pattern respects three constraints. (1) it is a clique (every two distinct vertices in the clique are adjacent), (2) it is almost contiguous (the clique is respected on almost consecutive timestamps) and (3) it is closed (the pattern is maximal, any subset of the pattern violates the connection constraint). More formally, we set $a_{t,v^1,v^2} = 1$, if there exists an edge between vertex v^1 and vertex v^2 at time t.

A δ-contiguous closed 3-clique is a triset $P = (T, V^1, V^2)$ where T is a set of timestamps, V is a set of vertices such that (1) P is connected and symmetric, i.e. $\forall(t, v^1, v^2) \in P$, $a_{t,v^1,v^2} = 1$ and $a_{t,v^2,v^1} = 1$; (2) P is δ-contiguous, i.e., $\forall t \in T, \exists t' \in T$ s.t. $|t - t'| < \delta$; (3) P is closed, i.e., $\nexists t \in \mathcal{T}$ and $\nexists v \in \mathcal{V} \mid (V^1 \cap V^2)$ s.t. $P \cup t$ or $P \cup v$ is connected.

2.8 Rules to describe the graph evolution

In (Holder and Cook, 2009), the authors study how a single graph structurally evolved over time. For this purpose, they specify graph rewriting rules that describe the evolution of two graphs. Fig. 2.10 (a) depicts graph rewriting rules between graph G_i and G_{i+1}. It includes removals (R_i) and additions (A_i) of edges between two graphs G_i and G_{i+1}. Then, a transformation rule which compresses rewriting rules and depicts structural changes (removals and additions) between graphs, is extracted. As shown in Fig. 2.10 (b), a transformation rule is simply defined as the common subgraph in removals and additions. (Holder and Cook, 2009) propose an algorithm to discover rewriting rules whose main challenge is the extraction of maximum common subgraphs between two graphs, which is a NP-complete task. However, as they use labeled graphs in this method, discovering maximum common subgraph becomes a quadratic problem.

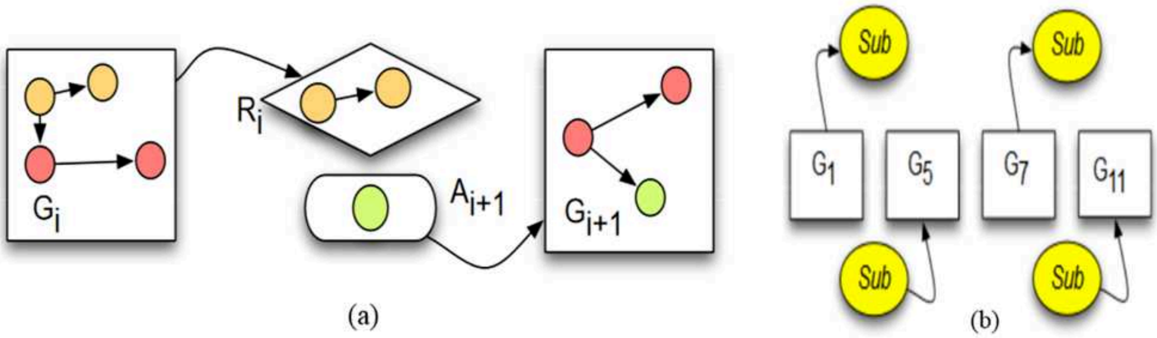
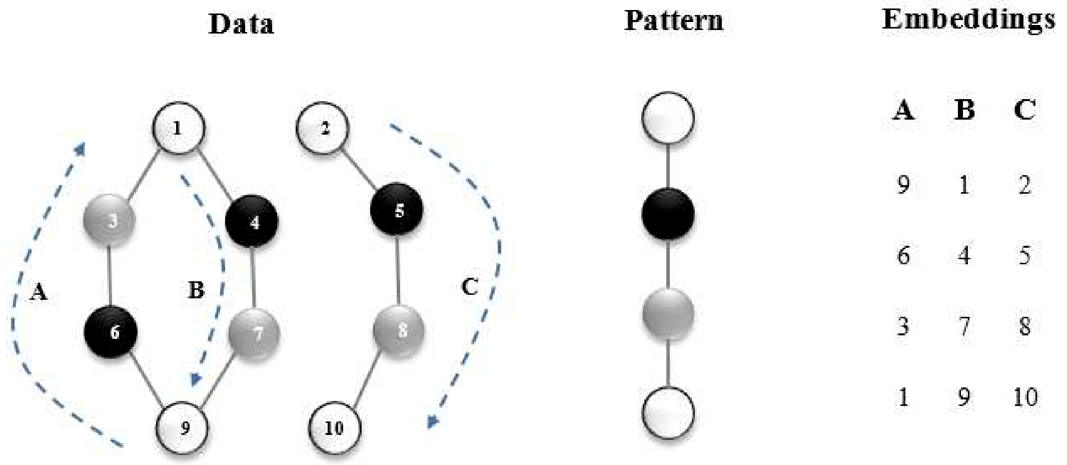


Figure 2.10 – (a) Graph rewriting rules between graph G_i and G_{i+1} (b) A transformation rule that compresses the graph rewriting rules (e.g., a subgraph is removed from G_i and then added in G_{i+1}). Notations: R_i, removals of edges between two graphs G_i and G_{i+1}, A_i, additions of edges between two graphs G_i and G_{i+1} (Holder and Cook, 2009)

In (Berlingerio *et al.*, 2009), the authors introduce another problem that consists in extracting graph evolution rules satisfying both a minimum support and a minimum confidence constraint. A graph evolution rule is composed of a rule *body* $\rightarrow$ *head* where body is a connected graph and head is a super-pattern of body. They develop an algorithm called GERM (Graph Evolution Rule Miner) based on a depth-first search strategy to mine all graph evolution rules satisfying user defined support and confidence thresholds. They use the minimum image based support measure proposed in (Bringmann and Nijssen, 2008). It depends on the number of unique nodes in the graph $G = (V_G, E_G, \lambda_G)$ which a node of the pattern $P = (V_P, E_P, \lambda_P)$ is mapped to. Fig. 2.11 depicts an example of minimum image based support. The support of the pattern is 2 although this pattern has three occurrences. That is because the two white vertices could only be mapped to the same vertices 1 and 8. The main advantage of this definition is to avoid a maximal independent set problem for each candidate pattern. A confidence measure is defined as the ratio of number of occurrences of head and body. It permits to calculate the likeness between steps of a graph evolution rule (head and body).



Embeddings

A	B	C
9	1	2
6	4	5
3	7	8
1	9	10

Figure 2.11 – Example of a pattern with three different occurrences

3. Pattern mining in dynamic attributed graph

Methods presented in the previous section aim to study a single dynamic labeled graph or a collection of dynamic labeled graphs. However, such graph representations are limited, because each vertex is described by a single label, i.e. a single information. However, in many applications, objects/vertices are characterised by several attributes. For example, in a co-authorship network, where vertices represent authors and edges depict the co-authorship between authors, vertices could be labeled by a set of attributes which represent the number of publications in different conferences or journals (instead of only one attribute: author's name). With additional information, we could analyze their research domain and their preferences. An aquaculture dataset (time series of satellite images) is another example. As already stated, with the development of remote sensing technology, we could obtain numerous vertex attributes, such as present/abscence of vegetation, presence/absence of water, presence/absence of aerator, etc. Thus, a dynamic attributed graph is proposed to model more complicated real-world phenomena where vertices are labeled by a set of attributes and both vertex attributes and edges could evolve over time. However, few methods have been proposed to mine such graphs. This task is complex because we have to consider both complexity of graph structure (e.g. connectivity, graph isomorphism etc.) and itemset complexity, which lead to a combinatorial explosion.

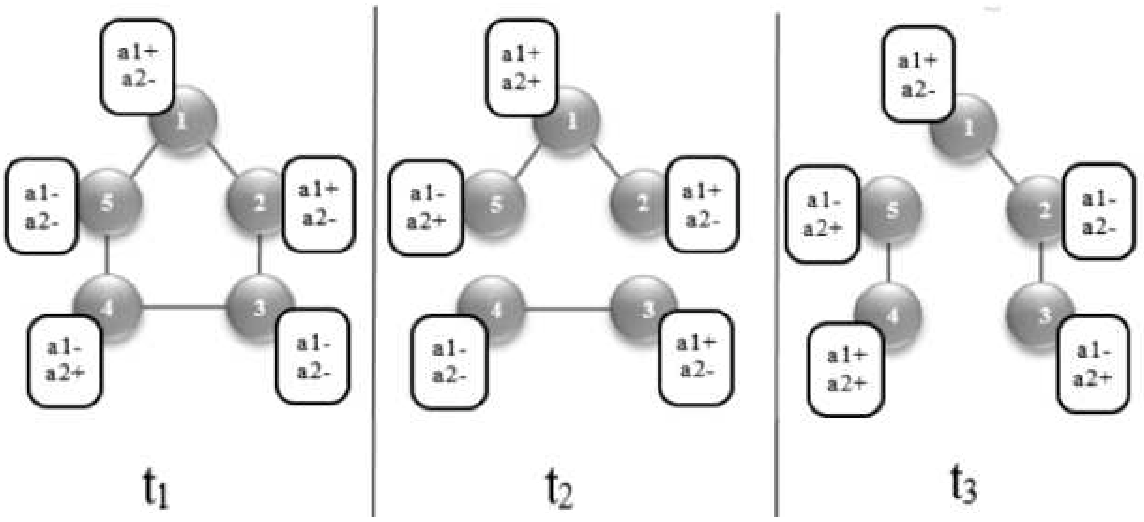


Figure 2.12 – Example of dynamic attributed graph

A dynamic attributed graph is a single sequence of attributed graphs $\mathcal{G} = \langle G_{t_1}, G_{t_2}, ..., G_{t_{max}} \rangle$ which represents the evolution of an attributed graph over a set of time $\mathcal{T} = \{t_1, \ldots, t_{max}\}$. As shown in Fig. 2.12, the set of vertices of $\mathcal{G}$ is denoted $\mathcal{V}$. Each vertex is described by a set of attributes A (numerical or categorical). Each attribute $a \in A$ is associated with a domain value $\mathcal{D}_a$. For each time $t \in T$, $G_t = (V_t, E_t, \lambda_t)$ is an attributed undirected graph where $V_t \subseteq \mathcal{V}$ is the set of vertices at time t, $E_t \subseteq V_t \times V_t$ is the set of edges at time t and $\lambda_t : V_t \rightarrow 2^{A\mathcal{D}}$ is a function that associates each vertex of V_t with a set of values $A\mathcal{D} = \bigcup_{a \in A}(a \times \mathcal{D}_a)$.

3.1 Triggering pattern mining

In (Kaytoue *et al.*, 2014), the authors define the triggering pattern problem which allows to find temporal relationships between vertex attributes and their topological properties (degree, betweeness, number of cliques etc.).

Given a dynamic attributed graph, let D be a set of descriptors (either vertex attributes or their topological properties), $S = \{+, -, \emptyset\}$, a set of symbols to denote increase, decrease and remain constant. A triggering pattern is defined as a sequence P $= < L, R >$ where L

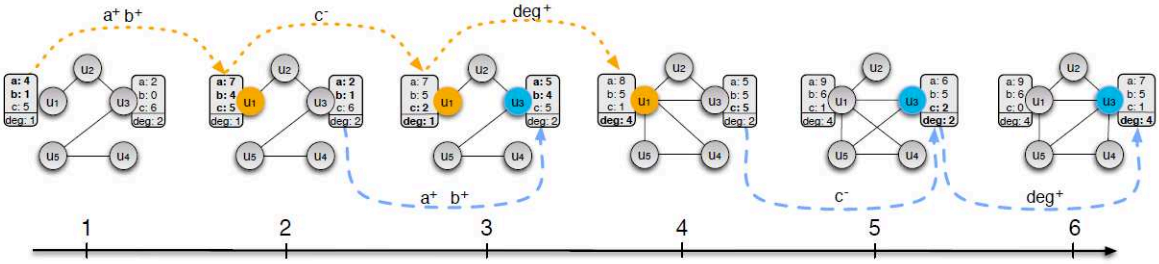


Figure 2.13 – A triggering pattern $< \{a^+, b^+\}, \{c^-\} \rightarrow \{deg^+\} >$ whose support equals 2 (orange line and blue line), (Kaytoue *et al.*, 2014)

is a sequence of descriptor variations sets ($L =< X_1, ..., X_k >$ with $X_j \subseteq (D \times S)$), and R is a single topological variation, $R \in (M \times S)$, where M is a set of topological attributes such as degree, closeness and betweenness etc. The authors define two interesting measures: the growth rate and the coverage.

Let $P =< L, R >$, Δ a set of all sequences representing evolution of each vertex and $\Delta^R \subseteq \Delta$ is the set of vertex descriptive sequences that contain R. The growth rate of P is given by: $GR(P, \triangle^R) {=} \frac{|SUPP(L, \triangle^R)|}{|\triangle^R|} \times \frac{|\triangle \backslash \triangle^R|}{SUPP(L, \triangle^R)}$ Coverage of a triggering pattern is defined as the set of vertices which support a pattern, i.e. $|COV(P, \triangle)|$. The problem is then to extract frequent triggering patterns, i.e., patterns that satisfy a minimum growth rate threshold minGR and a minimum coverage threshold minCov. Approaches for extracting triggering patterns have several possible applications. For example, Fig. 2.13 shows a social network whose vertices are users. Vertex attributes a, b and c describe number of updated blogs, positive opinions to other users and negative comments received from others respectively. For example, the sequence $< \{a^+, b^+\}, \{c^-\} \rightarrow \{deg^+\} >$ is supported by two vertices u_1 and u_3, it illustrates the fact that a blogger who updates blogs more often, gives more positive opinions to others and receives less negative comments, becomes often more popular.

The authors design an efficient algorithm TRIGAT to mine triggering patterns. First, TRIGAT generates all 1-item sequences satisfying coverage constraint. Then, it extends patterns using a pattern-growth strategy. The prefix sequences s can be extended by adding a single descriptor variation at the end of the sequence.

Extracted patterns allow to show the impacts of attributes variations on topological properties. However, this approach is limited, as it cannot consider the global graph structure (connectivity, diameter etc.) and every vertex is independently modeled as a sequence of itemsets composed of vertex attributes and topological properties (degree, betweeness, number of cliques etc.).

3.2 Cohesive co-evolution pattern mining

In (Desmier *et al.*, 2012), the authors define cohesive co-evolution patterns. Such patterns represent a set of vertices with the same attributes and a similar neighborhood over a set of timestamps (vertices and attributes were fixed). The input database is a single dynamic attributed graph $\mathcal{G} = \langle G_{t_1}, G_{t_2}, ..., G_{t_{max}} \rangle$ which represents the evolution of a graph over a set of time $\mathcal{T} = \{t_1, \dots, t_{max}\}$. The set of vertices of $\mathcal{G}$ is denoted $\mathcal{V}$. Each vertex is labeled by a set of attributes A (numerical or categorical). Each attribute $a \in A$ is associated with a domain value $\mathcal{D}_a = \{+, -, =\}$ which describes the evolution (increasing, decreasing and constant) of attribute values. For each time $t \in T$, $G_t = (V, E_t, \lambda_t)$ is an attributed undirected graph where $V \subseteq \mathcal{V}$ is the set of vertices, $E_t \subseteq V_t \times V_t$ is the set of edges at time t and and $\lambda_t : V_t \rightarrow 2^{A\mathbb{D}}$ is a function that associates each vertex of V_t with a set of values $A\mathbb{D} = \bigcup_{a \in A}(a \times \mathbb{D}_a)$.

Given a dynamic attributed graph, a cohesive co-evolution pattern is a triplet (N, T, P) where $N \subseteq V$, $T \subseteq \mathcal{T}$ is a set of not necessarily consecutive timestamps and P is a set of signed attributes, i.e. $P \subseteq A \times S$. This triplet must satisfy the following conditions: (1) each signed attribute $a^s \in P$ represents an attribute trend that has to be satisfied by every vertex $v \in V$ and at every timestep $t \in T$, (2) (N, T, P) is maximal: adding any vertex, any timestamp or any signed attribute leads to the violation of (1), (3) at each time $t \in T$, the vertices of the pattern have to be cohesive through the graph. Given a similarity threshold $\sigma \in [0, 1]$ and a similarity measure sim, a co-evolution pattern (N, T, P) is cohesive iff: $cohesive(N, T, P) \equiv \forall t \in T, \forall u, v \in N^2, sim(u, v, G_t) \geq \sigma$

Cohesiveness is important because it permits to consider the graph structure. For example, with a co-authorship network, this constraint allows to focus on the authors having co-author relationship. This constraint permits to assess vertex similarities by pairs, i.e. likeness of their neighbourhood. Any similarity measure can be considered. The authors choose Cosine (Tan, 2006) and Jaccard (Jaccard, 1912) measures considering only the direct neighbourhood of vertices. Another interesting measure named *volume* is defined to find interesting patterns. This measure evaluates the size of pattern as a three-dimensional volume: $volume((N, T, P)) = |N| \times |T| \times |P|$.

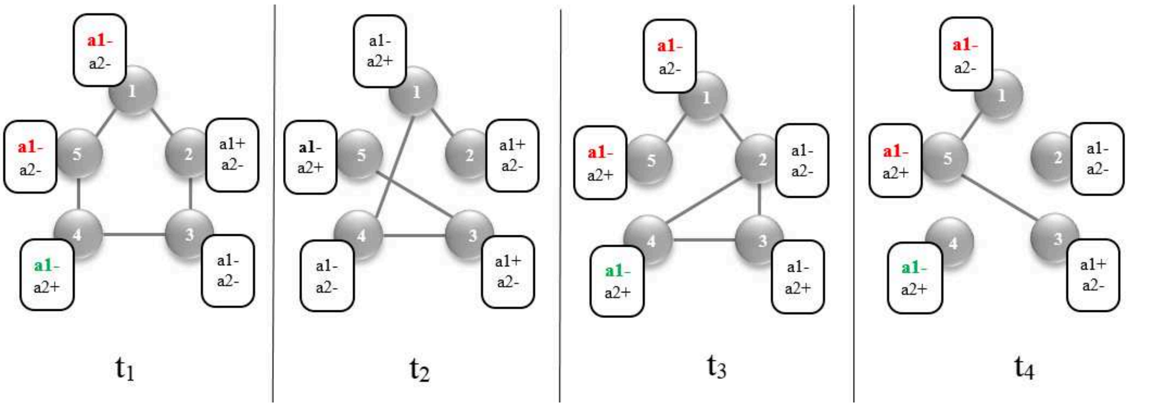


Figure 2.14 – Example of cohesive co-evolution pattern

Fig. 2.14 shows an example of cohesive co-evolution pattern (red bold) $(\{1, 5\}, \{t_1, t_3, t_4\}, \{a_1^-\})$, i.e. attribute a_1 of vertices 1 and 5 follows the same trend (decreasing) over three graphs G_1, G_2 and G_4. We can observe that attribute a_1 of vertex 4 (green bold) follows the same trend (decreasing) over G_1, G_2 and G_4. Yet, the pattern $(\{1, 4, 5\}, \{t_1, t_3, t_4\}, \{a_1^-\})$ is not cohesive, because vertex 4 does not have any neighbor in t_4.

Let $\mathcal{G}$ be a single dynamic attributed graph. Given a similarity measure sim, a minimum vertex similarity threshold σ and a minimum volume threshold θ, mining cohesive co-evolution patterns consists in finding the complete set of co-evolution patterns that have a volume no less than θ and that satisfy the cohesiveness constraint. The authors develop a novel algorithm to extract co-evolution patterns. They decompose the original search space into smaller pieces such that each portion can be independently computed in main memory. All valid triplets (N, T, P) are enumerated in a depth-first search manner. Finally, they union the three sets N, T and P extracted from each portion to generate the final co-evolution pattern.

The authors extended their work in (Desmier *et al.*, 2013) by integrating constraints on graph topology and on attribute values to extract maximal dynamic attributed subgraphs. Given a dynamic attributed graph $\mathcal{G} = \langle G_{t_1}, G_{t_2}, ..., G_{t_{max}} \rangle$, and a set of measures, the problem is to enumerate all trend sub-graphs (U, S, Ω) in a dynamic attributed graph (V, T, λ) where U is a subset of V, S is a subsequence of T and Ω is a subset of signed attributes λ. They define two more interesting measures compared to the former one (Desmier *et al.*, 2012): vertex specificity and trend relevancy. The first one allows to show the similarity between vertices outside the trend sub-graph and the ones inside this sub-graph. The second one allows to understand if attributes not belonging to Ω follow homogeneous trends on subgraph. The authors develop an algorithm based on a depth-first exploration strategy to find trend subgraph patterns.

These approaches are nevertheless limited because of the following reasons. Firstly, all vertex attributes have to follow the same trend over time. They do not consider patterns whose vertices and signed attributes follow different trend over time. Let us consider an airline network. If we want to study the impact of hurricanes on cancelled flight, an extracted co-evolution pattern could depict an increasing trend of cancelled flights when hurricane come. However, it is impossible to know how cancelled flights evolve in the following time when hurricane becomes weaker or disappears. Secondly, vertices are fixed, whereas, in many applications, vertices also evolve over time. For example, in an aquaculture dataset, a vertex (aquaculture pond) could be divided into several vertices and several vertices could be merged into one vertex over time.

Chapter 3

Contributions

Contents

1. Mining recurrent patterns in a dynamic attributed graph

As discussed in the state of the art, sequences and graphs become omnipresent models for analyzing data. Recently a richer model of graph, named a dynamic attributed graph, has received more attention. It can be used to model various real world datasets. For instance, an aquaculture dataset composed of satellite images time series can be modeled as a dynamic attributed graph. Vertices are objects (ponds) detected in the images, edges represent spatial relationships between objects and vertex attributes represent the characteristics such as presence of vegetation, activity, aerator etc in ponds. Another example is a social bookmarking system, where vertices represent users and edges describe mutual fan relationships. Users are characterized by bookmarks they shared for different categories (music, games, politics, etc.). Analyzing a dynamic attributed graph is interesting because of two main reasons. Firstly, this data structure permits to model more interesting and complex real world phenomena as vertices are labeled by a set of attributes instead of a single attribute. Secondly, not only the graph structure (edges) but also vertices and their attributes could evolve over time. Here, we extend the definition of a dynamic attributed graph which was initially proposed in (Desmier *et al.*, 2012) where all vertices are fixed. Because in many real world datasets, vertices may evolve over time (e.g. appear or disapper). For example, a vertex (e.g. an aquaculture pond) could be divided into several vertices, and several vertices could be merged into one vertex over time. To the best of our knowledge, few methods have been proposed to mine a dynamic attributed graph because it is a difficult task due to the complexity of such graph structure and the important number of attribute combinations. In (Desmier *et al.*, 2012), the authors mined cohesive co-evolutions in a dynamic attributed graph. These patterns represent a set of vertices with same values for a subset of attributes and a similar neighborhood over a set of timestamps (vertices and attributes were fixed). In our work, we define a more general pattern domain, called recurrent pattern, which describe recurrent evolutions in a dynamic attributed graph. It enables to capture not only vertices having same attribute values for periods of time such as in (Desmier *et al.*, 2012; Desmier *et al.*, 2013), but also to capture evolutions of attribute values and vertices over time. The patterns consist in connected subgraph sequences satisfying topological, frequency and non-redundancy constraints in the input data.

This chapter is organized as follows. In section 3.1, we introduce a dynamic attributed graph. Then we define our pattern domain and several constraints which allow users to filter interesting patterns. In section 3.2, we develop an original algorithm, called RPminer, based on graph intersections and a progressive extension of patterns over time. Section 3.3 reports experiments performed on artificial and real-world data that demonstrate the efficiency of the algorithm, its generality and interest of extracted patterns.

1.1 Dynamic attributed graph

The input database is a single dynamic attributed graph $\mathcal{G} = \langle G_{t_1}, G_{t_2}, ..., G_{t_{max}} \rangle$ which represents the evolution of a graph over a set of time $\mathcal{T} = \{t_1, \ldots, t_{max}\}$. The set of vertices of $\mathcal{G}$ is denoted $\mathcal{V}$. Each vertex is labeled by a set of attributes A (numerical or categorical). Each attribute $a \in A$ is associated with a domain value $\mathbb{D}_a$. For each time $t \in T$, $G_t = (V_t, E_t, \lambda_t)$ is an attributed undirected graph where: (1) $V_t \subseteq \mathcal{V}$ is the set of vertices at time t, (2) $E_t \subseteq V_t \times V_t$ is the set of edges at time t and (iii) $\lambda_t : V_t \rightarrow 2^{A\mathbb{D}}$ is a function that associates each vertex of V_t with a set of values $A\mathbb{D} = \bigcup_{a \in A}(a \times \mathbb{D}_a)$. Fig. 3.1 presents an example of dynamic attributed graph with $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$, $\mathcal{T} = \{T_1, T_2, T_3, T_4, T_5\}$ and $\mathcal{A} = \{a_1, a_2\}$. The value of the attribute a at time t for vertex v is denoted as $G_t(v, a)$. For instance, $G_1(v_5, a_1) = 5$.

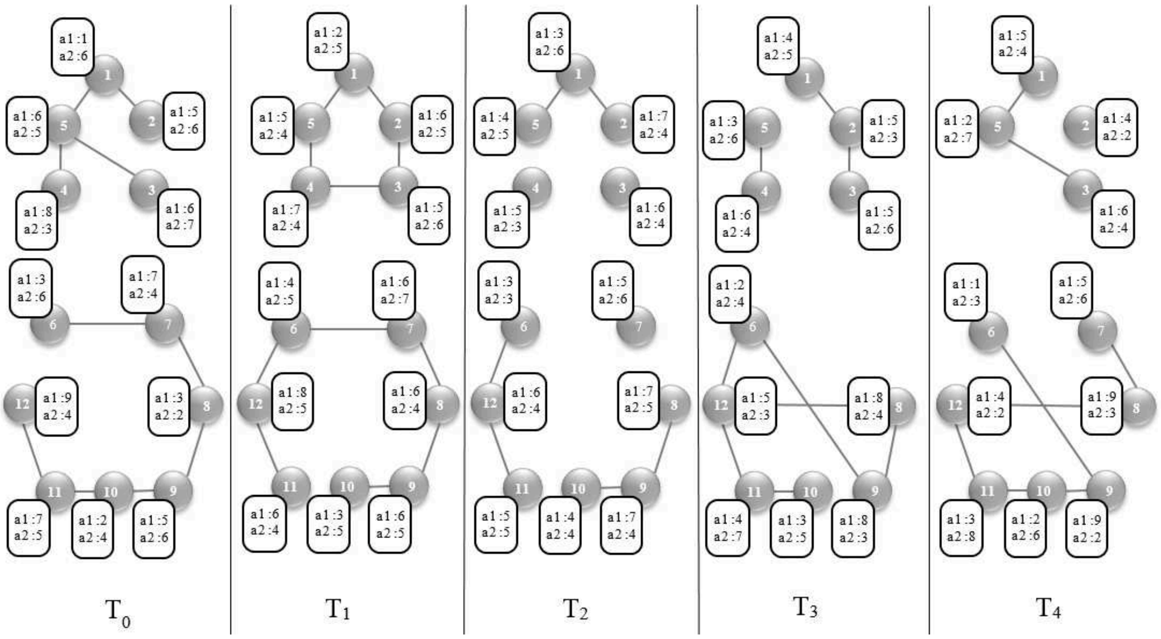


Figure 3.1 – An example of dynamic attributed graph $\mathcal{G}$

In this manuscript, in addition to numeric and categorical vertex values, we also study the trends of attributes (increasing, decreasing and constant), i.e., evolutions of the vertex attribute values between two consecutive timestamps T_i and T_{i+1}. Fig. 3.2 depicts the dynamic attributed graph after pre-processing. Edges for each time t_i are the edges of the timestamp T_{i+1}, and the trend is the evolution of attribute values between T_i and T_{i+1}. There is $|\mathcal{T}| - 1$ time steps. In the following, we consider $\mathbb{D}_a = \{+, -, =\}$ in order to simplify the given examples.

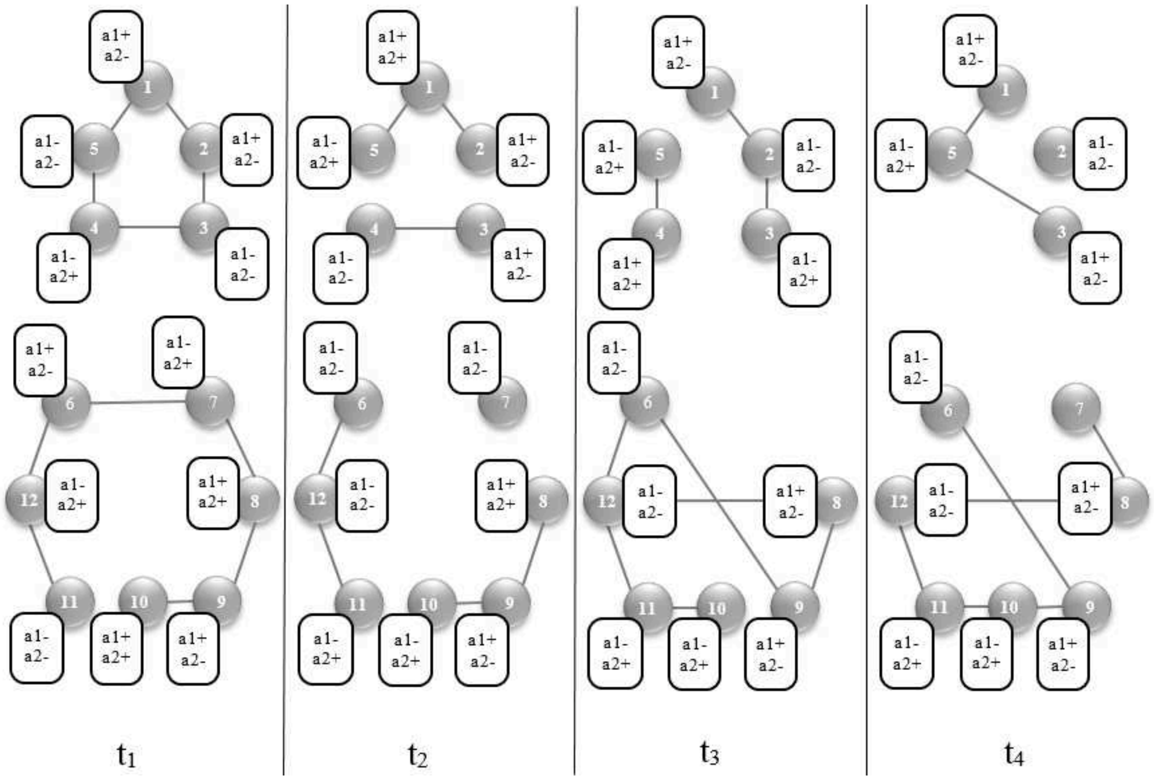


Figure 3.2 – Toy example: from values to trends

1.2 A new pattern domain and its constraints

1.2.1 Recurrent evolutions of vertices

Let (V, λ) be a subset of attributed vertices of $\mathcal{G}$ with $V \subseteq \mathcal{V}$ and $\lambda : V \rightarrow 2^{\mathcal{A}\mathbb{D}}$. (V, λ) can be considered as an attributed graph without edges. The definition of attributed subgraph presented in the previous section can be easily restrained to a set of attributed vertices. We then have $(V', \lambda') \sqsubseteq (V, \lambda)$, iff $V' \subseteq V$ and $\forall v' \in V' : \lambda'(v') \subseteq \lambda(v')$. To facilitate the reading of examples, (V, λ) can also be denoted $(v_1 : \lambda(v_1) \mid v_2 : \lambda(v_2) \mid ...)$, $\forall v_1, v_2 ... \in V$. As shown in Fig. 3.2, $(1 : a_1 + a_2 - \mid 2 : a_1 + a_2 - \mid 3 : a_1 - a_2 - \mid 4 : a_1 - a_2 + \mid 5 : a_1 - a_2 -)$ is a set of attributed vertices at time t_1.

An evolution of a subset of vertices of $\mathcal{G}$ starting at time $t \in \mathcal{T}$ is a sequence $S = \langle (V'_1, \lambda'_1)$ $(V'_2, \lambda'_2) \dots (V'_k, \lambda'_k) \rangle$, such as $\forall i \in \{1, 2, ..., k\}$, $\exists E'_i \subseteq E_{t+i-1}$, $(V'_i, E'_i, \lambda'_i) \sqsubseteq G_{t+i-1}$. For example, in Fig. 3.2, $\langle (1 : a_1 + a_2 - \mid 2 : a_1 + a_2 - \mid 3 : a_1 - a_2 - \mid 4 : a_1 - a_2 + \mid 5 : a_1 - a_2 -)$ $\langle (1 : a_1 + a_2 + \mid 2 : a_1 + a_2 - \mid 5 : a_1 - a_2 +) \rangle$ is an evolution starting at time t_1. $\langle (1 : a_1 + a_2 + \mid 2 : a_1 + a_2 - \mid 5 : a_1 - a_2 +)$ $(1 : a_1 + a_2 - \mid 2 : a_1 - a_2 - \mid 3 : a_1 - a_2 +)$ $(1 : a_1 + a_2 - \mid 3 : a_1 + a_2 - \mid 5 : a_1 - a_2 +) \rangle$ is an evolution starting at time t_2.

Let $T_P = \{t_{i_1}, ..., t_{i_m}\}$ be a set of times associated with the evolution $S_P = \langle (V'_1, \lambda'_1)$ $(V'_2, \lambda'_2) \dots (V'_k, \lambda'_k) \rangle$. A recurrent evolution of a subset of vertices of $\mathcal{G}$ starting at times T_P

according to the sequence S_P, is denoted $P = (S_P, T_P)$. In this case, the size of P is k. In Fig. 3.2, $(\langle(1:a_1+ \mid 2:a_1+a_2- \mid 5:a_1-)(1:a_1+ \mid 2:a_2-)\rangle, \{t_1, t_2\})$ is an example of recurrent pattern starting at times t_1 and t_2. This pattern represents a connected subgraph composed of v_1, v_2 and v_5 followed by a connected subgraph composed of v_1 and v_2. This pattern appears two times in the database, so its frequency is 2.
A relation of specialization/generalization can be defined on this pattern domain. Let $P1 = (\langle(V'_1, \lambda'_1)\ (V'_2, \lambda'_2) \ldots (V'_k, \lambda'_k)\rangle, T_{P1})$ and $P2 = (\langle(V''_1, \lambda''_1)\ (V''_2, \lambda''_2) \ldots (V''_l, \lambda''_l)\rangle, T_{P2})$ be two patterns representing two recurrent evolutions of $\mathcal{G}$. $P1$ is a recurrent evolution more general (resp. more specific) than $P2$, denoted $P1 \preceq P2$ (resp. $P1 \succeq P2$), if there exists $j \in \{0, ..., l-k\}$, such as $\forall i \in \{1, ..., k\}$, $(V'_i, \lambda'_i) \sqsubseteq (V''_{i+j}, \lambda''_{i+j})$. In Fig. 3.2, $(\langle 1:a_1+ \mid 2:a_1+a_2- \mid 5:a_1-)(1:a_1+ \mid 2:a_2-)\rangle, \{t_1, t_2\})$ is a recurrent pattern more specific than $(\langle(1:a_1+ \mid 2:a_1+a_2-)(1:a_1+)\rangle, \{t_1, t_2\})$.

1.2.2 Interesting measures and constraints

In this subsection, we define several measures and constraints to filter interesting patterns.

Firstly, we consider the graph structure. We propose three constraints taking into account the graph structure, i.e. connectivity of vertices, cohesiveness and volume of extracted patterns. In addition, we consider two temporal constraints: temporal continuity and gap. Finally, we consider the "classical" constraints such as frequency and non-redundancy.

Connectivity. In a graph, vertices often represent individuals/objects, and edges represent relationships between these individuals/objects. Integration of a connectivity constraint between vertices during pattern mining enables to focus on related objects. Let us consider for instance, in a dataset dealing with aquaculture dataset. A set of vertex-labels {Active/InActive, WithVegetation/WithoutVegetation} is assigned to each pond. It is interesting to understand how a set of adjacent ponds evolve over time, because adjacent ponds may interact with each other (for example, virus could spread from one to its neighbor, activities of adjacent ponds could be influenced by the same forest, mangrove, river or residential area etc.). $P = (\langle(V'_1, \lambda'_1)\ (V'_2, \lambda'_2) \ldots (V'_k, \lambda'_k)\rangle, T_P)$ is an evolution of connected vertices in $\mathcal{G}$ if $\forall t \in T_P$, $\forall i \in \{1, 2, ..., k\}$, $\exists E'_i \subseteq E_{t+i-1}$, $(V'_i, E'_i, \lambda'_i) \sqsubseteq_{conn} G_{t+i-1}$. In Fig. 3.2, $(\langle(1:a_1+ \mid 2:a_1+a_2- \mid 5:a_1-)(1:a_1+ \mid 2:a_2-)\rangle, \{t_1, t_2\})$ is an evolution of connected vertices.

Cohesiveness. This constraint (Desmier *et al.*, 2012) esures that the neighborhood of pairs of pattern vertices is cohesive. It calculates the similarity of the neighborhood or the neighborhood structure to extract a set of vertices which are closely related. For instance, in a DBLP dataset, this constraint permits to depict close working relationships between authors. Given a minimum similarity threshold $minsim \in [0, 1]$ and a similarity measure, a pattern $P = (\langle(V'_1, \lambda'_1)\ (V'_2, \lambda'_2) \ldots (V'_k, \lambda'_k)\rangle, T_P)$ is cohesive if $\forall v \in V'_i$, with $1 \leq i \leq k, \exists u \in V'_i$, such as $sim(v, u, V'_i) \geq minsim$

Here, any similarity measure can be used to discover patterns with different graph structures. We propose to use Cosine (Tan, 2006) and Jaccard (Jaccard, 1912) similarities which consider only the similarity of direct vertex neighborhood. Let $N(u)$ be the adjacent neighborhood of u.

$$Cosine(u, v) = \left(\frac{|(N(u)) \cap (N(v))|}{\sqrt{|N(u)| \times |N(v)|}} \right)$$

$$Jaccard(u, v) = \left(\frac{|(N(u)) \cap (N(v))|}{|N(u)| \cup |N(v)|} \right)$$

Volume. Volume is another measure commonly applied in the context of graph mining. It is defined as the number of vertices of a graph. It can represent, for instance, the size of a community in a social network. Let $vol(P) = min_{\forall i \in \{1 \dots k\}}(|V_i'|)$ be the volume of pattern $P = (\langle (V_1', \lambda_1') \dots (V_k', \lambda_k') \rangle, T_P)$. P is a sufficiently voluminous pattern iff $vol(P) \geq minvol$, where $minvol$ is a user-defined threshold. For example, pattern $(\langle (1 : a_1 + \mid 2 : a_1 + a_2 - \mid 5 : a_1 -)(1 : a_1 + \mid 2 : a_2 -) \rangle, \{t_1, t_2\})$ has a volume of 2.

Temporal continuity. By default, an evolution may include entirely different vertices at each step. In other words, if $P = (\langle (V_1', \lambda_1')\ (V_2', \lambda_2') \dots (V_k', \lambda_k') \rangle, T_P)$, then it is possible to have $\bigcap_{\forall i \in 1 \dots k} V_i' = \emptyset$. Interpreting such evolutions can be difficult for end users because there is actually no direct relation between individuals/objects (represented by vertices) at different timestamps. We propose a new constraint to target patterns which describe evolutions around a common core of individuals. Such a constraint allows to follow evolutions of a number of vertices over time while taking into account neighboring vertices (directly or indirectly). Let $P = (\langle (V_1', \lambda_1')\ (V_2', \lambda_2') \dots (V_k', \lambda_k') \rangle, T_P)$ be a pattern. Let $com(P) = |\bigcap_{\forall i \in 1 \dots k} V_i'|$ be the number of vertices occurring at all times in T_P. P is a continuous pattern over time iff $com(P) \geq mincom$, where $mincom$ is a user-defined threshold. For instrance, pattern $(\langle (1 : a_1 + \mid 2 : a_1 + a_2 - \mid 5 : a_1 -)(1 : a_1 + \mid 2 : a_2 -) \rangle, \{t_1, t_2\})$ has two common vertices at t_1 and t_2, i.e. $com(P) = 2$ while $com(P) = 0$ for the pattern $(\langle (1 : a_1 + \mid 2 : a_1 + a_2 - \mid 5 : a_1 -)(6 : a_1 - a_2 - \mid 11 : a_1 - a_2 + \mid 12 : a_1 - a_2 -) \rangle, \{t_1, t_2\})$.

Gap. Gap is a measure commonly applied in the context of sequence mining (Fournier-Viger *et al.*, 2008). It is defined as the time interval allowed between every two successive subgraphs of a recurrent pattern. It permits to study short term as well as long term evolutions. For example, if we study a DBLP co-authorship dataset, with gap=5 years, we could extract patterns describing general evolutions of authors, i.e., author A and author B firstly work with C and D, then in the following 5 years with F and G. While if we set gap to 2 years, we will get more specific evolutions over a smaller period of time: in the first 2 years A and B worked with authors C, D and E. Then, in the following 2 years, they worked with C, D, F and G. Next, they worked with C, F and G, and then with F, G and H, Finally they worked with F, G, H and I. Let gap(P)=extractgap, gap is a user-defined threshold. For example, given gap=2, $(\langle (6 : a_2 - \mid 11 : a_1 - \mid 12 : a_1 -)(6 : a_1 - a_2 - \mid 9 : a_1 + a_2 -) \rangle, \{t_1, t_2\})$ (extracted from Fig. 3.2) depicts a long term evolution in which time interval of every two consecutive graphs equals to 2.

Frequency. Minimum frequency is one of the most widely used constraints. It aims to filter patterns which occur more than a minimum number of times. It is commonly applied when a database is a collection of transactions. However, defining a frequency constraint is generally more challenging in a single graph context (Fiedler and Borgelt, 2007; Nijssen and Bringmann, 2008), mainly because of the presence of embedded overlappings. Nevertheless, frequency is easy to calculate in our case because of the nature of the extracted patterns. Indeed, the frequency of a pattern is simply the number of times at which a given evolution begins. It represents the number of recurrences of this evolution. Let $P = (S_P, T_P)$ be a pattern. Frequency of P is $sup(P) = |T_P|$. Consequently, P is a frequent evolution iff $sup(P) \geq minsup$, where $minsup$ is a user-defined threshold. For example, in Fig. 3.2, the frequency of $(\langle(6 : a_2- \mid 11 : a_1- \mid 12 : a_1-)(11 : a_1 - a_2+ \mid 12 : a_1 - a_2-)\rangle, \{t_1, t_2, t_3\})$ is 3 since it begins at t_1, t_2 and t_3.

Non-redundancy. A huge number of patterns can be extracted. However, some of these patterns may contain redundant information. For example, if two patterns $P1 = (S_{P1}, T_{P1})$ and $P2 = (S_{P2}, T_{P2})$ are such that $P1 \preceq P2$ and $T_{P1} = T_{P2}$, then it is not necessary to keep $P1$. Indeed, the sequence of attributed vertices of $P1$ is present in $P2$ and the two patterns occur exactly at the same times. The non-redundancy constraint is close to the notion of closure that has been applied to a large number of pattern domains (e.g. itemsets, sequences, trees) (Yan *et al.*, 2003; Huang *et al.*, 2006; Gomariz *et al.*, 2013; Wang *et al.*, 2007).
More formally, let Sol be a set of non-redundant pattern solutions. Let $P1 = (S_{P1}, T_{P1})$ and $P2 = (S_{P2}, T_{P2})$ be two recurrent patterns. If $P1 \in Sol$ then $\nexists P2 \in Sol$ such as $P1 \prec P2$ and $T_{P1} = T_{P2}$. In Fig. 3.2, $(\langle(1 : a_1+ \mid 2 : a_1 + a_2-)(1 : a_1+ \mid 2 : a_2-)\rangle, \{t_1, t_2\})$ is a redundant evolution with respect to $(\langle(1 : a_1+ \mid 2 : a_1+a_2- \mid 5 : a_1-)(1 : a_1+ \mid 2 : a_2-)\rangle, \{t_1, t_2\})$.

1.2.3 Problem setting

Given a dynamic attributed graph $\mathcal{G}$, the problem is to enumerate the complete set of recurrent evolutions in $\mathcal{G}$, denoted Sol, such that $\forall P \in Sol$: 1) vertices of P are connected or cohesive at each time (i.e. $cosine(P) \geq mincos$); 2) P is sufficiently voluminous (i.e. $vol(P) \geq minvol$); 3) P is frequent (i.e. $sup(P) \geq minsup$); 4) P is not redundant in Sol; 5) P is centered around a core of vertices sufficiently large (i.e. $com(P) \geq mincom$); and 6) the interval between every two successive subgraphs equals to gap, where $mincos$, $minsup$, $minvol$, $mincom$ and gap are user-defined thresholds. Intuitively, recurrent patterns are sequences of connected subgraphs which represent recurring evolutions of related subsets of nodes w.r.t. their attributes.

2. Algorithm

In this section, we introduce an enumeration strategy to extract recurrent patterns satisfying constraints stated above. Unlike a number of pattern mining algorithms, our approach is not based on a generate-test strategy (where candidate patterns are generated, tested and then combined). It performs neither a breadth-first nor a depth-first search. It is not based on a projection strategy either (such as *PrefixSpan*). Instead, our method is an iterative approach based on successive intersections and extensions of connected components occurring over time. As shown in Fig. 3.3, in each iteration, size-1 fragments are generated by processing time combinations T_i^k containing t_i, where $minsup \leq k \leq |\mathcal{T}|$ and then they are progressively combined to generate solutions. We thus get a set of solutions of different sizes at each iteration (at each time). The main advantage of this approach is to avoid generating a large number of patterns which do not satisfy the constraints. In the following subsection, we introduce the notion of intersection between attributed graphs and explain its interest w.r.t. our pattern mining problem.

2.1 Intersection of attributed graphs

Intersections of graphs permit to bring out two properties.

Intersection and frequency. Let us consider two times $i, j \in \mathcal{T}$. The intersection of two attributed graphs $G_i = (V_i, E_i, \lambda_i)$ and $G_j = (V_j, E_j, \lambda_j)$, $\forall G_i, G_j \in \mathcal{G}$ denoted by $G_i \sqcap G_j$, is an attributed graph $G = (V, E, \lambda)$ such as $V = V_i \cap V_j$, $E = E_i \cap E_j$, $\forall v \in V$, $\lambda(v) = \lambda_i(v) \cap \lambda_j(v)$. The result is a subgraph composed of vertices, edges and attribute values common to the two initial graphs. We can notice that every subgraph of G occurs at least two times in $\mathcal{G}$. In Fig. 3.4, let us consider only a part of $\mathcal{G}$ for clarity of presentation. The subgraph $c \sqsubset G_1 \sqcap G_3$ occurs at least 2 times (at t_1 and t_3).
This definition can be generalized to the intersection of k graphs, with $k \in \{2, 3, ...|\mathcal{T}|\}$. Let $T^k \subseteq \mathcal{T}$ be a subset of times of $\mathcal{G}$ such that $|T^k| = k$. The graphs intersection in $\mathcal{G}$ at the k times in T^k, denoted by $\underset{i \in T^k}{\sqcap} G_i$, is a graph $G = (V, E, \lambda)$, with $V = \underset{i \in T^k}{\cap} V_i$, $E = \underset{i \in T^k}{\cap} E_i$, $\forall v \in V$, $\lambda(v) = \underset{i \in T^k}{\cap} \lambda_i(v)$. The minimum frequency in $\mathcal{G}$ of all subgraphs of $\underset{i \in T^k}{\sqcap} G_i$ is k. Consequently, all patterns constructed from intersection of $minsup$ graphs of $\mathcal{G}$ will satisfy the minimum frequency constraint.

Intersection and non-redundancy. Intersections also have other properties. Let us study in particular connected components (i.e. maximal connected subgraphs) resulting from intersection of several graphs. We denote $\mathbb{C}_{i \sqcap j}$ the set of connected components obtained after intersection of graphs in $\mathcal{G}$ at times i and j, i.e. $G_i \sqcap G_j$. More formally, $\mathbb{C}_{i \sqcap j} = \{(V, E, \lambda) \mid (V, E, \lambda) \sqsubseteq_{conn} G_i \sqcap G_j$ and $\nexists (V', E', \lambda'), (V, E, \lambda) \sqsubset (V', E', \lambda')$ s.t. $(V', E', \lambda') \sqsubseteq_{conn} G_i \sqcap$

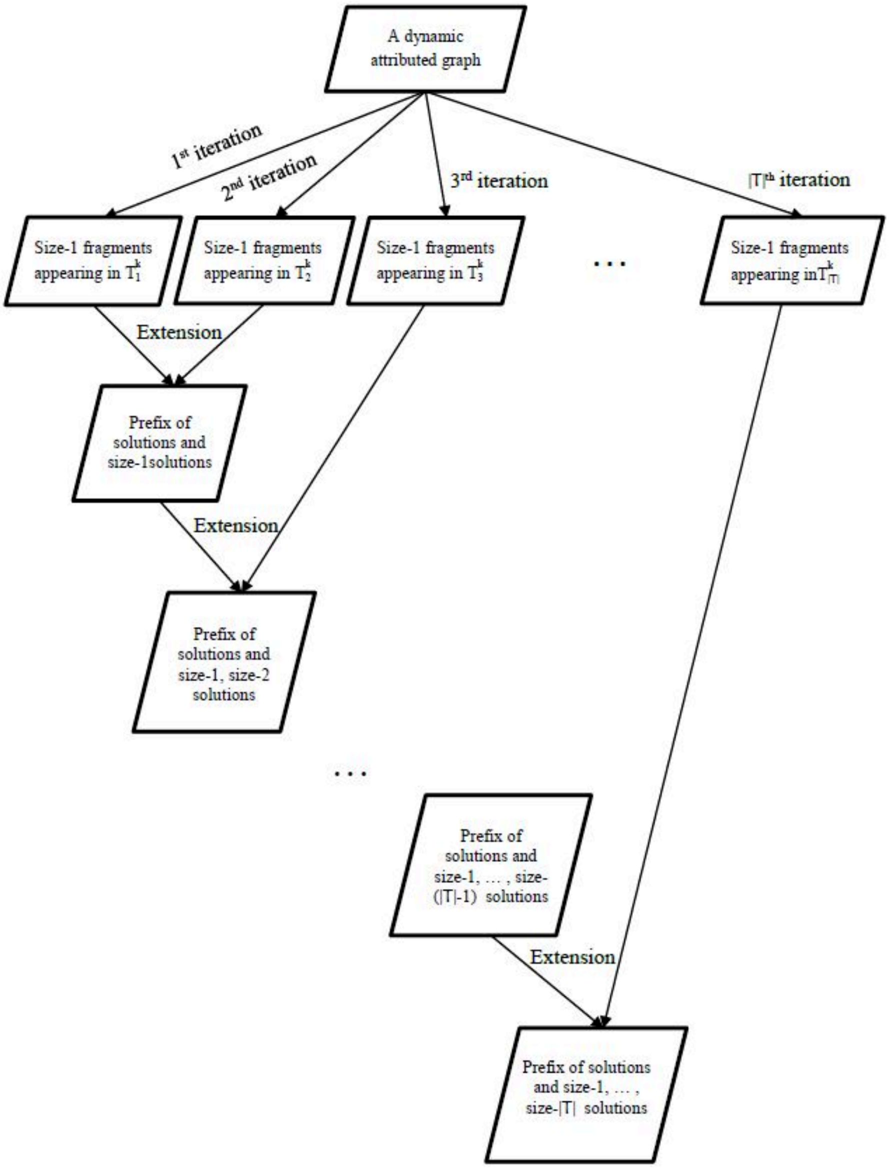

A dynamic attributed graph
1st iteration
2nd iteration
3rd iteration
|T|th iteration
Size-1 fragments appearing in T_1^k
Size-1 fragments appearing in T_2^k
Size-1 fragments appearing in T_3^k
...
Size-1 fragments appearing in $T_{|T|}^k$
Extension
Prefix of solutions and size-1solutions
Extension
Prefix of solutions and size-1, size-2 solutions
...
Prefix of solutions and size-1, ... , size-(|T|-1) solutions
Extension
Prefix of solutions and size-1, ... , size-|T| solutions

Figure 3.3 – Main process of our algorithm

$G_j\}$.

Let us then consider two connected components c and c' obtained after intersection of graphs in $\mathcal{G}$ at times $\{i,j\}$ and $\{k,l\}$ respectively, i.e. $c \in \mathbb{C}_{i \sqcap j}$ and $c' \in \mathbb{C}_{k \sqcap l}$, $\forall i,j,k,l \in \mathcal{T}$. Let $T_c = \{t \in \mathcal{T} \mid c \sqsubseteq G_t\}$ (resp. $T_{c'}$) be the subset of times in $\mathcal{T}$ when the connected component c (resp. c') occurs. It is not possible to have $c \sqsubset c'$ and $\boldsymbol{T_c = T_{c'}}$. Indeed it would imply that c' occurs at times $\{k,l\}$ but also at $\{i,j\}$. So, we would have $c' \sqsubseteq G_i \sqcap G_j$, which is impossible since c is a connected component of $G_i \sqcap G_j$ (thus it is maximal). In Fig. 3.4, the connected component $c_1 = (6 : a_2- \mid 11 : a_1- \mid 12 : a_1-)$ is in G_1, G_2 and G_3. Consequently, it appears in $G_1 \sqcap G_2$ and $G_1 \sqcap G_3$. In addition, there is no superset of vertices occurring at the same times. On the other hand, the subset $c_2 = (11 : a_1- \mid 12 : a_1-)$ can be obtained by performing $G_1 \sqcap G_4$, but it is not redundant to c_1 because it occurs at four times (t_1,

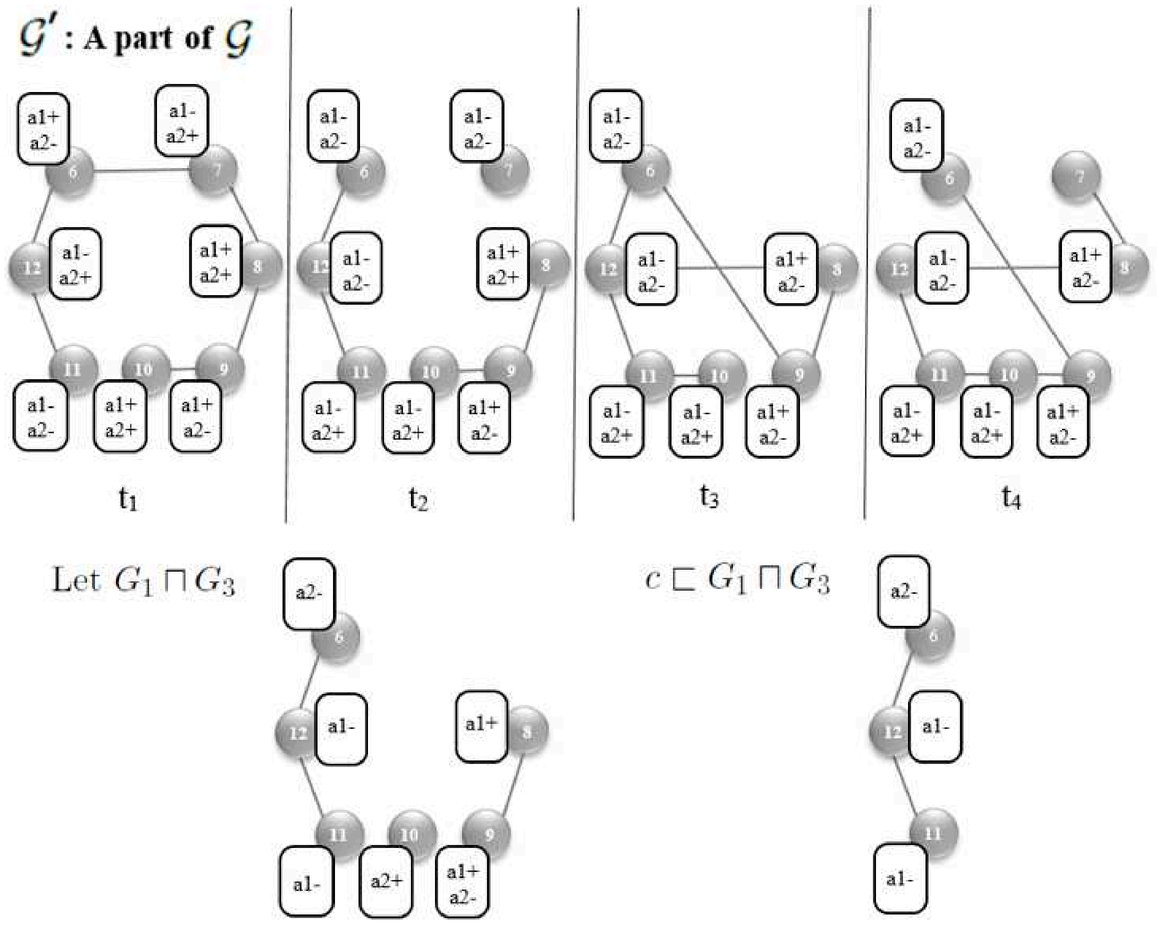


Figure 3.4 – Example of graph intersection

t_2, t_3 and t_4). To conclude, if $c = (V, E, \lambda)$, then the pattern $\big(\langle(V,\lambda)\rangle, T_c\big)$ satisfies the connectivity constraint (since it is a connected component), as well as the non-redundancy constraint (w.r.t. size-1 patterns). In other words, this pattern will be either a solution or a fragment of solution.

This property can be generalized to any set $T, T' \subseteq \mathcal{T}$. We note $\mathbb{C}_{\sqcap T}$ (resp. $\mathbb{C}_{\sqcap T'}$), the set of connected components obtained after intersection of graphs at times T (resp. T'), i.e $\underset{i\in T}{\sqcap} G_i$. If $c \in \mathbb{C}_{\sqcap T}$, then $\nexists c' \in \mathbb{C}_{\sqcap \mathbb{T}'}$ such as $c \sqsubset c'$ and $T_c = T_{c'}$. Size-1 patterns associated with those connected components satisfy both connectivity and non-redundancy constraints. The inverse of this proposition is also true. All size-1 solutions and all size-1 pattern fragments can be derived from connected components obtained after intersecting graphs in $\mathcal{G}$. In other words, these intersections provide the 'building blocks' to construct solutions.

The interest of these intersections is to avoid performing a large number of inclusion tests during pattern enumeration (to verify the frequency and non-redundancy constraints). The number of intersections is $2^{|\mathcal{T}|}$. Thus, it depends only on the number of times in $\mathcal{G}$, whereas the number of inclusion tests depends on the number of patterns generated, which is much higher.

2.2 Generation of a size-1 pattern

As shown in the previous section, size-1 patterns resulting from graph intersections directly satisfy frequency, connectivity and non-redundancy constraints. To extract final solutions, it is sufficient to verify volume and temporal continuity constraints. These constraints are simple and not costly to calculate as they are based on the studied pattern structure. Size-1 solutions or size-1 fragments can be defined as follows: $P = \{(\langle(V, \lambda)\rangle, T) \mid T \subseteq \mathcal{T}, |T| \geq$ $minsup, |V| \geq minvol,$ and $\exists c = (V, E, \lambda)$ such as $c \in \mathbb{C}_{\sqcap T}\}$.

A graph preprocessing is performed before intersections to reduce connectivity tests. It consists in finding all connected components of G_i $(1 \leq i \leq |\mathcal{T}|)$ whose volumes are greater than $minvol$, denoted as $\mathbb{C}_i$. As shown in Fig. 3.5, we firstly extract the set of connected components for G_1, i.e., $\mathbb{C}_1 = \{(v_1, v_2, v_3, v_4, v_5),\ (v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12})\}$. In the same way, $\mathbb{C}_2 = \{(v_1, v_2, v_5)$,$(v_3, v_4)$, (v_6, v_{11}, v_{12}), $(v_8, v_9, v_{10})\}$, $\mathbb{C}_3 = \{(v_1, v_2, v_3)$,$(v_4, v_5)$, $(v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12})\}$ and $\mathbb{C}_4 = \{(v_1, v_3, v_5),\ (v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12})\}$ are extracted. Intersections of connected components are then performed not on initial graphs but on their connected components.

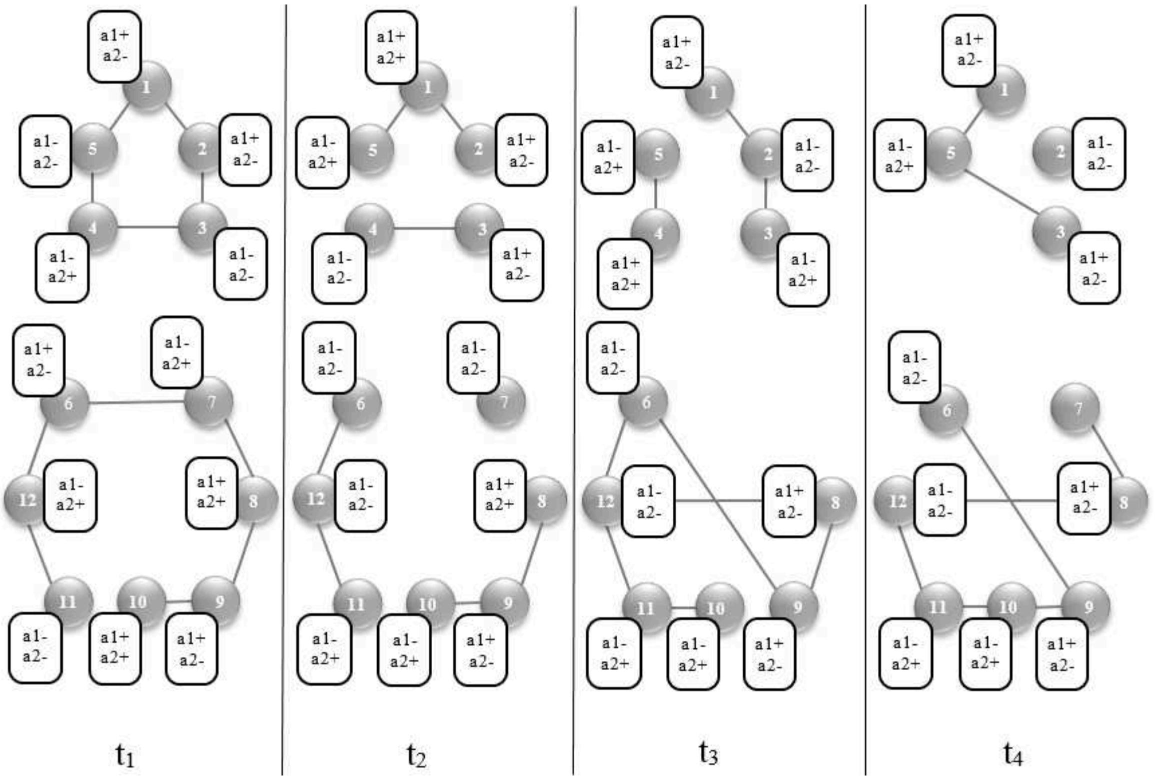


Figure 3.5 – Example of a dynamic attributed graph

Let us now present a two-step approach (algorithm 1) to intersect graphs G_i and G_j where $i, j \in \mathcal{T}$. Firstly, intersection is performed by simply finding the common sets of vertices $CandV$ and the common set of edges $CandE$ between $\mathbb{C}_i$ and $\mathbb{C}_j$ (Lines 1-6, algorithm 1). The function CommonVerticesEdges (algorithm 2) aims to extract sets of vertices $CandV$ with a sufficient volume, which avoids lots of connectivity tests and find the set of common edges $CandE$, which permits to verify the connectives of vertices in the following step (algorithm 3). The second step (Lines 7-19, algorithm 1) aims to calculate common attributes and verify vertex connectivity to determine final size-1 patterns. We browse each set V of $CandV$. For a vertex $v_l \in V$, if trends (or values) of at least one attribute a_l of v_l are the same at both t_i and t_j, the vertex and its attribute trend (value) will be added to the final set of connected components (size-1 patterns) $sol' \in P$ with $sol' = (V', \lambda')$. Otherwise, this vertex will be kept in a new set of connected component $sol^* \in P$. Then, in function CommonAttributes (algorithm 3), we conduct the same operations for its neighbors $N(v_l)$ such as $N(v_l) \in V$ and iteratively on neighbors of its neighbors $N(N(v_l))$ such as $N(N(v_l)) \in V$. This depth-first process (algorithm 3) continues until it has browsed all vertices in V and extracts the final size-1 patterns (complete sets of attributed connected components).

Algorithm 1: *ExtractIntersect*: mining size-1 patterns

```
Require: $\mathbb{C} = \{\mathbb{C}_i$ set of connected components of $G_i\}$, $T$: a set of times
Ensure: Set of size-1 patterns satisfying the constraints: $Cand$
 1: $CandV = \emptyset$
 2: $CandE = \emptyset$
 3: let $t' \in T$
 4: for each $c' \in \mathbb{C}_{t'}$, with $c' = (V', E', \lambda')$ do
 5:     $CommonVerticesEdges(V', E', T - t', \mathbb{C}, CandV, CandE)$
 6: end for
 7: for each $V \in CandV$ do
 8:     if $V \neq \emptyset$ then
 9:         for each $v \in V$ do
10:             $sol = \emptyset$
11:             $Cand\lambda(v) = \underset{t \in T}{\cap} \lambda_t(v)$
12:             $V = V - \{v\}$
13:             if $Cand\lambda(v) \neq \emptyset$ then
14:                 $CommonAttributes(v, V, CandE, T, sol, Cand)$
15:             end if
16:         end for
17:     end if
18: end for
19: return $Cand$
```

Algorithm 2: *CommonVerticesEdges*: mining candidates of size-1 patterns

```
Require: $V'$, $E'$, $T'$, $\mathbb{C}$, $CandV$, $CandE$
Ensure: $CandV$: Common sets of vertices, $CandE$: Common set of edges
if $T' = \emptyset$ then
    if $|V'| \geq minvol$ then
        $CandV = CandV \cup \{V'\}$
        $CandE = CandV \cup \{E'\}$
    end if
else
    let $t^* \in T'$
    for each $c^* \in \mathbb{C}_{t^*}$, with $c^* = (V^*, E^*, \lambda^*)$ do
        CommonverticesEdges($V' \cap V^*, E' \cap E^*, T' - \{t^*\}, \mathbb{C}, CandV, CandE$)
    end for
end if
```

For example, given a threshold $minvol = 2$, to extract subgraphs (size-1 patterns) of

Algorithm 3: *CommonAttributes*: mining final size-1 patterns

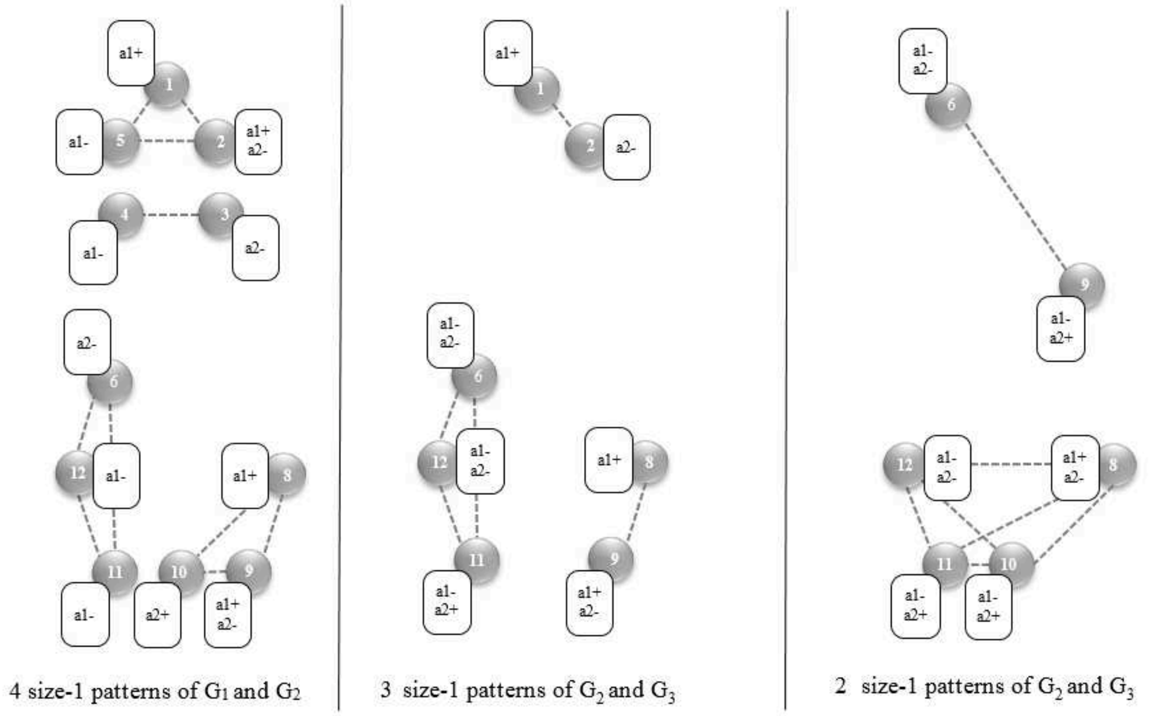
```
Require: $v$, $V$, $CandE$, $T$, $sol = (V, \lambda)$: an attribute subgraph , $Cand$: set of size-1 patterns
Ensure: $Cand$: set of size-1 patterns
if $V = \emptyset$ and $vol(sol) \geq minvol$ then
    $Cand = Cand \cup sol$
else
    for each neighbor of $v$, $N(v)$ such as $N(v) \in V$ and $(v, N(v)) \in CandE$ do
        $Cand\lambda(N(v)) = \bigcap_{t \in T} \lambda_t(N(v))$
        if $Cand\lambda(N(v)) \neq \emptyset$ then
            $V = V - \{N(v)\}$
            $sol = sol \cup \{N(v), Cand\lambda(N(v))\}$
            $CommonAttributes(N(v), V, CandE, T, sol, Cand)$
        end if
    end for
end if
```

G_1 and G_2, we calculate first the common sets of vertices $CandV$ and the common set of edges $CandE$ of $\mathbb{C}_1$ and $\mathbb{C}_2$, i.e., $CandV = \{\{v_1, v_2, v_5\},\{v_3, v_4\},\{v_6, v_{11}, v_{12}\},\{v_8, v_9, v_{10}\}\}$, $CandE = \{(v_1, v_2), (v_1, v_5), (v_3, v_4), (v_6, v_{12}), (v_8, v_9), (v_9, v_{10}), (v_{11}, v_{12})\}$. Then, for the candidate connected component (v_1, v_2, v_5), we calculate firstly the common attributes of vertex v_1 between G_1 and G_2. As we can see in Fig. 3.5, vertex v_1 shares the same evolution for a_1+ in graphs G_1 and G_2, so it is added to the final set of size-1 pattern sol'. Its neighbors v_2 ($v_2 \in V$ and $(v1, v2) \in CandE$) and v_5 ($v_5 \in V$ and $(v1, v5) \in CandE$) are then checked by calculating common attribute trends. This constructs the size-1 pattern $(\langle(1 : a_1+ \mid 2 : a_1 + a_2- \mid 5 : a_1-)\rangle, \{t_1, t_2\})$. As shown in Fig. 3.6, this approach enables to find other size-1 patterns of G_1 and G_2 : $(\langle(3 : a_2- \mid 4 : a_1-)\rangle, \{t_1, t_2\})$, $(\langle(6 : a_2- \mid 11 : a_1- \mid 12 : a_1-)\rangle, \{t_1, t_2\})$ and $(\langle(8 : a_1+ \mid 9 : a_1 + a_2- \mid 10 : a_2+)\rangle, \{t_1, t_2\})$. In this figure vertices of a subgraph are connected by a dotted line, because we do not consider how the vertices are connected, we only know that the subgraph is connected.

We illustrate intersections with another example based on G_2 and G_3. We first calculate the common sets of vertices $CandV$ and the common set of edges $CandE$ of $\mathbb{C}_2$ and $\mathbb{C}_3$, i.e., $CandV = \{\{v_1, v_2\},\{v_6, v_{11}, v_{12}\}, \{v_8, v_9, v_{10}\}\}$ and $CandE = \{(v_1, v_2), (v_6, v_{12}), (v_8, v_9), (v_{11}, v_{12})\}$. We can see that by calculating $CandV$, the vertices v_3, v_4 and v_5 are already deleted from the original sets which reduces connectivity tests. Then, for the candidate connected component $V = (v_1, v_2)$, we check the attribute trends of vertex v_1. The vertex v_1 shares the same evolution a_1+ in graphs G_2 and G_3. Its neighbor v_2 ($v_2 \in V$ and $(v1, v2) \in CandE$) is then verified by calculating common attribute trends and the size-1 pattern $(\langle(1 : a_1+ \mid 2 : a_2-)\rangle, \{t_2, t_3\})$ is recursively constructed. In the same way, we get another size-1 pattern $(\langle(6 : a_1 - a_2- \mid 11 : a_1 - a_2+ \mid 12 : a_1 - a_2-)\rangle, \{t_2, t_3\})$ from (v_6, v_{11}, v_{12}). For $V = (v_8, v_9, v_{10})$, we can notice that although v_8 v_9 and v_{10} are all in V, $(v_9, v_{10}) \notin CandE$. So v_{10} is rejected and we obtain the final size-1 pattern $(\langle(8 : a_1+ \mid 9 : a_1 + a_2-)\rangle, \{t_2, t_3\})$.

2.3 Extension of a size-1 pattern

Given a size-1 pattern, its possible extensions are generated by processing times (and corresponding graph intersections) incrementally. Next, size-1 patterns extracted by these intersections can be combined, according to the times when they occur, to built the solutions.



4 size-1 patterns of G1 and G2

3 size-1 patterns of G_2 and G_3

2 size-1 patterns of G_2 and G_3

Figure 3.6 – Size-1 pattern examples

This extension can be done by processing times incrementally.

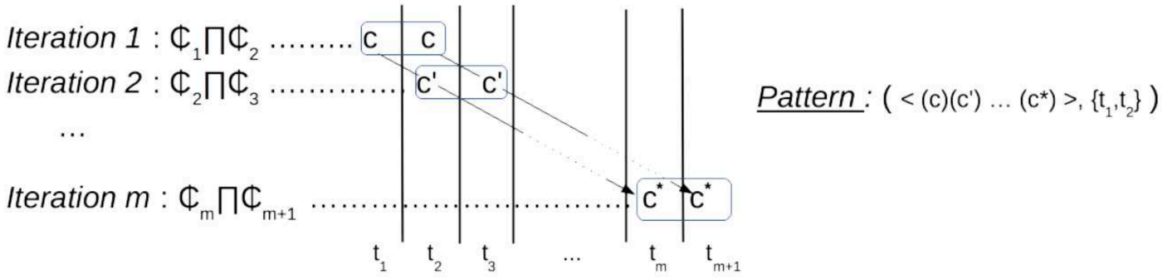

Iteration 1 : ₵1⊓₵2 ....
c
c
Iteration 2 : ₵2⊓₵3 ....
c'
c'
...
Iteration m : ₵m⊓₵m+1 ....
c*
c*
t1
t2
t3
...
tm
tm+1
Pattern : (< (c)(c') ... (c*) >, {t1,t2})

Figure 3.7 – Intersections and extensions in parallel of patterns from $\{t_1, t_2\}$

Fig. 3.7 illustrates this incremental generation starting from times t_1 and t_2. It displays parallel extensions of a pattern which occurs at t_1 and t_2. As the frequency constraint is directly related to the number of "intersected" times, we can conclude that minimum frequency in this example is 2. Suppose that there exists a solution $P = (\langle (V_1', \lambda_1')(V_2', \lambda_2') \ldots (V_n', \lambda_n') \rangle, \{t_1, t_2\})$. Let $\mathbb{C}_i$ and $\mathbb{C}_j$ be the sets of connected components of G_i and G_j. Intersection between $\mathbb{C}_1$ and $\mathbb{C}_2$ results in a graph composed of several connected components, such as $c = (V_1', E_1', \lambda_1')$, occurring at times t_1 and t_2. Pattern $P = (\langle (V_1', \lambda_1') \rangle, \{t_1, t_2\})$ can be generated based on that intersection. The first occurrence of this pattern is at time t_1, and the second one at time t_2. Candidate extensions for these occurrences can only be at t_2 and t_3 respectively (since gaps considered in this example is 1). Now let us consider times $\{t_2, t_3\}$. Let us suppose that $c' = (V_2', E_2', \lambda_2')$ is a connected component of $\mathbb{C}_2 \sqcap \mathbb{C}_3$. If c and c' share a sufficient number of vertices (temporal continuity constraint), then we can extend the pattern P to obtain $(\langle(V_1', \lambda_1')(V_2', \lambda_2')\rangle, \{t_1, t_2\})$. This process continues until no more extension can be performed. At each iteration, connected components can be used to extend patterns from the previous iteration, but they can also be "starting points" for new patterns. As a consequence, these successive extensions generate all solutions starting at time t_1, then all solutions starting at time t_2, etc.

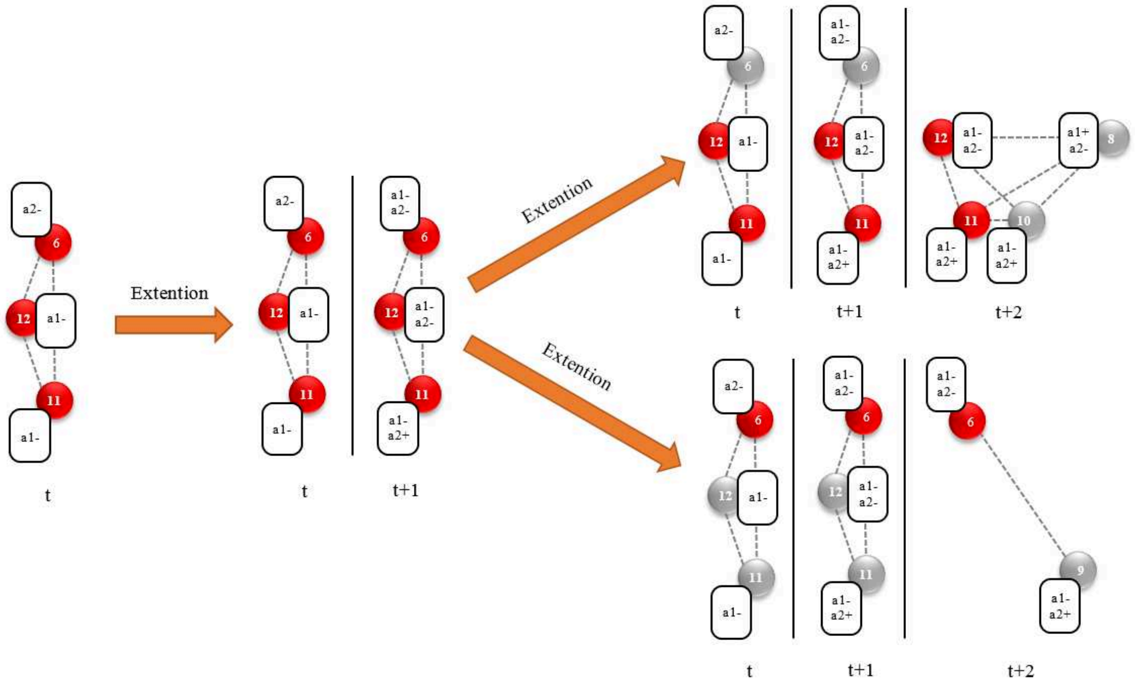


Figure 3.8 – An example of extension of a size-1 pattern

For example, let us consider the following size-1 patterns generated from the intersection of G_1 and G_2:

$(\langle(1 : a_1+ \mid 2 : a_1 + a_2- \mid 5 : a_1-)\rangle, \{t_1, t_2\})$,

$(\langle(3 : a_2- \mid 4 : a_1-)\rangle, \{t_1, t_2\})$,

$(\langle(6 : a_2- \mid 11 : a_1- \mid 12 : a_1-)\rangle, \{t_1, t_2\})$,

$(\langle(8 : a_1+ \mid 9 : a_1 + a_2- \mid 10 : a_2+)\rangle, \{t_1, t_2\})$.

We calculate firstly all size-1 patterns of G_2 and G_3:

$(\langle(1 : a_1+ \mid 2 : a_2-)\rangle, \{t_2, t_3\})$,

$(\langle(6 : a_1 - a_2- \mid 11 : a_1 - a_2+ \mid 12 : a_1 - a_2-)\rangle, \{t_2, t_3\})$,

$(\langle(8 : a_1+ \mid 9 : a_1 + a_2-)\rangle, \{t_2, t_3\})$.

Then, we extend size-1 patterns of G_1 and G_2 by verifying the temporal continuity constraint. Given a threshold $mincom = 1$, $(\langle(1 : a_1+ \mid 2 : a_1 + a_2- \mid 5 : a_1-)\rangle, \{t_1, t_2\})$ is extended with $(\langle(1 : a_1+ \mid 2 : a_2-)\rangle, \{t_2, t_3\})$ as they share two common vertices v_1 and v_2.

So we get a size-2 pattern $(\langle(1:a_1+ \mid 2:a_1+a_2- \mid 5:a_1-)(1:a_1+ \mid 2:a_2-)\rangle, \{t_1,t_2\})$. We note that $(\langle(3:a_2- \mid 4:a_1-)\rangle, \{t_1,t_2\})$ shares no common vertices with any set of connected components of G_2 and G_3. Thus it can not be extended and it is added to set of solutions. Other two size-1 patterns are extended in the same way, so we get:

$(\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)(6:a_1-a_2- \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_1,t_2\})$,

$(\langle(8:a_1+ \mid 9:a_1+a_2- \mid 10:a_2+)(8:a_1+ \mid 9:a_1+a_2-)\rangle, \{t_1,t_2\})$.

We then calculate all sets of size-1 patterns from G_3 and G_4:

$(\langle(6:a_1-a_2- \mid 9:a_1+a_2-)\rangle, \{t_3,t_4\})$,

$(\langle(8:a_1+a_2- \mid 10:a_1-a_2+ \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_3,t_4\})$.

We extend the extracted size-2 patterns by verifying the temporal continuity constraint. We notice that $(\langle(1:a_1+ \mid 2:a_1+a_2- \mid 5:a_1-)(1:a_1+ \mid 2:a_2-)\rangle, \{t_1,t_2\})$ shares no common vertices with any connected component sets of G_3 and G_4, so it can not be extended and is added to the solution set. For another size-2 pattern $(\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)(6:a_1-a_2- \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_1,t_2\})$, the following two size-1 patterns generated from G_3 and G_4 lead to this pattern satisfying the temporal continuity constraint:

$(\langle(6:a_1-a_2- \mid 9:a_1+a_2-)\rangle, \{t_3,t_4\})$,

$(\langle(8:a_1+a_2- \mid 10:a_1-a_2+ \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_3,t_4\})$.

The first one shares one common vertex v_6 and the second one shares two common vertices v_{11} and v_{12}. So two size-3 patterns are generated by those two extensions and we get:

$(\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)(6:a_1-a_2- \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)(6:a_1-a_2- \mid 9:a_1+a_2-)\rangle, \{t_1,t_2\})$,

$(\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)(6:a_1-a_2- \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)(8:a_1+a_2- \mid 10:a_1-a_2+ \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_1,t_2\})$.

Fig. 3.10 illustrates a detailed procedure for the extension of the size-1 pattern $(\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)\rangle, \{t_1,t_2\})$ to the final solutions. In the same way, $(\langle(8:a_1+ \mid 9:a_1+a_2- \mid 10:a_2+)(8:a_1+ \mid 9:a_1+a_2-)\rangle, \{t_1,t_2\})$ is extended to generate two size-3 patterns verifying the temporal continuity constraint:

$(\langle(8:a_1+ \mid 9:a_1+a_2- \mid 10:a_2+)(8:a_1+ \mid 9:a_1+a_2-)(6:a_1-a_2- \mid 9:a_1+a_2-)\rangle, \{t_1,t_2\})$,

$(\langle(8:a_1+ \mid 9:a_1+a_2- \mid 10:a_2+)(8:a_1+ \mid 9:a_1+a_2-)(8:a_1+a_2- \mid 10:a_1-a_2+ \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_1,t_2\})$.

Fig. 3.10 shows all extracted solutions (common vertices are in red) beginning from $\{t_1,t_2\}$ and satisfying user defined constraints $mincos = 0$, $minvol = 2$, $minsup = 2$, $mincon = 1$ and $gap = 1$.

2.4 Algorithm RPMiner

This method is detailed in algorithm 4. Line 1 corresponds to the extraction of connected components for each graph. Lines 3-7 construct size-1 patterns starting at time t_1 whose frequency are higher than the minimum threshold. For this purpose, the algorithm firstly calculates all time combinations T_1^k containing t_1 (algorithm 4 line 5). Then it generates size-1 patterns by performing intersections of connected components occurring at those times

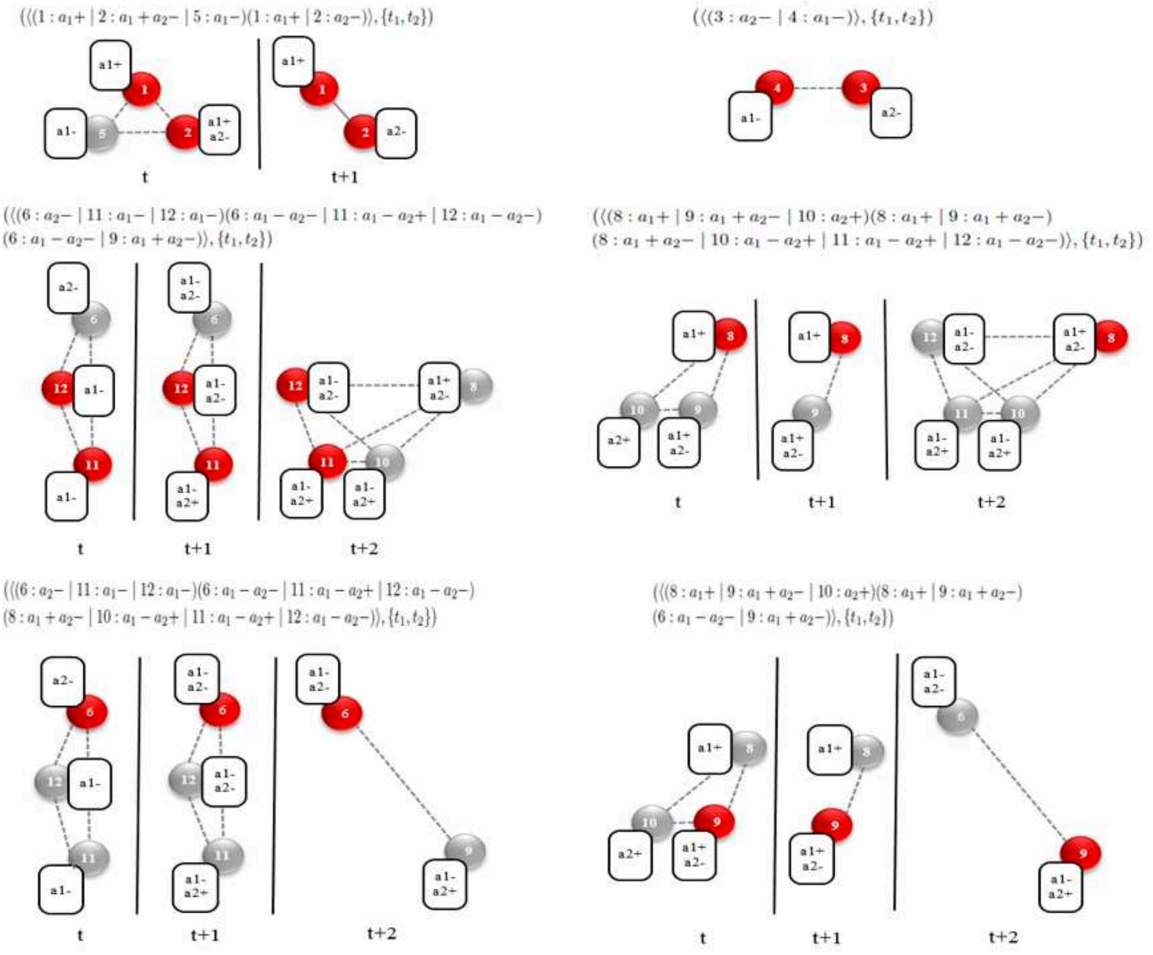

⟨⟨(1 : a1+ | 2 : a1 + a2− | 5 : a1−)(1 : a1+ | 2 : a2−)⟩, {t1, t2}⟩
⟨⟨(3 : a2− | 4 : a1−)⟩, {t1, t2}⟩
⟨⟨(6 : a2− | 11 : a1− | 12 : a1−)(6 : a1 − a2− | 11 : a1 − a2+ | 12 : a1 − a2−)
(6 : a1 − a2− | 9 : a1 + a2−)⟩, {t1, t2}⟩
⟨⟨(8 : a1+ | 9 : a1 + a2− | 10 : a2+)(8 : a1+ | 9 : a1 + a2−)
(8 : a1 + a2− | 10 : a1 − a2+ | 11 : a1 − a2+ | 12 : a1 − a2−)⟩, {t1, t2}⟩
⟨⟨(6 : a2− | 11 : a1− | 12 : a1−)(6 : a1 − a2− | 11 : a1 − a2+ | 12 : a1 − a2−)
(8 : a1 + a2− | 10 : a1 − a2+ | 11 : a1 − a2+ | 12 : a1 − a2−)⟩, {t1, t2}⟩
⟨⟨(8 : a1+ | 9 : a1 + a2− | 10 : a2+)(8 : a1+ | 9 : a1 + a2−)
(6 : a1 − a2− | 9 : a1 + a2−)⟩, {t1, t2}⟩
t
t+1
t+2

Figure 3.9 – All solutions beginning from $\{t_1, t_2\}$

(algorithm 4 line 5, method *ExtractIntersect*). This method has been detailed in algorithm 1. Line 5 of algorithm 1 (detailed in algorithm 2) constructs candidate sets of size-1 patterns, i.e. common sets of vertices $CandV$ and common sets of edges $CandE$ for each time combination T. Next, final size-1 patterns are generated by finding $CandV$ having the same attribute values (trends) in T (algorithm 1 line 6, detailed in algorithm 2). After that, the other times are processed incrementally. For each time t_i, **RPMiner** constructs all time combinations T_i^k containing t_i (algorithm 4 line 12), and extracts size-1 patterns P_i from intersections of connected components (algorithm 4 line 13). Then, it tries to extend each pattern P generated in the previous iteration with those size-1 patterns (algorithm 4 lines 14-15). If pattern P' resulting from the extension of P with P_i satisfies the temporal continuity constraint, it is added to the set of patterns generated at time t_i (algorithm 4 lines 16-17). Otherwise, P is added to the set of solutions, and P_i is saved for future extensions. In the end (line 26), all solutions generated at each time are put together and associated times are updated.

Fig. 3.11 depicts an example of algorithm execution with thresholds $mincos = 0$, $minvol = 2$, $minsup = 2$, $mincom = 1$ and $gap = 1$. The graph used is the example given in Fig. 3.2.

Algorithm 4: *RPMiner*: mining recurrent evolutions

```
Require: a dynamic attributed graph $\mathcal{G}$, $minsup$: minimum frequency threshold, $minvol$: minimum volume
    threshold, $mincom$: minimum number of common vertices over time, $mincos$: minimum similarity threshold, $gap$:
    interval allowed between every two successive subgraphs
Ensure: $Sol$: set of evolutions satisfying the constraints
$\mathbb{C} = \{\mathbb{C}_i$ set of connected components of $G_i \mid \forall c \in \mathbb{C}_i, c = (V, E, \lambda), |V| \geq minvol, \forall v \in V, \exists u \in V$ such that
    $cosine(v, u) \geq mincos\}$
$Cand_i = \emptyset, \forall i \in \{1, 2, ..., |\mathcal{T}|\}$
for $k = minsup$ to $|\mathcal{T}|$ do
    $T_i^k = \{t_{j1}, ..., t_{j_k} | t_{j1} < t_{j_k}$ and $t_{j1} = t_i\}$
    for each $T \subseteq T_1^k$ do
        $Cand_1 = Cand_1 \cup \{P_1 \in ExtractIntersect(\mathbb{C}, T)\}$
    end for
end for
$Sol_i = \emptyset, \forall i \in \{1, 2, ..., |\mathcal{T}|\}$
for $i = 1 + gap$ to $|\mathcal{T}|$ do
    for $k = minsup$ to $|\mathcal{T}|$ do
        for each $T \subseteq T_i^k$ do
            for each $P_i \in ExtractIntersect(\mathbb{C}, T)$ do
                for each $P = (S, T_P)$ such as $P \in Cand_{i-1}$ and $T_P = T$ do
                    $P' = ExtendWith(P, P_i)$
                    if $com(P') \geq mincom$ then
                        $Cand_i = Cand_i \cup \{P'\}$
                    else
                        $Sol_{i-1} = Sol_{i-1} \cup \{P\}$
                        $Cand_i = Cand_i \cup \{P_i\}$
                    end if
                end for
            end for
        end for
    end for
end for
$Sol = MergeUpdate(\bigcup_{\forall i \in \mathcal{T}} Sol_i)$
```

To keep the figure readable, We only display vertices associated to each pattern (attributes and times are omitted). At first, all size-1 patterns (column P_1 in Fig. 3.11) containing time t_1 are constructed (their frequency is higher than *minsup* and their volume is higher than *minvol*). Then, we extract all size-1 patterns starting at time t_2 (column "$P_1 + P_2$" in Fig. 3.11). Next, we extend P_1 with P_2 by verifying the temporal continuity constraint *mincom*. If they satisfy the constraint, they will be candidates for extension in the next iteration (column "$P_1 + P_2 + P_3$" in Fig. 3.11). Otherwise, P_1 is added to the set of solutions and patterns of P_2 are used for further extension. This processus continues until no more extensions can be performed. As shown in Fig. 3.11, red bold patterns are final solutions as they cannot be extended any more.

With this approach, a pattern will be generated and extended four times (from $\{t_1, t_2\}$, from $\{t_1, t_3\}$, from $\{t_2, t_3\}$, and from $\{t_1, t_2, t_3\}$). For each generation, pattern starting times are updated. Notice that even if the processing of $\{t_2, t_3\}$ and $\{t_1, t_2, t_3\}$ do not provide any new information, it can lead to generation of other patterns. All those combinations of intersections are thus necessary. That highlights the importance of our preprocessing to guarantee the scalability of tour approach.

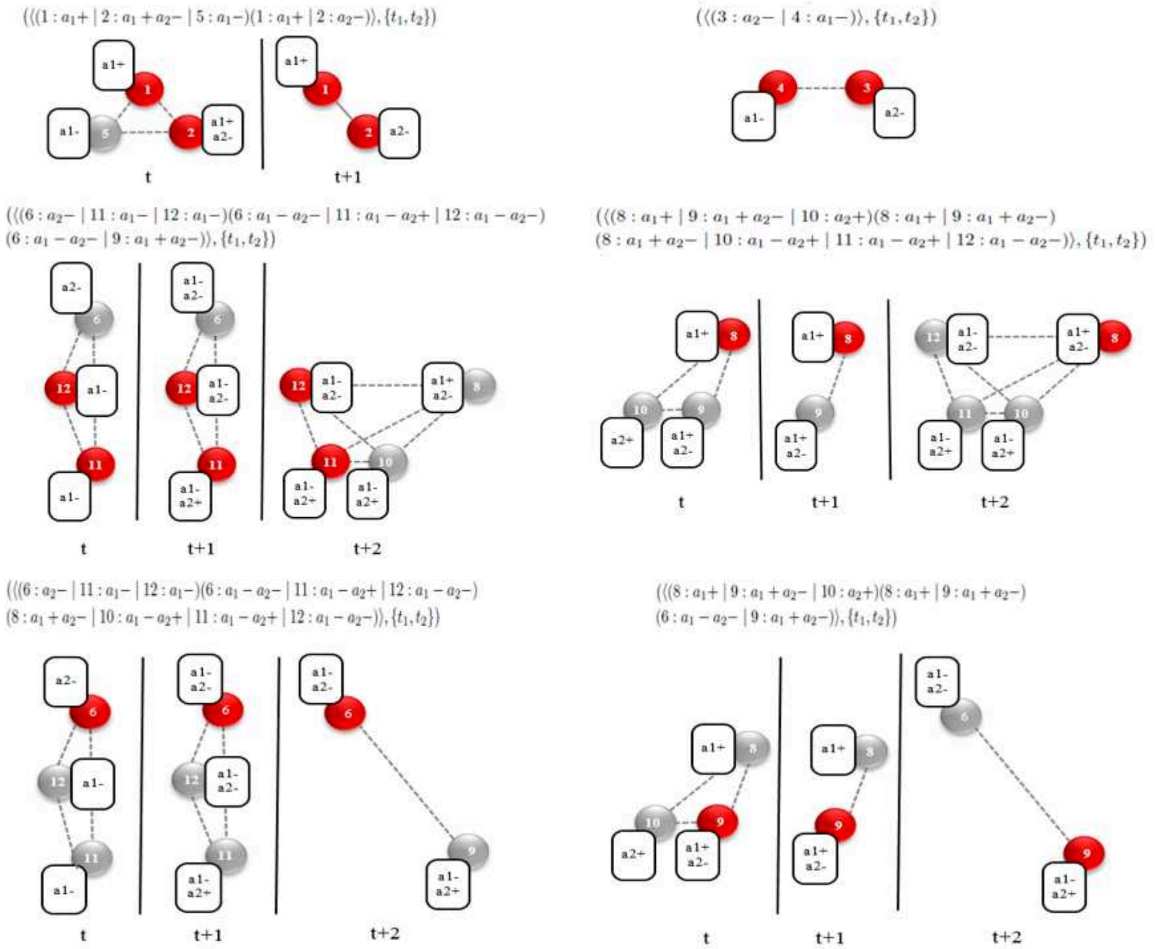

$\langle\langle(1:a_1+ \mid 2:a_1+a_2- \mid 5:a_1-)(1:a_1+ \mid 2:a_2-)\rangle, \{t_1,t_2\}\rangle$
t
t+1
$\langle\langle(3:a_2- \mid 4:a_1-)\rangle, \{t_1,t_2\}\rangle$
$\langle\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)(6:a_1-a_2- \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)$
$(6:a_1-a_2- \mid 9:a_1+a_2-)\rangle, \{t_1,t_2\}\rangle$
t
t+1
t+2
$\langle\langle(8:a_1+ \mid 9:a_1+a_2- \mid 10:a_2+)(8:a_1+ \mid 9:a_1+a_2-)$
$(8:a_1+a_2- \mid 10:a_1-a_2+ \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_1,t_2\}\rangle$
t
t+1
t+2
$\langle\langle(6:a_2- \mid 11:a_1- \mid 12:a_1-)(6:a_1-a_2- \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)$
$(8:a_1+a_2- \mid 10:a_1-a_2+ \mid 11:a_1-a_2+ \mid 12:a_1-a_2-)\rangle, \{t_1,t_2\}\rangle$
t
t+1
t+2
$\langle\langle(8:a_1+ \mid 9:a_1+a_2- \mid 10:a_2+)(8:a_1+ \mid 9:a_1+a_2-)$
$(6:a_1-a_2- \mid 9:a_1+a_2-)\rangle, \{t_1,t_2\}\rangle$
t
t+1
t+2

Figure 3.10 – Example of solutions

2.5 Algorithm time complexity and completeness

Complexity. To calculate time complexity for RPMiner, we frist consider the complexity of one intersection of k graphs, where $minsup \leq k \leq |\mathcal{T}|$. In the worst case, the complexity of browsing vertices for all connected components of one graph is equal to $|V| + |E|$ (Alho *et al.*, 1987). To calculate all intersected connected components of k graphs, we perform two by two intersections of graphs. It thus requires $(k-1)$ intersections of two graphs. So the complexity related to intersections of k graphs is equal to $(|V_{max}| + |E_{max}|) * 2 * (k-1)$, where $V_{max} = \underset{t \in \mathcal{T}}{\cup} V_t$ and $E_{max} = \underset{t \in \mathcal{T}}{\cup} E_t$. Then, we need to calculate common attributes for each vertex of those intersected connected components, so the complexity of intersection of k attributed graphs is: $|A| * (|V_{max}| + |E_{max}|) * 2 * (k-1)$.

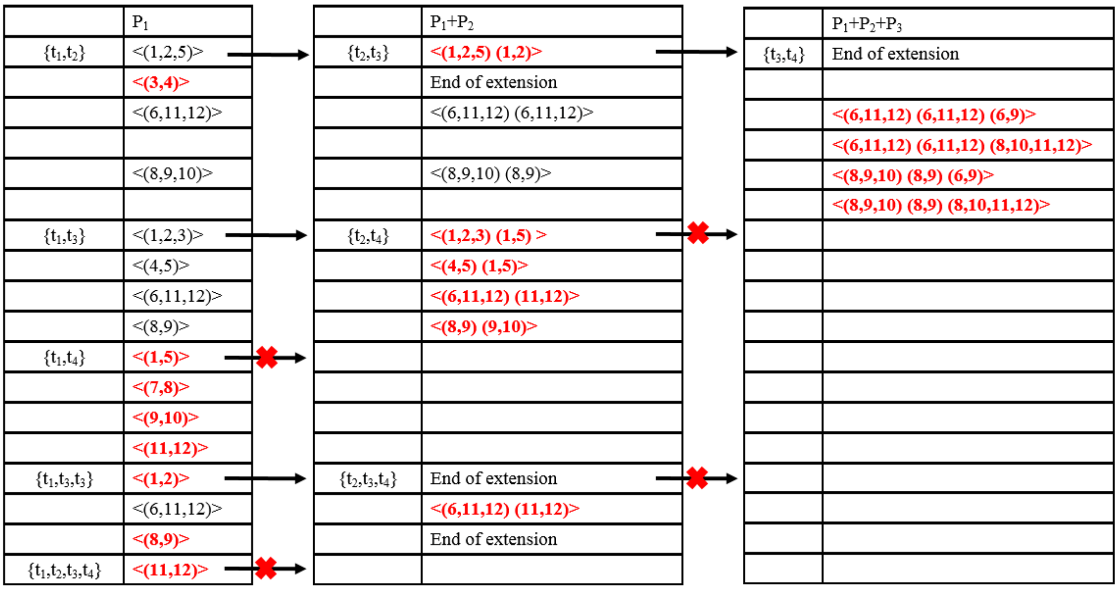
Next, we calculate the total number of time combinations. In the worst case $minsup = 1$ and $gap = 1$, we have to process all time combinations. It equals to $\sum_{i=1}^{|\mathcal{T}|} \binom{|\mathcal{T}|}{i}$. Then, we note that for every time combination $T \subseteq T_i^k$, we have to perform intersections $i-1$ times. So, the complexity of all intersection of graphs, denoted by $Com_{Intersection}$, is $Com_{Intersection} = \sum_{i=1}^{|\mathcal{T}|} (i-1) \binom{|\mathcal{T}|}{i} * 2 * (|V_{max}| + |E_{max}|) * |A| = ((|\mathcal{T}|-2) * 2^{|\mathcal{T}|-1} + 1) * 2 * (|V_{max}| + |E_{max}|) * |A|$.

Then we consider the complexity of extensions. In the worst case, the maximal number of connected components extracted in a timestamp is $|V_{max}|$ (that is to say, each vertex is a connected component). Thus, the maximal number of patterns that can be generated by the successive extensions equals to $|V_{max}|^{|T|-1}$. As discussed above, the total number of time combinations is $\sum_{i=1}^{|\mathcal{T}|}\binom{|\mathcal{T}|}{i} = (2^{|\mathcal{T}|} - 1)$. Thus, the complexity of generating all extensions is $Com_{Extension} = (2^{|\mathcal{T}|} - 1) * |V_{max}|^{|T|-1}$. Note that in practice, execution times of this part of the algorithm are quite low thanks to volume and temporal continuity constraints.

Based on the complexity of all graph intersections and the complexity of all extensions of patterns, the complexity of our algorithm is $Com_{Total} = Com_{Intersection} + Com_{Extension} = ((|\mathcal{T}| - 2) * 2^{|\mathcal{T}|-1} + 1) * 2 * (|V_{max}| + |E_{max}|) * |A| + (2^{|\mathcal{T}|} - 1) * |V_{max}|^{|T|-1}$

Completeness.

The algorithm correctness and completeness can be justified based on the properties of graph intersections: (1) all patterns constructed from the intersection of minsup graphs of $\mathcal{G}$ will satisfy the minimum frequency constraint; (2) all size-1 patterns satisfy non-redundancy constraints. These two properties guarantee that our algorithm extracts the complete set of size-1 patterns for each graph intersection. Furthermore, as mentioned in previous paragraph, our algorithm generates all the possible time combinations. They guarantee that our algorithm extracts the complete set of recurrent patterns.



	P_1		P_1+P_2		P_1+P_2+P_3
$\{t_1,t_2\}$	<(1,2,5)>	$\{t_2,t_3\}$	<(1,2,5) (1,2)>	$\{t_3,t_4\}$	End of extension
	<(3,4)>		End of extension		
	<(6,11,12)>		<(6,11,12) (6,11,12)>		<(6,11,12) (6,11,12) (6,9)>
					<(6,11,12) (6,11,12) (8,10,11,12)>
	<(8,9,10)>		<(8,9,10) (8,9)>		<(8,9,10) (8,9) (6,9)>
					<(8,9,10) (8,9) (8,10,11,12)>
$\{t_1,t_3\}$	<(1,2,3)>	$\{t_2,t_4\}$	<(1,2,3) (1,5) >		
	<(4,5)>		<(4,5) (1,5)>		
	<(6,11,12)>		<(6,11,12) (11,12)>		
	<(8,9)>		<(8,9) (9,10)>		
$\{t_1,t_4\}$	<(1,5)>				
	<(7,8)>				
	<(9,10)>				
	<(11,12)>				
$\{t_1,t_3,t_3\}$	<(1,2)>	$\{t_2,t_3,t_4\}$	End of extension		
	<(6,11,12)>		<(6,11,12) (11,12)>		
	<(8,9)>		End of extension		
$\{t_1,t_2,t_3,t_4\}$	<(11,12)>				

Figure 3.11 – An example of execution of algorithm

3. Experimental results

The algorithm was implemented in $C++$. Experiments were performed on PC with a 3.5GHz processor and 24 Gbytes of RAM. We used two real-world datasets and twenty synthetic datasets for our tests.

3.0.1 Datasets

Synthetic datasets. Graph sequences were generated by varying different parameters such as number of vertices per timestamp, number of attributes, number of edges and sequence size. Algorithm 5 illustrates the synthetic datasets generation. Firstly, we create $|\mathcal{G}|$ graphs (line 1 in algorithm 5). For each graph, we create $|V|$ vertices and then we associate each vertex with $|A|$ attributes following an uniform distribution. Then for each graph, we create $|E|$ pairs of vertices (edges), based on an uniform distribution.

Algorithm 5: *Generation of synthetic datasets*

```
Require: $|V|$: number of vertices per graph, $|E|$: number of edges per graph, $|A|$: number of attributes per vertex,
    $|\mathcal{G}|$: number of graphs, MaxValue: max value of attribute
Ensure: $\mathcal{G}$: an attributed dynamic graph
 1: for $i = 1$ to $|\mathcal{G}|$ do
 2:     Create $G_i = (V, E, \lambda), V = \emptyset, E = \emptyset$
 3:     for $j = 1$ to $|V|$ do
 4:         Create $v_j$, $v_j \in V$
 5:         for $k = 1$ to $|A|$ do
 6:             $a_k$=UniformDistribution(MaxValue), $a_k \in \lambda(v_j)$
 7:         end for
 8:     end for
 9:     for $l = 1$ to $|E|$ do
10:         $e_l = (v_y, v_{y'}), v_y \in V, v_{y'} \in V, y = UniformDistribution(|V|), y' = UniformDistribution(|V|)$
11:     end for
12: end for
```

DBLP dataset. This dataset used in (Desmier *et al.*, 2012) represents DBLP authors and their co-publications between 1990 and 2009. This period is divide into 9 timesteps ([[1990-1993][1992-1995]...[2006-2009]) where each timestep depicts co-authorship relationships and authors' number of publications over 4 years. Vertices represent authors who published more than 10 papers. Edges exist between authors who published at least one paper together during this period. Each vertex is labeled by 43 attributes representing the number of publications in 43 different conferences and journals belonging to the Data Mining and Databases communities. The dataset is composed of 2,723 vertices per timestamp (authors), 10,737 edges in average (co-publications), 43 attributes (a set of selected conferences/journals) and 9 timestamps ([1990-1993][1992-1995]...[2006-2009]).

Domestic US Flight dataset. This dataset used in (Kaytoue *et al.*, 2014) represents airport traffic in the US during the Katrina hurricane period (from 01/08/2005 to 25/09/2005). Hurricane Katrina was an extremely destructive and deadly tropical cyclone. It was also one of the costliest natural disasters and one of the five deadliest hurricanes in the the United States history. The aim is to study the impact of Katrina hurricane on the US flights. Vertices represent US airports, each edge links two airports having at least one flight between them. Each vertex is associated to a set of attributes depicting traffic aspects (number of departures/arrivals, number of canceled flights, number of flights whose destination airport has been diverted, mean delay of departure/arrival and ground waiting time departure/arrival). Instead of using numeric values of attributes, we calculate the trend for each attribute over time, i.e. +,- and =, which mean that a value increases, decreases or remains constant over two consecutive timestamps. To summarize, DBLP dataset is composed of 280 vertices per timestamp (airports), 1206 edges in average (flight connections), 8 attributes and 8 timestamps (data is aggregated by weeks).

3.0.2 Quantitative Results

Impact of number of vertices and edges Fig. 3.12, Fig. 3.13 and Fig. 3.14 present execution times, number of solutions and maximum memory usage for twelve synthetic datasets with a growing number of vertices and edges (number of attributes is set to 50 and number of timestamps to 8). It can be noticed that our algorithm remains efficient when analyzing sequence consisting of 20000 vertices per graph and per date, with a very small *minsup* (2).

Impact of number of timestamps Fig. 3.15, Fig. 3.16 and Fig. 3.17 show the impact of number of timestamps on our algorithm. We can observe that execution times, number of solutions and maximum memory usage increase exponentially with regard to the number of timestamps. This impact is gigh because we browse all combinations of graph intersections.

Impact of number of attributes Fig. 3.18, Fig. 3.19 and Fig. 3.20 show the impact of attribute number. As we can see, execution time increases linearly according to the number of attributes. **RPMiner** could process up to 1000 attributes per vertex, therefore permitting to model complex data. We can notice that **RPMiner** scales well on synthetic data according to number of vertices, number of edges and number of attributes, while the execution time increases exponentially w.r.t. number of graphs (timestamps).

Impact of *mincos* Fig. 3.21 reports the performance of **RPMiner** on synthetic data when varying the parameter *mincos* (minimum similarity threshold). We can observe that number of solutions and execution time increase quickly when *mincos* is set to 0.2. It is because many big connected components are divided into smaller connected components which are more cohesive (whose vertices share more common neighbors). However, when *mincos* keeps on increasing, we remark that the number of solutions decreases sharply,

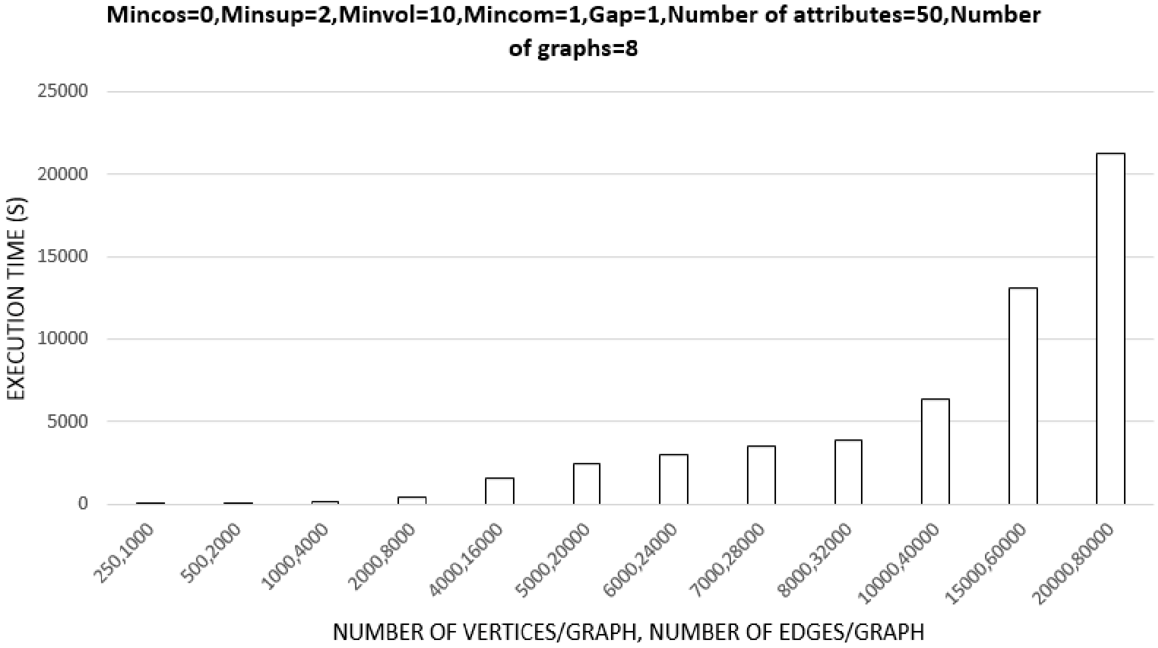


Figure 3.12 – Impact of number of vertices and edges per graph on the execution time (synthetic data)

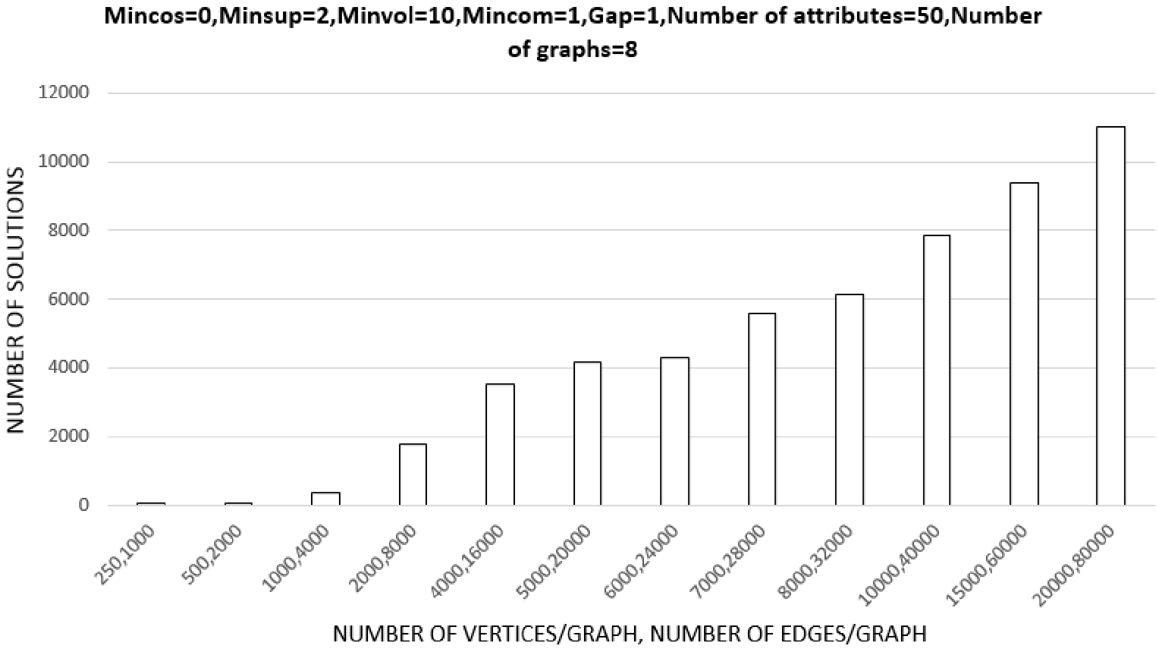


Figure 3.13 – Impact of number of vertices and edges per graph on the number of solutions (synthetic data)

because there are much less connected components which are strongly cohesive (most of vertex neighbors in a given connected component are the same).

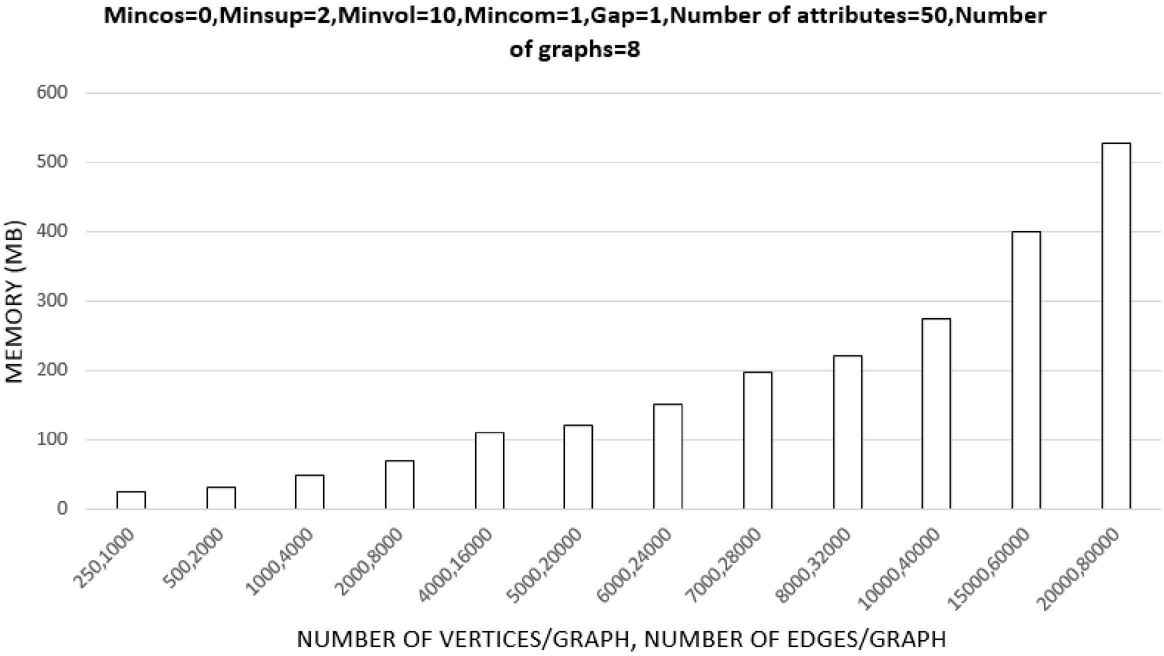


Figure 3.14 – Impact of number of vertices and edges per graph on the memory (synthetic data)

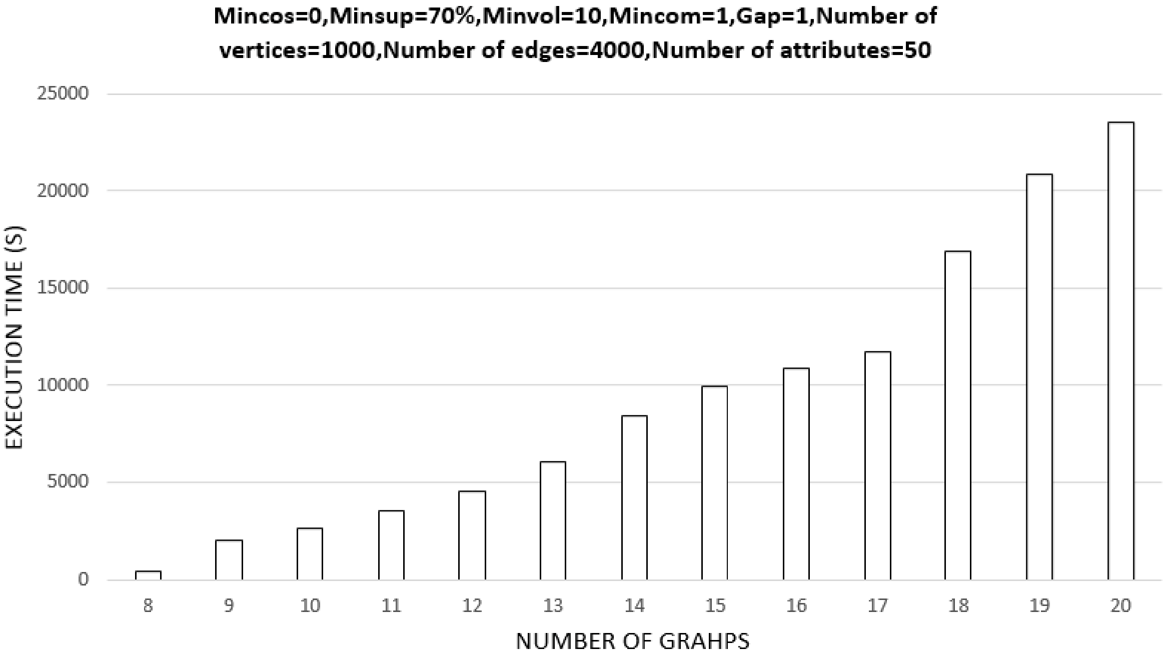


Figure 3.15 – Impact of number of graphs (timestamps) on the execution time (synthetic data)

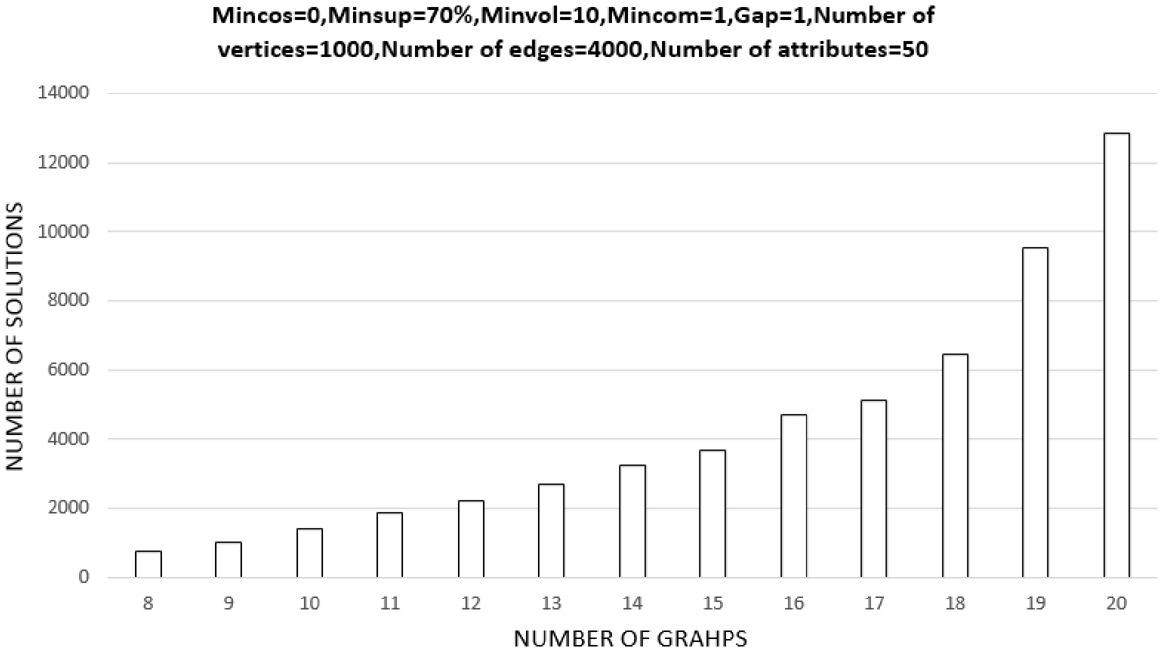


Figure 3.16 – Impact of number of graphs (timestamps) on the number of solutions (synthetic data)

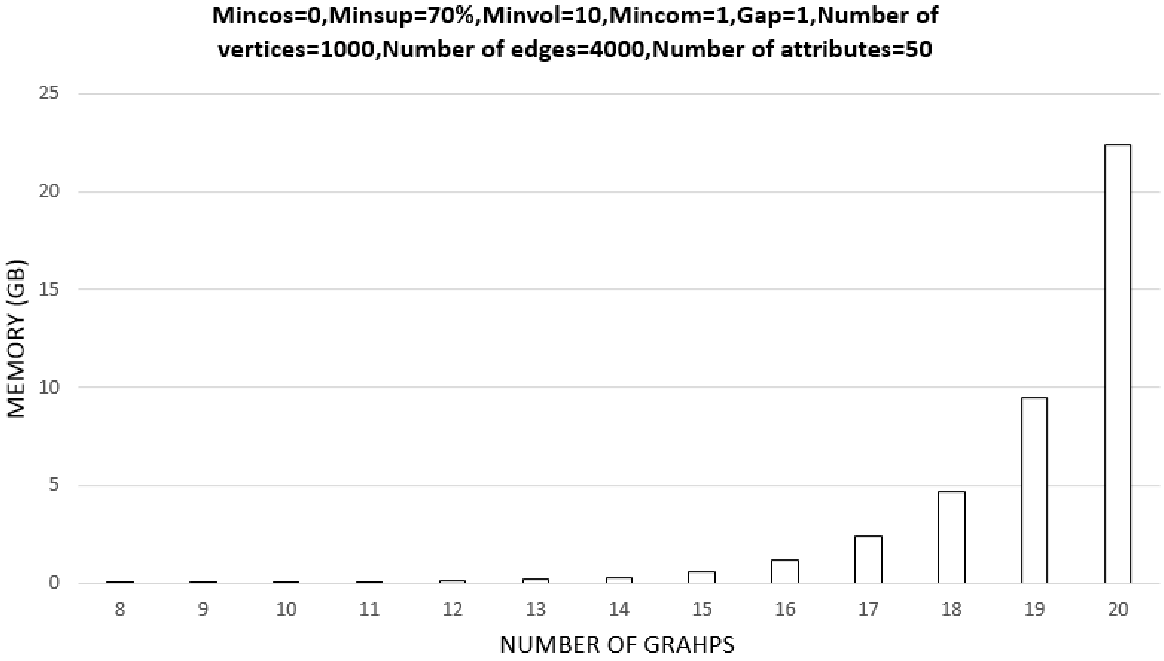


Figure 3.17 – Impact of number of graphs (timestamps) on the memory (synthetic data)

Impact of *minsup* We also study the performance of **RPMiner** on synthetic data and DBLP dataset w.r.t. different frequency thresholds. As shown in Fig. 3.22, the frequency

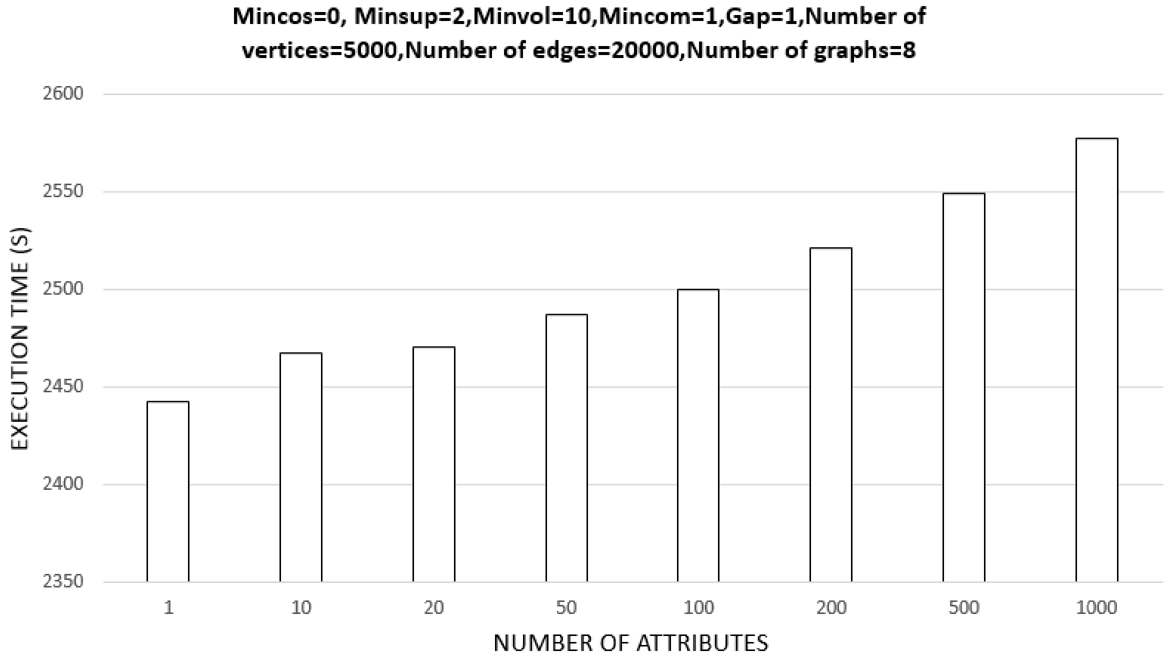


Figure 3.18 – Impact of number of attributes per vertex on the execution time (synthetic data)

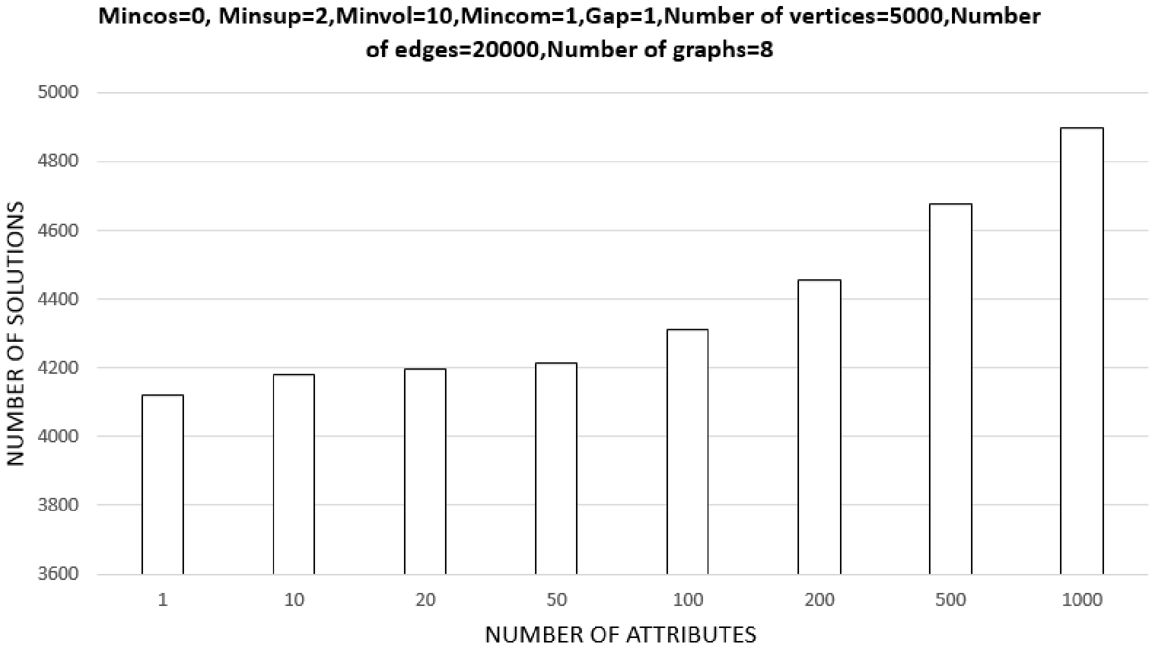


Figure 3.19 – Impact of number of attributes per vertex on the number of solutions (synthetic data)

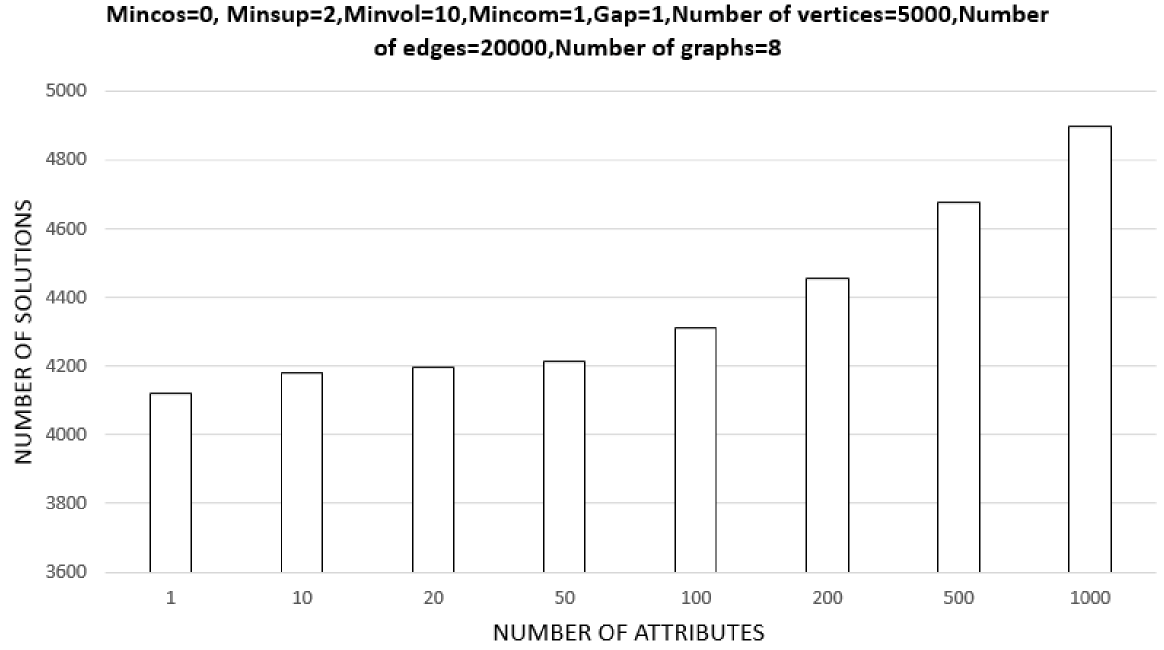


Figure 3.20 – Impact of number of attributes per vertex on memory (synthetic data)

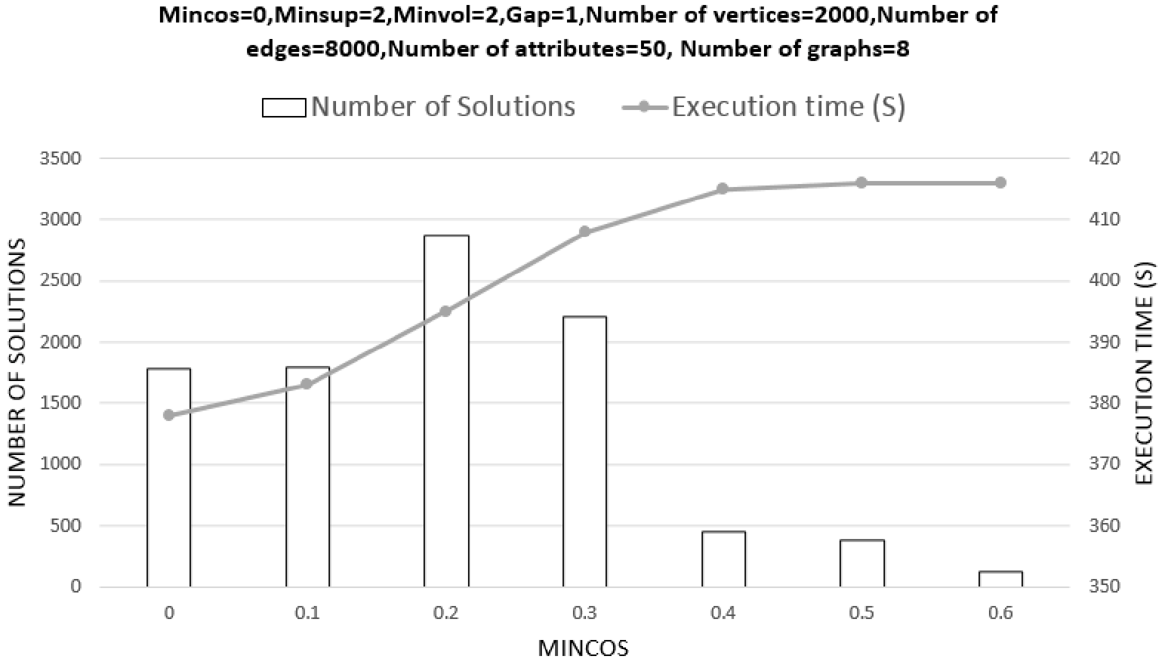


Figure 3.21 – Impact of *mincos* on the number of solutions and the execution time (synthetic data)

constraint permits to significantly reduce execution time and number of solutions. It is due to the fact that only the intersection of no less than *minsup* graphs are performed, which completely depends on the minimum frequency threshold. As shown in Fig. 3.23, **RPMiner** is still efficient on this real-world dataset (e.g. DBLP dataset) even for low thresholds.

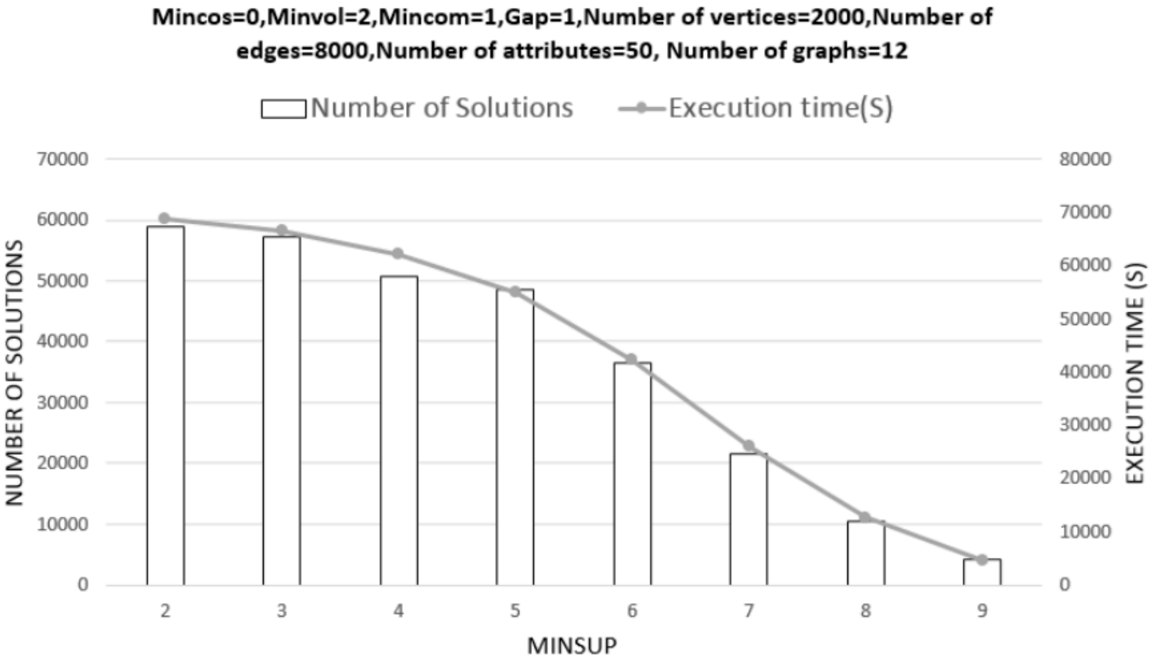


Figure 3.22 – Impact of *minsup* on the number of solutions and the execution time (synthetic data)

Impact of *minvol* Fig. 3.24 shows performance on synthetic data w.r.t. different volume thresholds. The impact of volume threshold is quite significant. Indeed we can observe that this constraint has an effective impact on patterns whose volume is less than 15 while it has barely any effect on patterns larger than 15. This is because there exists numerous small connected components (composed of less than 15 vertices) whereas there are a few groups composed of more than 15 vertices.

Fig. 3.25 shows performance on DBLP dataset w.r.t. different volume thresholds. We can observe that this constraint has an effective impact on the patterns whose volume is less than 5 while it has barely any effect on the patterns larger than 5. This is because there exists a great number of small groups of authors working together (composed of 2 or 3 authors) while there are only few groups composed of more than 10 authors.

Impact of *mincom* Fig. 3.26 shows performance on synthetic data w.r.t. different *mincom* thresholds. Impact of this threshold is less important compared with frequency and volume. We can observe that this constraint has an effective impact on patterns whose *mincom* is less than 5 whereas it has barely any effect on patterns larger than 5. This is because most of patterns evolve around a common core of less than 5 vertices while there are much fewer groups evolving around a common core of more than 5 vertices.

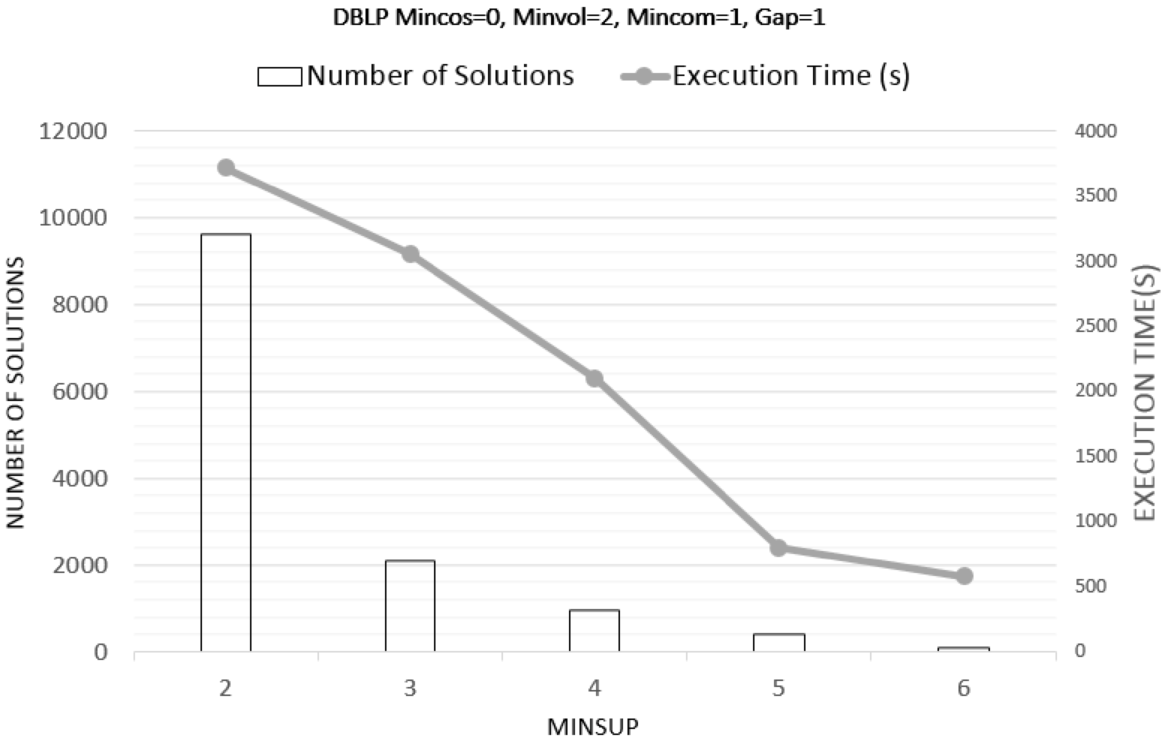


Figure 3.23 – Impact of *minsup* on the number of solutions and the execution time (DBLP dataset)

Impact of *gap* Finally, we study **RPMiner** performance on synthetic data w.r.t. different gap thresholds. As shown in Fig. 3.27, impact of this threshold is very important. We can observe that number of solutions and execution time drop by half when the *gap* threshold varies from 1 to 4. This is because we divided by two the number of studied timestamps.

3.0.3 Qualitative interpretation

DBLP dataset We have also carried out a qualitative analysis of patterns extracted from real-world dataset DBLP. For this experiment, parameters were firstly set to $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0$ and $mincom = 2$. Fig. 3.28 shows an example of pattern extracted in the that data:

$\big(\langle$(Henry Tirri: KDD, ICML | Petri Myllymaki: KDD, ICML)
(Henry Tirri: KDD, IntellDtAnal | Petri Myllymaki: KDD, IntellDtAnal)
(Henry Tirri: ECMLPKDD | Petri Myllymaki: ECAI) $\rangle, \{[90-93], [94-97]\}\big)$.

Here, vertex attributes are the names of different conferences and journals which signify the corresponding authors published at least one article in this conference or journal during the period under consideration. The pattern depicts the evolution of a co-author network of Henry Tirri and Petri Myllymaki. This is a sequence of size 4 which represents an evolution over 3 timestamps. This sequence is repeated twice. First from 1990 to 2005 (i.e. timesptamps [90-93], [96-99] and [02-05]) and second from 1994 to 2009 (i.e. timesptamps [94-97], [00-03] and [06-09]).

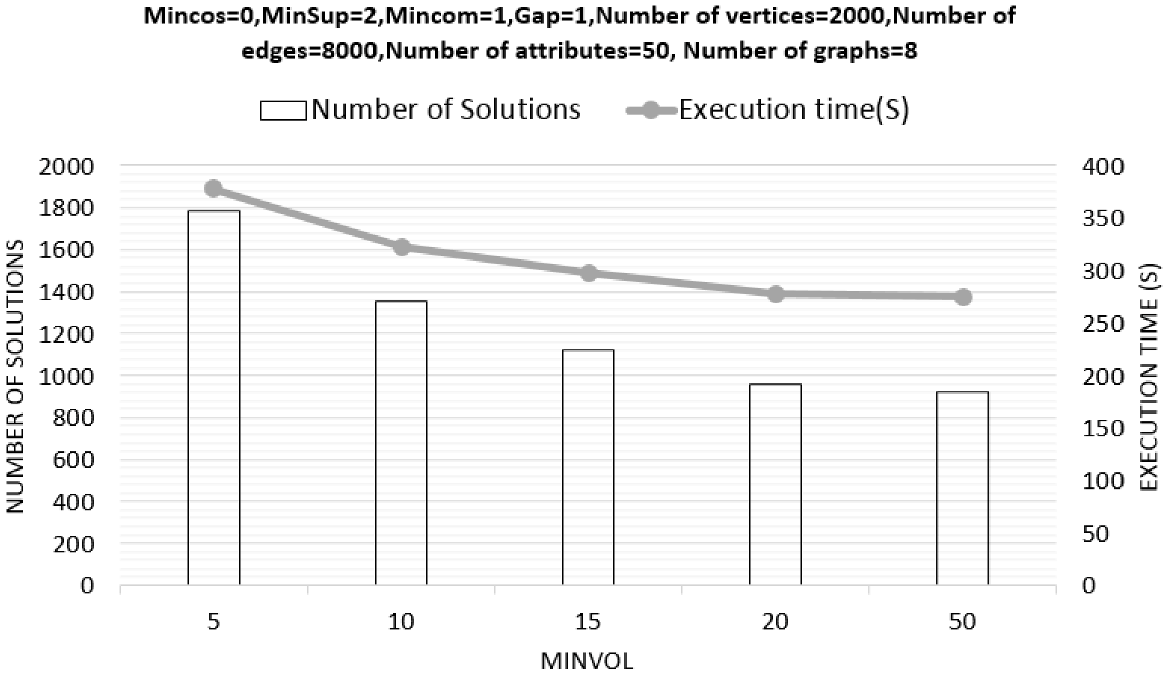


Figure 3.24 – Impact of *minvol* on the number of solutions and the execution time (synthetic data)

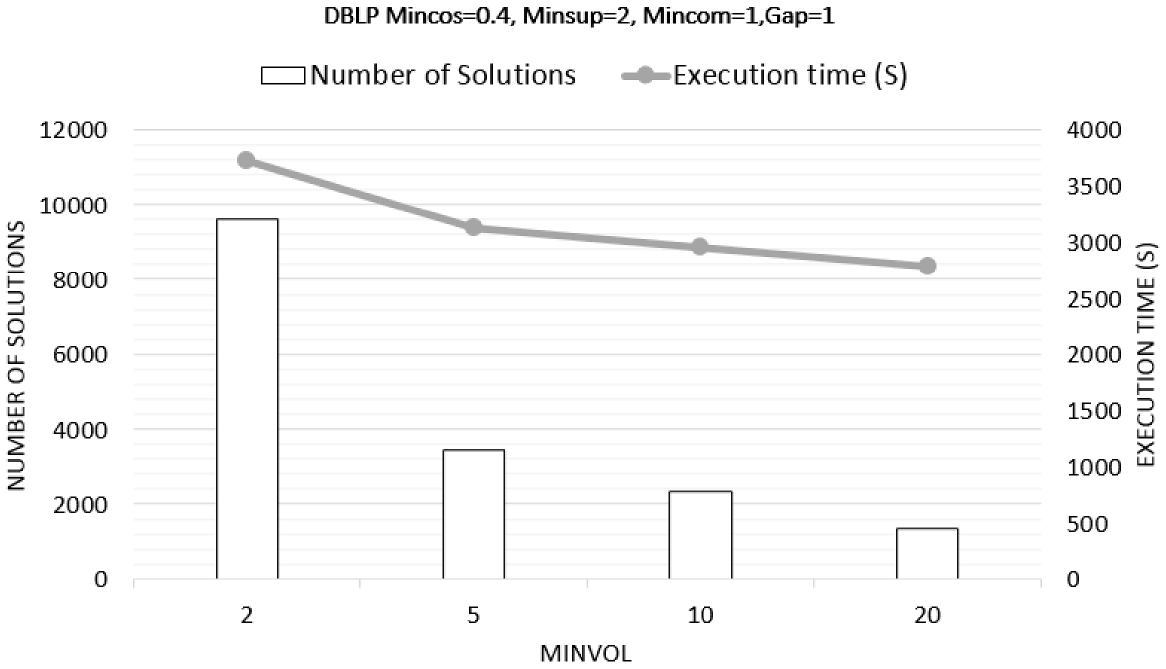


Figure 3.25 – Impact of *minvol* on the number of solutions and the execution time (DBLP dataset)

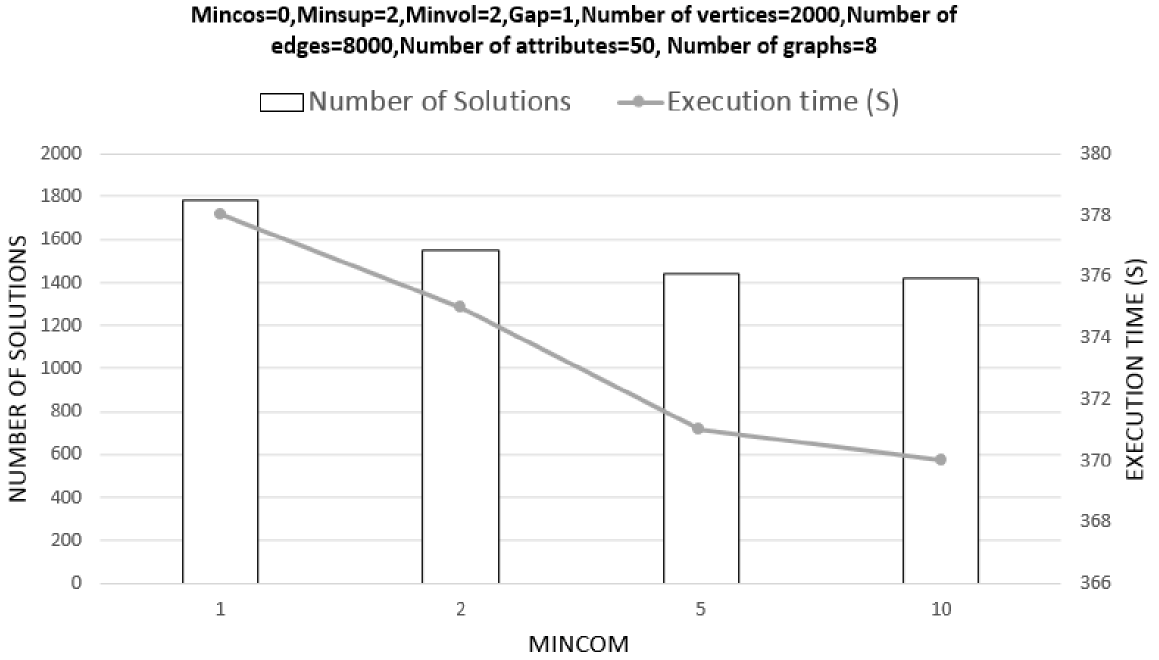


Figure 3.26 – Impact of *mincom* on the number of solutions and the execution time (synthetic data)

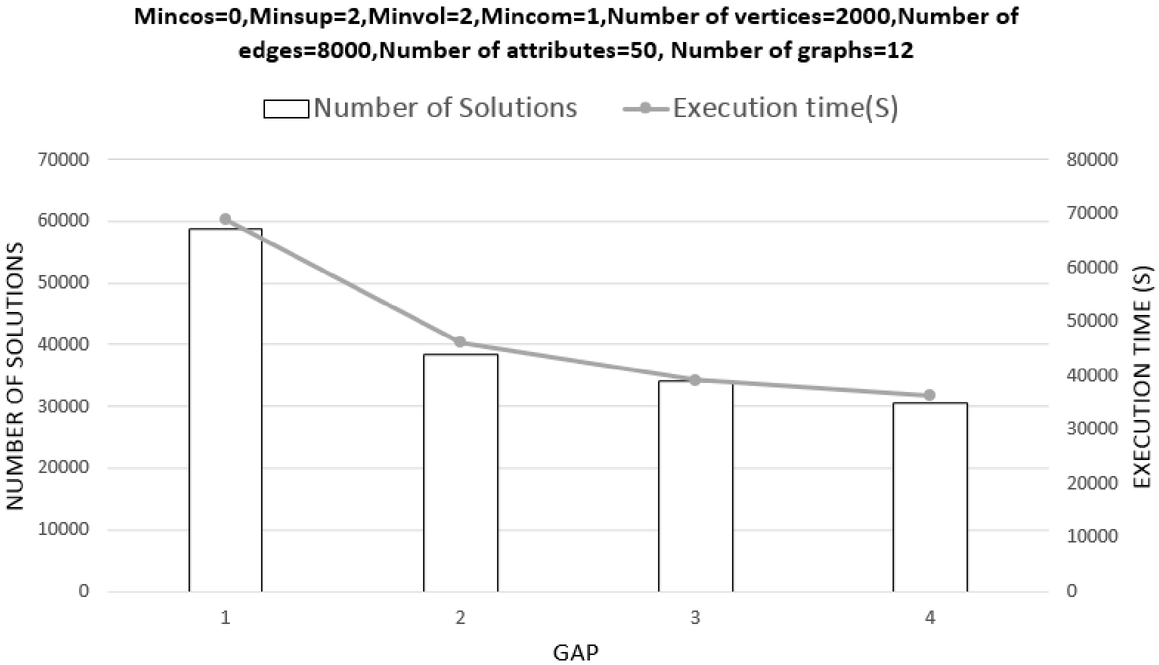


Figure 3.27 – Impact of *gap* on the number of solutions and the execution time (synthetic data)

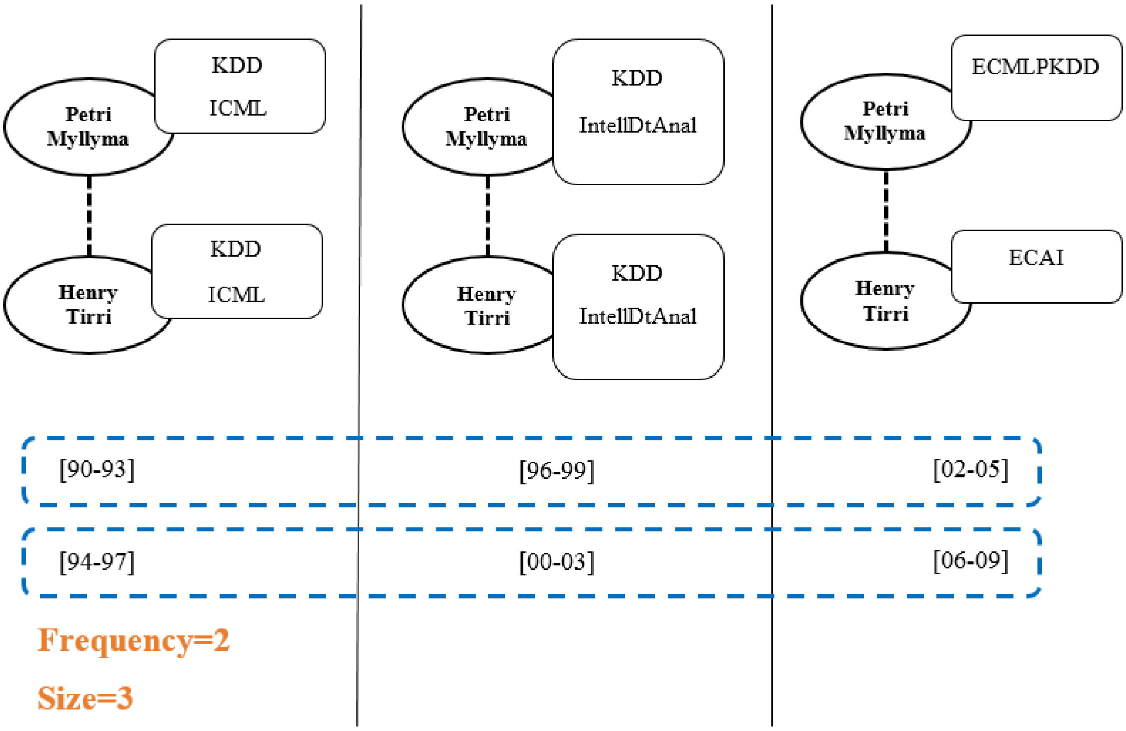


Figure 3.28 – First pattern extracted from DBLP with the parameters $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0$ and $mincom = 2$

This pattern shows preferences of co-authors Henry Tirri and Myllymaki over time. Between 1990 and 1993, they published together in KDD and ICML conferences. Then, from 1996 to 1999, besides KDD, they also succeeded in publishing articles together in IntellDtAnal. Between 2002 and 2005, Henry Tirri published in ECMLPKDD and Myllymaki published in ECAI proceedings. This recurrent pattern appears another time (from 1994 to 2009). RPMiner permits to extract all evolutions in co-author networks. However, when we set *mincos* to 0 (i.e. we only take into account the connectivity constraint), we could extract some patterns which describe evolutions of very large and sparse co-author networks (a network of more than 50 authors) where most of authors do not have direct co-authorship. It is difficult to interpret the evolution of such a big network because it may be composed of several smaller groups of co-authors. For this purpose, here we set *mincos* to a rather high value to focus on evolutions of cohesive and dense groups of co-authors (connected vertices), that work closely together.

So parameters were then set to $minvol = 2$, $minsup = 2$, $gap = 5$ $mincos = 0.4$ and $mincom = 2$. As shown in Fig. 3.29, an example of pattern extracted from DBLP dataset is

$\big(\langle(WeiyiMeng : IEEETransKnowlDtEn, ICDE \mid ClementT.Yu : IEEETransKnowlDtEn, ICDE)$

$(WeiyiMeng : IEEETransKnowlDtEn, CIKM, VLDB, KnowlInfSyst, VLDBJ, SIGMOD, DataKnlEng \mid ClementT.Yu : IEEETransKnowlDtEn, CIKM, VLDB, KnowlInfSyst, VLDBJ, SIGMOD, DataKnlEng \mid AnHaiDoan : SIGMOD)$

$\rangle, \{[90-93], [94-97]\}\big).$

This pattern describes the evolution of Weiyi Meng and Clement T. Yu co-author network. This is a sequence of size 2 which represents an evolution over 2 timestamps. This sequence is repeated twice, first from 1990 to 2003 (i.e. timesptamps [90-93] and [00-03]), and then from 1994 to 2007 (i.e. timesptamps [94-97] and [04-07]). This pattern describes evolution of a group of authors over time at co-authorship level. Between 1990 and 1997, only Weiyi Meng and Clement T. Yu worked together. Then, from 2000 to 2007, we observe that their co-author network became bigger with Weiyi Meng joining their team. On the other hand, this pattern highlights evolutions of their publications over time. We can see that between 1990 and 1993, Weiyi Meng and Clement T. Yu published together only in two conferences IEEETransKnowlDtE and ICDE. Then, from 2000 to 2003, besides EEE-TransKnowlDtEn, Weiyi Meng and Clement T. Yu had much more publications together in other conferences, i.e. CIKM, VLDB, KnowlInfSyst, VLDBJ and DataKnlEng. In addition, we can notice that Weiyi Meng and Clement T. Yu published together with Weiyi Meng in SIGMOD conference. This pattern appears another time (from 1994 to 2007).

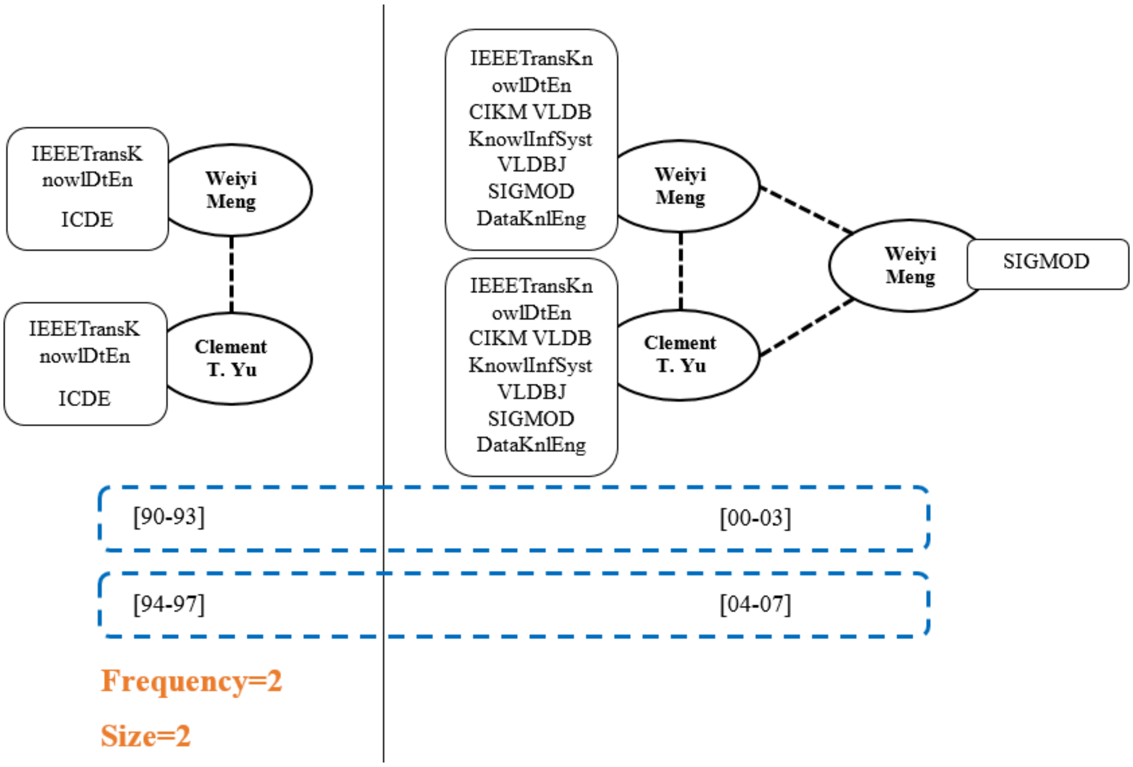


Figure 3.29 – Second pattern extracted from DBLP with the parameters $minvol = 2$, $minsup = 2$, $gap = 5$ $mincos = 0.4$ and $mincom = 2$

Then we change the parameter *gap* to 3 to study the evolution of the same group of co-authors Weiyi Meng and Clement T.Yu. As shown in Fig. 3.30, we obtain the pattern

(⟨(Weiyi Meng: IEEETransKnowlDtEn, ICDE | Clement T.Yu: IEEETransKnowlDtEn, ICDE | Won Kim: ICDE | Son Dao: ICDE)

(Weiyi Meng: IEEETransKnowlDtEn, CIKM, VLDB, VLDBJ | Clement T.Yu: IEEETransKnowlDtEn, CIKM, VLDB, VLDBJ | King-Lup Liu: CIKM)

(Weiyi Meng: ICDE, CIKM, VLDB, SIGMOD, DataKnlEng | Clement T.Yu: ICDE, CIKM, VLDB, SIGMOD, DataKnlEng | AnHai Doan: SIGMOD | A. Prasad Sistla: ICDE | Abdur Chowdhury: SIGMOD | Fang Liu: SIGMOD)⟩, $\{[90-93],[94-97]\}\big)$.

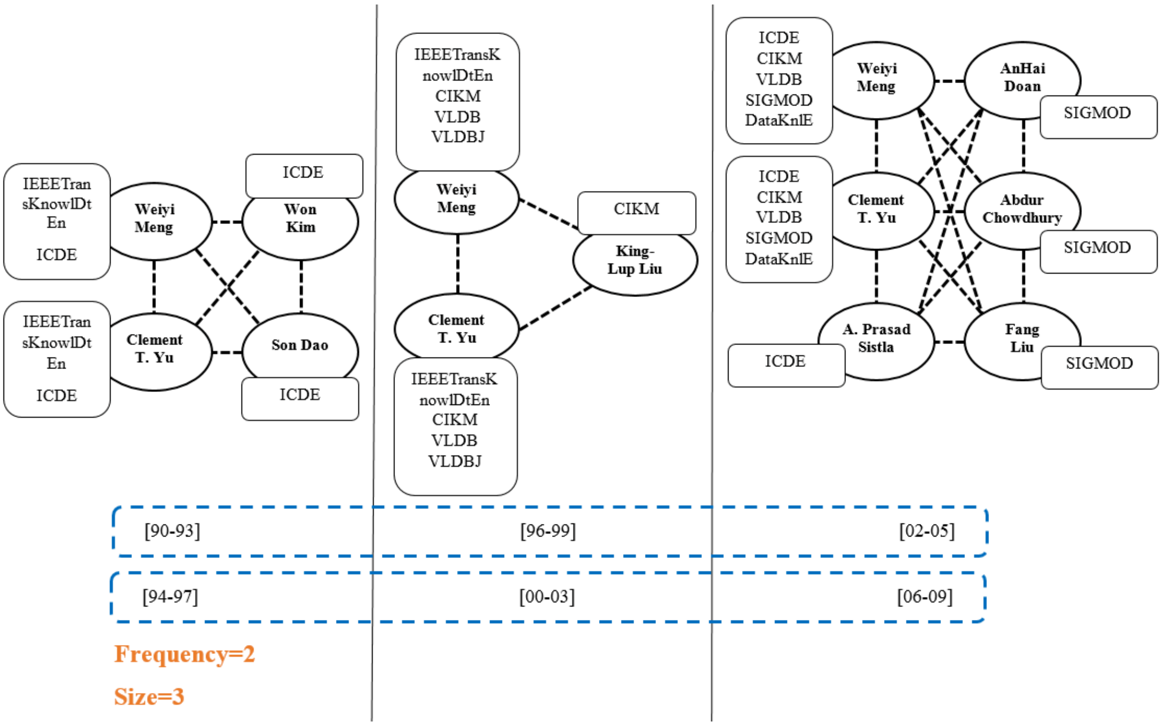


Figure 3.30 – Third pattern extracted from DBLP with the parameters $minvol = 2$, $minsup = 2$, $gap = 3$ $mincos = 0.4$ and $mincom = 2$

With a gap threshold of 3, the pattern highlights publication evolution of this co-author network in a shorter term. This sequence of size 3 represents an evolution over 3 timestamps. It is repeated two times from 1990 to 2005 (i.e. timesptamps [90-93],[96-99] and [02-05]) and from 1994 to 2009 (i.e. timesptamps [94-97], [00-03] and [06-09]). Firstly, we analyse the evolution of this group of authors at co-authorship level, we can see that at first Weiyi Meng and Clement T. Yu worked together with two authors Won Kim and Son Dao. Then, they changed their co-authorship by publishing with King-Lup Liu. From 2002 to 2009, they did not publish with King-Lup Liu any more and began to work with AnHai Doan, A. Prasad Sistla, Abdur Chowdhury and Fang Liu. In addition, we can focus on the evolution of their publications in different conferences/journals over time. Between 1990 and 1993, Weiyi Meng and Clement T. Yu published with Won Kim and Son Dao in ICDE conference. Besides, they also published together in IEEETransKnowlDtEn. Then from 1996 to 1999, Weiyi Meng and Clement T. Yu produced much more articles together, they published in IEEETransKnowlDtEn, VLDB and VLDBJ. They also published an article in CIKM with King-Lup Liu. From 2002 to 2005, this pair of co-authors continued to publish together in CIKM and VLDB conferences. Besides, they also published with A. Prasad Sistla in ICDE and co-authored with AnHai Doan, Abdur Chowdhury and Fang Liu in SIGMOD conference. This pattern also appears another time (from 1994 to 2009).

Fig. 3.31 depicts another example of pattern

$\big(\langle$(Myoung-Ho Kim: DEXA, DASFAA, CIKM | Jae Soo Yoo: DEXA | Yoon-Joon Lee: DEXA, DASFAA | Jae-Woo Chang: DEXA)

(Myoung-Ho Kim: DASFAA, DataKnlEng | Jae Soo Yoo :DASFAA, DataKnlEng | Yoon-Joon Lee: DASFAA)

(Myoung-Ho Kim: JIntellInfSys | Jae Soo Yoo : JIntellInfSys)$\rangle, \{[90-93], [94-97]\}\big)$

with thresholds $minvol = 2$, $minsup = 2$, $gap = 3$ $mincos = 0.5$ and $mincom = 2$. This pattern depicts the evolution of a co-author network for Myoung-Ho Kim and Jae Soo Yoo. This is a size 3 sequence which represents an evolution over 3 timestamps. This sequence is repeated twice: once from 1990 to 2005 (i.e. timesptamps [90-93], [96-99] and [02-05]) and again from 1994 to 2009 (i.e. timesptamps [94-97], [00-03] and [06-09]). Firstly, we analyse the evolution of this group of authors over time at co-authorship level. We can observe that at first, Myoung-Ho Kim and Jae Soo Yoo worked together with Jae-Woo Chang. Next, they changed their co-authorship by publishing together with Yoon-Joon Lee. Then, they did not publish with Yoon-Joon Lee any more. Regarding the evolution of their publications in different conferences/journals over time, we can make the following remarks. Between 1990 and 1993, Myoung-Ho Kim,Jae Soo Yoo, Jae-Woo Chang and Yoon-Joon Lee published together in DEXA. Besides, Myoung-Ho Kim and Yoon-Joon Lee co-authored in DASFAA. Then between 1996 and 1999, Myoung-Ho Kim, Jae Soo Yoo and Yoon-Joon Lee published together in DASFAA. Moreover, Myoung-Ho Kim and Jae Soo Yoo had publications in DataKnlEng. From 2002 to 2005, Myoung-Ho Kim and Jae Soo Yoo published together in the JIntellInfSys journal. This pattern also appears again (from 1994 to 2009). It shows Myoung-Ho Kim and Jae Soo Yoo's preference of conferences/journals as well as their publication strategy. Overall we could notice that researchers often firstly publish articles in conferences, and then publish their work in journals later.

Discussions on results of DBLP dataset Here we compare our results with patterns extracted in (Desmier *et al.*, 2012). Firstly, **RPMiner** extracts all possible evolutions of subgraphs (connected components). In the preprocessing stage, we extract more than 100 connected components from the first graph, where each component represents a co-authors network. That means there are at least more than 100 different network evolutions. However, (Desmier *et al.*, 2012) can extract maximum 12 co-evolution patterns which depict only 12 different co-evolutions of co-author networks. It is because (Desmier *et al.*, 2012) consider only evolution of the same set of vertices over time. In comparison, **RPMiner** extracts all possible evolutions (more than 9000 recurrent patterns) of different components (co-author networks) because we consider evolutions of different vertices. For example, new co-authors can join a group or previous co-authors may no longer publish together. Secondly, we can notice that co-evolution patterns focus only on patterns having the same attributes over time. However, for DBLP dataset and many other real world datasets, attributes may also change over time. For example, the trend of authors' publications number may evolve over time. To summarize, our algorithm **RPMiner** breaks through these two main limits by considering evolutions of both vertices and attributes.

US Flight dataset For the US Flight dataset, extracted patterns highlight the impact of hurricanes on US airport traffic. Fig. 3.32 shows an example of pattern. This pattern shows

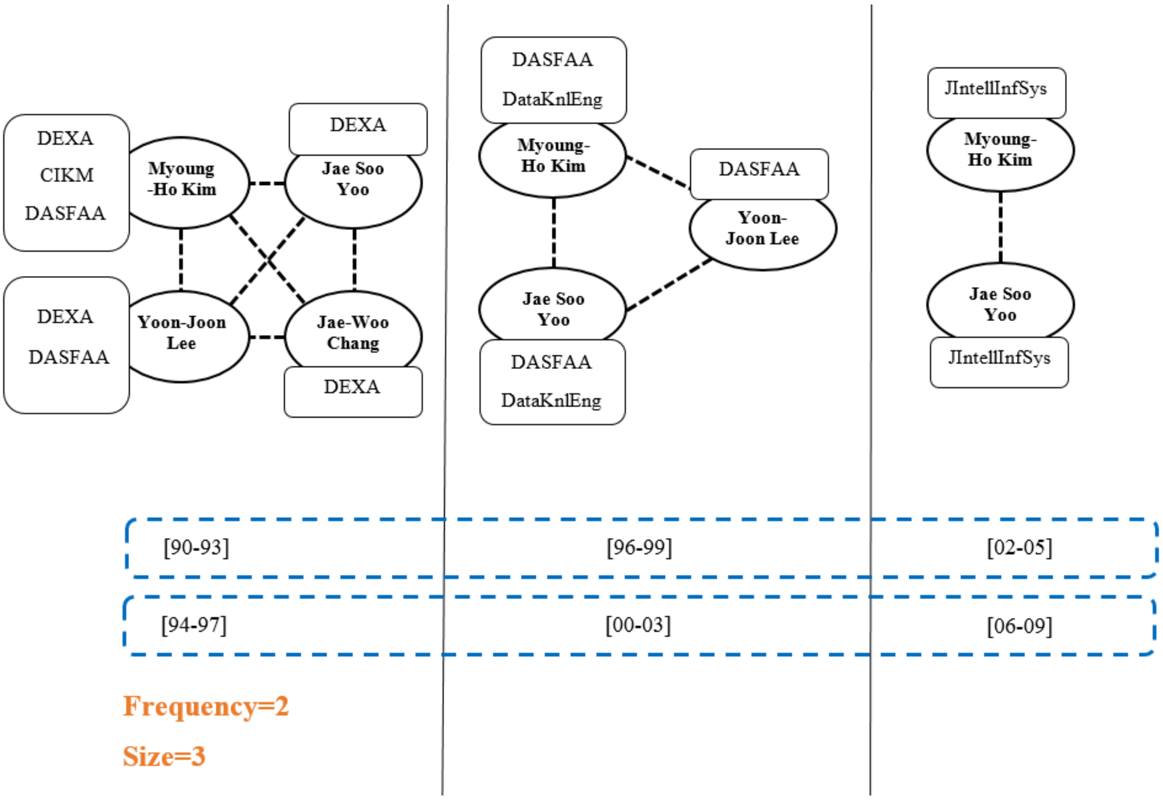


Figure 3.31 – Forth pattern extracted from DBLP with the parameters $minvol = 2$, $minsup = 2$, $gap = 3$ $mincos = 0.4$ and $mincom = 2$

the impact of hurricanes on cancellation, diverted flights, mean delay of departure/arrival and ground waiting time departure/arrival. At first, cancellations, diverted flights and delays increased. Then, cancellations diverted flights and delays decreased the following week when the hurricane became weaker. We can observe that this pattern occurred three times: [08/08-15/08], [22/08-29/08] and [05/09-12/09] which does not correspond to the period of Category 5 Katrina hurricane (as shown in Fig. 3.35), i.e., from 23/08 to 31/08). Actually just before Katrina hurricane, a category 2 hurricane called Irene (as shown in Fig. 3.34) hit the East Coast of United States from 04/08 to 18/08. Moreover, just after Katrina hurricane, another category 1 hurricane, called Ophelia (as shown in Fig. 3.36), reached in the East Coast of United States from 06/09 and disappeared on 17/09. These three periods correspond exactly to the pattern beginning times and we could notice that all hurricanes stronger than category 1 could result in the cancellations and delays of flights. Their influences decrease when hurricanes become weaker. For readability of extracted patterns, we repost its description in Table 3.1 of which first column represents name of airport, the following columns represent attributes trends is consecutive time steps. If a table cell is empty, it means that corresponding airport is not included in this connected component (a set of airports connected by flights). As we can see in Fig. 3.33, this pattern contains 13 airports (in blue) all over the United States where 10 over 13 airports are located in the East Coast of United States (region strongly affected by the three hurricanes). The other three airports

(Providence, Reno and Rochester) are located in the West Coast of United States (far away from the region strongly affected by Irene hurricane). However, these three airports were also affected by these hurricanes. As we can see from Table 3.1, the canceled flights (from airport Providence and airport Rochester to East Coast airports) increased in the first week. It is probably because the hurricanes were too strong in the destinations that they have to cancel the flights in the airports of departure.

Table 3.1 – First recurrent pattern extracted from US Flight dataset

Airport	First time step	Second time step
Pittsburgh	Canceled+, Diverted=, WaitingTimeArrival+	Canceled=, Diverted-, WaitingTimeDeparture=
Portland	Canceled+, DelayDeparture+	DelayDeparture-, WaitingTimeDeparture-
Bend/ Redmond	Canceled=, Diverted=, WaitingTimeDeparture=, WaitingTimeArrival=	Diverted=, WaitingTimeDeparture=
Raleigh/ Durham		DelayArrival-, WaitingTimeArrival-
Richmond	Diverted=, DelayDeparture+	WaitingTimeDeparture=
Roanoke	Diverted=, DelayDeparture+	
Savannah	Canceled+, DelayArrival+	
Louisville	Diverted=	
San Diego		Diverted=
Shreveport	Diverted=	Diverted=, DelayDeparture-
Providence	Canceled+, Diverted=, WaitingTimeDeparture+	Diverted-, WaitingTimeDeparture=, WaitingTimeArrival-
Reno		WaitingTimeDeparture=, WaitingTimeArrival=
Rochester	Canceled+	

Fig. 3.37 shows another example of pattern. This pattern shows the impact of hurricanes on canceled/diverted flights and delays. First, the number of canceled flights and delays increased at these airports. Then, cancellations and delays decreased the following week when the hurricane became weaker. This pattern occurred at the beginning of August and then again at the beginning of September, because there were two hurricanes during this period. As we can see in Fig. 3.38, this pattern contains 11 airports (in blue) all over the United States where 9 over 11 airports are located in the East Coast of United States which is the region strongly affected by Irene hurricane and Ophelia hurricane. The other two airports Medford and Minot (located far away from East Coast of United States) were also affected by the hurricanes. Moreover, we note also that cancellations and delays increased when hurricanes came, while diverted flight remained always the same. It shows that hurricanes have strong impact on cancellations but have hardly impact on diverted flight. It may

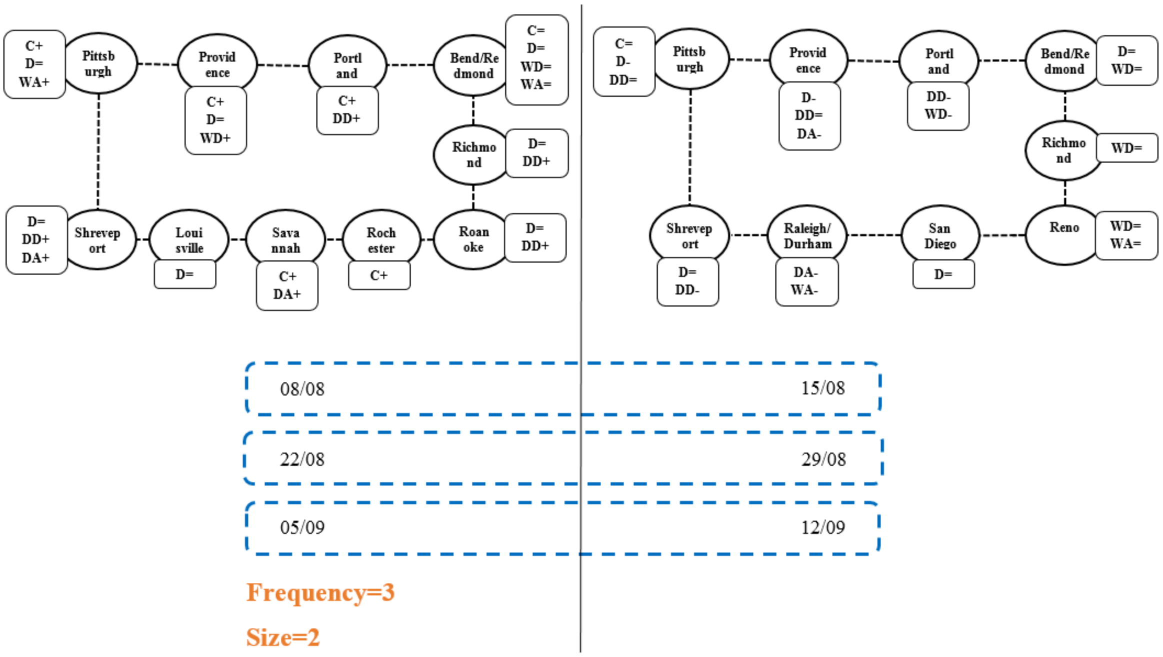


Figure 3.32 – First pattern extracted from Domestic US Flight dataset with the parameters $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0.4$ and $mincom = 2$. C: cancellation, D: diverted flights, DD: the mean delay of departure, DA: the mean delay of arrival, WD: the ground waiting time departure, WA: the ground waiting time arrival

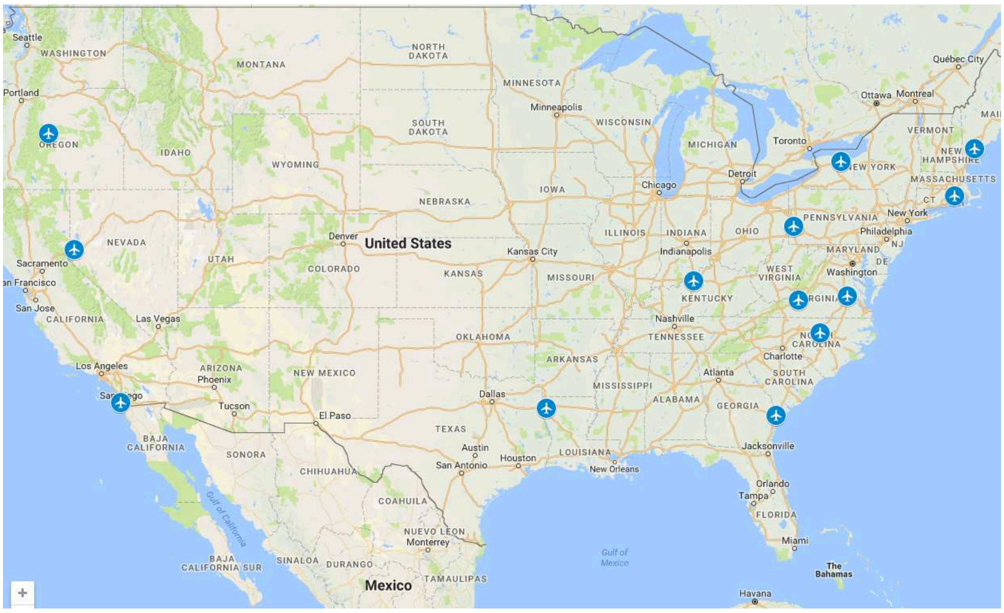


Figure 3.33 – First pattern extracted from Domestic US Flight dataset with the parameters $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0.4$ and $mincom = 2$

be due to the following three reasons: (1) Delays and cancellations are the most frequent result of this disruptive event. For the airlines, these are the least damaging in terms of

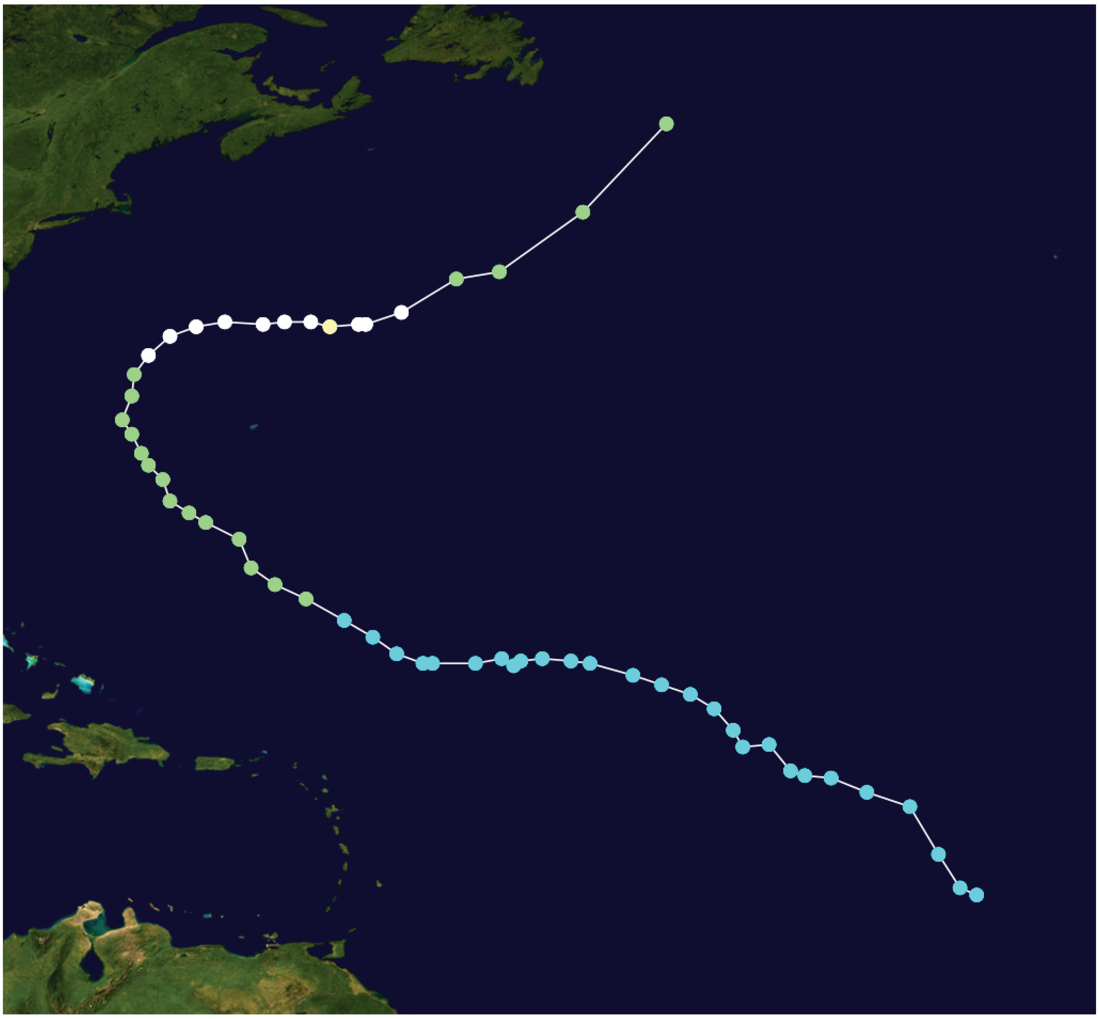


Figure 3.34 – The Irene' track (from 04/08/2005 to 18/08/2005) (Nilfanion, 2005a)

creating schedule disruption and customer dissatisfaction. Aircraft are at least located at a point that was included in their itineraries and the crews assigned to them are likely to be available for services. If the delayed or the cancelled flight is at a hub, Emirates airline have the alternatives for accommodating affected passengers and services (Abdi and Sharma, 2007). (2) Diversions can be extremely disruptive to an airline because they put aircraft, air crews, passengers, and baggage out of place. Passengers who were waiting to board the diverted aircraft also need to be accommodated (Wright, 2010). (3) Air company have to pay higher than usual price for fuel, catering services and access to gates at which it can off-load passengers. In the worst case, flights wait in the air (holding in air) till the weather is suitable for landing (Abdi and Sharma, 2007). So traffic managers usually cancel directly the flights other than not cancel the flights but take the risk to divert the flights when there are strong hurricanes.

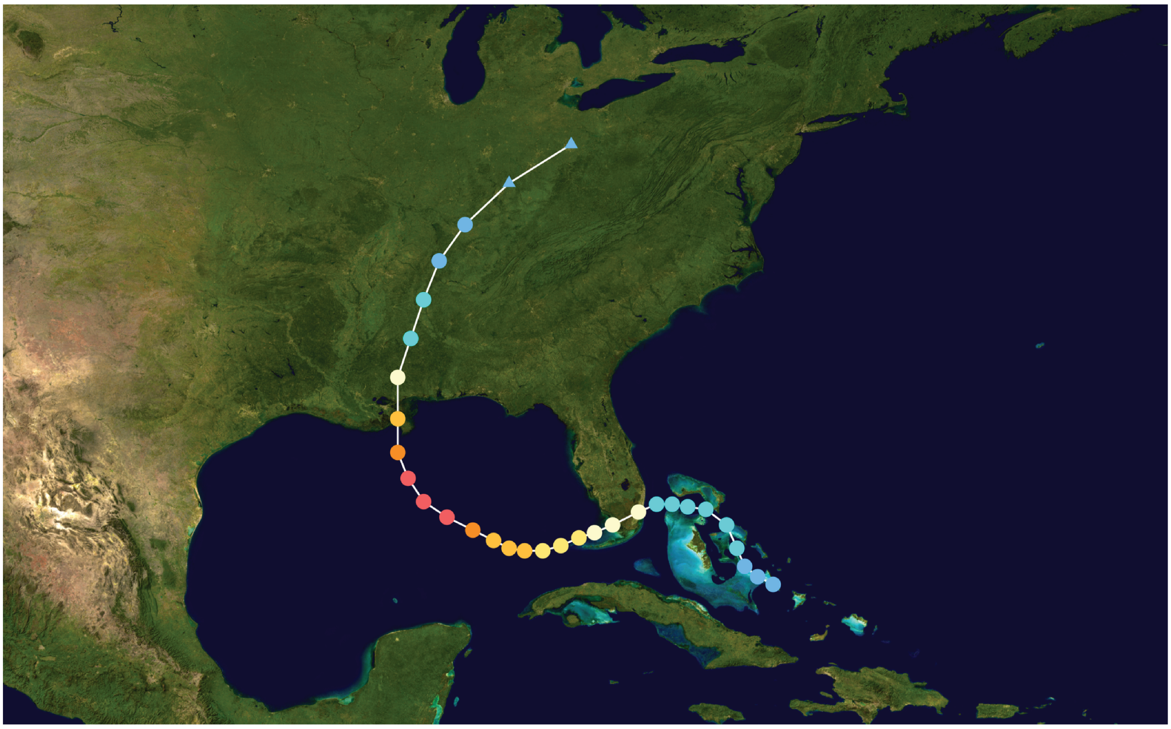


Figure 3.35 – The Katrina' track (from 23/08/2005 to 31/08/2005) (Nilfanion, 2005b)

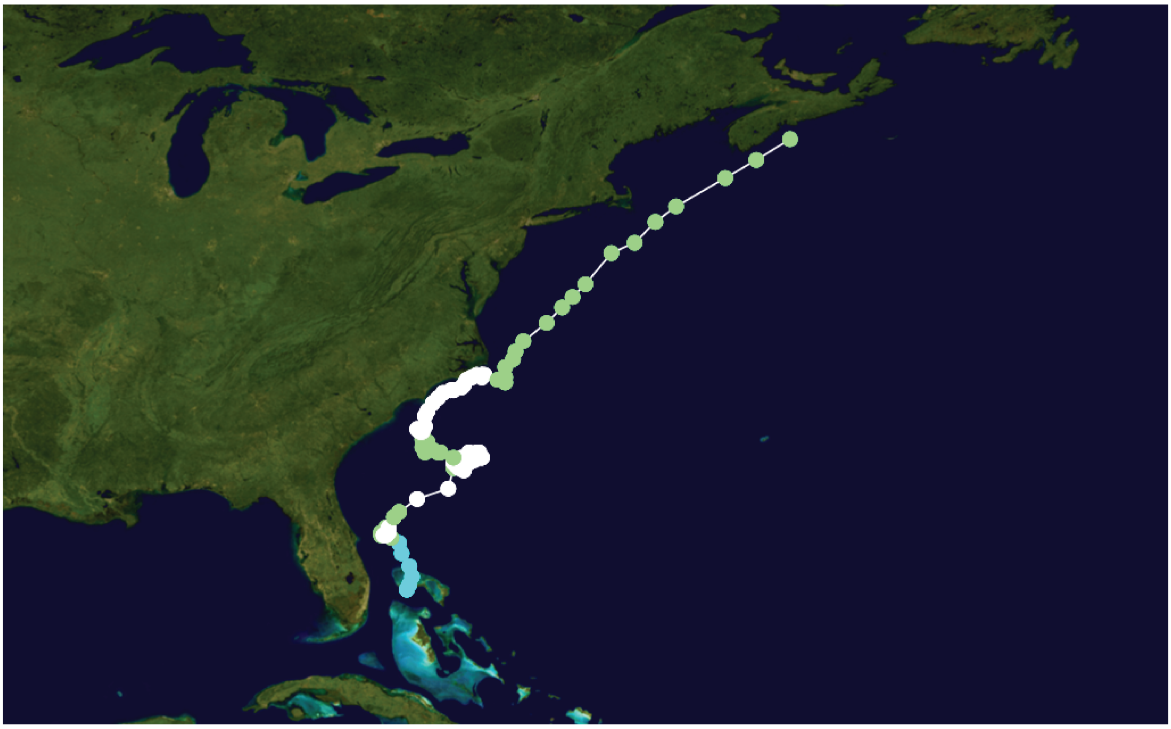


Figure 3.36 – The Ophelia' track (from 06/09/2005 to 17/09/2005) (Nilfanion, 2005c)

Table 3.2 – Second recurrent pattern extracted from US Flight dataset

Airport	First time step	Second time step
New York	Canceled+, Diverted=, DelayDeparture+, DelayArrival+, WaitingTimeDeparture+, WaitingTimeArrival+	Canceled-, Diverted=, DelayDeparture-, WaitingTimeDeparture-
Orlando	Diverted=, DelayDeparture+, DelayArrival+, WaitingTimeArrival+	Diverted=, DelayDeparture-, DelayArrival-
Chicago	Diverted=, DelayDeparture+, DelayArrival+, WaitingTimeArrival+	DelayDeparture-, DelayArrival-, WaitingTimeDeparture-
Meridian	Canceled+, DelayDeparture+, DelayArrivals+, WaitingTimeDeparture=, WaitingTimeArrival+	Canceled=, Diverted-
Medford	Canceled=, Diverted=, DelayDepartures+, DelayArrivals+, WaitingTimeDeparture=, WaitingTimeArrival+	Diverted=, WaitingTimeDeparture=, WaitingTimeArrival-
Montgomery	Diverted=, DelayDeparture+, WaitingTimeDeparture=	WaitingTimeDeparture=, WaitingTimeArrival=
Miami	Canceled+, DelayDeparture+, DelayArrival+, WaitingTimeDeparture=, WaitingTimeArrival+	Diverted-, DelayDeparture-, WaitingTimeDeparture=
Moline	Canceled+, Diverted=, WaitingTimeArrival=	Diverted=
Monroe	Diverted=, DelayDeparture+, DelayArrival+	Diverted=
Minot	Diverted=, DelayDeparture+, WaitingTimeDeparture=	Diverted=
Marquette	Canceled=	Canceled=, Diverted=, DelayDeparture-

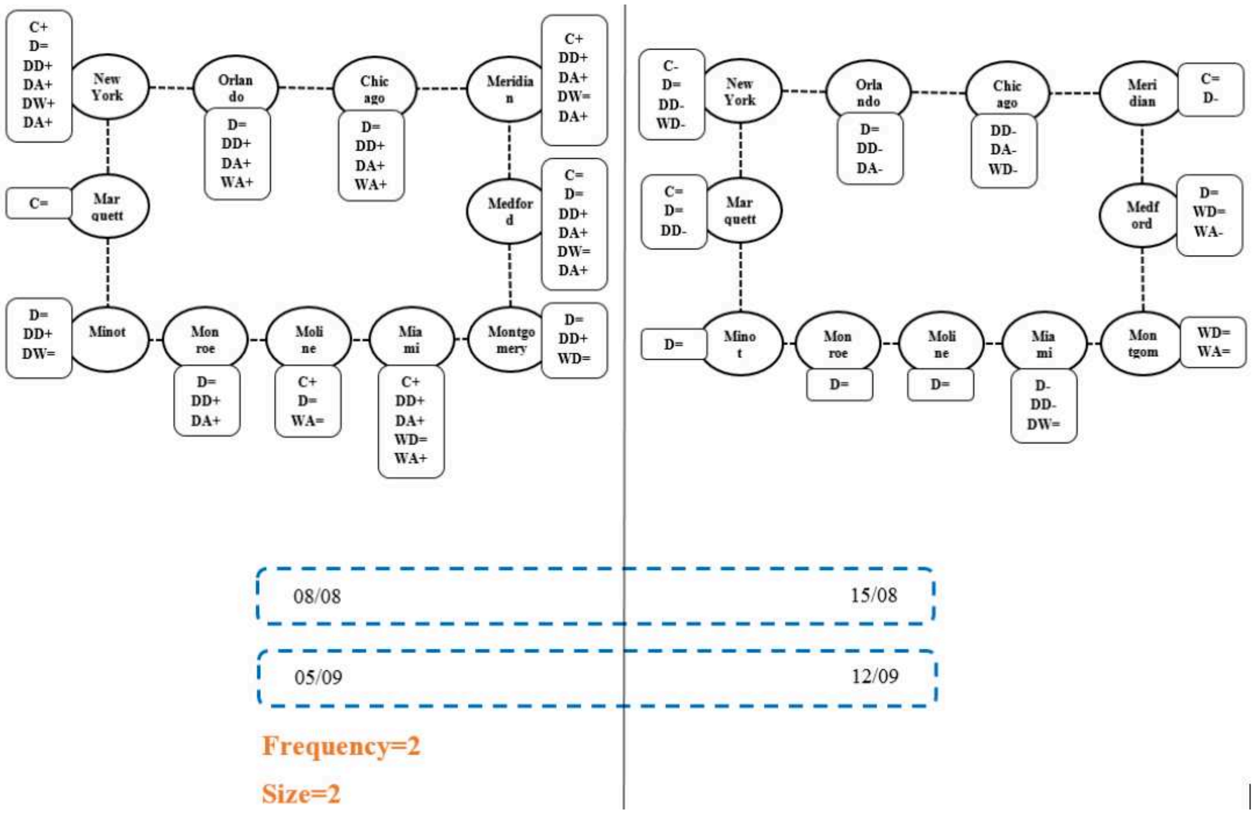


Figure 3.37 – Second pattern extracted from Domestic US Flight dataset with the parameters $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0.4$ and $mincom = 2$. C: cancellation, D: diverted flights, DD: the mean delay of departure, DA: the mean delay of arrival, WD: the ground waiting time departure, WA: the ground waiting time arrival

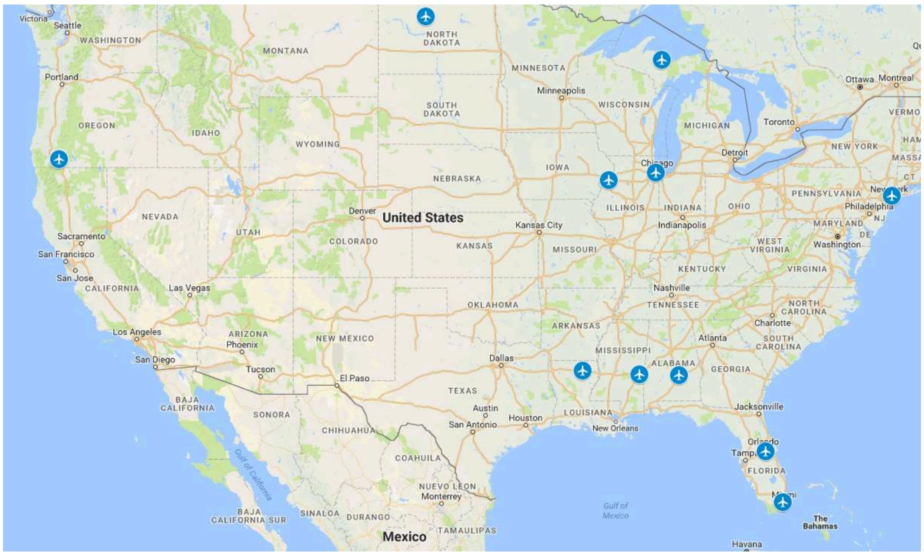


Figure 3.38 – Second pattern extracted from Domestic US Flight dataset with the parameters $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0.4$ and $mincom = 2$

Chapter 4

Application to spatio-temporal data analysis

Contents

In this part, we propose to study a real-world problem in which we will use spatio-temporal patterns to analyze trends and evolutions of interesting objects (aquaculture ponds). This work was conducted in the framework of the "INDESO" project, which is dedicated to developing tools and methods to better manage marine and coastal resources in Indonesia. This project is financed by the Indonesian Ministry of Marine Affairs and Fisheries (KKP) and coordinated by CLS (Collecte Localisation Satellites). This work was done in collaboration with Niken Financia GUSMAWATI (as a part of her thesis), and Hugues Lemonnier and Benoit Soulard of the LEAD/IFREMER team .

In that context, we developed a complete KDD process (Fig. 4.1): from pre-processing to visualization and interpretation of results. In that task, input data is composed of a satellite image time series crossed with ground truth data generated by experts.

The data preparation step required a major contribution and use of image analysis methods. Depending on objects to be detected, identification of interesting objects to analyse could be very complex and require to adapt the most efficient segmentation methods.

In the data mining step, we have chosen two types of pattern domains:

1. Sequential patterns, which permit to describe frequent or rare temporal evolutions of objects without taking account relations between objects.

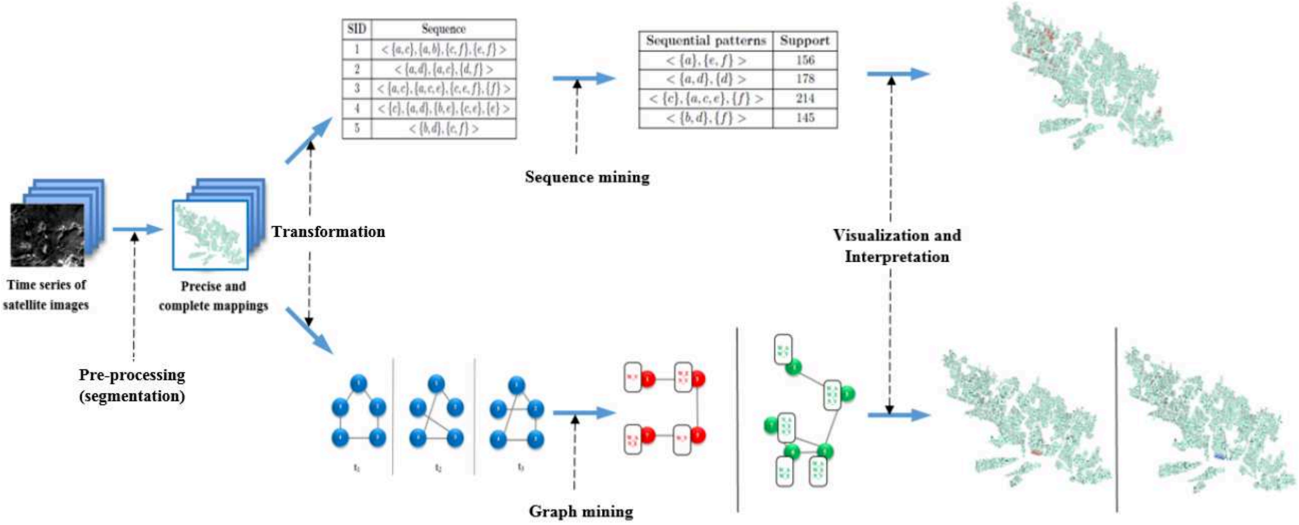
2. Our method **RPMiner**. It permits to mine recurrent patterns in a dynamic attributed graph. This new pattern domain allows to efficiently study recurrent evolutions of a set of objects which are closely connected.

Image data transformation is necessary for the data mining phase. On the one hand, it permits to transform information, i.e. temporal images extracted by object identification and calculation of their characteristics, into transactional data when using sequential patterns. On the other hand, it allows to construct a sequence of attributed graphs representing objects, their spatial relationships and their characteristics.

The pattern visualization phase (sequential and recurrent patterns) occurs naturally in images, by identifying objects and following their evolution. This step provides to experts with the possibility to choose a pattern from the list of extracted patterns, and and visualizes objects (in images by date) corresponding to that pattern evolution.

1. Problematic

In the past few years, Indonesia's production in shrimp farming sector has experienced strong growth due to large expansion of areas. This activity contributes to the national income and optimize food security. However, its development has generated negative ecological and social effects. As consequences of diseases and environmental degradations, 250,000 ha of ponds have been abandoned. Sustainable practice in aquaculture farming is thus a high priority for the Indonesian government, in order to reduce those impacts. Consequently, useful tools need



SID	Sequence
1	$<\{a,c\},\{a,b\},\{c,f\},\{e,f\}>$
2	$<\{a,d\},\{a,c\},\{d,f\}>$
3	$<\{a,c\},\{a,c,e\},\{c,e,f\},\{f\}>$
4	$<\{c\},\{a,d\},\{b,e\},\{c,e\},\{e\}>$
5	$<\{b,d\},\{c,f\}>$

Sequential patterns	Support
$<\{a\},\{e,f\}>$	156
$<\{a,d\},\{d\}>$	178
$<\{c\},\{a,c,e\},\{f\}>$	214
$<\{b,d\},\{f\}>$	145

Time series of satellite images
Pre-processing (segmentation)
Precise and complete mappings
Transformation
Sequence mining
Visualization and Interpretation
Graph mining
t_1
t_2
t_3

Figure 4.1 – Complete KDD process to study evolutions of aquaculture ponds

to be developed to improve aquaculture farming sustainability, coastal resource preservation and human activity management.

We have contributed to this project by proposing automatic analysis tools integrated in a complete KDD process. The objective is to monitor aquaculture pond evolution. Fig. 4.2 shows details of this KDD process. Firstly, we designed a process to provide a complete and precise aquaculture mapping by segmenting and classifying satellite images. Secondly, we developed methods to identify pond attributes (vegetation, water etc). Thirdly, we proposed methods to transform this satellite image time series into sequential dataset and a dynamic attributed graph. Next, we applied two different kinds of algorithms to study aquaculture pond evolution: a sequential mining algorithm which aims to study temporal evolutions; and our algorithm **RPMiner** which permits to study both spatial and temporal evolutions of aquaculture ponds. Finally, we visualized results on original satellite images. When interpreted results permit to describe, understand and manage shrimp farming.

2. Data description

The dataset is composed of fourteen very high-resolution images of Perancak estuary (located in Bali province, Indonesia) taken between 2001 and 2015. Perancak estuary system is almost 5 km long and covers an area of approximately 1800 ha. It is located at northwest of Denpasar in Bali Province. This area was chosen because it was a complex zone consisting of aquaculture ponds of various sizes and types. They included active shrimp ponds in traditional, semi-intensive and intensive culture systems, abandoned shrimp ponds with water, without

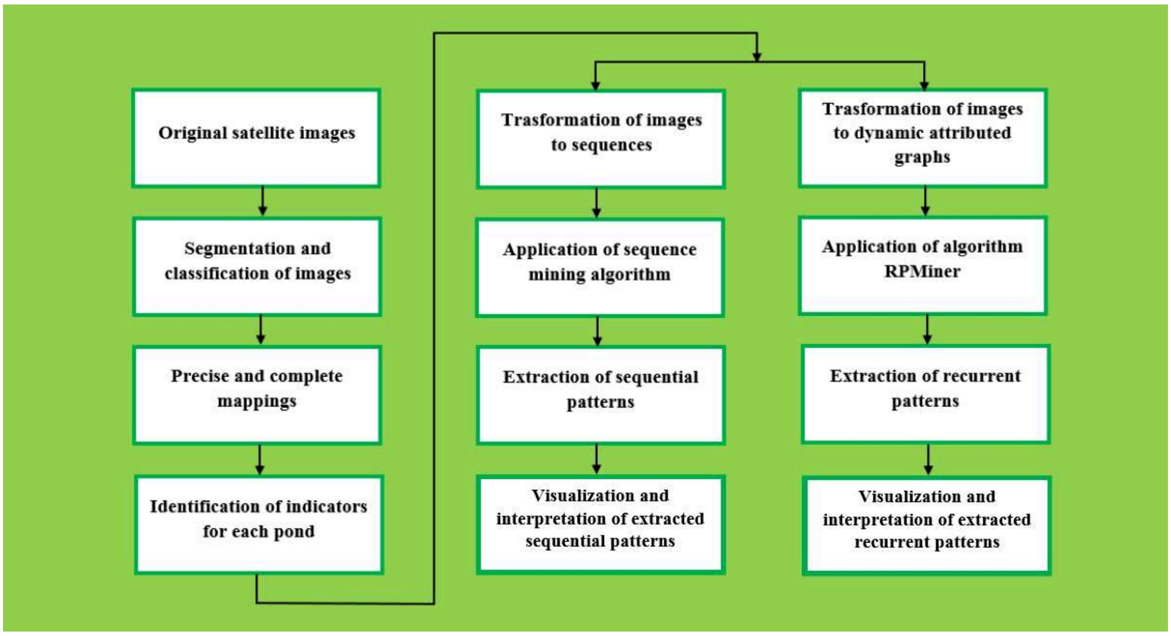

Original satellite images
Segmentation and classification of images
Precise and complete mappings
Identification of indicators for each pond
Trasformation of images to sequences
Application of sequence mining algorithm
Extraction of sequential patterns
Visualization and interpretation of extracted sequential patterns
Trasformation of images to dynamic attributed graphs
Application of algorithm RPMiner
Extraction of recurrent patterns
Visualization and interpretation of extracted recurrent patterns

Figure 4.2 – Complete process to study evolutions of aquaculture ponds

water, with natural vegetation and mangrove plantation conditions, fish ponds, and polyculture ponds (algae/fish and shrimp). Those optical images came from five different satellite sensors, namely IKONOS (images acquired in 12/10/2001, 09/03/2002, 21/02/2003, and 27/06/2003), Quickbird (22/09/2007, 19/07/2008, and 09/07/2009),Worldview-2 (16/08/2010, 15/04/2011, 23/10/2012, 10/12/2013, and 26/03/2014), GeoEye-1 (11/10/2014), and Worldview-3 (16/04/2015). Pixel size varied from 30 cm to 1m for panchromatic channels whereas resolution of multispectral channels ranged between 1.2 and 4 m.

3. Identification of aquaculture ponds

An automatic precise, and efficient generation of maps is important for experts because (1) to provide spatiotemporal information to decision support tools for sustainable fisheries policy, (2) experts could use them to develop indicators to monitor and assess ecosystems (Revenga, 2005).

In this section, the objective is to identify aquaculture ponds and generate a corresponding cartography by using segmentation and classification methods. (Burnett and Blaschke, 2003) proposed a segmentation and classification method using Object-Based Image Analysis (OBIA). This approach is particularly suitable for analyzing medium and high resolution satellite images. Its mapping capability has been extended by incorporating spectral, contextual, textural and shape information of homogenous pixel sets (Meinel *et al.*, 2001; Shackelford and Davis, 2003; Blaschke, 2010). However, OBIA faces some challenges: (1) within-class heterogeneity and irrelevant features may increase classification uncertainty (Kim *et al.*, 2011; Chettri *et al.*, 2013; Dronova *et al.*, 2015).(2) OBIA classification can hardly detect and delineate fine-scale elements even for very high resolution satellite images (Yoshino *et al.*, 2014).

On the other hand, traditional Pixel-Based Image Analysis (PBIA) has encountered issues with high-resolution imagery, resulting in 'a salt and pepper appearance' that leads to very general land cover information, or limited accuracy in thematic maps (Zhu *et al.*, 2000).

In the present work, two segmentation and classification methods were firstly applied to build the cartography of ponds (Gusmawati *et al.*, 2016): (1) Region Growing Segmentation algorithm followed by ISOSEG unsupervised classification method (RGT) implemented in SPRING software and (2) Isocluster Unsupervised Classification method (IUC) implemented in ArcGIS software. RGT method, which has been widely used in many remote sensing applications, can extract closed contours (Espindola *et al.*, 2006). It permits to extract water surface and pond embankment limits (Virdis, 2014). IUC has been used to classify aquaculture zones and generate aquaculture farm cadastre (Hossain *et al.*, 2002). However, these existing methods could not provide an accurate mapping. Broken structure of embankments, low contrast between soil and pond embankments in dried-up ponds, ongoing pond development, abundance of algae as well as mangrove vegetation inside ponds made it difficult to extract enclosed contours with high accuracy. To provide an accurate automatic mapping, we proposed a method called Edge Detection Based (EDB) segmentation (Gusmawati *et al.*, 2016).

3.1 IUC Method

IUC is a segmentation and classification method composed of three steps:

Image pre-processing and classification After contrast enhancement, several classes of land cover are identified in the multi-spectral image using Iso Cluster, without knowledge of class type. These classes are re-classified into pond and non-pond (2 classes) to generate two unique labels. Iso Cluster algorithm (K-means) is an iterative process that assigns each candidate cell to a cluster based on the minimum Euclidean distance. The process starts with arbitrary means, one for each cluster (users dictate the number of clusters). Every cell is assigned to the cluster with the closest mean. Next, new means are recalculated for each cluster, based on attribute distances of cells belonging to the cluster after the first iteration. The process is repeated: each cell is assigned to the closest mean in multidimensional attribute space, and new means are calculated for each cluster based on the new member cells. After running the specified number of iterations, the migration of cells from one cluster to another is minimal; therefore, all the clusters become stable.

Automatic vectorization ArcScan, as an extension tool in ArcGIS, that provides automatic vectorization. It is an of outline vectorization which generates vector polygon features on raster cell borders.

Post-vectorization refinement We can use other editing tools, such as topology, advanced editing, and spatial adjustment, to further refine vectorization results, whenever necessary. For instance, several connected ponds which might appear in results could be separated by manual editing.

Fig. 4.3 shows steps for aquaculture ponds detection using IUC method.

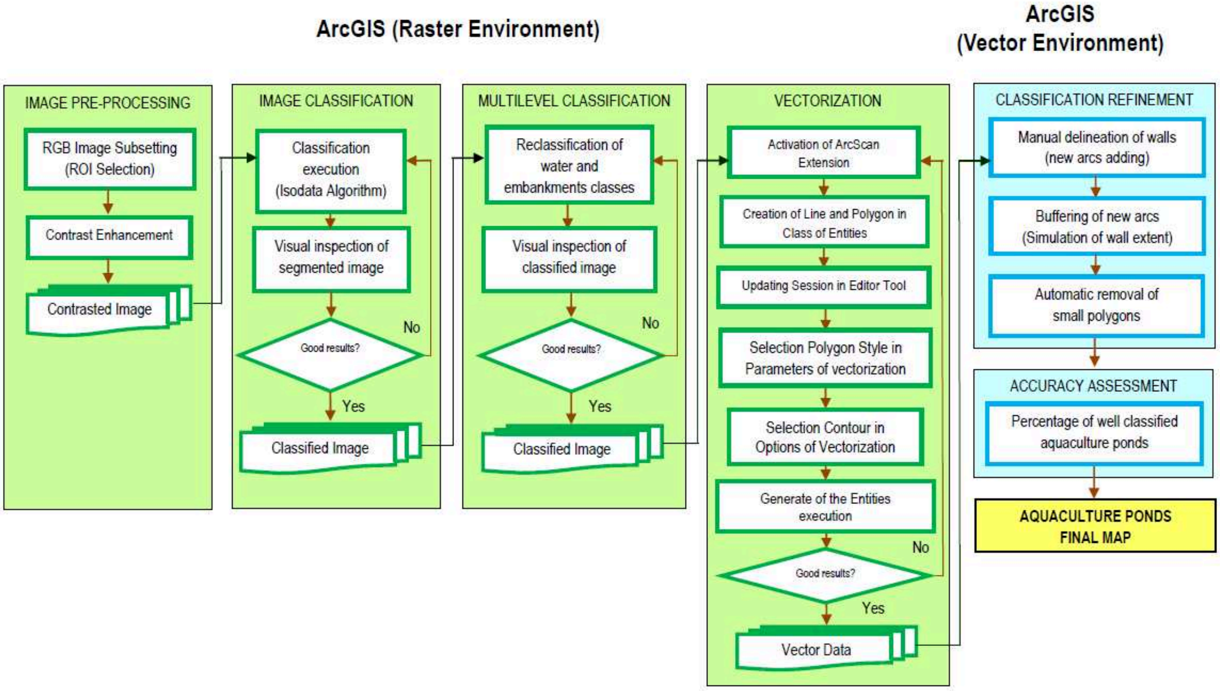

ArcGIS (Raster Environment)
ArcGIS (Vector Environment)
IMAGE PRE-PROCESSING
RGB Image Subsetting (ROI Selection)
Contrast Enhancement
Contrasted Image
IMAGE CLASSIFICATION
Classification execution (Isodata Algorithm)
Visual inspection of segmented image
Good results?
No
Yes
Classified Image
MULTILEVEL CLASSIFICATION
Reclassification of water and embankments classes
Visual inspection of classified image
Good results?
No
Yes
Classified Image
VECTORIZATION
Activation of ArcScan Extension
Creation of Line and Polygon in Class of Entities
Updating Session in Editor Tool
Selection Polygon Style in Parameters of vectorization
Selection Contour in Options of Vectorization
Generate of the Entities execution
Good results?
No
Yes
Vector Data
CLASSIFICATION REFINEMENT
Manual delineation of walls (new arcs adding)
Buffering of new arcs (Simulation of wall extent)
Automatic removal of small polygons
ACCURACY ASSESSMENT
Percentage of well classified aquaculture ponds
AQUACULTURE PONDS FINAL MAP

Figure 4.3 – Aquaculture ponds detection using IUC method

3.2 RGT Method

RGT a segmentation and classification method composed of four steps:

Image pre-processing and filtering The panchromatic image was spatially filtered by a sliding kernel with one iteration, to enhance the contrast.

Image segmentation RGT method was performed with different similarity criteria and minimum area (pixels) thresholds to achieve optimal segmented output. Similarity controls grouping of similar pixels in a segment (region) while the minimum area threshold filters smallest segments (areas).

Image classification A clustering algorithm ISOSEG available in SPRING software (Iterative Self Organizing Data Analysis Technique) allows to assign each image segment to different classes. ISOSEG initially assigns the same class to all segments, and then reduces variability by creating new classes. Several thresholds values and various iteration numbers are tested. Classification output is converted into vector layers.

Post-vectorization refinement After classification and vectorization, manual refinement using editing tools can be conducted using ArcGIS, as for IUC post-vectorization.

Fig. 4.4 shows steps for aquaculture ponds detection using RGT method.

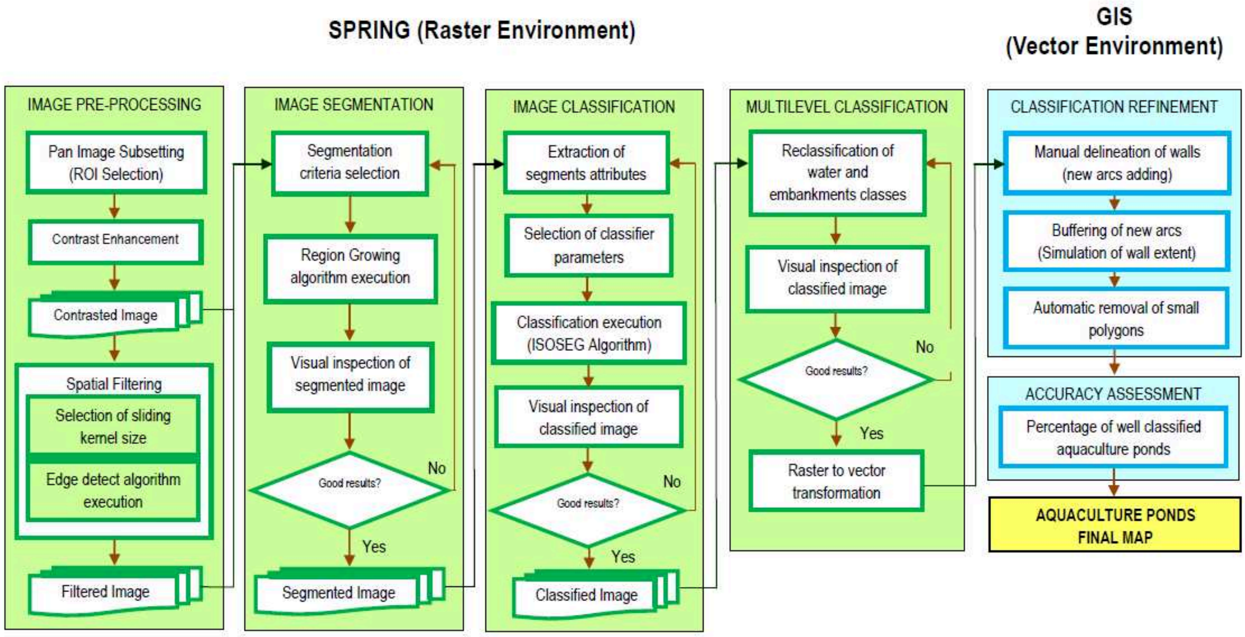

SPRING (Raster Environment)
GIS (Vector Environment)
IMAGE PRE-PROCESSING
Pan Image Subsetting (ROI Selection)
Contrast Enhancement
Contrasted Image
Spatial Filtering
Selection of sliding kernel size
Edge detect algorithm execution
Filtered Image
IMAGE SEGMENTATION
Segmentation criteria selection
Region Growing algorithm execution
Visual inspection of segmented image
Good results?
No
Yes
Segmented Image
IMAGE CLASSIFICATION
Extraction of segments attributes
Selection of classifier parameters
Classification execution (ISOSEG Algorithm)
Visual inspection of classified image
Good results?
No
Yes
Classified Image
MULTILEVEL CLASSIFICATION
Reclassification of water and embankments classes
Visual inspection of classified image
Good results?
No
Yes
Raster to vector transformation
CLASSIFICATION REFINEMENT
Manual delineation of walls (new arcs adding)
Buffering of new arcs (Simulation of wall extent)
Automatic removal of small polygons
ACCURACY ASSESSMENT
Percentage of well classified aquaculture ponds
AQUACULTURE PONDS FINAL MAP

Figure 4.4 – Aquaculture ponds detection using RGT method

3.3 EDB Method

EDB is an automatic method composed of four steps:

Image pre-processing and filtering Panchromatic image is firstly filtered using a Gaussian filter to remove noise and obtain a smooth image.

Edge detection based segmentation and adaptive thresholds In this part, an improved Canny edge detection method is employed to process the panchromatic image. Canny algorithm introduces two thresholds, which allows significant adaptation to local content in images. The higher threshold (Th) is calculated by Ostu method and the lower threshold (Tl) is determined by using Canny initial formula $Tl = 0.5Th$, which guarantees contour completeness and accuracy.

Pixel values above Th are classified as edge whereas pixel values lower than Tl are discarded. Pixel value between those thresholds would be recognized as contours if they are already connected to an accepted pixel already classified as contour.

Color based segmentation and image fusion For the purpose of extracting drained ponds, a color classification using a color threshold of 90 is performed using Hue-Saturation-Value (HSV) system. Segmented panchromatic and segmented multispectral images are then merged to get a complete map of ponds.

Shape Recognition and Image Classification A shape recognition method is then applied to filter noise and other objects such as rivers and vegetation. An area threshold ranging from 300 to 10,000 pixels has been used to identify aquaculture ponds. Moreover, a shape factor (SF), derived from the ratio of perimeter to area showed promising results in identifying aquaculture ponds based on their elongated shape ($SF \geq 1.2$). Vectorization was then conducted using ArcGIS tool.

Steps for aquaculture ponds detection using EDB method is shown in Fig. 4.5.

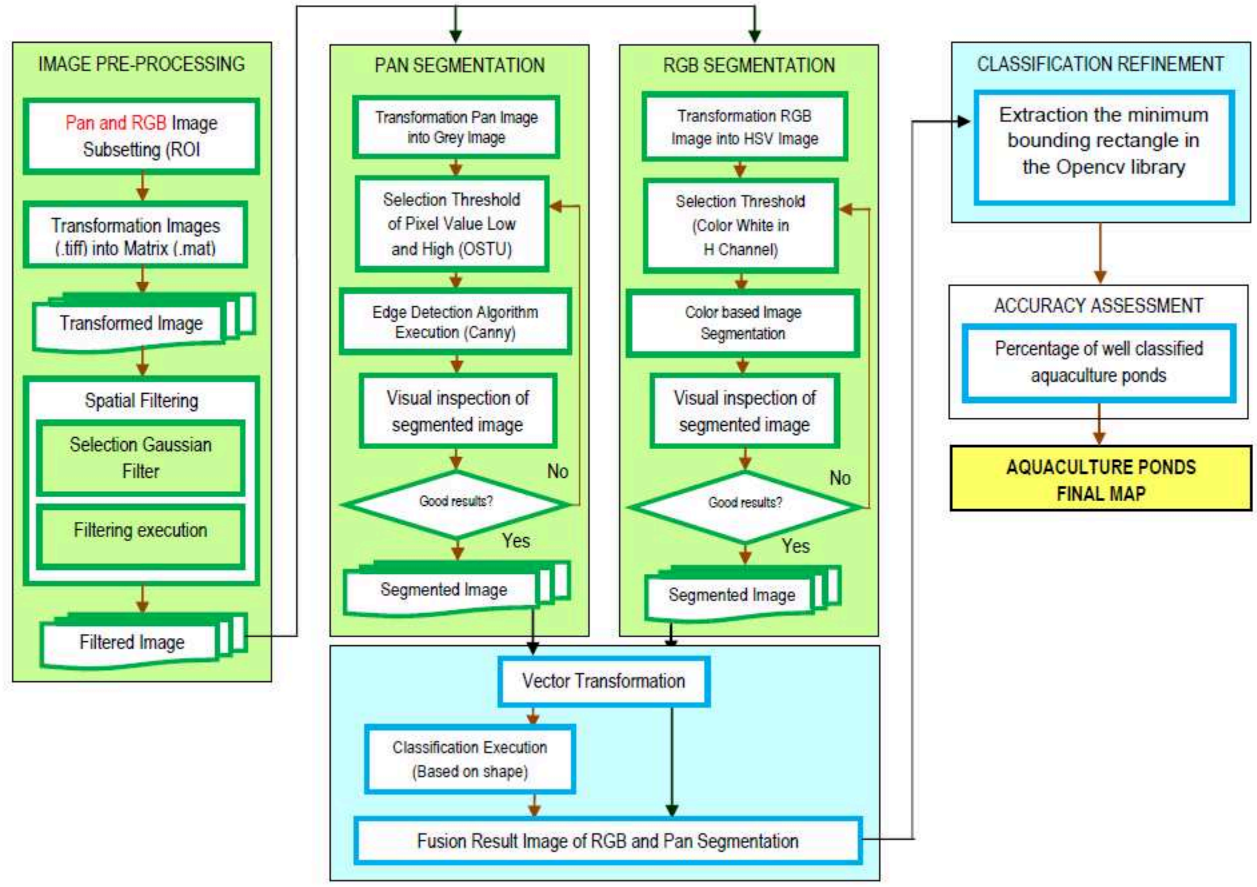


IMAGE PRE-PROCESSING
Pan and RGB Image Subsetting (ROI
Transformation Images (.tiff) into Matrix (.mat)
Transformed Image
Spatial Filtering
Selection Gaussian Filter
Filtering execution
Filtered Image
PAN SEGMENTATION
Transformation Pan Image into Grey Image
Selection Threshold of Pixel Value Low and High (OSTU)
Edge Detection Algorithm Execution (Canny)
Visual inspection of segmented image
Good results?
No
Yes
Segmented Image
RGB SEGMENTATION
Transformation RGB Image into HSV Image
Selection Threshold (Color White in H Channel)
Color based Image Segmentation
Visual inspection of segmented image
Good results?
No
Yes
Segmented Image
Vector Transformation
Classification Execution (Based on shape)
Fusion Result Image of RGB and Pan Segmentation
CLASSIFICATION REFINEMENT
Extraction the minimum bounding rectangle in the Opencv library
ACCURACY ASSESSMENT
Percentage of well classified aquaculture ponds
AQUACULTURE PONDS FINAL MAP

Figure 4.5 – Aquaculture ponds detection using EDB method

3.4 Results

To calculate the accuracy of the maps generated by different segmentation and classification methods, a reference map is firstly generated. It is composed of all pond boundaries which were delineated manually according to field surveys. Then, the accuracy was calculated by using a confusion matrix and kappa agreement coefficient and the proportion of correctly identified ponds (Foody, 2004).

Fig. 4.6 shows results from the three segmentation and classification methods used to generate an aquaculture map. As we can see, RGT method (upper right image) works well in homogeneous high contrast areas (Region g) by producing closed polygons. However, other areas such as dry ponds or low-level water ponds (region a region b and region f), as well as all textured areas such as mangrove in abandoned ponds (Region d) are under-segmented. Besides, RGT is not an automatic method and could not segment large images because it extracts a large number of ponds whose contours are connected (region c and region e) which requires a great amount of manual contour refinement.

Unsupervised classification implemented in IUC (bottom left image) provided a better overall mapping. However, as RGT method, IUC also under-segment highly heterogeneous areas (region a, region b and region d). Besides, pond connectivities are even more severe (as shown in region c and in region e). It needs many manual refinements which makes it impossible to provide a precise map.

As shown in Fig. 4.6 (bottom right image), EDB gave great results in all difficult situations. It permitted to detect and delineate ponds in heterogeneous and textured areas that were hardly extracted by RGT and IUC. Moreover, it also solved the connectivity problem. As we can see, EDB method provides the most precise maps.

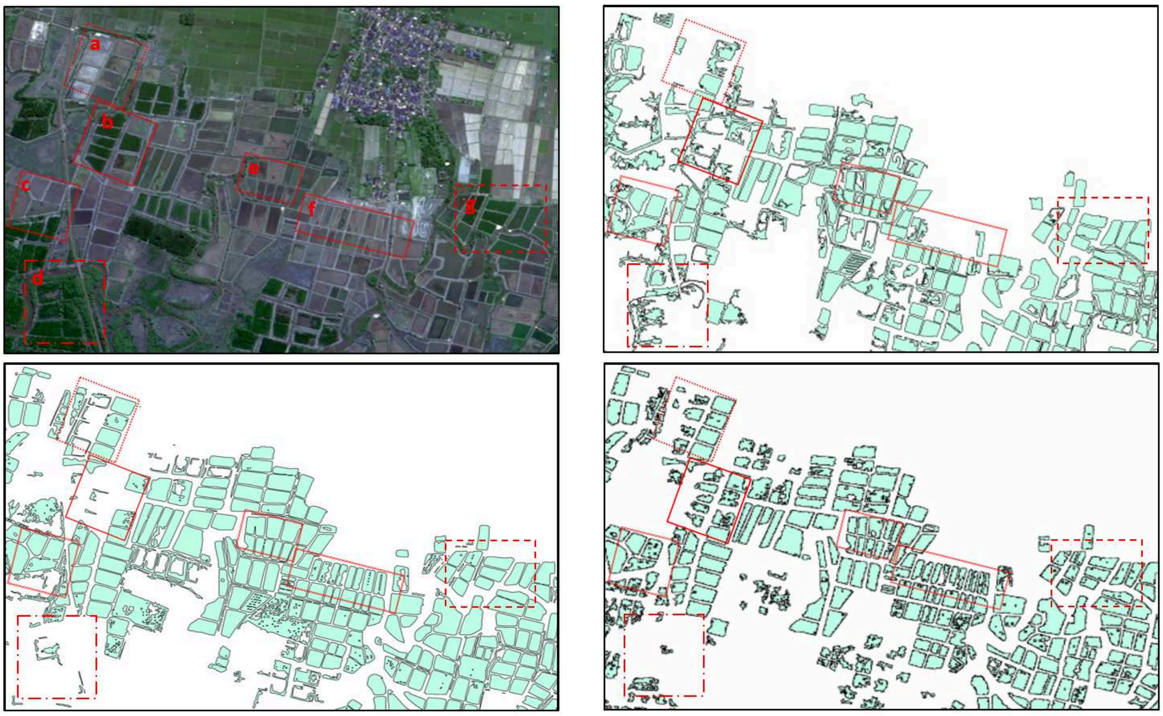


Figure 4.6 – Aquaculture map obtained using three classification methods. Upper left image: World View-2 image; Upper right image: RGT; Bottom left image: IUC; Bottom right image: EDB. Region a:dry active pond; Region b: abandoned pond with young vegetation; Region c, Region e and Region f: dry abandoned pond; Region d: watered active pond; Region g: abandoned pond with mature vegetation

Percentages of ponds identified by RGT, IUC and EDB, are 62% (after manual contour refinement), 81% (after contour manual refinement), and 96% (without manual contour refinement), respectively (Table 4.1). EDB accuracy is thus the highest while other methods could not provide a precise map even when followed by manual contour refinements. Accuracy assessments of these three methods are shown in Table 4.2. EDB overall accuracy is 84%, with a Kappa statistic of 0.68. As shown in this table, both of EDB accuracy and Kappa statistic are higher than RGT and IUC.

Table 4.1 – The proportion of identified and unidentified aquaculture ponds

Aquaculture ponds	**RGT**	**IUC**	**EDB**
Identified ponds (unit)	835	1091	1295
Unidentified ponds (unit)	517	261	57
Proportion of ponds identified (%)	62	81	96

Table 4.2 – Accuracy assessment for different methods

Classification	**Overall Accuracy (%)**	**Kappa coefficient**
RGT	65	0.3
IUC	77	0.54
EDB	84	0.68

Discussions on the results As shown in Fig. 4.6, RGT was hardly able to generate complete ponds in heterogeneous areas. It could not locate object boundaries and edges because neighboring pixels with same or similar values could not be clustered in the same region. When over-segmentation occurred, the actual edge pixels might be joined to other pixels in their neighborhood (Jain and Singh, 2011). The WV-2 image segmentation process, which took more than 3 hours, made the operation and data handling more complicated, while steps consisting of cadastre integration into map tasks were easily performed with an external GIS software.

IUC could create a smooth cadastre of aquaculture ponds. A fast processing time and integration with GIS spatial analysis added advantages to this method. That method depends on spectral signature and statistical information in images, and users do not have control over the clustering process. As an unsupervised classification method, IUC is not sensitive to variation covariations in object spectral signature, especially in the case of the low contrast images. That method could miss some ponds due to their connectivities which inevitably lead to a lower map accuracy (Fraisse *et al.*, 2001).

EDB approach achieves a high segmentation and classification accuracy. EDB overcomes the limitations related to sensitivity to noise and low image contrast by using a Gaussian filter and Canny operator. It did not need manual refinement of contours and it could improve segmentation accuracy in a significant manner.

The EDB approach possesses two key advantages over both IUC and RGT. Firstly, the Canny operator improved by the Ostu method permits to improve segmentation accuracy by effectively retaining details and slight borders of objects. Secondly, EDB approach considers not only spectral properties but also shape features that provide a dominant factor for classification.

4. Automatic identification of pond indicators

To study aquaculture pond evolution, we need to identify pond indicators. It is important to provide information to support decision makers with regard to abandoned pond rehabilitation, at pond and ecosystem scales. For this purpose, (Gusmawati *et al.*, 2017) identified four boolean criteria for aquaculture ponds by analysing satellite images: (1) presence of water, (2) presence of vegetation, (3) presence of aerator(s), (4) presence of wooden feeding bridge(s). Then they developed an activity indicator based on these four criteria to monitor change in shrimp farm activity. However, it requires to perform systematic ground truth surveys, which are tedious and very expensive. To overcome this issue, an automatic method is needed to identify pond indicators on a large scale. As detailed next, we developed such automatic methods.

To detect water in ponds, we used the normalized difference water index (NDWI) defined by (Gao, 1996). NDWI is defined as follows:

$$NDWI = \left(\frac{Green - NIR}{Green + NIR}\right)$$

where $Green$ is reflectance of the green wavelength band and NIR is the near-infrared wavelength band reflectance. This index is designed to (1) maximize water reflectance by using green wavelengths; (2) minimize the low reflectance of NIR by water features; and (3) take advantage of the high reflectance of NIR by vegetation and soil features. So water features having positive values are enhanced, while vegetation and soil usually have zero or negative values and therefore are suppressed.

Then, we used a threshold of 0.5 to identify water indicator (Zhai *et al.*, 2015). If the mean NDWI value of a pond is greater than that threshold, that pond contains water. Otherwise, there is no water in that pond. Fig. 4.7 shows ponds with water (in red) on one original satellite image. The indicator accuracy (the ratio of correctly identified ponds with water to total number of ponds with water) is 96%.

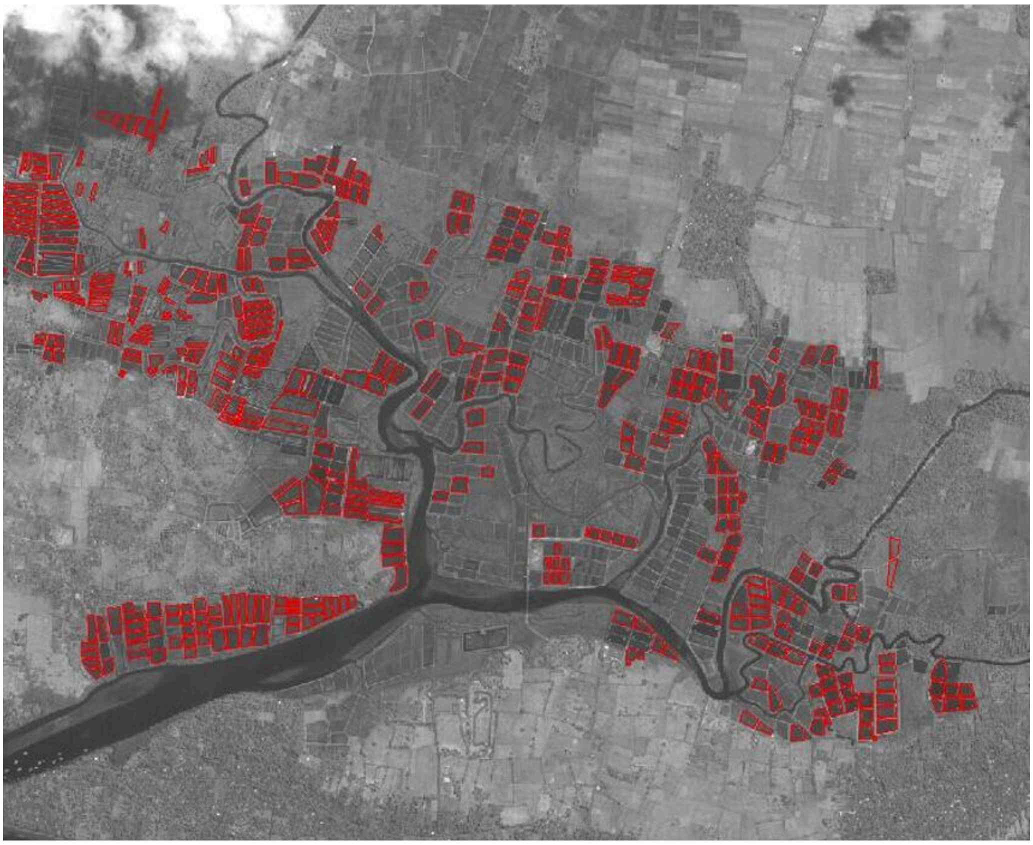


Figure 4.7 – Identification of ponds with Water

To detect vegetation in ponds, we used the Landscape Normalized Difference Vegetation Index (NDVI) which is firstly proposed by (Tucker, 1979). The NDVI was defined as follows:

$NDVI = \left(\frac{NIR-Red}{NIR+Red}\right)$

where NIR is the near-infrared wavelength band reflectance and Red is the red wavelength band reflectance. It has been widely used to detect vegetation (Nouri *et al.*, 2014; Nouri *et al.*, 2017; Alam *et al.*, 2018). The mean NDVI value of vegetation areas is between 0.3 and 0.8 (Nouri *et al.*, 2014).

Thus, we used a threshold between 0.3 and 0.8 to identify vegetation. If the mean NDVI value for a pond is greater than 0.3 and less than 0.8, there is some vegetation in the pond. Otherwise, there is no vegetation. Fig. 4.8 shows ponds with vegetation (in red) detected by these thresholds. Accuracy of this indicator (ratio of correctly identified ponds with vegetation to the total number of ponds with vegetation) is 94%.

To detect pond aerators, we firstly detect all the pond contours (Fig. 4.10) from original image (Fig. 4.9), Then an area threshold ranging from 20 to 50 pixels was applied to filter other small and big objects which are not aerators. Fig. 4.11 shows the aerators extracted from original image (Fig. 4.9). Accuracy (ratio of correctly identified ponds with aerators to total number of ponds with aerators)of this indicator is 96%.

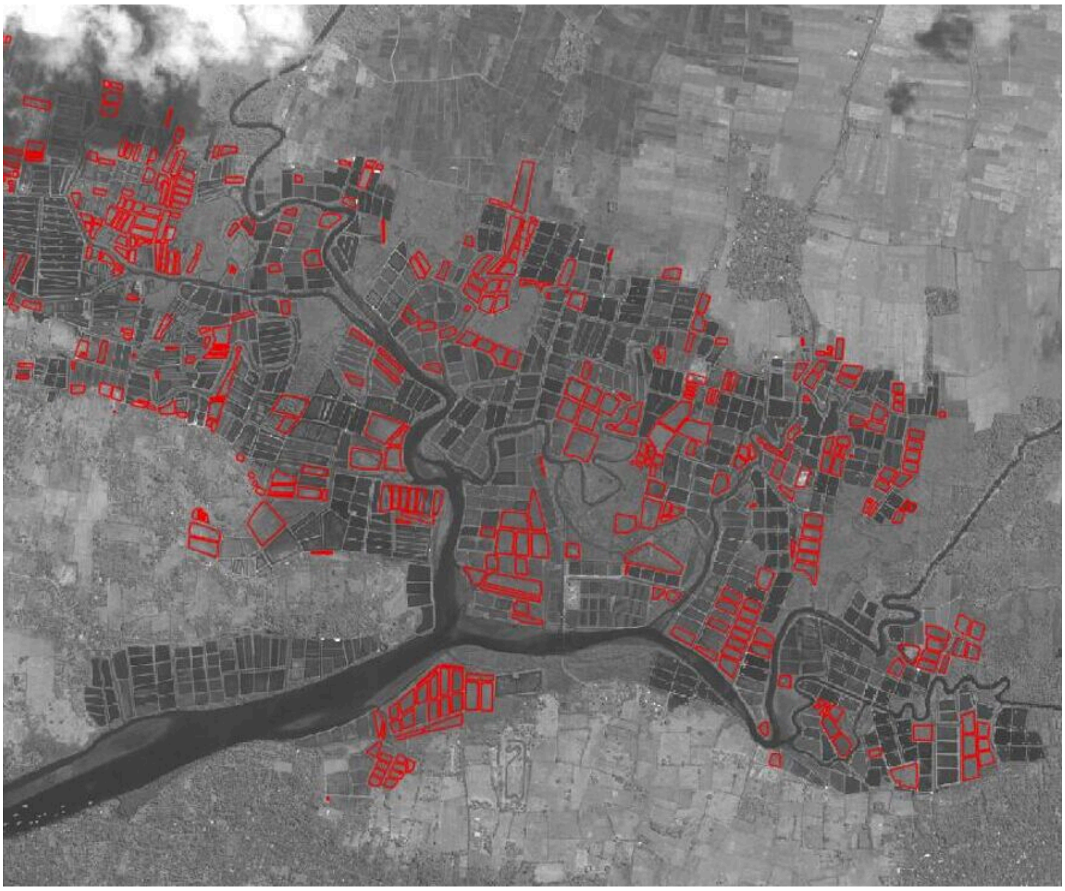


Figure 4.8 – Identification of ponds with vegetation

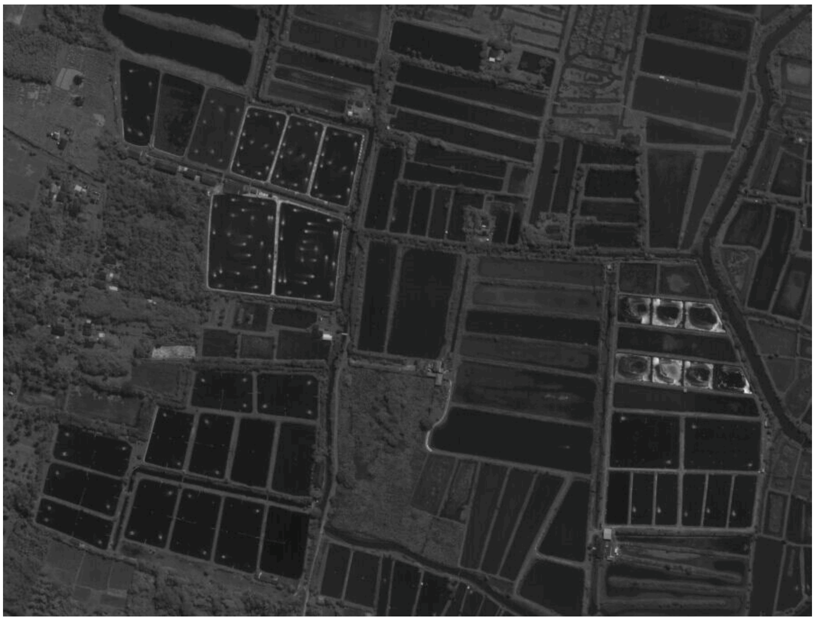


Figure 4.9 – Original satellite image

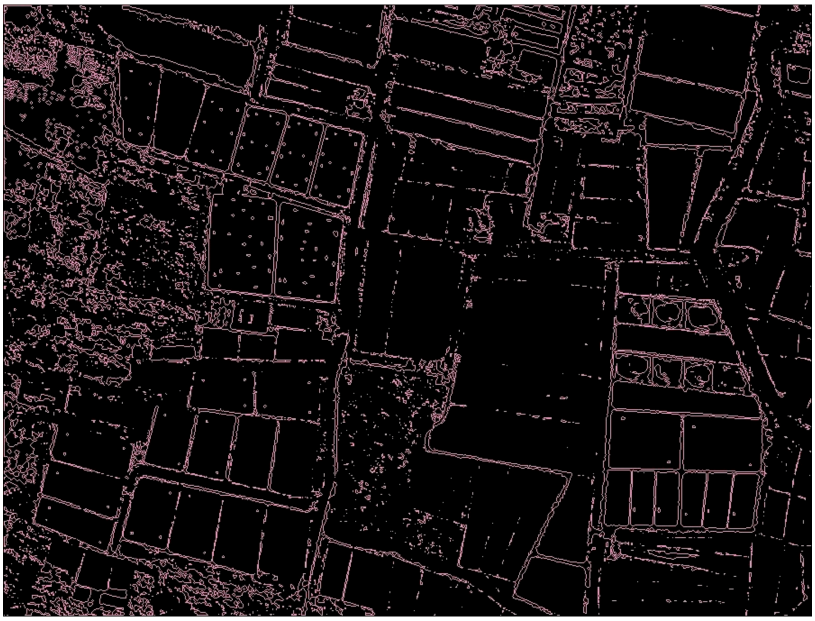


Figure 4.10 – Pond contour detection from Fig. 4.9

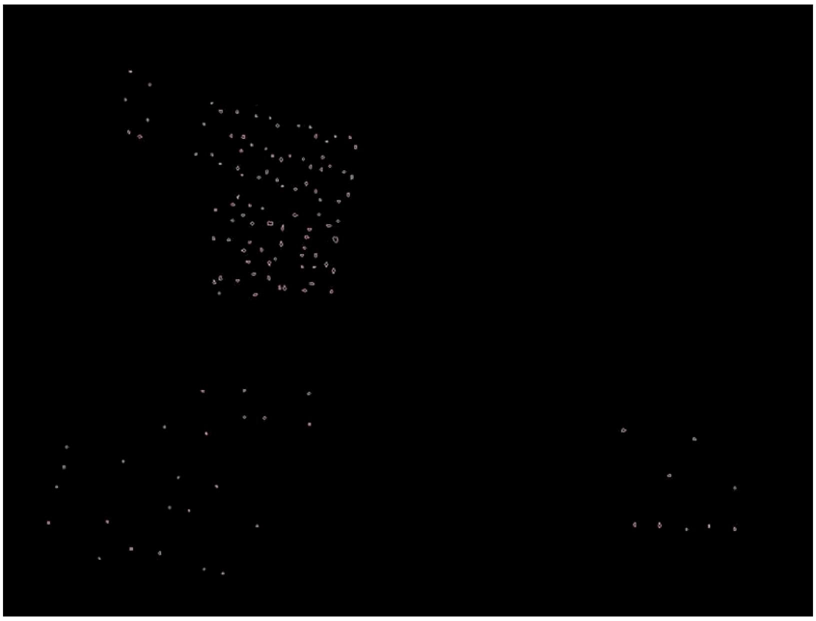


Figure 4.11 – Detected aerators in Fig. 4.9

Bridge indicator was identified by analysing satellite images visually. It is hard to detect pond bridge automatically, because as shown in Fig. 4.12, bridges are placed on dikes where contrast is very low. Besides, their sizes are extremely small (about $0.5 - 1$ m $\times$ $3 - 5m$) which makes it very difficult to identify bridges in ponds.

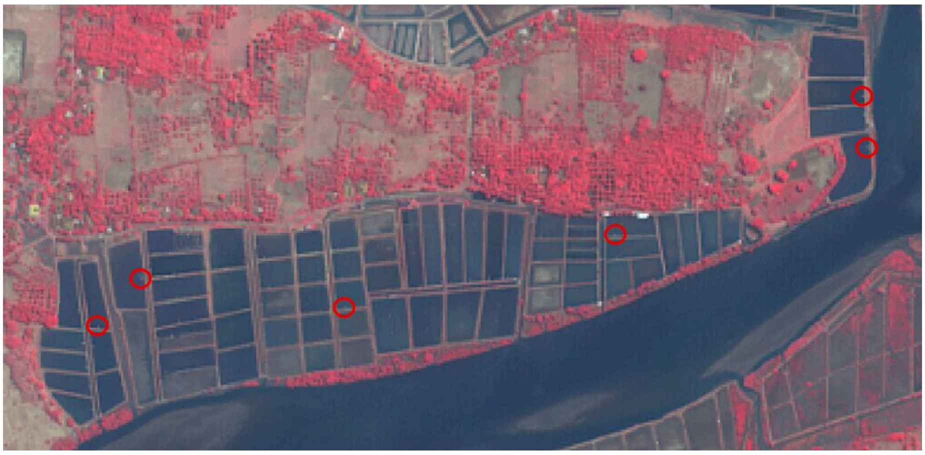


Figure 4.12 – Bridges of ponds

The activity indicator has been identified by conducting field surveys and analyzing satellite images visually (Gusmawati *et al.*, 2017). In her thesis, she verified that activity indicator is closely related to other four criteria mentioned above.

In conclusion, we have identified interesting objects which are aquaculture ponds spatially located in satellite image. Each pond is depicted by four characteristics (indicators): presence of water, vegetation, aerator(s) and wooden feeding bridge(s). Those characteristics discriminate ponds with activity and abandoned ponds.

5. Image dataset transformation

We transform images information into two data representations to analyze pond evolutions: a dynamic attributed graph and sequential data. It allows to extract recurrent patterns and frequent sequential patterns depicting pond evolution.

5.1 From cartographies to dynamic attributed graphs

Generation of vertices To generate vertices, we extract all the ponds in each satellite image and save their id for each image. For example, as there are 1513 ponds in satellite image for the first date, so the pond id in first image goes from 1 to 1513. However, as ponds evolve over time, a pond could have different id in two consecutive images. As a consequence, we need to find the temporal relationships of ponds in two consecutive images. Firstly, we browse ponds in the first image. For every pond in this first image, we identify all ponds in the second image that are intersected with it (having a common region). This step permits to detect all five possible pond evolutions: (1) fusion (several ponds correspond to one pond), (2) division (one pond corresponds to several ponds), (3) appearance (zero to one: zero pond corresponds to one pond), (4) disappearance (one to zero: one pond corresponds to zero pond) and (5) one-to-one (one pond corresponds to one pond). Then we repeat this step for all the following 12 dated images. Table 4.3 shows evolutions of the first 20 ponds. As we can see, this table is composed of 15 columns where the first column displays final ids of ponds and the remaining 14 columns contain original pond ids in the 14 satellite images. Each row represents the evolution of one pond over 14 times. If a pond is absent in an image, its id is set to "-1". Let us consider, for example, the pond in the first row. We can notice that this pond is absent in the first image. Then it appears in the second image with id 453. Next, it becomes pond 460 in the third image and pond 463 in the forth image etc. Finally, we standardise the first pond id by assigning it the id "1" (in the first column named "Final id"). To generate vertex attributes, we replace pond original ids by their corresponding attributes. Table 4.4 shows attributes of the first 20 vertices, where the first column presents the final id of ponds and the other columns represent the attributes of ponds for each date (image). As we can see, each table cell is composed of five numbers representing the five attributes of ponds: "1" for "No Aerator", "2" for "With Aerator", "3" for "No Bridge", "4" for "With Bridge", "5" for "No Vegetation", "6" for "With Vegetation", "7" for "No Water", "8" for "With Water", "9" for "No Activity", "10" for "With Activity". We note that several table cells are composed of "-1". It represents that there are no attributes for this pond, as it is absent in this image.

Generation of edges We create edges by considering spatial relationships of ponds (vertices) in each image. For every pond in a given image, we define a ROI (region of interest) to calculate all its neighbors in this region. This ROI is a circle region whose center is the pond center and its radius is equal to the *MajorAxisLength*, i.e., the major axis length (in pixels) of the ellipse that has the same normalized second central moments as the region. Then, we calculate the distance between this pond and every other aquaculture pond in this ROI (region of interest). If the distance is less than a user defined threshold (50 pixels), we create an edge between these two vertices (adjacent ponds). For example, Fig. 4.13 (right) shows the circular ROI (red) for the pond with red contour in Fig. 4.13 (left). With this ROI, we only need to calculate its possible neighbors in this region instead of considering all the other ponds of the map.

We illustrate this transformation with an example. Fig. 4.14 shows 3 consecutive satellite images (from 2011 to 2013). As we can see, from 2011 to 2012, pond 1 (in 2011) was divided into pond 1 and pond 2 (in 2012), pond 2 (in 2011) was divided into pond 3 and pond 4 (in 2012). Then, from 2012 to 2013, pond 7 (in 2012) was divided into pond 7 and pond 8 (in 2013). Based on these temporal relationships between vertices (ponds), we can firstly construct all vertices. The first column of Table 4.5 shows the final ids of vertices and the other columns represent their original id in each image. To generate the vertex attributes, we only need to replace the original pond id in Table 4.5 by their corresponding attributes. Table 4.6 shows all attributes of this graph. Then, we generate all edges for each image (Table 4.7). Based on correspondences between the final id and the original id of ponds (as shown in Table 4.5), we transform Table 4.7 by using the final pond ids to obtain the final ids of edges (as shown in Table 4.8).

By using this method, we transform this 14 images time series into a dynamic attributed

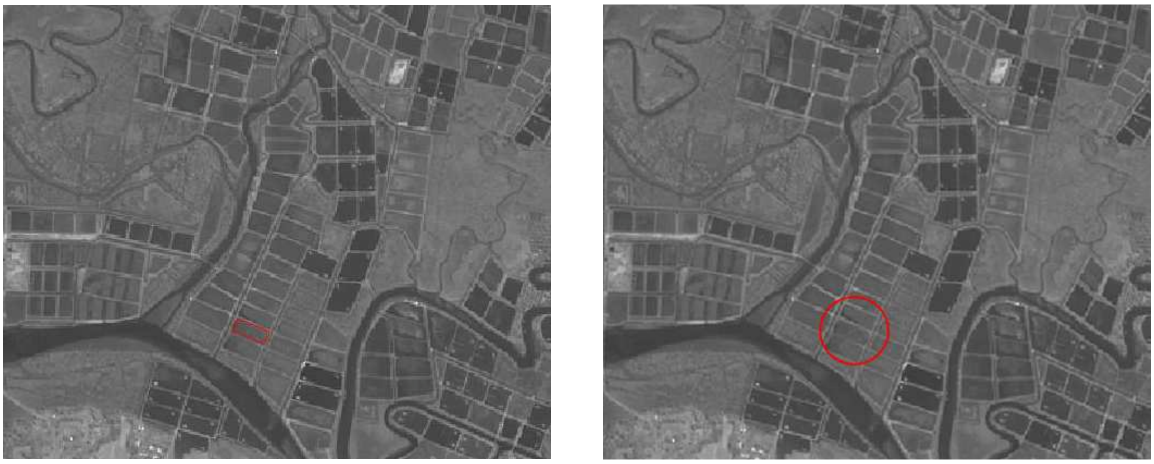
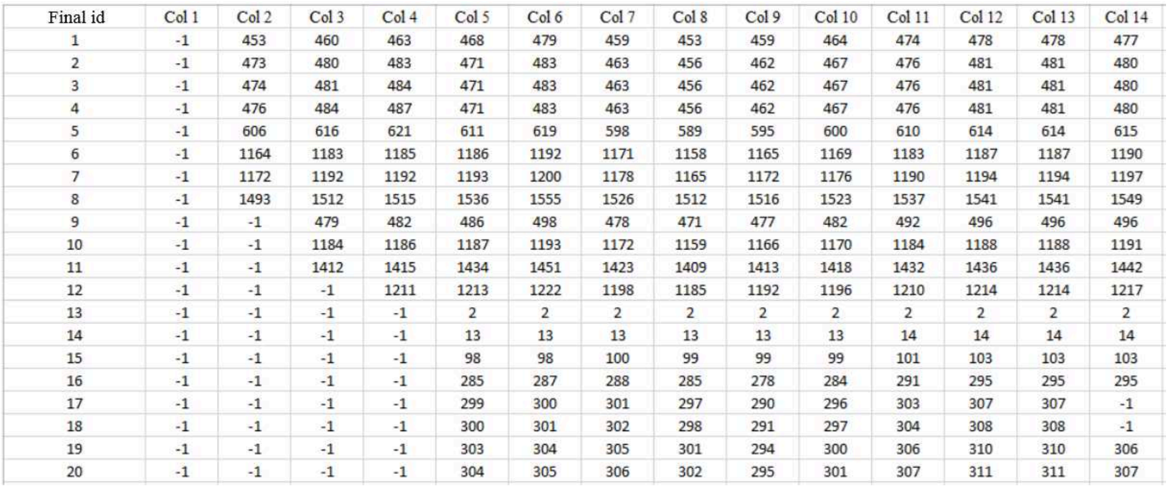


Figure 4.13 – Example of ROI



Final id	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10	Col 11	Col 12	Col 13	Col 14
1	-1	453	460	463	468	479	459	453	459	464	474	478	478	477
2	-1	473	480	483	471	483	463	456	462	467	476	481	481	480
3	-1	474	481	484	471	483	463	456	462	467	476	481	481	480
4	-1	476	484	487	471	483	463	456	462	467	476	481	481	480
5	-1	606	616	621	611	619	598	589	595	600	610	614	614	615
6	-1	1164	1183	1185	1186	1192	1171	1158	1165	1169	1183	1187	1187	1190
7	-1	1172	1192	1192	1193	1200	1178	1165	1172	1176	1190	1194	1194	1197
8	-1	1493	1512	1515	1536	1555	1526	1512	1516	1523	1537	1541	1541	1549
9	-1	-1	479	482	486	498	478	471	477	482	492	496	496	496
10	-1	-1	1184	1186	1187	1193	1172	1159	1166	1170	1184	1188	1188	1191
11	-1	-1	1412	1415	1434	1451	1423	1409	1413	1418	1432	1436	1436	1442
12	-1	-1	-1	1211	1213	1222	1198	1185	1192	1196	1210	1214	1214	1217
13	-1	-1	-1	-1	2	2	2	2	2	2	2	2	2	2
14	-1	-1	-1	-1	13	13	13	13	13	13	14	14	14	14
15	-1	-1	-1	-1	98	98	100	99	99	99	101	103	103	103
16	-1	-1	-1	-1	285	287	288	285	278	284	291	295	295	295
17	-1	-1	-1	-1	299	300	301	297	290	296	303	307	307	-1
18	-1	-1	-1	-1	300	301	302	298	291	297	304	308	308	-1
19	-1	-1	-1	-1	303	304	305	301	294	300	306	310	310	306
20	-1	-1	-1	-1	304	305	306	302	295	301	307	311	311	307

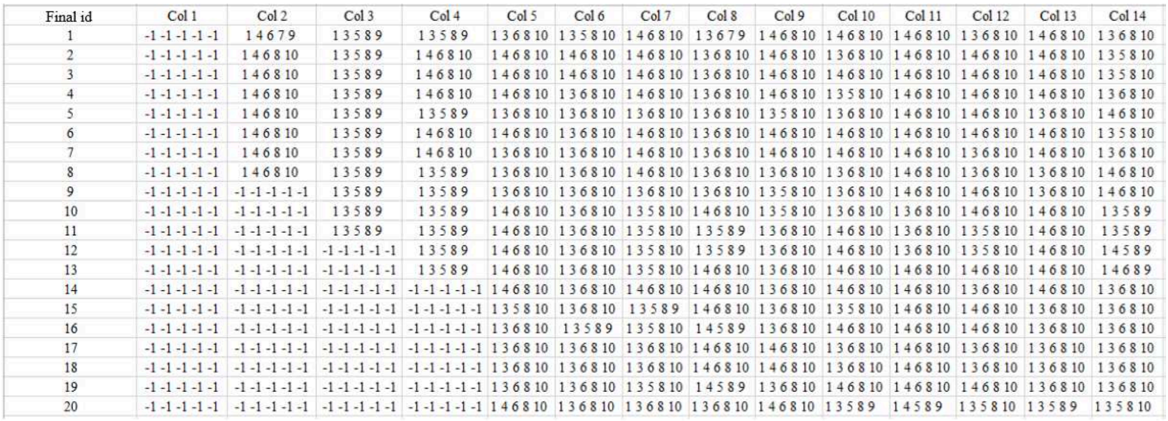
Table 4.3 – Rearrangement of pond id

graph. It is composed of 1915 vertices where vertices represent ponds, each vertex is associated to 10 attributes (*WithActivity, WithoutActivity, WithBridge, WithoutBride, WithVegetation, WithoutVegetation, WithWater, WithoutWater, WithAerator, WithoutAerator*). Each edge links two ponds which are spatially adjacent. To summarize, this graph consists of 1915 vertices (ponds), 4350 edges in average, 10 attributes and 14 timestamps. This representation enables to study the spatio-temporal evolutions of ponds.

5.2 From cartographies to sequential data

To generate sequential data, we only have to use the Table 4.4 constructed in previous step. Table. 4.9 shows the first 20 sequences generated from this dataset. As we can see, this sequential data consists of 1915 sequences (ponds). Each sequence is composed of 14 itemsets in which each item describes one pond characteristic.

This data representation considers all kinds of temporal evolutions such as pond appearance,



Final id	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10	Col 11	Col 12	Col 13	Col 14
1	-1 -1 -1 -1 -1	1 4 6 7 9	1 3 5 8 9	1 3 5 8 9	1 3 6 8 10	1 3 5 8 10	1 4 6 8 10	1 3 6 7 9	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10
2	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 5 8 10
3	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 5 8 10
4	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 5 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10
5	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 3 5 8 9	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 5 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10
6	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 5 8 10
7	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10
8	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 5 8 9	1 3 5 8 9	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10
9	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 5 8 9	1 3 5 8 9	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 5 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10
10	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 5 8 9	1 3 5 8 9	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 4 6 8 10	1 3 5 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 5 8 9
11	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 5 8 9	1 3 5 8 9	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 3 5 8 9	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 4 6 8 10	1 3 5 8 9
12	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 5 8 9	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 3 5 8 9	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 4 6 8 10	1 4 5 8 9
13	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 5 8 9	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 9
14	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10
15	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 5 8 10	1 3 6 8 10	1 3 5 8 9	1 4 6 8 10	1 3 6 8 10	1 3 5 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10
16	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 6 8 10	1 3 5 8 9	1 3 5 8 10	1 4 5 8 9	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10
17	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10
18	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10
19	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 3 6 8 10	1 3 6 8 10	1 3 5 8 10	1 4 5 8 9	1 3 6 8 10	1 4 6 8 10	1 4 6 8 10	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10
20	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	-1 -1 -1 -1 -1	1 4 6 8 10	1 3 6 8 10	1 3 6 8 10	1 3 6 8 10	1 4 6 8 10	1 3 5 8 9	1 4 5 8 9	1 3 5 8 10	1 3 5 8 9	1 3 5 8 10

Table 4.4 – Attributes and vertices of dynamic attributed graph. -1: disappeared pond, 1: No Aerator, 2: With Aerator, 3: No Bridge, 4: With Bridge, 5: No Vegetation, 6: With Vegetation, 7: No Water, 8: With Water, 9: No Activity, 10: With Activity

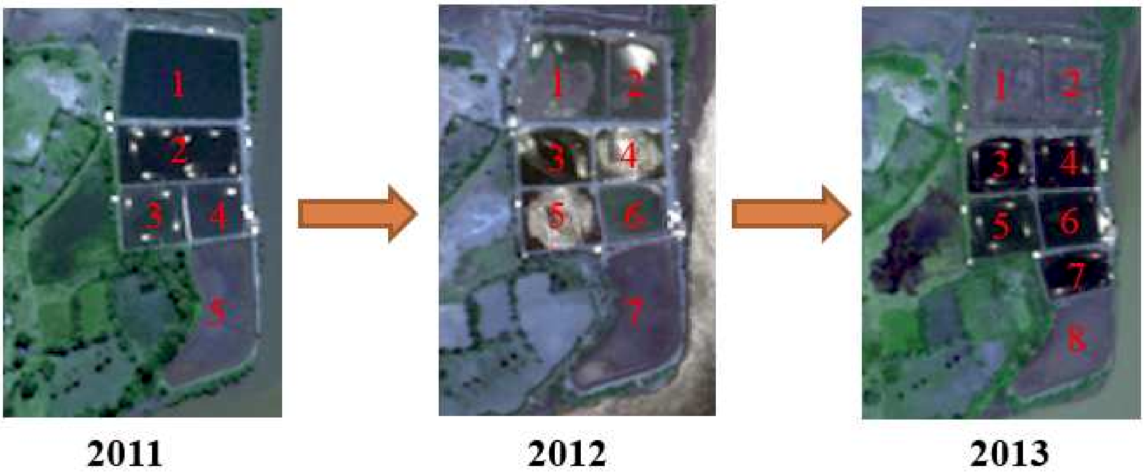


Figure 4.14 – An example of ROI over 3 consecutive time (from to 2011 to 2013)

disappearance, fusion and division. For example, let consider the evolution of pond 5 in 2011 (Fig. 4.14). This pond (in 2011) was evolved to pond 7 (in 2012). Then, pond 7 (in 2012) was divided into pond 7 (in 2013) and pond 8 (in 2013). Consequently, we generate two sequences:
$< \{WAc, NB, NV, NW, NAe\}, \{WAc, NB, NV, NW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$ $(pond5 \rightarrow pond7 \rightarrow pond7)$ and
$< \{WAc, NB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, NB, NV, WW, NAe\} >$ $(pond5 \rightarrow pond7 \rightarrow pond8)$.
Table 4.10 shows all the constructed sequences for this example (Fig. 4.14). We have to notice that sequence mining algorithms could introduce biases to study such data (support of extracted patterns could be higher than its real support). For example, let us consider the two sequences generated above. Given a support threshold=2, one of the frequent sequences of length 2 is $< \{WAc, NB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$. However, its real

Table 4.5 – Vertices and attributes of dynamic attributed graph. WAc:WithActivity, NAc:WithoutActivity, WB:WithBridge, NB:WithoutBridge, WV:WithVegetation, NV:WithoutVegetation, WW:WithWator, NAc:WithoutWator, WAe:WithAerator, NAe:WithoutAerator

Final id	id in 2011	id in 2012	id in 2013
Pond1	Pond1	Pond1	Pond1
Pond2	Pond1	Pond2	Pond2
Pond3	Pond2	Pond3	Pond3
Pond4	Pond2	Pond4	Pond4
Pond5	Pond3	Pond5	Pond5
Pond6	Pond4	Pond6	Pond6
Pond7	Pond5	Pond7	Pond7
Pond8	Pond5	Pond7	Pond8

Table 4.6 – Vertices and attributes of dynamic attributed graph. WAc:WithActivity, NAc:WithoutActivity, WB:WithBridge, NB:WithoutBridge, WV:WithVegetation, NV:WithoutVegetation, WW:WithWator, NAc:WithoutWator, WAe:WithAerator, NAe:WithoutAerator

Vertices	Attributes (2011)	Attributes (2012)	Attributes (2013)
Pond1	WAc, NB, NV, WW, NAe	WAc, WB, NV, WW, NAe	WAc, NB, NV, NW, NAe
Pond2	WAc, NB, NV, WW, NAe	WAc, WB, NV, WW, NAe	WAc, WB, NV, NW, NAe
Pond3	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe
Pond4	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe
Pond5	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe
Pond6	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe	WAc, WB, NV, WW, NAe
Pond7	WAc, NB, NV, NW, NAe	WAc, NB, NV, NW, NAe	WAc, WB, NV, WW, NAe
Pond8	WAc, NB, NV, NW, NAe	WAc, NB, NV, NW, NAe	WAc, NB, NV, WW, NAe

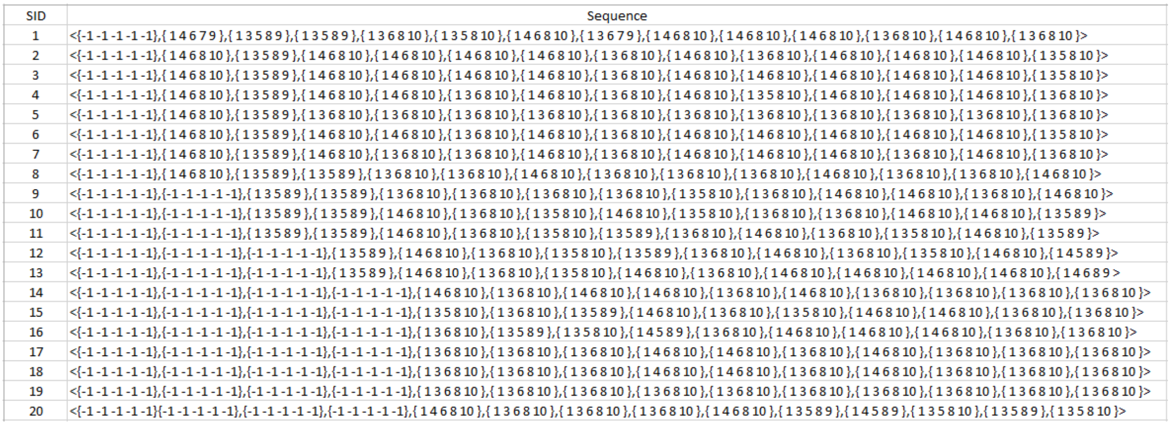
Table 4.7 – Set of edges of each image

	Edges
2011	(1,2), (2,3), (2,4), (3,4), (4,5)
2012	(1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (3,5), (3,6), (5,6), (5,7), (6,7)
2013	(1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (3,5), (3,6), (5,6), (5,7), (6,7), (7,8)

support is just 1, because these two occurrences represent the same evolution $pond5 \rightarrow pond7$. To deal with this problem, we have to add a post-processing method to calculate the real support of extracted patterns.

Table 4.8 – Final set of Edges of dynamic attributed graph

	Edges
2011	(1,3), (3,5), (3,6), (5,6), (6,7)
2012	(1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (3,5), (3,6), (5,6), (5,7), (6,7)
2013	(1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (3,5), (3,6), (5,6), (5,7), (6,7), (7,8)



SID	Sequence
1	<{-1 -1 -1 -1 -1},{1 4 6 7 9},{1 3 5 8 9},{1 3 5 8 9},{1 3 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 3 6 7 9},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10}>
2	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 5 8 10}>
3	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 5 8 10}>
4	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10}>
5	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
6	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 5 8 10}>
7	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10}>
8	<{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 5 8 9},{1 3 5 8 9},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10}>
9	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 5 8 9},{1 3 5 8 9},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10}>
10	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 5 8 9},{1 3 5 8 9},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 3 5 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 5 8 9}>
11	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 5 8 9},{1 3 5 8 9},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 3 5 8 9},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 3 5 8 9}>
12	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 5 8 9},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 3 5 8 9},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 4 5 8 9}>
13	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 5 8 9},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 9 >
14	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
15	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 5 8 10},{1 3 6 8 10},{1 3 5 8 9},{1 4 6 8 10},{1 3 6 8 10},{1 3 5 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
16	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 6 8 10},{1 3 5 8 9},{1 3 5 8 10},{1 4 5 8 9},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
17	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
18	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
19	<{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10}>
20	<{-1 -1 -1 -1 -1}{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{-1 -1 -1 -1 -1},{1 4 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 3 6 8 10},{1 4 6 8 10},{1 3 5 8 9},{1 4 5 8 9},{1 3 5 8 10},{1 3 5 8 9},{1 3 5 8 10}>

Table 4.9 – Sequence data

Table 4.10 – Transformed sequential dataset

SID	**Sequence**
1	$< \{WAc, NB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, BB, NV, NW, NAe\} >$
2	$< \{WAc, NB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, NW, NAe\} >$
3	$< \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$
4	$< \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$
5	$< \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$
6	$< \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$
7	$< \{WAc, WB, NV, WW, NAe\}, \{WAc, NB, NV, NW, NAe\}, \{WAc, WB, NV, WW, NAe\} >$
8	$< \{WAc, NB, NV, NW, NAe\}, \{WAc, NB, NV, NW, NAe\}, \{WAc, NB, NV, WW, NAe\} >$

6. Pond evolution by sequential pattern mining

Firstly, we use sequence mining algorithms (Srikant and Agrawal, 1996; Pei *et al.*, 2004; Fournier-Viger *et al.*, 2008; Fournier-Viger *et al.*, 2014) to study temporal evolutions of aquaculture ponds. In this work, we choose the algorithm developed by (Fournier-Viger *et al.*, 2008) to study aquaculture data because it extracts closed subsequences and it integrates five useful constraints to filter uninteresting patterns: (1) minimum support (minsup), (2) minimum time interval allowed between two successive itemsets, (3) maximum time interval allowed between two successive itemsets, (4) minimum time interval allowed between the first and the last itemset of a sequential pattern (min_whole_interval) and (5) maximum time interval allowed between the first and the last itemset of a sequential pattern (max_whole_interval). The second and third constraints are especially important and interesting to study aquaculture pond evolution. Indeed, they permit to extract consecutive evolutions of ponds. In addition, patterns extracted without those constraints could lead to ambiguity and be difficult to interpret.

We conducted a large number of experiments by varying different parameters. We present the most interesting patterns obtained with parameters $minsup = 0.2$ (20%), minimum time interval=1, maximum time interval=1, min_whole_interval=2 and max_whole_interval=6. The minimum and the maximum time interval are both set to 1 to extract consecutive evolutions of ponds.

To better analyze and interpret sequential patterns, we developed a visualisation tool using Matlab. For each sequential pattern, we visualize all its instances in the 14 consecutive images. For example, an extracted pattern $\langle \{WithActivity\},\{WithoutActivity\},\{WithoutActivity\},\{WithoutActivity\},\{WithoutActivity\} \rangle$ of length 5 is shown in Fig. 4.15 (from 2001 to 2008) and Fig. 4.16 (from 2009 to 2012). That pattern is displayed in two figures because the 10 corresponding images could not be displayed on one page. As shown in those figures, each red pond represents an instance of this pattern over 5 consecutive times. The time where each red contour appears represents the beginning time of this instance. This pattern depicts the evolution that an active pond became inactive and remained inactive in four consecutive timestamps. It is verified by 668 ponds (frequency=668). As we can see from figure C and figure D in Fig. 4.15, most of ponds in the center became inactive from 2003 due to abandonment of those ponds by farmers after disease outbreaks. Then, we can notice that between 2010 and 2011, numerous ponds in the periphery became inactive (figure H and figure I in Fig. 4.16). This can be explained by strong temperature anomalies (-3°C) during this period (Gusmawati *et al.*, 2017). It reveals a gradient of abandonment from the center to the periphery of this agrosystem. (Gusmawati *et al.*, 2017) obtained the same result by using a manual approach. To analyze such temporal evolutions of ponds, they display the dates of last activity of each pond in one map (Fig. 4.17). This map gives us a global view of ponds evolutions. However, we could not know when and where an evolution locally occurs.

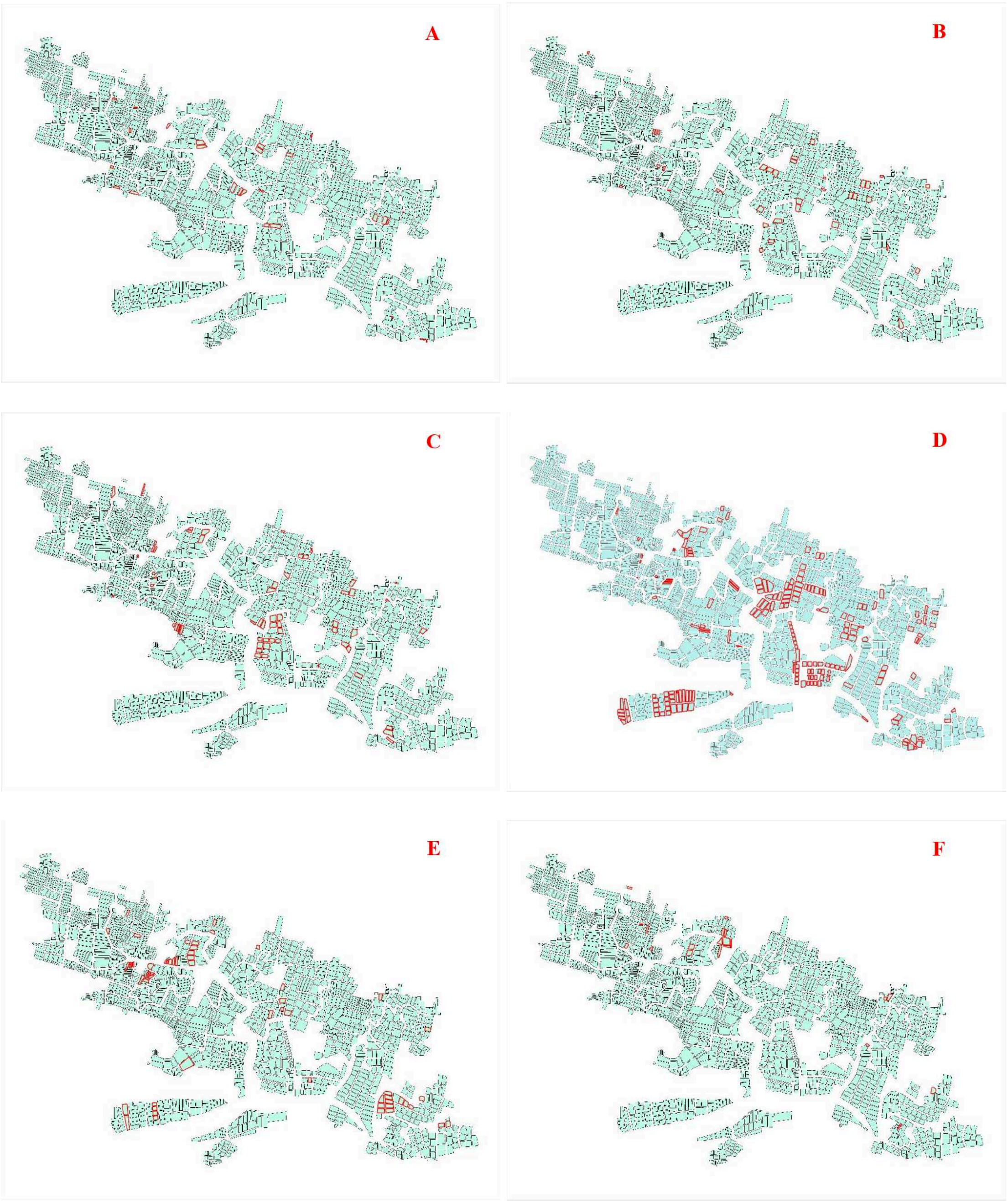


Figure 4.15 – First sequential pattern ⟨ {WithActivity}, {WithoutActivity}, {WithoutActivity}, {WithoutActivity}, {WithoutActivity} ⟩ (from 2001 to 2008). Red contours represent active ponds which became inactive in the 4 consecutive years. A: 12/10/2001, B: 09/03/2002, C: 21/02/2003, D: 27/06/2003, E: 22/09/2007, F: 19/07/2008

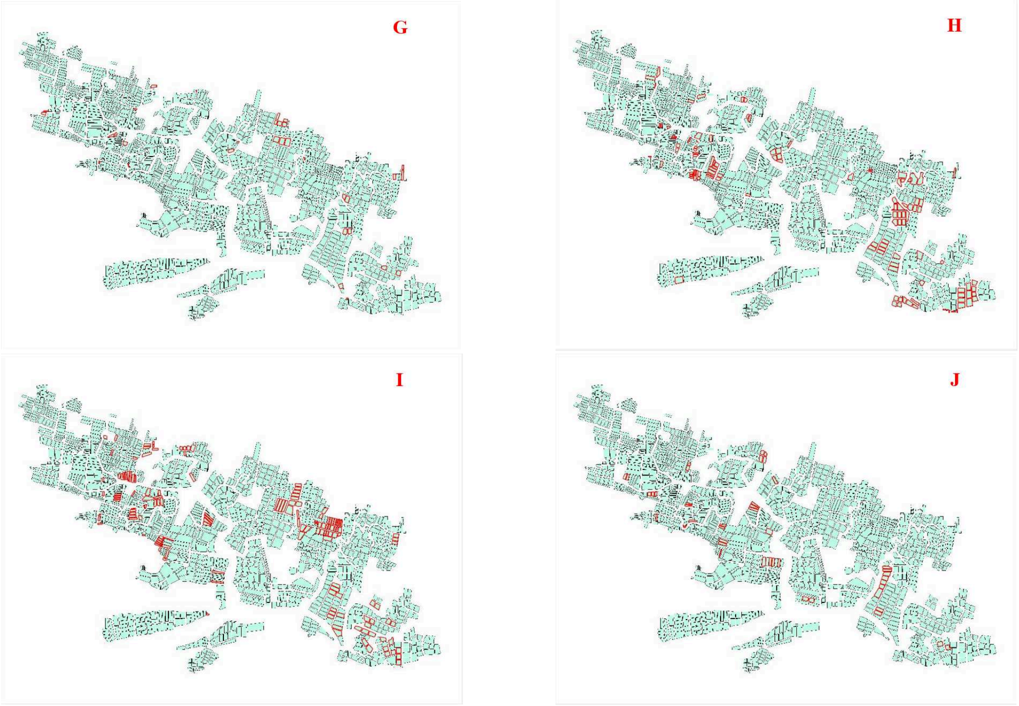


Figure 4.16 – First sequential pattern ⟨ {WithActivity}, {WithoutActivity}, {WithoutActivity}, {WithoutActivity}, {WithoutActivity} ⟩ (from 2009 to 2012). Red contours represent active ponds which became inactive in the 4 consecutive years. G: 09/07/2009, H:16/08/2010, I:15/04/2011, J:23/10/2012

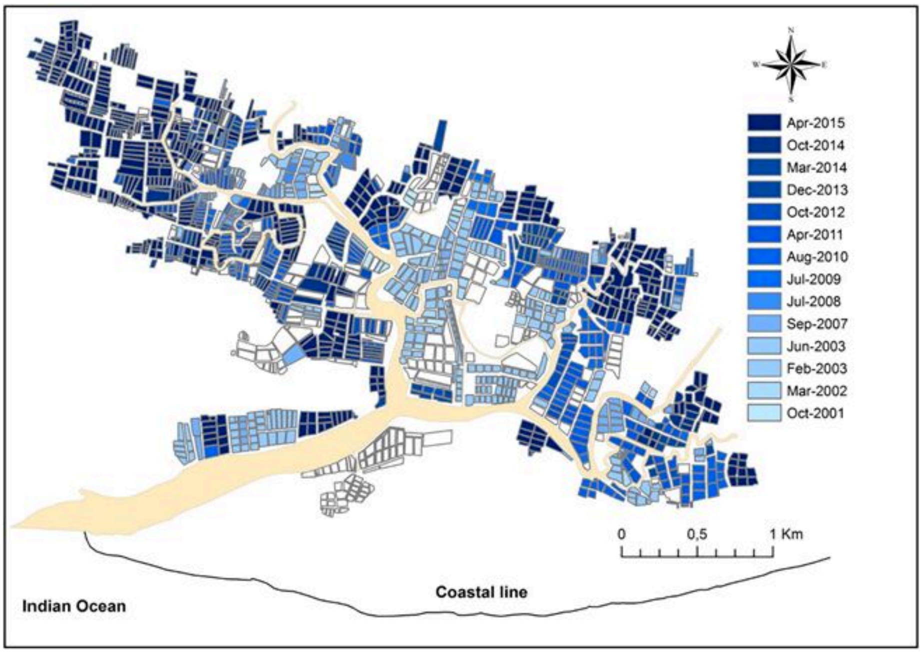


Figure 4.17 – Cadastre of the last activity detected in ponds between 2001 and 2015 in Perancak estuary, based on Integrated Pond Activity Indicator (Gusmawati *et al.*, 2017)

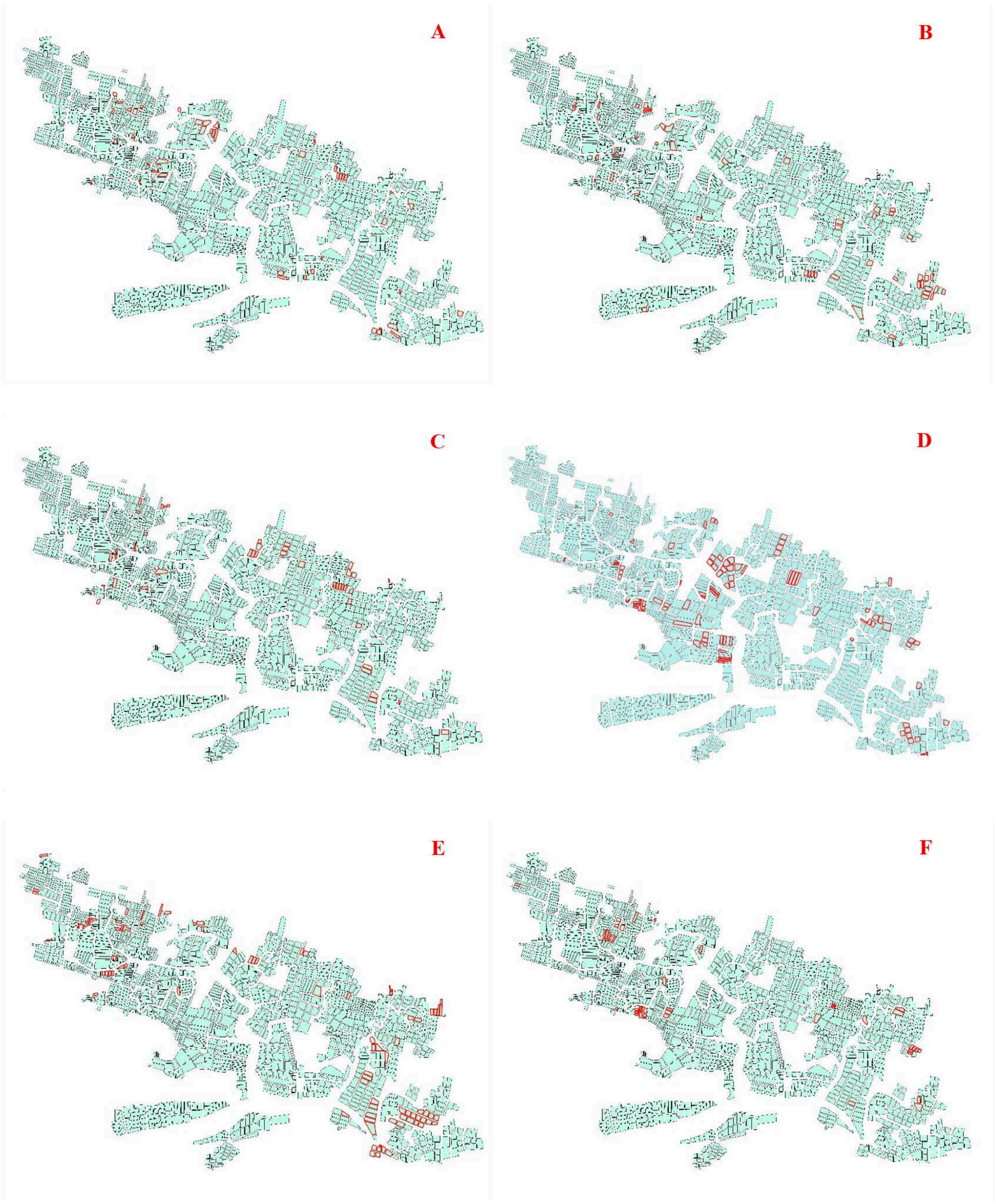


Figure 4.18 – Second sequential pattern ⟨ {WithActivity}, {WithoutActivity}, {WithActivity} ⟩ (from 2001 to 2008). Red contours represent active ponds became inactive in the second years and then became active again in the third year. A: 12/10/2001, B: 09/03/2002, C: 21/02/2003, D: 27/06/2003, E: 22/09/2007, F: 19/07/2008

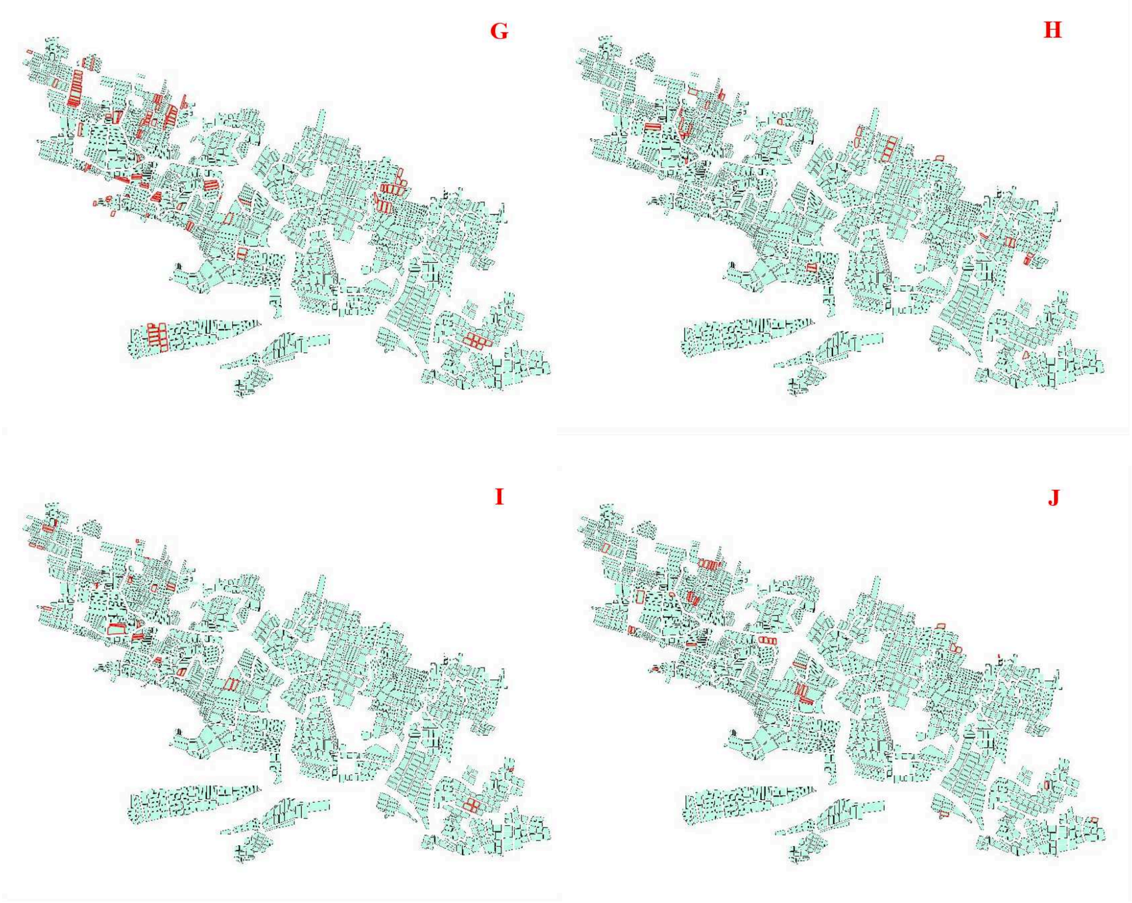


Figure 4.19 – Second sequential pattern ⟨ {WithActivity}, {WithoutActivity}, {WithActivity} ⟩ (from 2009 to 2012). Red contours represent active ponds became inactive in the second years and then became active again in the third year. G: 09/07/2009, H:16/08/2010, I:15/04/2011, J:23/10/2012

Fig. 4.18 (from 2001 to 2008) and Fig. 4.19 (from 2009 to 2012) show another example of sequential pattern ⟨ $\{WithActivity\}$,$\{WithoutActivity\}$,$\{WithActivity\}$,⟩ of length 3. It is verified by 478 ponds. That pattern represents an evolution over 3 consecutive timestamps. It describes an active pond became inactive in the second year and became active again in the third year. We can observe that this pattern is only present in periphery zones. It is due to the fact that farmers dry their ponds regularly to improve sediments.

Fig. 4.20 (from 2001 to 2008) and Fig. 4.21 (from 2009 to 2011) show another example of sequential pattern ⟨ $\{WithoutActivity\}$,$\{WithoutActivity\}$,$\{WithoutActivity\}$, $\{WithActivity\}$ ⟩ which is verified by 258 ponds. It depicts an evolution over 4 consecutive timestamps. It represents an inactive pond remained inactive in the three consecutive years and then became active in the forth year. We observe that in periphery zones, a large amount of inactive ponds became active in the second year. This is due to two kinds of human activities. Firstly, farmers felled trees to activate ponds. Secondly, some ponds were rehabilitated to produce fish or shrimp after a period of abandonment (Gusmawati *et al.*, 2017).

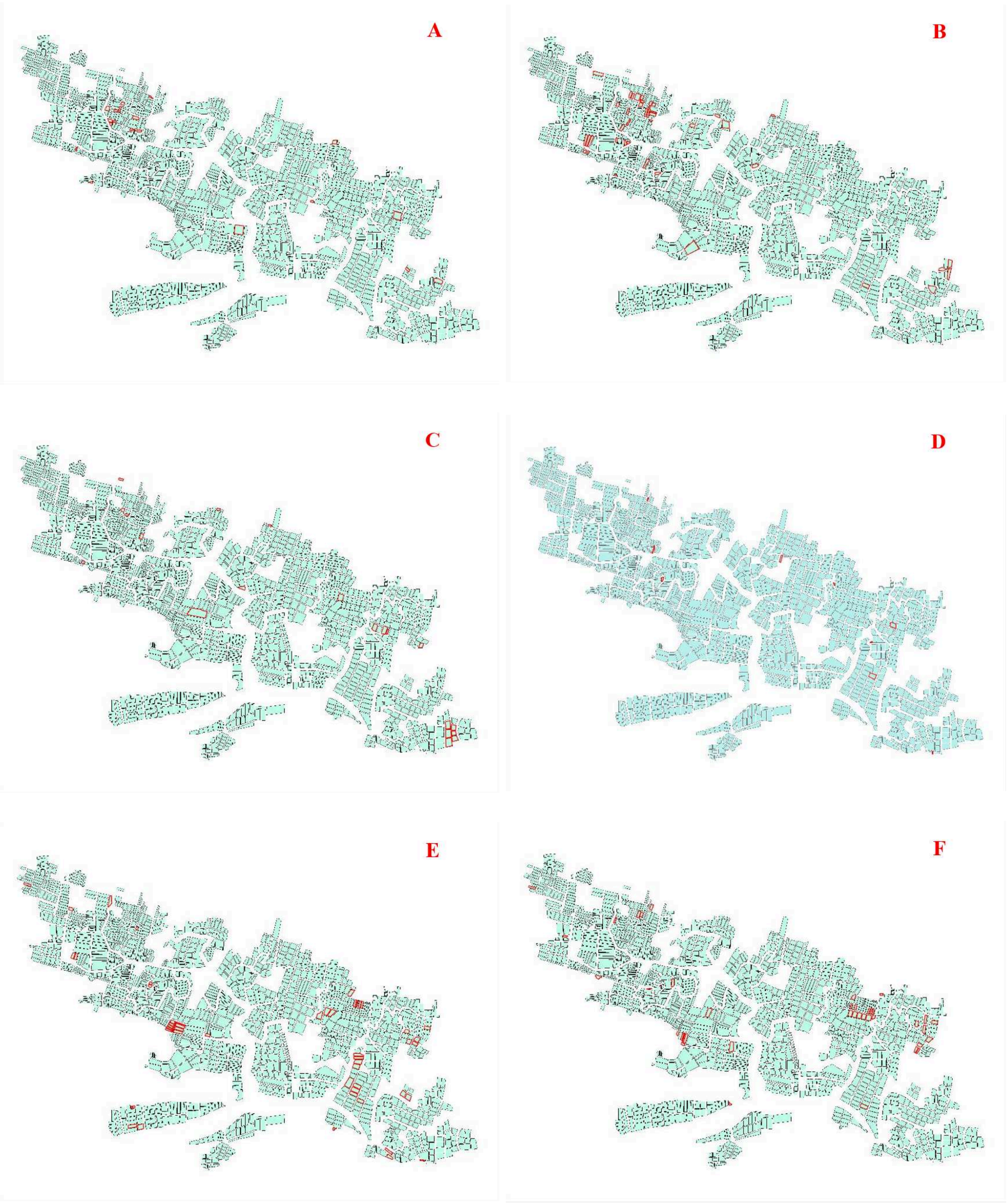


Figure 4.20 – Third sequential pattern ⟨ {WithoutActivity}, {WithoutActivity}, {WithoutActivity}, {WithActivity} ⟩ (from 2001 to 2008). Red contours represent inactive ponds remained inactive in the two consecutive years and then became active in the forth year. A: 12/10/2001, B: 09/03/2002, C: 21/02/2003, D: 27/06/2003, E: 22/09/2007, F: 19/07/2008

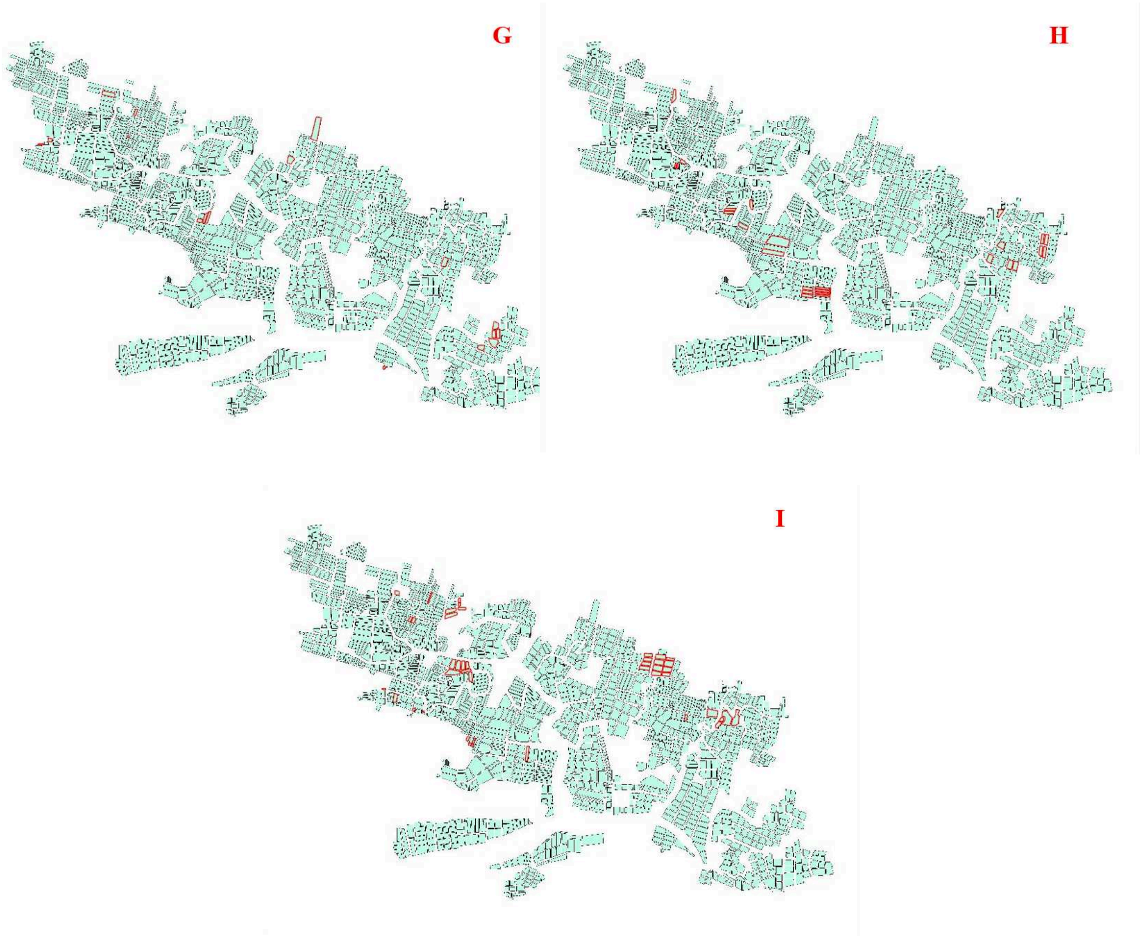


Figure 4.21 – Third sequential pattern $\langle$ {WithoutActivity}, {WithoutActivity}, {WithoutActivity}, {WithActivity} $\rangle$ (from 2009 to 2011). Red contours represent inactive ponds remained inactive in the two consecutive years and then became active in the forth year. G: 09/07/2009, H: 16/08/2010, I: 15/04/2011

Sequential mining algorithms also enable experts to use all attributes to analyze pond evolutions which could not be performed by the manual approach (considering only one attribute at a time) (Gusmawati *et al.*, 2017). Fig. 4.22 and Fig. 4.23 show an example of sequential pattern $\langle\ \{WithoutActivity, WithVegetation\},\{WithActivity, WithoutVegetation\}\ \rangle$ which is verified by 114 ponds. It represents an evolution over 2 consecutive timestamps. It represents inactive pond with vegetation becomes active pond without vegetation the next year. This is because mangrove trees were cut to rehabilitate aquaculture activity in those ponds. Moreover, wood can be used as a resource for farmers. This kind of patterns also allow experts to study automatically management of mangroves instead of time consuming manual image analysis (Proisy *et al.*, 2018).

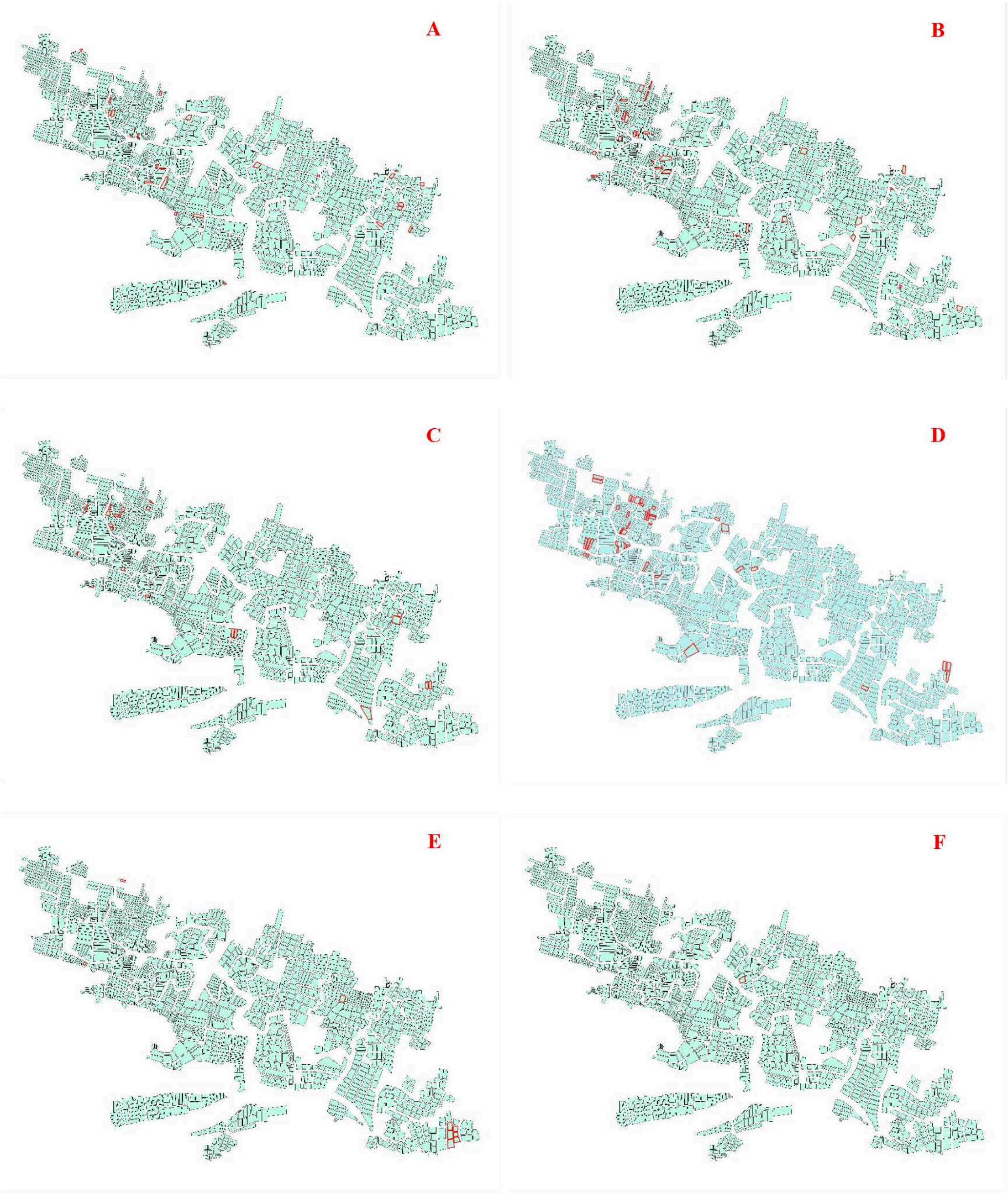


Figure 4.22 – Forth sequential pattern ⟨ {WithoutActivity, WithVegetation}, {WithActivity, WithoutVegetation} ⟩ (from 2001 to 2008). Red contours represent inactive pond with vegetation becomes active pond without vegetation in the next year. A: 12/10/2001, B: 09/03/2002, C: 21/02/2003, D: 27/06/2003, E: 22/09/2007, F: 19/07/2008

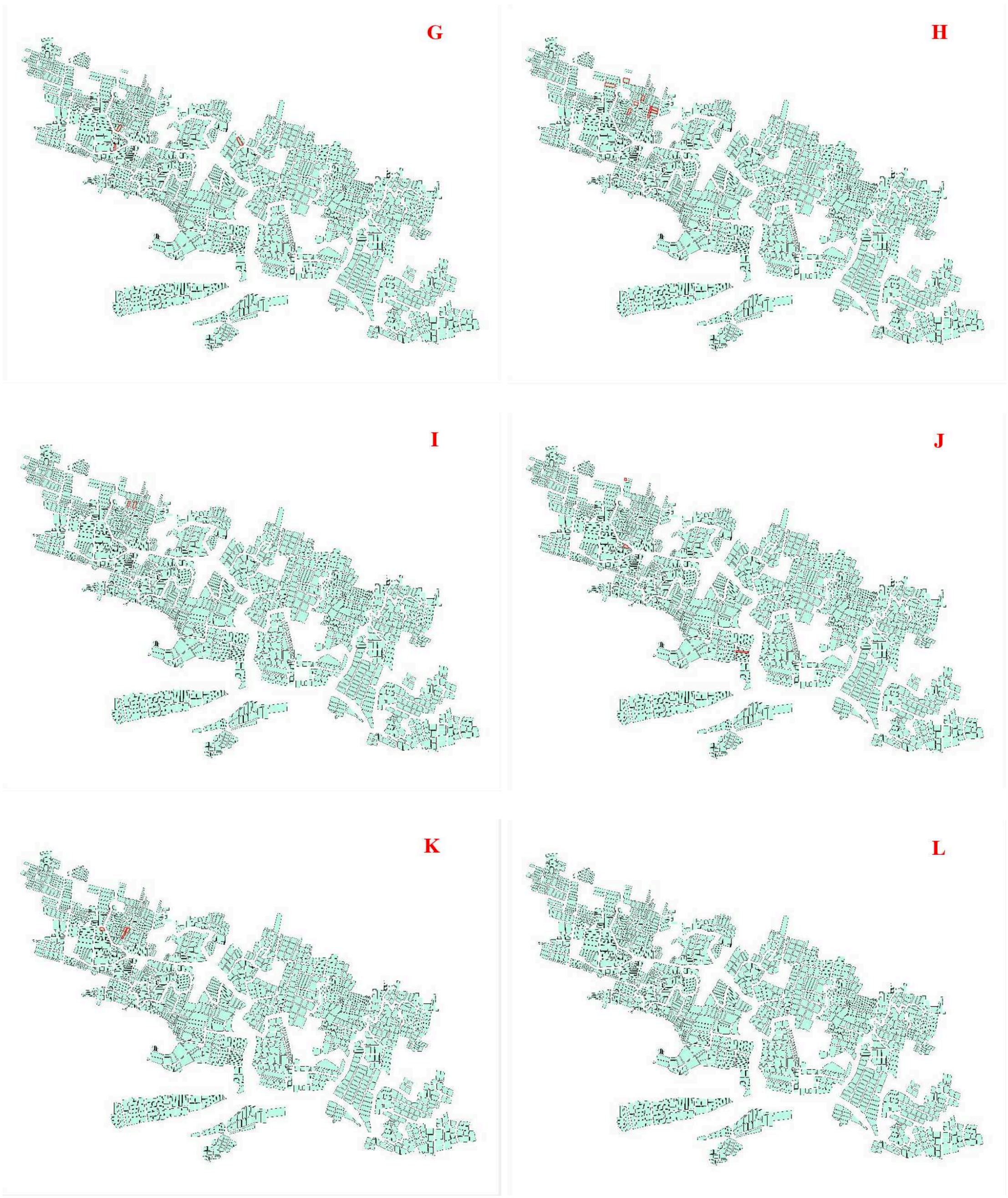


Figure 4.23 – Forth sequential pattern ⟨ {WithoutActivity, WithVegetation}, {WithActivity, WithoutVegetation} ⟩ (from 2009 to 2014). Red contours represent inactive pond with vegetation becomes active pond without vegetation in the next year. G: 09/07/2009, H: 16/08/2010, I: 15/04/2011, J: 23/10/2012 K: 10/12/2013, L: 26/03/2014

Conclusions and Discussions Sequence mining methods provide to domain experts a semi-automatic tool to analyze temporal pond and mangrove evolutions. Compared with traditional analysis done manually by experts, application of sequence mining methods have two significant advantages. Firstly, it takes experts a large amount of time to analyze ponds one by one manually, whereas sequence mining methods permit to extract interesting patterns found in (Gusmawati *et al.*, 2017) automatically. Secondly, experts could only consider one attribute at a time to analyze evolutions, while sequence mining methods enable to consider all attributes. However, these methods have also some limitations. Firstly, we cannot take into account the spatial relationships between objects. Secondly, division and fusion of objects could lead to biases of extracted pattern frequencies (could be higher than real frequency). Thirdly, frequency constraint permits to know how many sequences (objects) verify extracted pattern. Moreover, we do not know how many times extracted patterns appear in a single sequence. To deal with those limitations, we apply our approach **RPMiner** to this dataset.

7. Pond evolution by graph mining

Compared with sequence mining algorithm in (Fournier-Viger *et al.*, 2008), our algorithm **RPMiner** permits to analyze both temporal and spatial evolutions. More precisely, instead of studying evolution of individual pond over time, recurrent patterns enable experts to study how a set of connected ponds (spatially) evolve together over time (temporally). Based on those extracted recurrent patterns, we could generate aquaculture farm cadastres, identify different farmers' practices and analyze mangrove dynamic etc.

We conducted a large number of experiments by varying different parameters. We firstly present several interesting patterns extracted with parameters $minvol = 2$, $minsup = 2$, $gap = 1$ $mincos = 0$ and $mincom = 2$. $mincos$ is set to 0, because we focus on evolutions of groups of connected ponds, which usually represent farms.

To better analyze and interpret recurrent patterns, we also developed another visualisation tool using Matlab. We only visualize the first occurrence for each recurrent pattern because all occurrences of a recurrent pattern are the same. Fig. 4.24 shows an example of size-2 recurrent pattern. As mentioned previously, a recurrent pattern represents a set of connected vertices. However, we do not know how vertices are connected between each other. Thus, we visually represent a pattern by sequence of graphs with dotted lines for edges. Fig. 4.25 shows its first occurrence. That pattern depicts the evolution of 10 adjacent ponds. These adjacent inactive ponds (blue) became active (red) in the next year. The pattern appears two times, first one from 2009 to 2010 and other one from 2012 to 2013. That pattern enables experts to identify farms and establish aquaculture farm cadastre, because farms are usually managed in the similar manner by holders and recurrent patterns could depict these similar managements over time. We summarize this pattern in Table 4.11 in which the first column represents vertices of the farm (a set of adjacent ponds), the following columns represent their attributes values in consecutive timestamps. An other example of pattern is presented in Fig. 4.26 and its first occurrence is displayed in Fig. 4.27. This size-2 pattern depicts a set of active ponds (red) became inactive (blue) in the second year. It appears two times, firstly from 2007 to 2008 and secondly from 2011 to 2012. This is because 2008 and 2012 were two drying periods in which farmers oxidize harmful chemical substances and eliminate undesirable species (e.g. sulfides). Table 4.12 shows details of this pattern.

Table 4.11 – First recurrent pattern (Fig. 4.25). The first column represents vertices of the farm (a set of adjacent ponds), the following columns represent detailed attributes information of ponds in consecutive timestamps. It depicts a set of inactive ponds became active, this recurrent pattern repeats two times

Ponds	Attributes of ponds in first image (timestamp)	Attributes of ponds in second image (timestamp)
Pond474	WithoutActivity, WithoutVegetation, WithoutAerator	WithoutWater, WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond484	WithoutWater, WithoutActivity, WithBridge, WithoutVegetation, WithoutAerator	WithWater, WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond487	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithWater, WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond492	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithWater, WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond493	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator	WithWater, WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond495	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator	WithoutWater, WithoutAerator
Pond544	WithoutActivity, WithoutVegetation, WithoutAerator	WithWater, WithAcitivity, WithBridge WithoutVegetation, WithoutAerator
Pond548	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond533	WithoutActivity, WithoutVegetation, WithoutAerator	WithWater, WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond559	WithoutActivity, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator

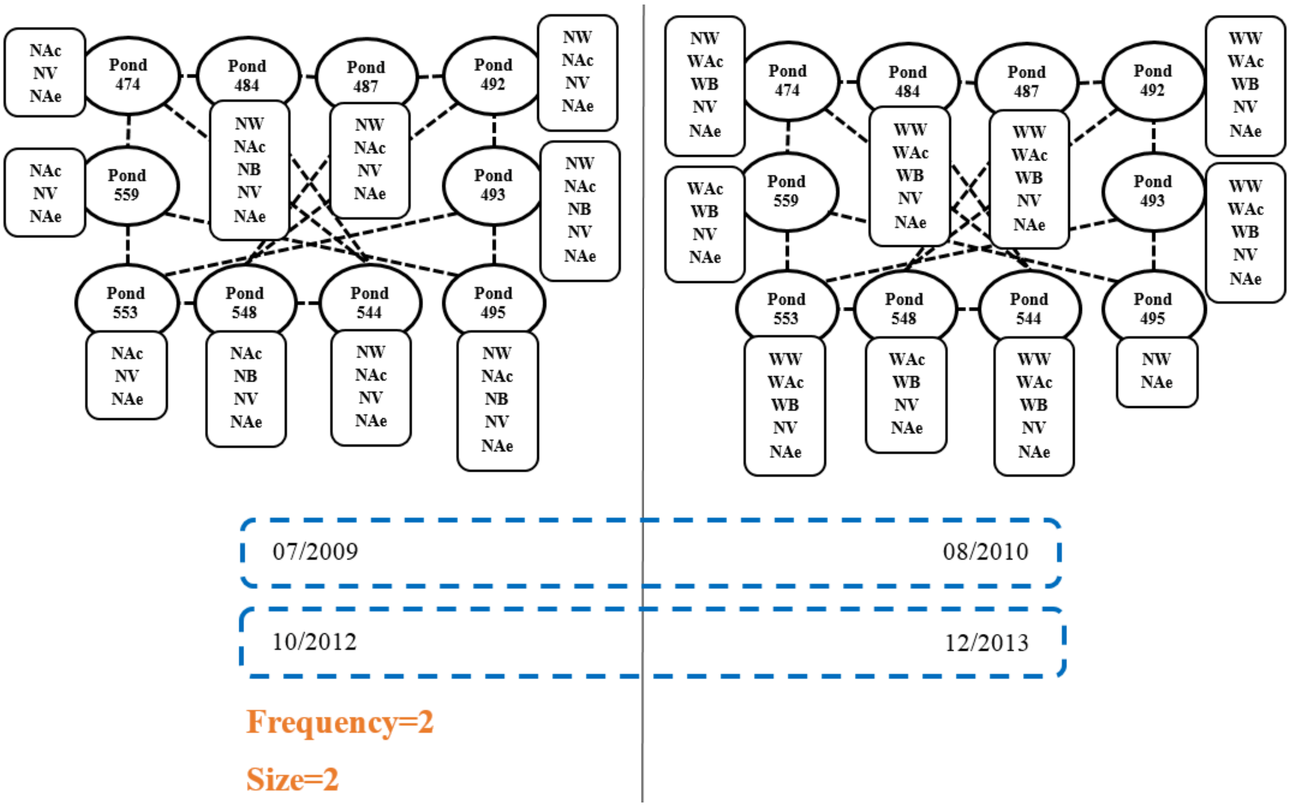


Figure 4.24 – First recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

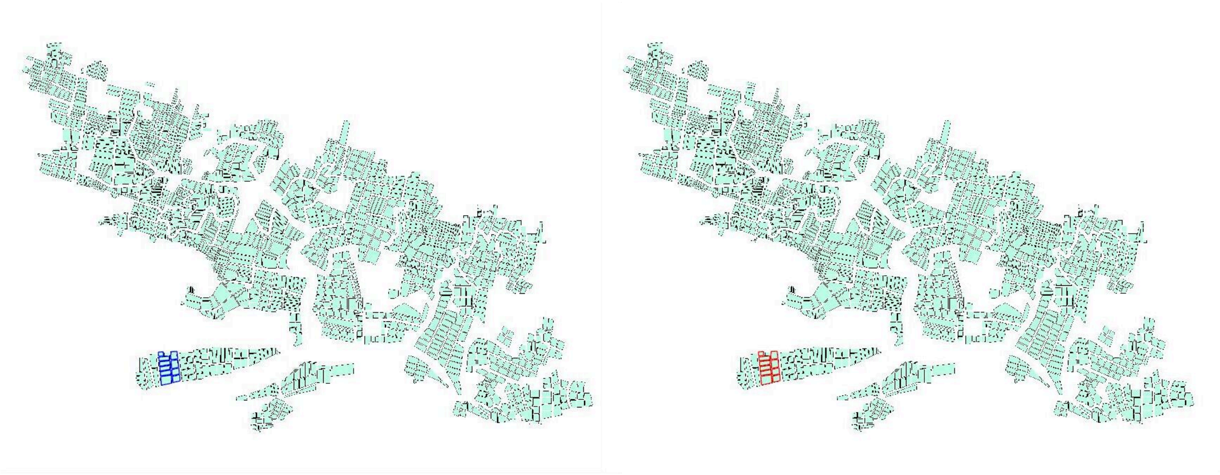


Figure 4.25 – First recurrent pattern. It depicts a set of 10 adjacent inactive (blue) ponds became active (red) in the next year. It appears two times, one from 2009 to 2010 and the other from 2012 to 2013

Table 4.12 – Second recurrent pattern (Fig. 4.27). The first column represents vertices of the farm (a set of adjacent ponds), the following columns represent the detailed attributes information of ponds in consecutive timestamps. It depicts a set of active ponds became inactive, this recurrent pattern repeats two times.

Ponds	Attributes of ponds in first image (timestamp)	Attributes of ponds in second image (timestamp)
Pond1559	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1567	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1578	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1585	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1590	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1600	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1611	WithWater, WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1654	WithoutVegetation, WithoutAerator	WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator

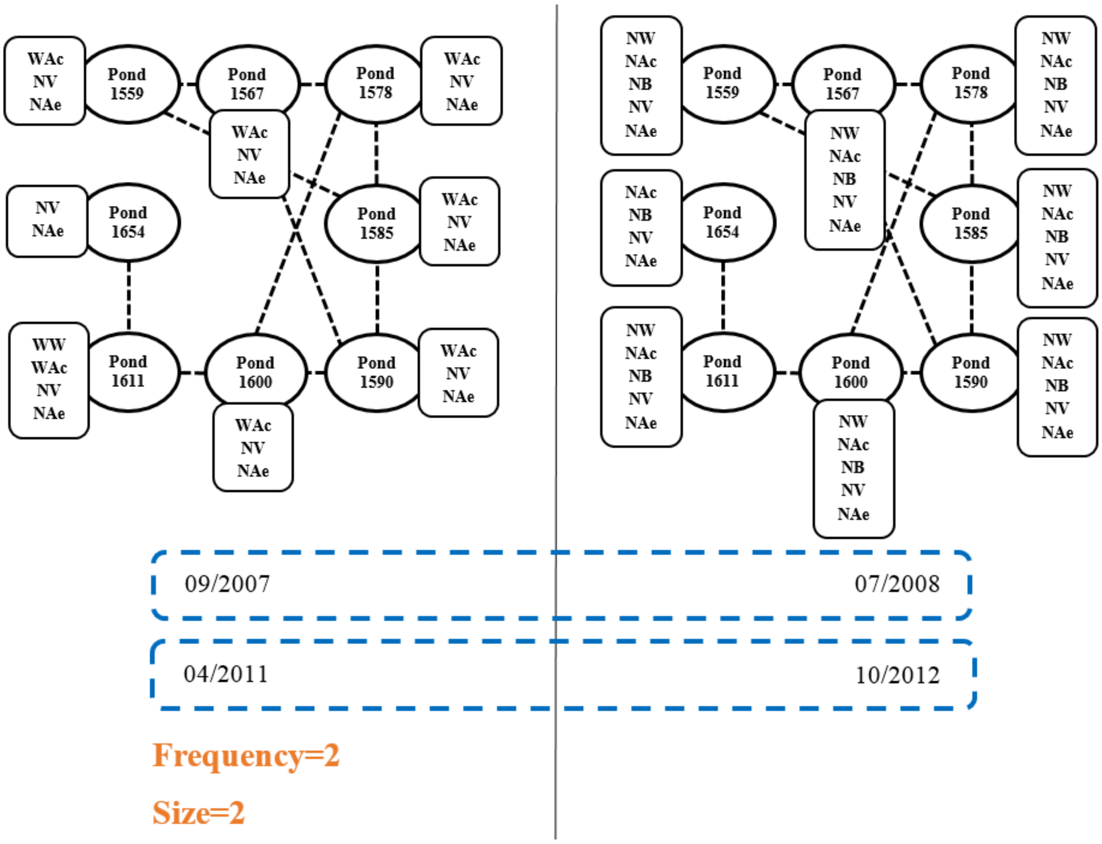


Figure 4.26 – Second recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

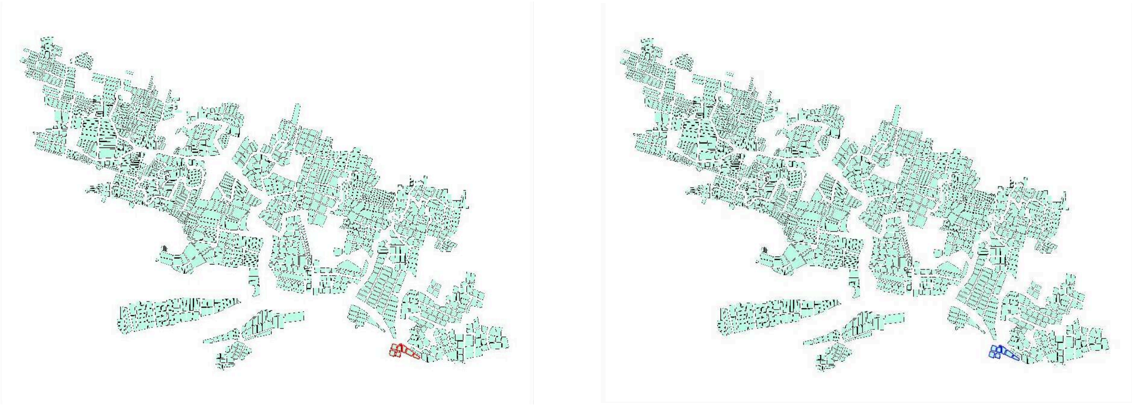


Figure 4.27 – Second recurrent pattern. It depicts a set of active ponds (red) became inactive (blue) in the next year. It appears twice, one from 2007 to 2008 and the other from 2001 to 2012.

Fig. 4.28 displays another example of size-3 recurrent pattern. Fig. 4.29 shows its first occurrence. This pattern depicts the evolution of 8 adjacent ponds. It appears 3 times, from 2007 to 2009, from 2010 to 2012 and from 2014 to 215. It allows experts not only to identify farms but also to understand another traditional management by aquaculture holders. As we can see, 7 out of 8 adjacent ponds (red) remained active over time. However, one pond of this group remained inactive. This is because aquaculture holders usually use one pond to stock water to serve other active ponds. So this pattern can be used to identify specific farm practices. (Table 4.13) gives this pattern details.

Table 4.13 – Third recurrent pattern (Fig. 4.29). The first column represents vertices of the farm (a set of adjacent ponds), the following columns represent detailed attributes information of ponds in consecutive timestamps. It depicts an evolution of 8 ponds, where 7 out of 8 adjacent ponds (red) remained active over time while one pond (blue) of this group remained inactive. This recurrent pattern repeats three times

Ponds	Attributes of ponds in first image (timestamp)	Attributes of ponds in second image (timestamp)	Attributes of ponds in third image (timestamp)
Pond123	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithoutVegetation, WithoutAerator
Pond130	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond136	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond159	WithoutWater, WithoutActivity, WithoutBridge, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutAerator
Pond170	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond171	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithoutWater, WithAcitivity, WithBridge, WithoutVegetation	WithBridge, WithoutVegetation, WithoutAerator
Pond178	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond182	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator

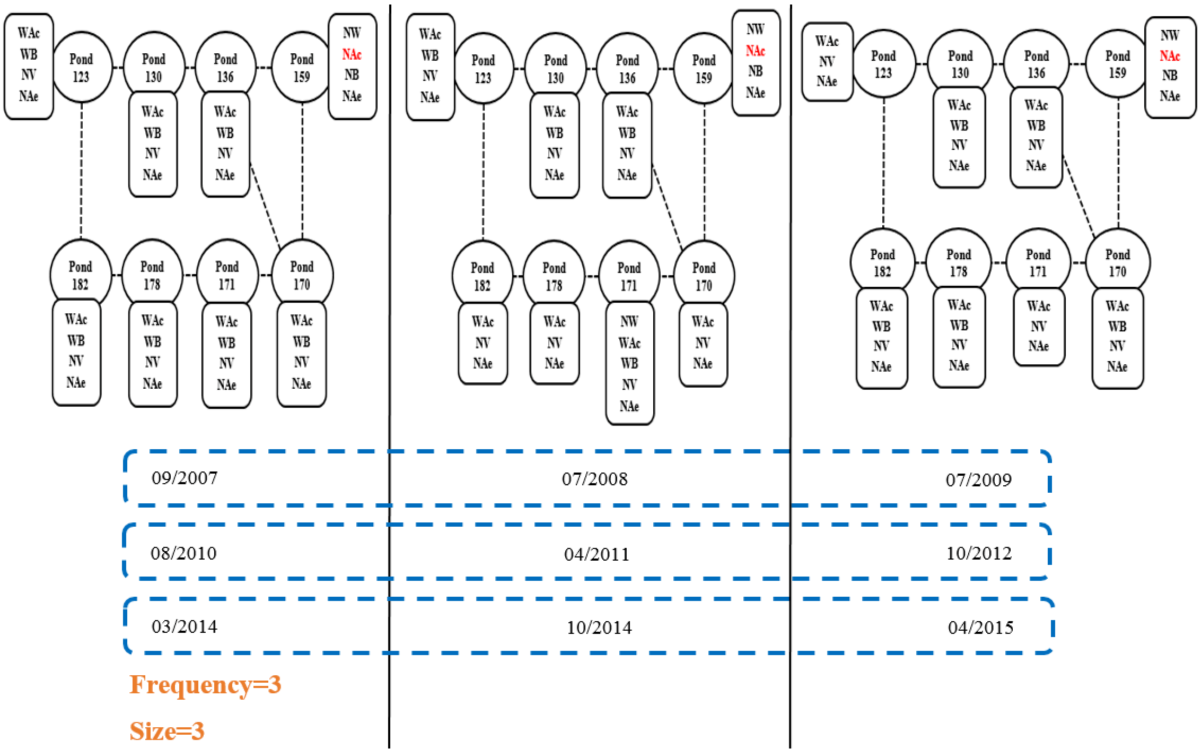


Figure 4.28 – Third recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

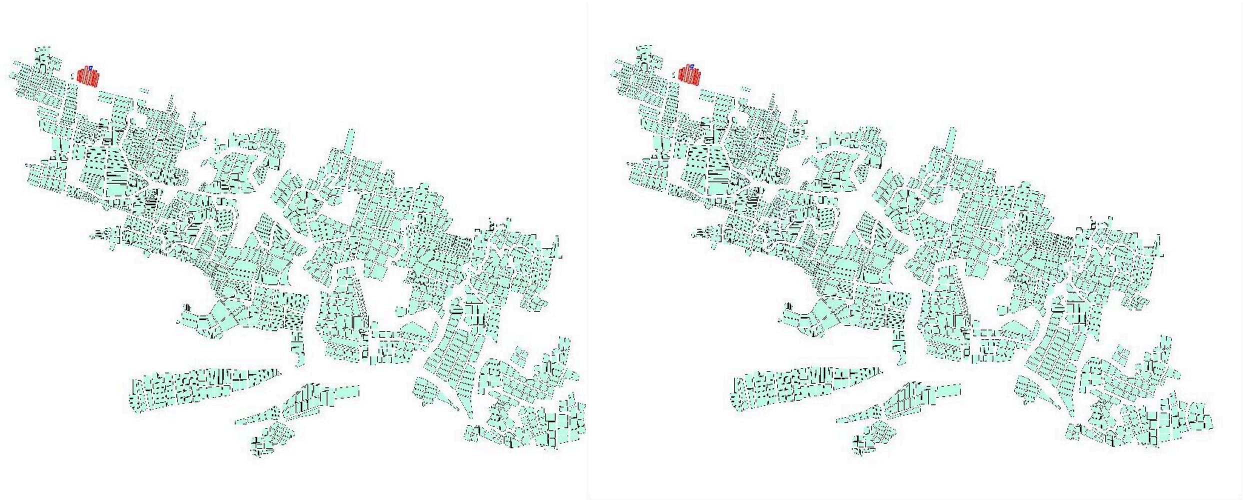


Figure 4.29 – Third recurrent pattern. It depicts the evolution of 8 ponds, where 7 of 8 adjacent ponds (red) remained active over time while one pond (blue) of this group remained inactive. This recurrent pattern repeats three times.

Fig. 4.30 depicts another recurrent pattern extracted in the aquaculture dataset. Fig. 4.31 shows its first occurrence. Again, this size-2 pattern allows experts to validate another management of farmers. As we can see, it describes the evolution of 6 adjacent ponds (6 inactive ponds in blue became active in red in the next year). This pattern appears two times: from 2007 to 2008 and then from 2009 to 2010. Moreover, as shown in Table 4.14, the pond1762 is divided into pond1762 and pond1763, and pond1767 is divided into pond1767 and pond1768. This is because it is more difficult to manage and control diseases in large ponds compared to small ponds. Therefore, to improve pond management and to control these diseases more efficiently, holders divided one pond into two or more smaller ponds. This pattern particularly shows one of the main advantages of our approach: they permit to find evolutions at vertex level, i.e. appearance, disappearance, fusion, division of ponds over time.

Table 4.14 – Forth recurrent pattern (Fig. 4.31). The first column represents vertices of the farm (a set of adjacent ponds), the following columns represent detailed attributes information of ponds in consecutive timestamps. It depicts an evolution of 6 adjacent ponds (6 inactive ponds became active in the next year) This recurrent pattern repeats two times

Ponds	Attributes of ponds in first image (timestamp)	Attributes of ponds in second image (timestamp)
Pond1753	WithoutWater, WithoutActivity, WithoutBridge, WithoutAerator	WithoutWater, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1762	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithoutVegetation, WithoutAerator
Pond1763		WithAcitivity, WithoutVegetation, WithoutAerator
Pond1767	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator	WithoutWater, WithAcitivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1768		WithAcitivity, WithoutVegetation, WithoutAerator
Pond1775	WithoutWater, WithoutActivity, WithoutBridge, WithoutAerator	WithoutWater, WithBridge, WithoutVegetation, WithoutAerator
Pond1785	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithBridge, WithoutVegetation, WithoutAerator
Pond1786	WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator	WithAcitivity, WithoutVegetation, WithoutAerator

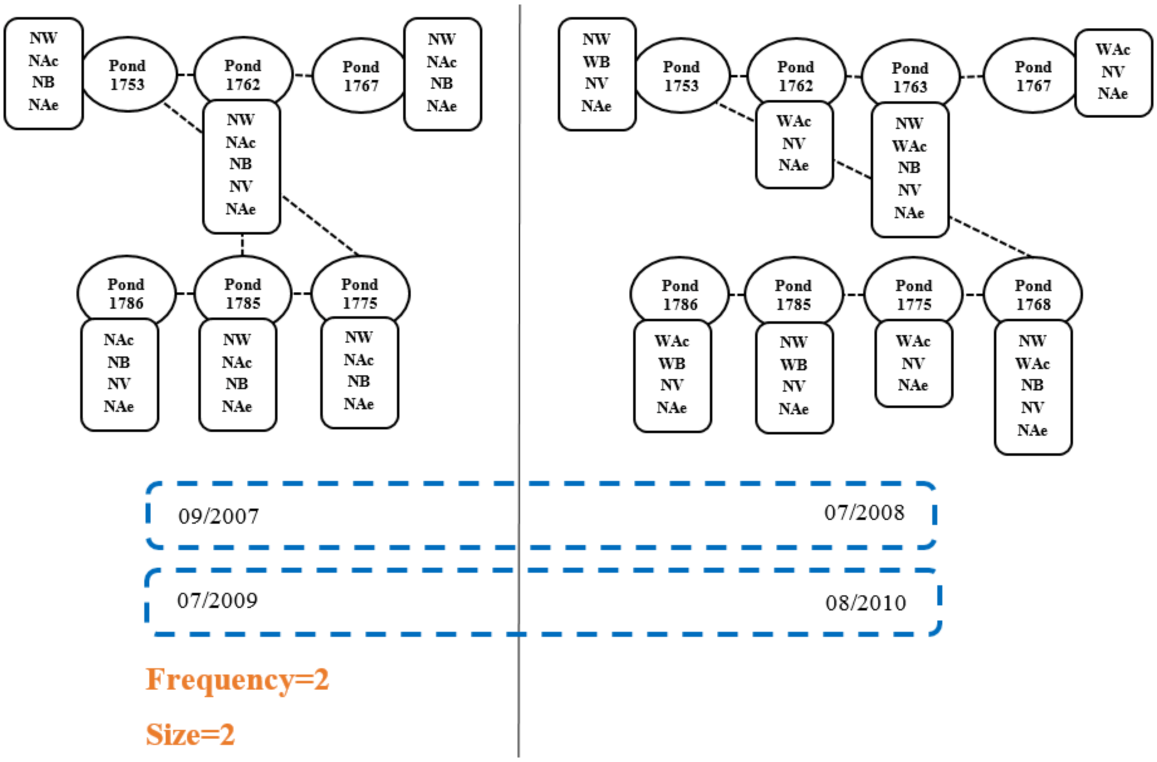


Figure 4.30 – Forth recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

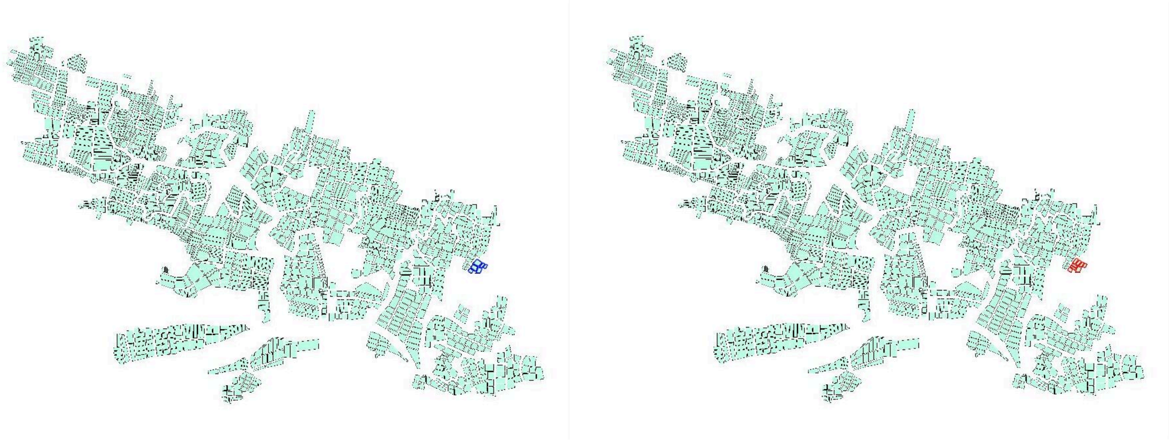


Figure 4.31 – Forth recurrent pattern. It depicts the evolution of 6 adjacent ponds (6 inactive ponds in blue became active in red in the next year). This pattern appears two times: from 2007 to 2008 and then from 2009 to 2000.

Another example of recurrent pattern is shown in Fig. 4.32. Fig. 4.33 shows its first occurrence. This size-3 pattern outlines a set of 6 active adjacent ponds (red) in the center region that became inactive (blue) in the following two years. It appears two times, from 2007 to 2009 and from 2013 to 2014. As shown from the first sequential pattern above, almost all ponds in center region were abandoned by farmers from 2003. However, this pattern shows that a set of ponds became active again in 2007 and in 2013. This is because these ponds were specifically used by scientists to conduct experimental trials. It also shows that recurrent patterns permit to detect unusual behaviors (evolutions). Table 4.15 gives details for this pattern.

Table 4.15 – Fifth recurrent pattern (Fig. 4.33). The first column represents vertices of the farm (a set of adjacent ponds), the following columns represent detailed attributes information of ponds in consecutive timestamps. It depicts a set of 6 adjacent active ponds in the center region became inactive in the two following years This recurrent pattern repeats two times

Ponds	Attributes of ponds in first image (timestamp)	Attributes of ponds in second image (timestamp)	Attributes of ponds in third image (timestamp)
Pond1165	WithWater, WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1167	WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithBridge, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1205	WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1221	WithWater, WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator
Pond1224	WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator
Pond1245	WithWater, WithAcitivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutVegetation, WithoutAerator	WithoutWater, WithoutActivity, WithoutBridge, WithoutVegetation, WithoutAerator

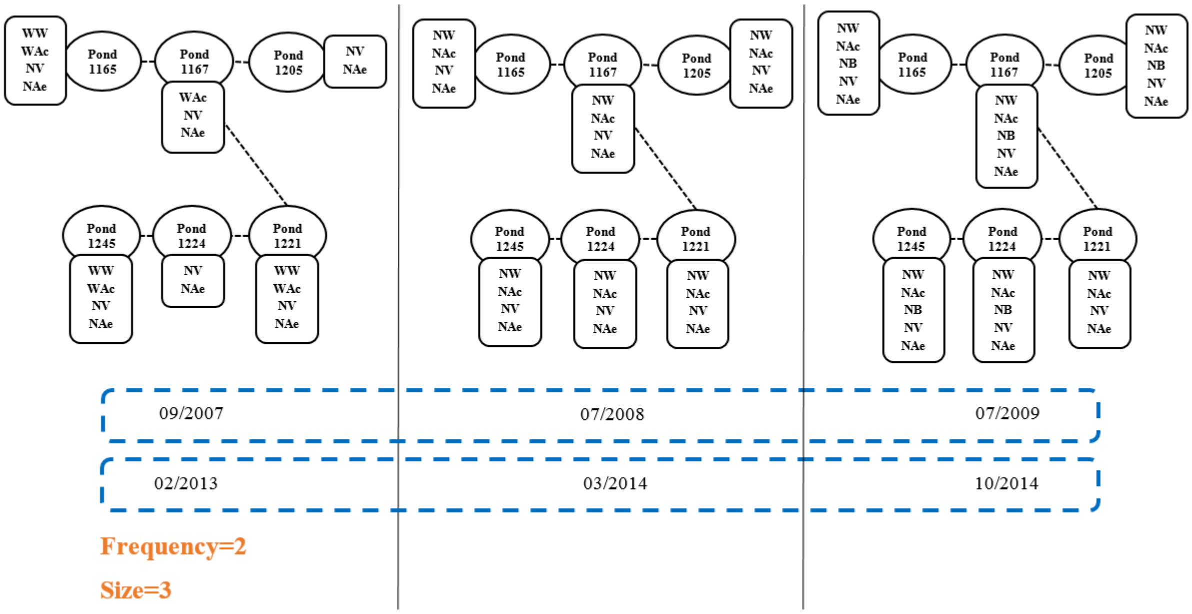


Figure 4.32 – Fifth recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

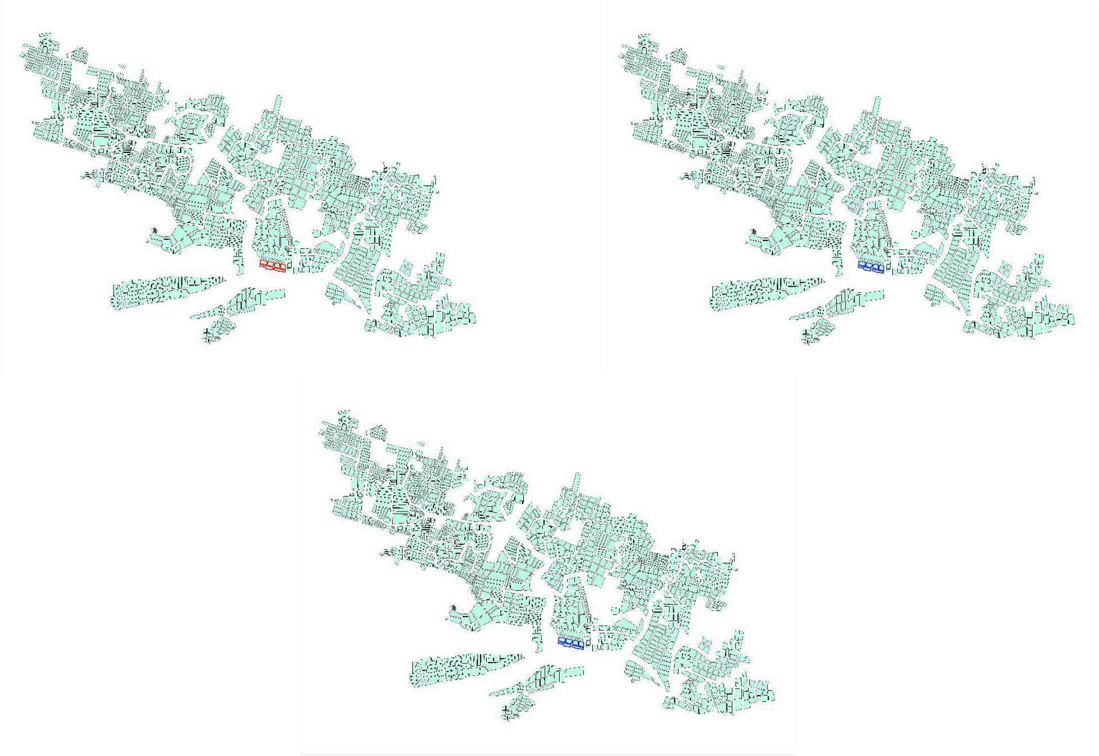


Figure 4.33 – Fifth recurrent pattern. It depicts a set of 6 adjacent active ponds (red) in the center region became inactive (blue) in the two following years. It appears two times, one from 2007 to 2009 and the other from 2013 to 2014.

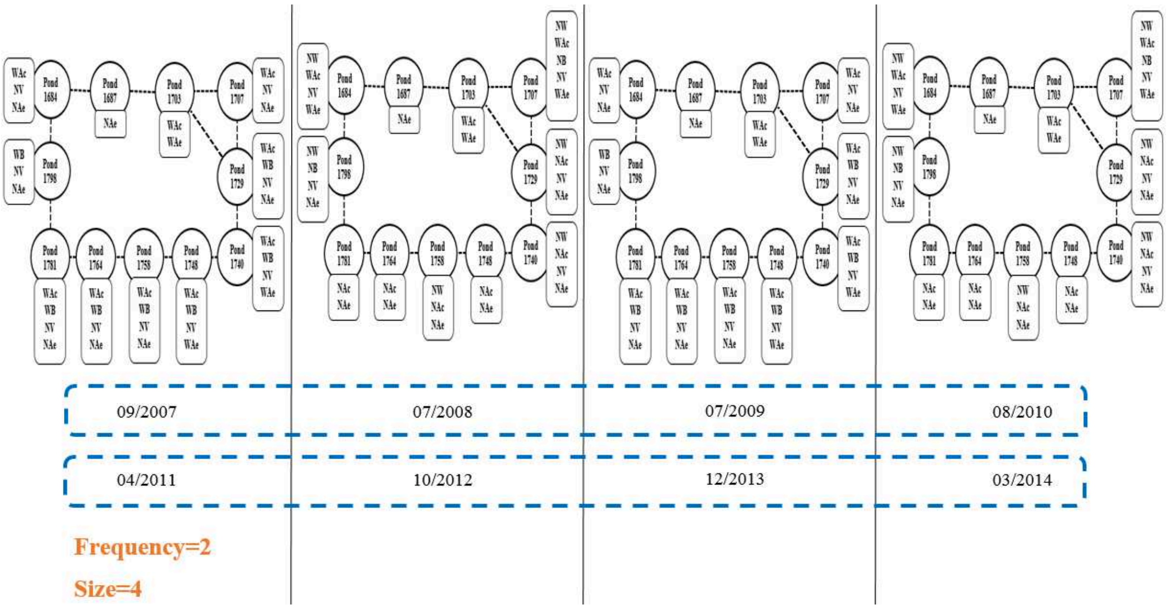


Figure 4.34 – Sixth recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

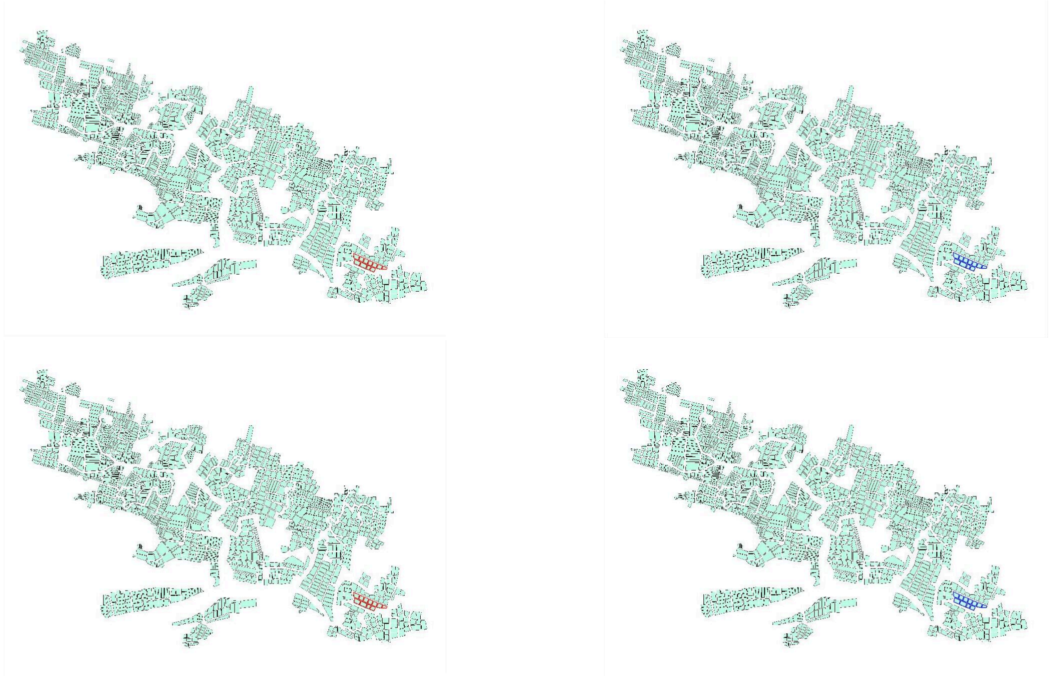


Figure 4.35 – Sixth recurrent pattern. It depicts the evolution of a set of 11 adjacent ponds over four timestamps. Most of these ponds were active (red) firstly became inactive (blue) in the second year, then became active again (red) in the third year and finally became inactive (blue) in the forth year. This pattern appears two times: from 2007 to 2010 and from 2011 to 2014.

Fig. 4.34 displays another recurrent pattern extracted from the aquaculture dataset. Fig. 4.35 shows its first occurrence. This size-4 pattern depicts the recurrent evolution of a set of 11 adjacent ponds over four timestamps. As we can see, most of those ponds were active (red) at first, and then became inactive (blue), active (red) and inactive again (blue) during the following three years (blue). This pattern appears two times: from 2007 to 2010 and from 2011 to 2014. This pattern highlights that aquaculture holders dry their ponds regularly to improve pond sediment.

Besides activity, recurrent patterns also permit to study the activity intensity level of aquaculture ponds. Fig. 4.36 describes another example of recurrent pattern. Fig. 4.37 shows its first occurrence. That pattern depicts the evolution of a farm composed of 8 adjacent ponds. As we can see in this figure, most of active ponds with aerators (red) had no more aerators (blue) in the next year. This size-2 pattern appears two times: from 10/2001 to 03/2002 and from 02/2003 to 06/2003. Farmers add and reduce the number of aerators of active ponds regularly because of these three following reasons: (1) Aerators are used to oxygenate the water column . They can raise the dissolved oxygen (DO) level to maintain oxygen level for animals and to permit aerobic bacteria to reduce biochemical oxygen demand thus improving water quality (Moulick *et al.*, 2002). (2) Mixing of pond water by aerators can reduce temperature vertical stratification and chemical substances (e.g. sulfides) (Boyd and Clay, 1998). (3) Changing culture system from intensive pond into semi-intensive pond may decrease the risk of disease emergence (Alapide *et al.*, 2010).

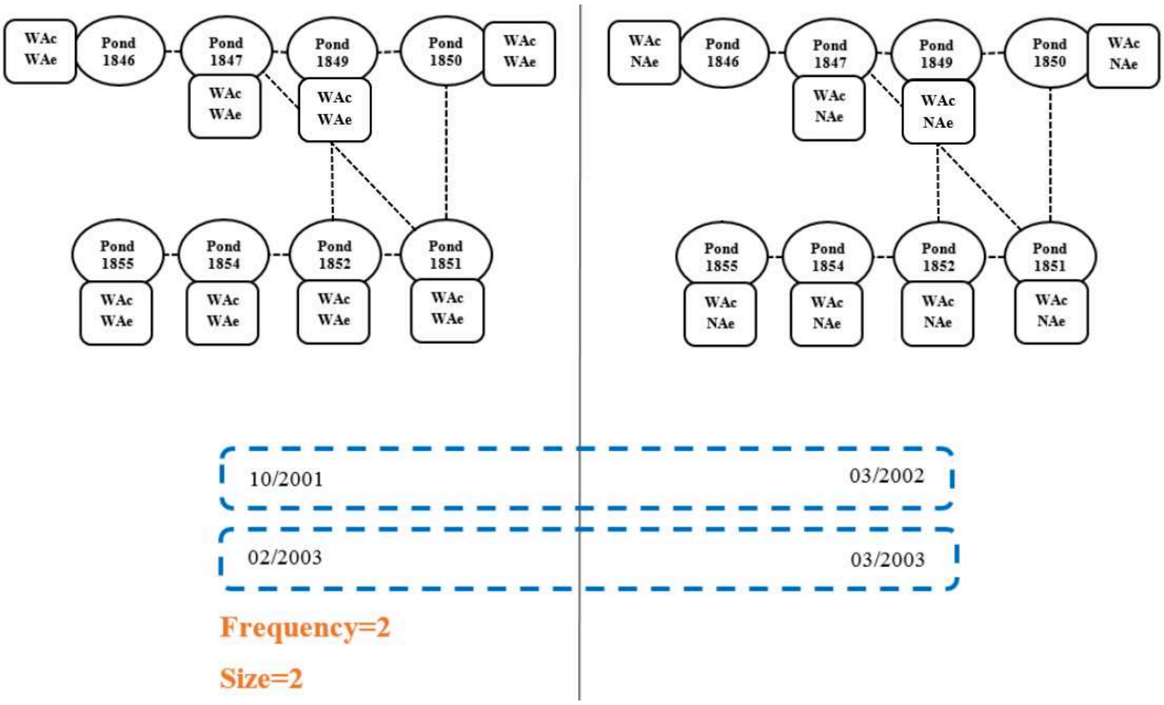


Figure 4.36 – Seventh recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

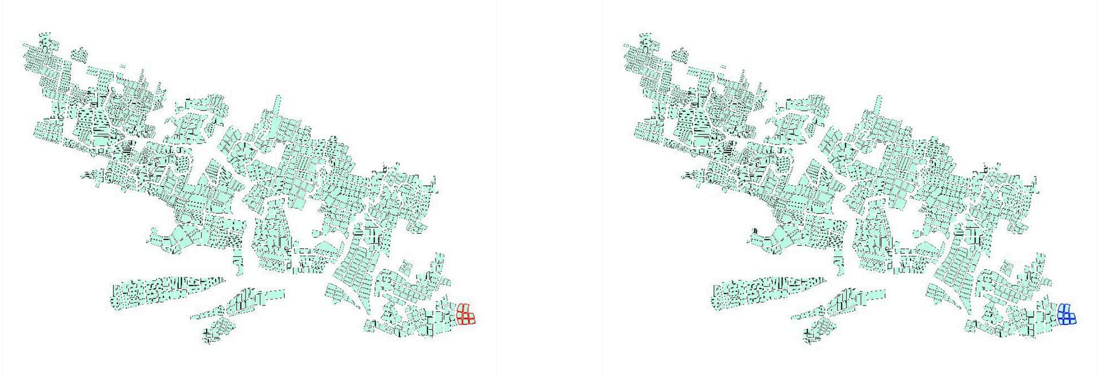


Figure 4.37 – Seventh recurrent pattern. It depicts the evolution of a farm composed of 8 adjacent ponds. Most of the active ponds with aerators (red) had no more aerators (blue) in the next year. This pattern appears two times: from 10/2001 to 03/2002, from 02/2003 to 06/2003.

Fig. 4.39 describes another example of recurrent pattern. As shown in Fig. 4.38, this size-2 pattern appears two times: from 10/2001 to 03/2002 and from 02/2003 to 06/2003. It shows that farmers usually adopt similar measures (adding and reducing aerators regularly) to manage active ponds.

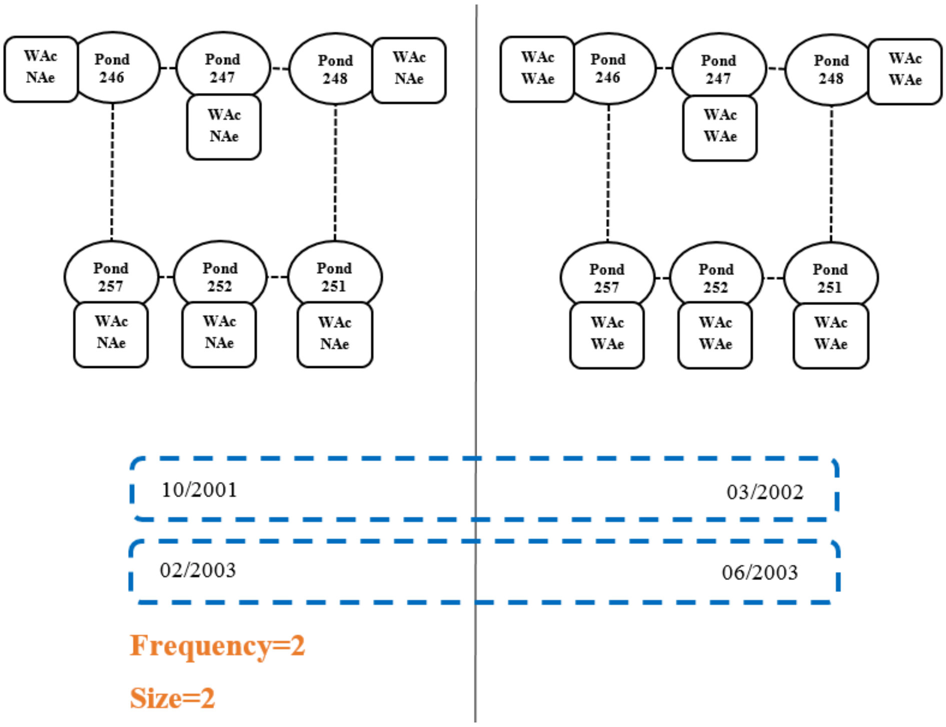


Figure 4.38 – Eighth recurrent pattern. WAc: WithActivity, NAc: WithoutActivity, WB: WithBridge, NB: WithoutBridge, WV: WithVegetation, NV: WithoutVegetation, WW: WithWator, NAc: WithoutWator, WAe: WithAerator, NAe: WithoutAerator

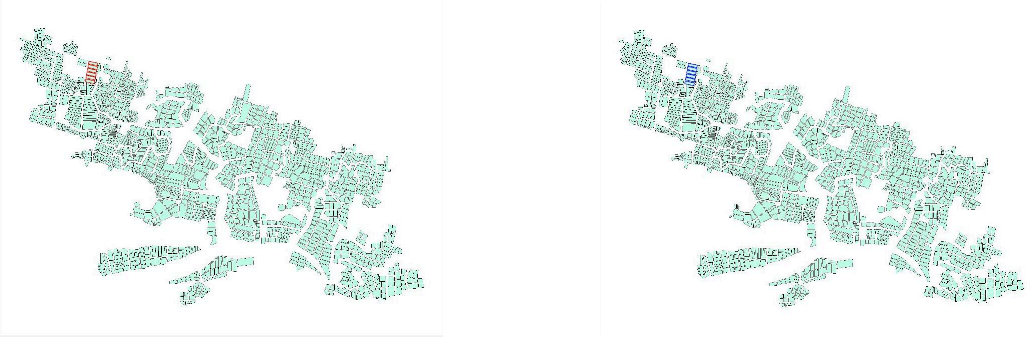


Figure 4.39 – Eighth recurrent pattern. It depicts the evolution of a farm composed of 6 adjacent ponds. These active ponds with aerators (red) had no more aerators (blue) in the next year. This pattern appears two times: from 10/2001 to 03/2002, from 02/2003 to 06/2003

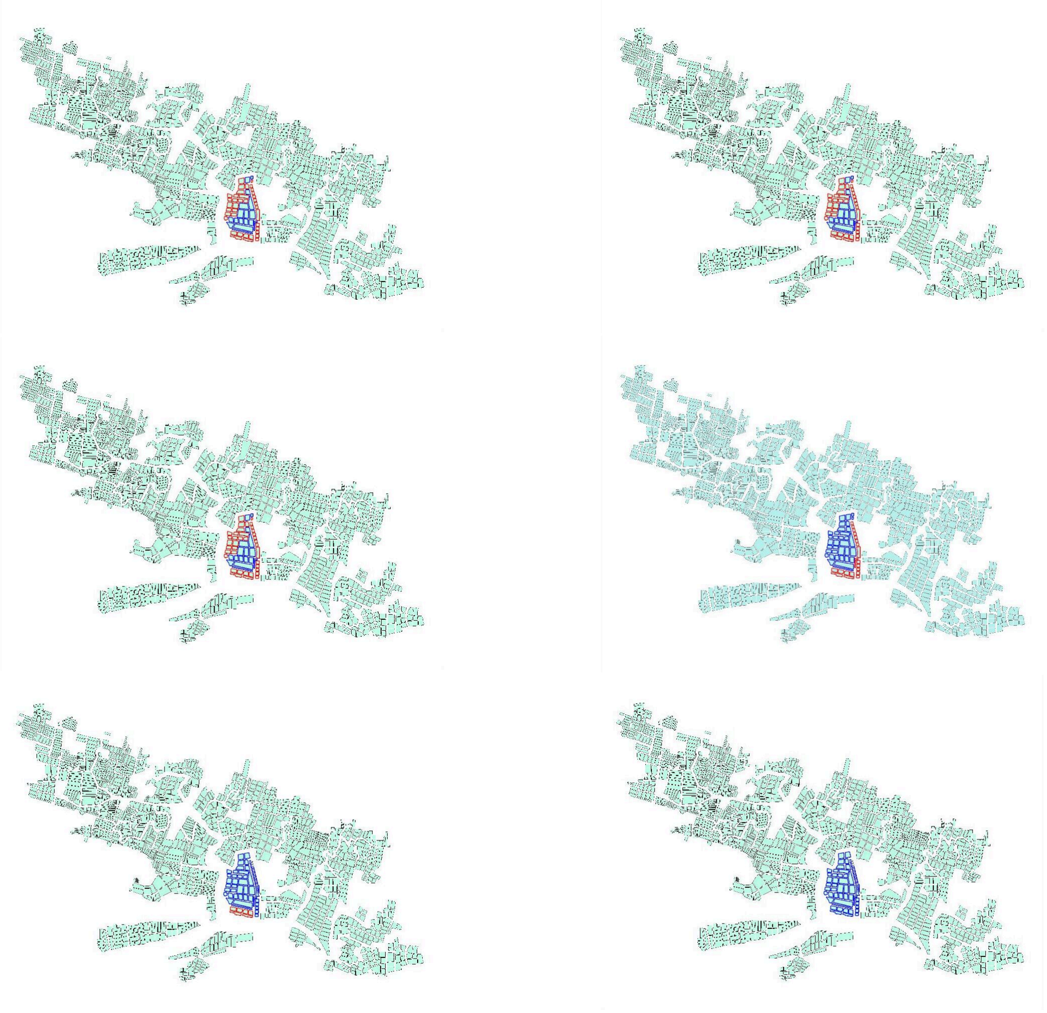


Figure 4.40 – Ninth recurrent pattern. It depicts the evolution activities of 55 adjacent ponds from 2001 to 2008, where red ponds represent active ponds and blue ponds represent inactive ponds

We conducted another experiments with parameters *minvol* $= 10$, *minsup* $= 1$, *gap* $= 1$ *mincos* $= 0$ and *mincom* $= 5$. In that context, extracted patterns could only occur one time. However, they are of great value to study viral disease and mangrove spread. Fig. 4.40 shows an example: $(\langle$ (Pond1142, ... , Pond1858: WithAcitivity | Pond1150, ... , Pond1859: WithoutAcitivity) ... (Pond1142, ... , Pond1859: WithoutAcitivity) $\rangle, \{10/2001\})$. It describes the activity evolution of an adjacent pond set during 6 consecutive times (from 2001 to 2008). To make this pattern easier to read, we grouped ponds (vertices) having the same attributes. As we can see, in 2003, most of the ponds (38 out of 55) were active (red) and 17 ponds were inactive (blue) in center region. Then, more and more active ponds adjacent to inactive ponds became inactive year after year. We can observe that, even though farmers split several ponds into smaller ponds to eliminate diseases and activate those ponds in 2007, all 59 ponds became inactive in 2008. It is due to the fact that disease transmission often occurs in the ecosystem(Salama and Murray, 2011). This pattern shows powerful impact of disease on pond activity.

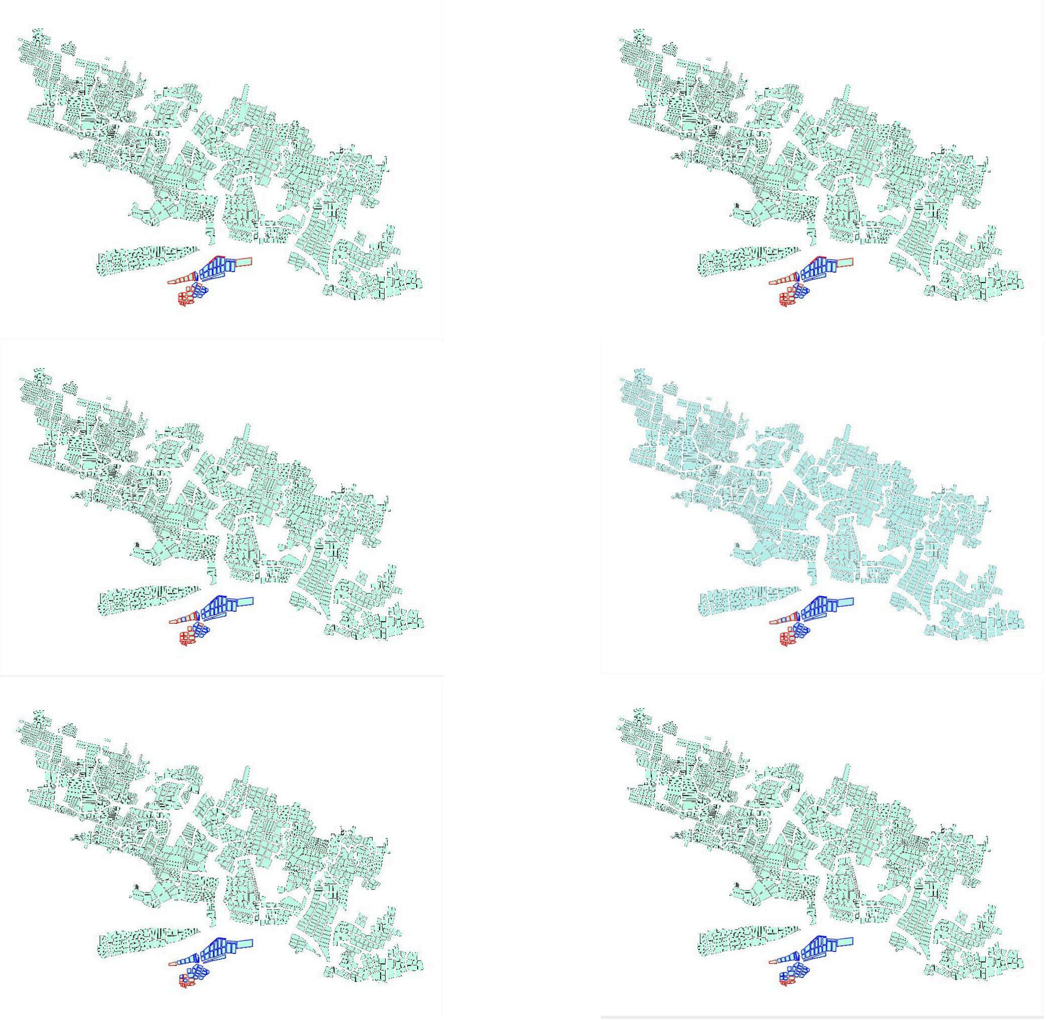


Figure 4.41 – Tenth recurrent pattern. This pattern shows mangrove spread in a farm composed of 53 ponds over 6 consecutive times (from 2001 to 2008), where red ponds represent the ponds without vegetation and blue ponds represent the ponds with vegetation

Fig. 4.41 displays a recurrent pattern describing mangroves evolution: $(\langle$ (Pond888, ... , Pond1207: WithoutVegetation | Pond1000, ... , Pond1174: WithVegetation) ... (Pond888, ... , Pond1207: WithVegetation) $\rangle, \{10/2001\})$. This pattern shows mangrove development in a farm composed of 53 ponds over 6 consecutive times (from 2001 to 2008). As we can see, in 2003, nearly half of ponds (24 out of 53) did not have vegetation (red). Then more and more ponds without vegetation and adjacent to ponds with vegetation began to be colonized by mangrove year after year. In 2008, almost all ponds (45 out of 53) were colonized by mangrove (blue). This kind of patterns can thus be used to follow pond colonization by mangrove trees.

Conclusions and Discussions Results show that recurrent patterns extracted by our method **RPMiner** could help domain experts to identify farms, understand various managements of farmers and study disease and mangrove spread over adjacent ponds etc. **RPMiner** provides domain experts a new insight to study how a set of connected ponds evolve over time. As presented above, most of extracted patterns are not very "big" in term of size and frequency (one pattern with frequency 3 and size 3, one pattern with frequency 2 and size 4, some others with frequency 2 and size 2). In fact we have extracted numerous frequent and long patterns. We present here patterns selected by experts based on their domain interest and their interpretability. Moreover, evolution circles are long for some aquaculture monitoring. As a consequence, a less frequent pattern could be meaningful for domain experts. In future works, we could study aquaculture ponds in a short term (within one production circle of shrimps). Instead of yearly satellite images, we could use monthly images to study how farmers manage (add and reduce) aerators to control diseases, improve water quality and eventually increase the production of shrimp. With those monthly data, we may extract more frequent, longer but also meaningful patterns.

In the pre-processing stage, bridges are very difficult to detect because of theirs sizes and the low contrast of image. Currently, they are detected manually by a domain expert. Thus, it would be interesting to propose methods to identify this indicator automatically. In this application, recurrent patterns depict evolutions of sets of adjacent ponds. They are local patterns, describing recurrent phenomena depending on their locations. In the future work, we would develop a new algorithm to extract more general patterns: frequent evolution of connected ponds, considering all pattern occurrences.

When comparing our approach with sequence mining algorithms (Fournier-Viger *et al.*, 2008; Fournier-Viger *et al.*, 2014; Pei *et al.*, 2007), we can see that **RPMiner** considers not only temporal relationships but also spatial relationships between vertices. Instead of studying evolutions individually, **RPMiner** permits to study evolutions of a set of connected vertices over time. Moreover, **RPMiner** considers more complicated evolutions, such as appearance, disappearance, fusion and division of vertices, while we have to introduce biases to study such data with sequence mining algorithms. Another difference is that the frequency defined by sequence mining methods is the number of sequences verifying the pattern, whereas frequency defined by **RPMiner** shows how many times that a recurrent pattern occurs over time. Those two frequency definitions give experts two different angles to understand and analyze evolutions.

Chapter 5

Conclusions and perspectives

Contents

1. Conclusions

This thesis focuses on spatio-temporal data mining and its application to aquaculture monitoring. The application of aquaculture monitoring aims at describing, understanding and monitoring shrimp farming in Indonesia based on time series of satellite images. Several works have used sequence mining and graph mining algorithms to analyze such complex data. However, these approaches are limited. Sequence mining algorithms don't consider spatial relationships between vertices or only consider direct neighboring environment, while most graph mining algorithms can study only one attribute per object instead of considering more than one attribute. For this purpose, we use a more general model, dynamic attributed graph, to study spatio-temporal data. However, neither classical sequence mining algorithms nor graph mining algorithms enable to study dynamic attributed graph. Traditional algorithms could not mine this model because mining dynamic attributed graph is much more complex compared with sequential data or dynamic labeled graph. For this purpose, we proposed a novel algorithm, **RPMiner**, to study such graph. Different from other strategies that are based on depth-first search, breadth-first search, successive projections of data or generate-test strategies, our algorithm is based on successive intersections of graphs. With connected subgraphs generated from graph intersections, patterns are progressively extended. The advantage of this approach it to avoid the generation of a large number of patterns which do not verify constraints and to explore the graph in an incremental manner. Our algorithm extracts recurrent patterns. They are in some ways sequences of connected subgraphs verifying several constraints. Those constraints aim to reduce the search space and to extract meaningful patterns. We use two constraints considering the graph structure, i.e. the connectivity and the cohesiveness. We consider also a minimum frequency constraint to filter unfrequent patterns. This constraint is based on the number of recurrences of a pattern over time. Two temporal constraints have also been used: temporal continuity and time gap. Temporal continuity enables to target patterns which describe evolutions around a common individual core. Fixed time gap allows to study evolutions in a short term and in a long term.

To evaluate our method, we have done an experimental study on both syndetic and real-world datasets. Our algorithm scales well on synthetic data according to the number of vertices, the number of edges and the number of attributes (up to 1000 attributes per vertex) while the execution time increases exponentially with the number of graphs. Our algorithm can mine bigger graphs than most algorithms proposed for mining dynamic labeled graphs (Inokuchi and Washio, 2008; Inokuchi and Washio, 2010b; Yan and Han, 2002). If we compare recurrent patterns with patterns extracted by other dynamic attributed graph mining algorithms on the same dataset (Kaytoue *et al.*, 2014; Desmier *et al.*, 2012), we observe that our approach have results of the same order of magnitude although the extracted patterns are more general (and so more costly to extract). It demonstrates the interest of the proposed approach and

its efficiency.
Our approach has been used to study evolutions of aquaculture farming. In that context, we developed a complete KDD process: from pre-processing to visualization and interpretation of results. We proposed an automatic and accurate method to extract aquaculture ponds from a low contrast satellite images. Then, we developed several methods to identify ponds' attributes. Two automatic processes have been developed to transform images of aquaculture ponds to sequential data and dynamic attributed graph. We applied a sequence mining algorithm to study temporal evolutions, and compare extracted patterns to the ones extracted with our algorithm **RPMiner** (which considers both spatial and temporal aspects). Finally, extracted patterns were visualized on original satellite images and validated by domain experts. This application showed that our approach could give experts a new insight to study spatio-temporal phenomena. It extracts recurrent evolutions of groups of adjacent ponds. Moreover, our approach permits to study complex spatio-temporal phenomena by considering appearance, disappearance, fusion and division of vertices over time.

2. Perspectives

2.1 Using other strategies

We developed a new algorithm, different from others based on depth-first search, breadth-first search or successive projections strategies. Our algorithm is an incremental approach based on successive intersections and extensions of connected components occurring over time. It requires a large mount of memory because during this extension process, we have to keep lots of patterns in memory to generate and extend patterns in the next iteration. In the future work, we propose to explore the search space in a depth-first manner. Instead of generating all patterns incrementally, we could generate only a part of size-1 patterns by processing one time combination T ($T \subseteq T_1^k$) and extend them until no more patterns could be generated or extended. Then we could generate another part of size-1 patterns by processing another time combination T' ($T' \subseteq T_i^k$-T). This process could generate patterns progressively until T_i^k becomes empty. It can reduce memory usage and allows to study a longer sequence of dynamic attributed graphs.

2.2 Parallel computing

Parallel execution of an algorithm on multi-core architecture enables to dramatically increase performance (Gepner and Kowalik, 2006; Negrevergne *et al.*, 2014). The original search space could be divided into several portions where each portion can be independently computed. In our approach, an important step is to calculate the graph intersections. The computation of all possible size-1 patterns could be parallelized for each connected component.
Besides, distributed systems can scale our algorithm. For example, Hadoop (Hadoop, 2011)

is a widely-used software framework for distributed storage and processing of large data sets. However, it processes data in and out of the disk. Besides, it is not so efficient for iterative processing, as Hadoop does not support cyclic data flow. Spark (Zaharia *et al.*, 2016) is a parallel data processing framework. It permits to run multiple tasks in parallel. Moreover, it provides in-memory processing which could run 100 times faster than Hadoop's MapReduce. It permits applications to access data from RAM instead of disk. In the future work, Spark could be used to improve the performance of our algorithm.

2.3 Mining more global patterns

Patterns extracted by our algorithm **RPMiner** represent recurrent evolutions of sets of connected vertices. Those patterns are in some ways local patterns, as it depicts recurrent evolution of specific sets of connected vertices. We do not know whether there exits other sets of vertices following same evolutions. Thus, a perspective of our work is to extract more global patterns in dynamic attributed graph (e.g. frequent patterns instead of recurrent). For this, we can use a post-processing approach or study a new pattern domain (and develop a new mining algorithm).

Clustering A post-processing could be performed to group similar recurrent patterns. It could facilitate interpretation and permit to seek for more general patterns, independent of vertices. Comparing sequence similarity measures has been much studied, (Saneifar *et al.*, 2008) proposed an adjustable similarity measure to group similar sequences. This measure considers not only itemset similarities but also their positions in sequences. In recent years, the problem of determining the similarity or distance between graphs has raised much more attention (Cha, 2007; Papadimitriou *et al.*, 2010; Jeh and Widom, 2002). (Pelillo, 1999) used graph isomorphism to evaluate the graph similarity. Two graphs are similar if they are isomorphic between these two graphs. (Zager and Verghese, 2008) assessed the similarity between two graphs by calculating similarity scores of vertices and edges. However, those measures have to be adapted to recurrent patterns because we are working with sequence of sets of itemsets, representing evolutions of connected graphs and not only sequence of itemsets. In the future work, new similarity measure and an algorithm could be proposed to group similar recurrent patterns.

New pattern domain and new mining algorithm A second approach could be to extract more general patterns: frequent subgraph evolutions, considering all pattern occurrences in dynamic attributed graph. The main advantage of these patterns is to consider evolutions independently from vertices in which they occur. In a spatio-temporal context, it means that such patterns would highlight phenomena independently from their locations. In the future work, we will study how to adapt our strategy to extract such patterns.

Bibliography

Nilfanion (2005a). *LaTeX — Wikipedia, The Free Encyclopedia* (cited page 70).

— (2005b). *LaTeX — Wikipedia, The Free Encyclopedia* (cited page 71).

— (2005c). *LaTeX — Wikipedia, The Free Encyclopedia* (cited page 71).

Abdi, M Reza and Sanjay Sharma (2007). "Strategic/tactical information management of flight operations in abnormal conditions through Network Control Centre". In: *International Journal of Information Management* 27.2, pages 119–138 (cited page 70).

Aggarwal, Charu C. and Haixun Wang, editors (2010). *Managing and Mining Graph Data.* Volume 40. Springer (cited page 13).

Agrawal, Rakesh and Ramakrishnan Srikant (1995). "Mining sequential patterns". In: *Data Engineering, 1995. Proceedings of the Eleventh International Conference on.* IEEE, pages 3–14 (cited page 10).

Ahmed, Rezwan and George Karypis (2015a). "Algorithms for mining the coevolving relational motifs in dynamic networks". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.1, page 4 (cited pages 13, 22, 23).

— (2015b). "Mining coevolving induced relational motifs in dynamic networks". In: *Proceedings of the 2nd SDM Workshop on Mining Networks and Graphs: A Big Data Analytic Challenge* (cited page 22).

Alam, Muhammad Shahinur, David W Lamb, and Muhammad Moshiur Rahman (2018). "A refined method for rapidly determining the relationship between canopy NDVI and the pasture evapotranspiration coefficient". In: *Computers and electronics in agriculture* 147.C, pages 12–17 (cited page 87).

Alapide, E Tendencia, RH Bosma, and JAJ Verreth (2010). "WSSV risk factors related to water physico-chemical properties and microflora in semi-intensive P. monodon culture ponds in the Philippines". In: *Aquaculture* 302.3-4, pages 164–168 (cited page 117).

Alatrista-Salas, Hugo, Sandra Bringay, Frédéric Flouvat, Nazha Selmaoui-Folcher, and Maguelonne Teisseire (2012). "The pattern next door: Towards spatio-sequential pattern discovery". In: *Advances in Knowledge Discovery and Data Mining.* Springer, pages 157–168 (cited page 4).

Alho, AV, JE Hopcroft, and JD Ullman (1987). "Data Structures and Algorithms. Addision-Wesley". In: *Reading (Mass.)* (cited page 50).

Apostolico, Alberto, Manuel Barbares, and Cinzia Pizzi (2011). "Speedup for a periodic subgraph miner". In: *Information Processing Letters* 111.11, pages 521–523 (cited page 22).

Aseervatham, Sujeevan, Aomar Osmani, and Emmanuel Viennet (2006). "bitSPADE: A lattice-based sequential pattern mining algorithm using bitmap representation". In: *Data*

Mining, 2006. ICDM'06. Sixth International Conference on. IEEE, pages 792–797 (cited page 11).

Aydin, Berkay and Rafal A Angryk (2016). "Spatiotemporal event sequence mining from evolving regions". In: *Pattern Recognition (ICPR), 2016 23rd International Conference on.* IEEE, pages 4172–4177 (cited page 4).

Ayres, Jay, Jason Flannick, Johannes Gehrke, and Tomi Yiu (2002). "Sequential pattern mining using a bitmap representation". In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, pages 429–435 (cited pages 10, 11).

Berlingerio, Michele, Francesco Bonchi, Björn Bringmann, and Aristides Gionis (2009). "Mining graph evolution rules". In: *Machine learning and knowledge discovery in databases*, pages 115–130 (cited pages 13, 24).

Berlingerio, Michele, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi (2011). "Foundations of multidimensional network analysis". In: *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on.* IEEE, pages 485–489 (cited page 13).

Bickel, Peter, Chao Chen, Jaimyoung Kwon, John Rice, Pravin Varaiya, and Erik van Zwet (2003). "Traffic flow on a freeway network". In: *Nonlinear Estimation and Classification.* Springer, pages 63–81 (cited page 15).

Blaschke, Thomas (2010). "Object based image analysis for remote sensing". In: *ISPRS journal of photogrammetry and remote sensing* 65.1, pages 2–16 (cited page 80).

Bogdanov, Petko, Misael Mongiovì, and Ambuj K Singh (2011). "Mining heavy subgraphs in time-evolving networks". In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on.* IEEE, pages 81–90 (cited pages 13, 15).

Borgwardt, Karsten M, Hans-Peter Kriegel, and Peter Wackersreuther (2006). "Pattern mining in frequent dynamic subgraphs". In: *Data Mining, 2006. ICDM'06. Sixth International Conference on.* IEEE, pages 818–822 (cited page 13).

Boyd, Claude E and Jason W Clay (1998). "Shrimp aquaculture and the environment". In: *Scientific American* 278.6, pages 58–65 (cited page 117).

Bringmann, Björn and Siegfried Nijssen (2008). "What is frequent in a single graph?" In: *Advances in Knowledge Discovery and Data Mining*, pages 858–863 (cited page 25).

Burnett, Carolyn and Thomas Blaschke (2003). "A multi-scale segmentation/object relationship modelling methodology for landscape analysis". In: *Ecological modelling* 168.3, pages 233–249 (cited page 79).

Celik, Mete (2015). "Partial spatio-temporal co-occurrence pattern mining". In: *Knowledge and Information Systems* 44.1, pages 27–49 (cited page 4).

Celik, Mete, Shashi Shekhar, James P Rogers, and James A Shine (2006). "Sustained emerging spatio-temporal co-occurrence pattern mining: A summary of results". In: *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on.* IEEE, pages 106–115 (cited page 4).

— (2008). "Mixed-Drove Spatio-Temporal Co-occurrence Pattern Mining". In: *network* 11, page 15 (cited page 4).

Cerf, Loïc, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut (2008). "Data-Peeler: Constraint-based closed pattern mining in n-ary relations". In: *proceedings of the 2008 SIAM International conference on Data Mining*. SIAM, pages 37–48 (cited page 23).

— (2009a). "Closed patterns meet n-ary relations". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.1, page 3 (cited page 23).

Cerf, Loïc, Tran Bao Nhan Nguyen, and Jean-François Boulicaut (2009b). "Discovering relevant cross-graph cliques in dynamic networks". In: *International symposium on methodologies for intelligent systems*. Springer, pages 513–522 (cited pages 15, 24).

Cha, Sung-Hyuk (2007). "Comprehensive survey on distance/similarity measures between probability density functions". In: *City* 1.2, page 1 (cited page 127).

Chettri, Nakul, Kabir Uddin, Sunita Chaudhary, and Eklabya Sharma (2013). "Linking spatio-temporal land cover change to biodiversity conservation in the Koshi Tappu Wildlife Reserve, Nepal". In: *Diversity* 5.2, pages 335–351 (cited page 80).

Cook, Diane J and Lawrence B Holder (2006). *Mining graph data*. John Wiley & Sons (cited page 13).

Damiand, Guillaume, Colin De La Higuera, Jean-Christophe Janodet, Émilie Samuel, and Christine Solnon (2009). "A polynomial algorithm for submap isomorphism". In: *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, pages 102–112 (cited page 21).

Deshpande, Mukund, Michihiro Kuramochi, Nikil Wale, and George Karypis (2005). "Frequent substructure-based approaches for classifying chemical compounds". In: *IEEE Transactions on Knowledge and Data Engineering* 17.8, pages 1036–1050 (cited page 13).

Desmier, Elise, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut (2012). "Cohesive co-evolution patterns in dynamic attributed graphs". In: *International Conference on Discovery Science*. Springer, pages 110–124 (cited pages 3, 4, 27, 29, 33, 36, 52, 66, 125).

— (2013). "Trend mining in dynamic attributed graphs". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 654–669 (cited pages 3, 4, 29, 33).

Diesner, Jana, Terrill L Frantz, and Kathleen M Carley (2005). "Communication networks from the Enron email corpus âĂIJIt's always about the people. Enron is no differentâĂİ". In: *Computational & Mathematical Organization Theory* 11.3, pages 201–228 (cited page 15).

Diot, Fabien, Elisa Fromont, Baptiste Jeudy, Emmanuel Marilly, and Olivier Martinot (2012). "Graph mining for object tracking in videos". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 394–409 (cited page 21).

Dronova, Iryna, Peng Gong, Lin Wang, and Liheng Zhong (2015). "Mapping dynamic cover types in a large seasonally flooded wetland using extended principal component analysis and object-based classification". In: *Remote Sensing of Environment* 158, pages 193–206 (cited page 80).

Espindola, GM, Gilberto Câmara, IA Reis, LS Bins, and AM Monteiro (2006). "Parameter selection for region-growing image segmentation algorithms using spatial autocorrelation". In: *International Journal of Remote Sensing* 27.14, pages 3035–3040 (cited page 80).

Fayyad, Usama M, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy (1996). “Advances in knowledge discovery and data mining”. In: (cited page 2).

Fiedler, Mathias and Christian Borgelt (2007). “Subgraph support in a single large graph”. In: *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, pages 399–404 (cited page 38).

Foody, Giles M (2004). “Thematic map comparison”. In: *Photogrammetric Engineering & Remote Sensing* 70.5, pages 627–633 (cited page 84).

Fournier-Viger, Philippe, Roger Nkambou, and Engelbert Mephu Nguifo (2008). “A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems”. In: *Mexican International Conference on Artificial Intelligence*. Springer, pages 765–778 (cited pages 12, 37, 96, 105, 121).

Fournier-Viger, Philippe, Antonio Gomariz, Ted Gueniche, Espérance Mwamikazi, and Rincy Thomas (2013). “TKS: efficient mining of top-k sequential patterns”. In: *International Conference on Advanced Data Mining and Applications*. Springer, pages 109–120 (cited page 12).

Fournier-Viger, Philippe, Antonio Gomariz, Manuel Campos, and Rincy Thomas (2014). “Fast vertical mining of sequential patterns using co-occurrence information”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 40–52 (cited pages 10, 11, 96, 121).

Fraisse, CW, KA Sudduth, and NR Kitchen (2001). “Delineation of site-specific management zones by unsupervised classification of topographic attributes and soil electrical conductivity”. In: *Transactions of the ASAE* 44.1, page 155 (cited page 85).

Gao, Bo-Cai (1996). “NDWIâĂŤA normalized difference water index for remote sensing of vegetation liquid water from space”. In: *Remote sensing of environment* 58.3, pages 257–266 (cited page 86).

Gao, Chuancong, Jianyong Wang, Yukai He, and Lizhu Zhou (2008). “Efficient mining of frequent sequence generators”. In: *Proceedings of the 17th international conference on World Wide Web*. ACM, pages 1051–1052 (cited page 12).

Gepner, Pawel and Michal Filip Kowalik (2006). “Multi-core processors: New way to achieve high system performance”. In: *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*. IEEE, pages 9–13 (cited page 126).

Gomariz, Antonio, Manuel Campos, Roque Marin, and Bart Goethals (2013). “ClaSP: an efficient algorithm for mining frequent closed sequences”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 50–61 (cited page 38).

Günnemann, Stephan and Thomas Seidl (2010). “Subgraph mining on directed and weighted graphs”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 133–146 (cited page 16).

Gusmawati, Niken, Benoît Soulard, Nazha Selmaoui-Folcher, Christophe Proisy, Akhmad Mustafa, Romain Le Gendre, Thierry Laugier, and Hugues Lemonnier (2017). “Surveying shrimp aquaculture pond activity using multitemporal VHSR satellite images-case study from the Perancak estuary, Bali, Indonesia”. In: *Marine pollution bulletin* (cited pages 2, 86, 90, 96, 98, 100, 102, 105).

Gusmawati, Niken F, Cheng Zhi, Benoît Soulard, Hugues Lemonnier, and Nazha Selmaoui-Folcher (2016). “Aquaculture Pond Precise Mapping in Perancak Estuary, Bali, Indonesia”. In: *Journal of Coastal Research* 75.sp1, pages 637–641 (cited page 80).

Hadoop, Apache (2011). *Apache hadoop* (cited page 126).

Halder, Sajal, Md Samiullah, and Young-Koo Lee (2017). “Supergraph based periodic pattern mining in dynamic social networks”. In: *Expert Systems with Applications* 72, pages 430–442 (cited page 22).

Han, Jiawei, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu (2000). “FreeSpan: frequent pattern-projected sequential pattern mining”. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 355–359 (cited page 12).

Holder, Lawrence B, Diane J Cook, *et al.* (2009). “Learning patterns in the dynamics of biological networks”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 977–986 (cited pages 15, 24).

Hossain, Md Zakir, W Muttitanon, M Phillips, and NK Tripathi (2002). “Monitoring shrimp farming development from the space: A RS and GIS approach in Kandleru Creek area, Andhra Pradesh, India”. In: *Proceedings of the Map Asia* (cited page 80).

Huan, Jun, Wei Wang, and Jan Prins (2003). “Efficient mining of frequent subgraphs in the presence of isomorphism”. In: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, pages 549–552 (cited page 13).

Huang, Kuo-Yu, Chia-Hui Chang, Jiun-Hung Tung, and Cheng-Tao Ho (2006). “COBRA: closed sequential pattern mining using bi-phase reduction approach”. In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer, pages 280–291 (cited pages 12, 38).

Huang, Yan, Liqin Zhang, and Pusheng Zhang (2007). “A framework for mining sequential patterns from spatio-temporal event data sets”. In: *IEEE Transactions on Knowledge & Data Engineering* 4, pages 433–448 (cited page 4).

Inokuchi, Akihiro and Takashi Washio (2008). “A fast method to mine frequent subsequences from graph sequence data”. In: *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*. IEEE, pages 303–312 (cited pages 13, 17, 19, 125).

— (2010a). “GTRACE2: Improving performance using labeled union graphs”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 178–188 (cited page 18).

— (2010b). “Mining frequent graph sequence patterns induced by vertices”. In: *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, pages 466–477 (cited pages 4, 13, 18, 125).

Inokuchi, Akihiro, Takashi Washio, and Hiroshi Motoda (2000). “An apriori-based algorithm for mining frequent substructures from graph data”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pages 13–23 (cited pages 13, 21).

Iváncsy, Renáta and István Vajk (2006). “Frequent pattern mining in web log data”. In: *Acta Polytechnica Hungarica* 3.1, pages 77–90 (cited page 9).

Jaccard, Paul (1912). “The distribution of the flora in the alpine zone.” In: *New phytologist* 11.2, pages 37–50 (cited pages 28, 37).

Jain, Monika and SK Singh (2011). “A survey on: content based image retrieval systems using clustering techniques for large data sets”. In: *International Journal of Managing Information Technology* 3.4, page 23 (cited page 85).

Jaschke, Robert, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme (2006). “TRIAS–An Algorithm for Mining Iceberg Tri-Lattices”. In: *Data Mining, 2006. ICDM’06. Sixth International Conference on.* IEEE, pages 907–911 (cited page 23).

Jeh, Glen and Jennifer Widom (2002). “SimRank: a measure of structural-context similarity”. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, pages 538–543 (cited page 127).

Ji, Liping, Kian-Lee Tan, and Anthony KH Tung (2006). “Mining frequent closed cubes in 3D datasets”. In: *Proceedings of the 32nd international conference on Very large data bases.* VLDB Endowment, pages 811–822 (cited page 23).

Jiang, Chuntao, Frans Coenen, and Michele Zito (2010). “Frequent sub-graph mining on edge weighted graphs”. In: *International Conference on Data Warehousing and Knowledge Discovery.* Springer, pages 77–88 (cited pages 15, 16).

Kaytoue, Mehdi, Yoann Pitarch, Marc Plantevit, and Céline Robardet (2014). “Triggering Patterns of Topology Changes in Dynamic Graphs”. In: *The 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.* Edited by Guandong Xu Xindong Wu Martin Ester. Beijing, China, China: IEEE/ACM, 55:1–55:17 (cited pages 3, 4, 26, 27, 53, 125).

Khan, Arijit, Xifeng Yan, and Kun-Lung Wu (2010). “Towards proximity pattern mining in large graphs”. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data.* ACM, pages 867–878 (cited page 13).

Kim, Minho, Timothy A Warner, Marguerite Madden, and Douglas S Atkinson (2011). “Multi-scale GEOBIA with very high spatial resolution digital aerial imagery: scale, texture and image objects”. In: *International Journal of Remote Sensing* 32.10, pages 2825–2850 (cited page 80).

Kiran, R Uday and P Krishna Reddy (2009). “Mining rare periodic-frequent patterns using multiple minimum supports”. In: *work* 5.6, pages 7–8 (cited page 12).

Kiran, R Uday, Masaru Kitsuregawa, and P Krishna Reddy (2016). “Efficient discovery of periodic-frequent patterns in very large databases”. In: *Journal of Systems and Software* 112, pages 110–121 (cited page 12).

Kuramochi, Michihiro and George Karypis (2004). “An efficient algorithm for discovering frequent subgraphs”. In: *IEEE Transactions on Knowledge and Data Engineering* 16.9, pages 1038–1051 (cited page 13).

Lahiri, Mayank and Tanya Y Berger-Wolf (2008). “Mining periodic behavior in dynamic social networks”. In: *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on.* IEEE, pages 373–382 (cited page 21).

— (2010). “Periodic subgraph mining in dynamic networks”. In: *Knowledge and Information Systems* 24.3, pages 467–497 (cited page 13).

Lan, Guo-Cheng, Tzung-Pei Hong, Vincent S Tseng, and Shyue-Liang Wang (2014). “Applying the maximum utility measure in high utility sequential pattern mining”. In: *Expert Systems with Applications* 41.11, pages 5071–5081 (cited page 12).

Lee, Gangin and Unil Yun (2012). “Mining Weighted Frequent Sub-graphs with Weight and Support Affinities.” In: *MIWAI*. Springer, pages 224–235 (cited page 17).

Lin, Nancy P, Wei-Hua Hao, Hung-Jen Chen, Hao-En Chueh, Chung-I Chang, *et al.* (2007). “Fast mining maximal sequential patterns”. In: *Proceedings of the 7th International Conference on Simulation, Modeling and Optimization, September*, pages 15–17 (cited page 12).

Lo, David, Siau-Cheng Khoo, and Jinyan Li (2008). “Mining and ranking generators of sequential patterns”. In: *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, pages 553–564 (cited page 12).

Lu, S and C Li (2004). “AprioriAdjust: An efficient algorithm for discovering the maximum sequential patterns”. In: *Proc. Intern. Workshop knowl. Grid and grid intell* (cited page 12).

Luo, Congnan and Soon M Chung (2005). “Efficient mining of maximal sequential patterns using multiple samples”. In: *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, pages 415–426 (cited page 12).

Meinel, Gotthard, Marco Neubert, and Johannes Reder (2001). “The potential use of very high resolution satellite data for urban areasâĂŤFirst experiences with IKONOS data, their classification and application in urban planning and environmental monitoring”. In: *Regensburger Geographische Schriften* 35, pages 196–205 (cited page 79).

Moser, Flavia, Recep Colak, Arash Rafiey, and Martin Ester (2009). “Mining cohesive patterns from graphs with feature vectors”. In: *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, pages 593–604 (cited pages 3, 13).

Moulick, Sanjib, BC Mal, and S Bandyopadhyay (2002). “Prediction of aeration performance of paddle wheel aerators”. In: *Aquacultural Engineering* 25.4, pages 217–237 (cited page 117).

Muzammal, Muhammad and Rajeev Raman (2010). “On probabilistic models for uncertain sequential pattern mining”. In: *International Conference on Advanced Data Mining and Applications*. Springer, pages 60–72 (cited page 12).

— (2011). “Mining sequential patterns from probabilistic databases”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 210–221 (cited page 12).

Negrevergne, Benjamin, Alexandre Termier, Marie-Christine Rousset, and Jean-François Méhaut (2014). “Para Miner: a generic pattern mining algorithm for multi-core architectures”. In: *Data Mining and Knowledge Discovery* 28.3, pages 593–633 (cited page 126).

Nijssen, Siegfried and Björn Bringmann (2008). “What is Frequent in a Single Graph?” In: *PAKDD* (cited page 38).

Nijssen, Siegfried and Joost N Kok (2004). “A quickstart in frequent structure mining can make a difference”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 647–652 (cited page 21).

Nouri, Hamideh, Simon Beecham, Sharolyn Anderson, and Pamela Nagler (2014). “High spatial resolution WorldView-2 imagery for mapping NDVI and its relationship to temporal urban landscape evapotranspiration factors". In: *Remote sensing* 6.1, pages 580–602 (cited page 87).

Nouri, Hamideh, Sharolyn Anderson, Paul Sutton, Simon Beecham, Pamela Nagler, Christopher J Jarchow, and Dar A Roberts (2017). "NDVI, scale invariance and the modifiable areal unit problem: An assessment of vegetation in the Adelaide Parklands". In: *Science of the total environment* 584, pages 11–18 (cited page 87).

Ozaki, Tomonobu and Minoru Etoh (2011). "Closed and maximal subgraph mining in internally and externally weighted graph databases". In: *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on.* IEEE, pages 626–631 (cited page 16).

Ozaki, Tomonobu and Takenao Ohkawa (2009). "Discovery of Correlated Sequential Subgraphs from a Sequence of Graphs." In: *ADMA*. Springer, pages 265–276 (cited pages 4, 13, 19).

Papadimitriou, Panagiotis, Ali Dasdan, and Hector Garcia-Molina (2010). "Web graph similarity for anomaly detection". In: *Journal of Internet Services and Applications* 1.1, pages 19–30 (cited page 127).

Pasquier, Claude, Jérémy Sanhes, Frédéric Flouvat, and Nazha Selmaoui-Folcher (2013). "Frequent pattern mining in attributed trees". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, pages 26–37 (cited page 13).

Pei, Jian, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu (2004). "Mining sequential patterns by pattern-growth: The prefixspan approach". In: *IEEE Transactions on knowledge and data engineering* 16.11, pages 1424–1440 (cited pages 12, 96).

Pei, Jian, Jiawei Han, and Wei Wang (2007). "Constraint-based sequential pattern mining: the pattern-growth methods". In: *Journal of Intelligent Information Systems* 28.2, pages 133–160 (cited pages 12, 121).

Pelillo, Marcello (1999). "Replicator equations, maximal cliques, and graph isomorphism". In: *Advances in Neural Information Processing Systems*, pages 550–556 (cited page 127).

Pinto, Helen, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal (2001). "Multi-dimensional sequential pattern mining". In: *Proceedings of the tenth international conference on Information and knowledge management.* ACM, pages 81–88 (cited page 12).

Pokou, Yao Jean Marc, Philippe Fournier-Viger, and Chadia Moghrabi (2016). "Authorship Attribution Using Small Sets of Frequent Part-of-Speech Skip-grams." In: *FLAIRS Conference*, pages 86–91 (cited page 9).

Prado, Adriana, Baptiste Jeudy, Élisa Fromont, and Fabien Diot (2013). "Mining spatiotemporal patterns in dynamic plane graphs". In: *Intelligent Data Analysis* 17.1, pages 71–92 (cited page 19).

Prakash, B Aditya, Jilles Vreeken, and Christos Faloutsos (2014). "Efficiently spotting the starting points of an epidemic in a large graph". In: *Knowledge and information systems* 38.1, pages 35–59 (cited page 13).

Proisy, Christophe, Gaëlle Viennois, Frida Sidik, Ariani Andayani, James Anthony Enright, Stéphane Guitet, Niken Gusmawati, Hugues Lemonnier, Gowrappan Muthusankar, Adewole Olagoke, *et al.* (2018). “Monitoring mangrove forests after aquaculture abandonment using time series of very high spatial resolution satellite images: A case study from the Perancak estuary, Bali, Indonesia”. In: *Marine pollution bulletin* 131, pages 61–71 (cited page 102).

Ren, Jia-Dong, Jing Yang, and Yan Li (2008). “Mining weighted closed sequential patterns in large databases”. In: *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on.* Volume 5. IEEE, pages 640–644 (cited page 12).

Revenga, Carmen (2005). “Developing indicators of ecosystem condition using geographic information systems and remote sensing”. In: *Regional Environmental Change* 5.4, pages 205–214 (cited page 79).

Salama, Nabeil KG and Alexander G Murray (2011). “Farm size as a factor in hydrodynamic transmission of pathogens in aquaculture fish production”. In: *Aquaculture Environment Interactions* 2.1, pages 61–74 (cited page 120).

Saneifar, Hassan, Sandra Bringay, Anne Laurent, and Maguelonne Teisseire (2008). “S 2 MP: similarity measure for sequential patterns”. In: *Proceedings of the 7th Australasian Data Mining Conference-Volume 87.* Australian Computer Society, Inc., pages 95–104 (cited page 127).

Sanhe, JÃľrÃľmy (2014). “Contribution Ãă la fouille de donnÃľes spatio-temporelles : application Ãă l'ÃľtAude de l'Ãľrosion”. PhD thesis. NoumÃľa, Nouvelle CalÃľdonie: UniversitÃľ de la Nouvelle CalÃľdonie, page 140 (cited page 9).

Sanhes, Jérémy, Frédéric Flouvat, Claude Pasquier, Nazha Selmaoui-Folcher, and Jean-François Boulicaut (2013). “Weighted Path as a Condensed Pattern in a Single Attributed DAG.” In: *IJCAI.* Volume 13, pages 1642–1648 (cited pages 9, 13).

Shackelford, Aaron K and Curt H Davis (2003). “A combined fuzzy pixel-based and object-based approach for classification of high-resolution multispectral data over urban areas”. In: *IEEE Transactions on GeoScience and Remote sensing* 41.10, pages 2354–2363 (cited page 80).

Songram, Panida and Veera Boonjing (2008). “Closed multidimensional sequential pattern mining”. In: *International Journal of Knowledge Management Studies* 2.4, pages 460–479 (cited page 12).

Srikant, Ramakrishnan and Rakesh Agrawal (1996). “Mining sequential patterns: Generalizations and performance improvements”. In: *International Conference on Extending Database Technology.* Springer, pages 1–17 (cited pages 9, 10, 96).

Tan, Pang-Ning, Vipin Kumar, and Jaideep Srivastava (2002). “Selecting the right interestingness measure for association patterns”. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, pages 32–41 (cited page 19).

Tan, Pang-Ning *et al.* (2006). *Introduction to data mining.* Pearson Education India (cited pages 28, 37).

Tanbeer, Syed Khairuzzaman, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee (2009). “Discovering periodic-frequent patterns in transactional databases”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, pages 242–253 (cited page 12).

Tucker, Compton J (1979). “Red and photographic infrared linear combinations for monitoring vegetation”. In: *Remote sensing of Environment* 8.2, pages 127–150 (cited page 87).

Virdis, Salvatore Gonario Pasquale (2014). “An object-based image analysis approach for aquaculture ponds precise mapping and monitoring: a case study of Tam Giang-Cau Hai Lagoon, Vietnam”. In: *Environmental monitoring and assessment* 186.1, pages 117–133 (cited page 80).

Wang, Jianyong, Jiawei Han, and Chun Li (2007). “Frequent closed sequence mining without candidate maintenance”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.8, pages 1042–1056 (cited pages 9, 12, 38).

Wright, Kenneth (2010). “Aviation system performance during the summer convective weather season”. In: *Journal of the Transportation Research Forum*. Volume 45. 3 (cited page 70).

Yan, Xifeng and Jiawei Han (2002). “gspan: Graph-based substructure pattern mining”. In: *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on.* IEEE, pages 721–724 (cited pages 16, 21, 125).

Yan, Xifeng, Jiawei Han, and Ramin Afshar (2003). “CloSpan: Mining: Closed sequential patterns in large datasets”. In: *Proceedings of the 2003 SIAM international conference on data mining.* SIAM, pages 166–177 (cited pages 12, 38).

Yi, Shengwei, Tianheng Zhao, Yuanyuan Zhang, Shilong Ma, and Zhanbin Che (2011). “An effective algorithm for mining sequential generators”. In: *Procedia Engineering* 15, pages 3653–3657 (cited page 12).

Yin, Junfu, Zhigang Zheng, and Longbing Cao (2012). “USpan: an efficient algorithm for mining high utility sequential patterns”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, pages 660–668 (cited page 12).

Yoshino, Kunihiko, Sayuri Kawaguchi, Fusayuki Kanda, Keiji Kushida, and Fuan Tsai (2014). “Very high resolution plant community mapping at High Moor, Kushiro Wetland”. In: *Photogrammetric Engineering & Remote Sensing* 80.9, pages 895–905 (cited page 80).

Yun, Un-Il (2007). “WIS: Weighted interesting sequential pattern mining with a similar level of support and/or weight”. In: *ETRI journal* 29.3, pages 336–352 (cited page 19).

Yun, Unil and John J Leggett (2006). “WSpan: Weighted Sequential pattern mining in large sequence databases”. In: *Intelligent Systems, 2006 3rd International IEEE Conference on.* IEEE, pages 512–517 (cited page 12).

Zager, Laura A and George C Verghese (2008). “Graph similarity scoring and matching”. In: *Applied mathematics letters* 21.1, pages 86–94 (cited page 127).

Zaharia, Matei, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, *et al.* (2016). “Apache spark: a unified engine for big data processing”. In: *Communications of the ACM* 59.11, pages 56–65 (cited page 127).

Zaki, Mohammed J (2001). “SPADE: An efficient algorithm for mining frequent sequences”. In: *Machine learning* 42.1-2, pages 31–60 (cited page 10).

Zhai, Ke, Xiaoqing Wu, Yuanwei Qin, and Peipei Du (2015). “Comparison of surface water extraction performances of different classic water indices using OLI and TM imageries in different situations". In: *Geo-spatial Information Science* 18.1, pages 32–42 (cited page 86).

Zhao, Zhou, Da Yan, and Wilfred Ng (2014). "Mining probabilistically frequent sequential patterns in large uncertain databases". In: *IEEE transactions on knowledge and data engineering* 26.5, pages 1171–1184 (cited page 12).

Zhu, Zhillang, Limin Yang, Stephen V Stehman, Raymond L Czaplewski, *et al.* (2000). "Accuracy assessment for the US Geological Survey regional land-cover mapping program: New York and New Jersey region". In: *Photogrammetric Engineering and Remote Sensing* 66.12, pages 1425–1438 (cited page 80).