Contribution to the interpretation of evolving communities in complex networks : Application to the study of social interactions

Keziban Orman

Numéro d'ordre 2014-ISAL-0072
Année 2014

Thèse présentée par
Keziban Orman

Devant
L'Institut National des Sciences Appliquées de Lyon

Pour obtenir
Le grade de docteur

Formation doctorale
Mathématiques et Informatique (InfoMaths)

Spécialité
Informatique

Contribution to the Interpretation of Evolving Communities in Complex Networks:
Application to the Study of Social Interactions

Soutenu publiquement le 16 juillet 2014 devant le jury composé de :

Pr. Jean-François BOULICAUT	INSA de Lyon	Co-directeur
Pr. Éric GAUSSIER	Université Joseph Fourier	Examinateur
Dr. Jean-Loup GUILLAUME	Université Pierre et Marie Curie	Examinateur
Pr. François JACQUENET	Université Jean Monnet	Rapporteur
Dr. Vincent LABATUT	Université Galatasaray	Co-directeur
Pr. Céline ROUVEIROL	Université Paris-Nord	Président
Dr. Maguelonne TEISSEIRE	IRSTEA Maison de la Télédetection	Examinatrice
Pr. Emmanuel VIENNET	Université Paris-Nord	Rapporteur

Bu tez, onurlarıyla var olmaya çalışan tüm kadınlara adanmıştır.

Acknowledgements

I would like to express my special appreciation and gratitude to my advisor Prof. Jean-François Boulicaut. I would like to thank you for encouraging my research. After all our meetings, I have felt that a new perspective has opened in my mind not only about my research but also about science and universe. Since we met, I have always felt lucky to have such an experienced and talented advisor. I would also like to express my deepest appreciation to my co-advisor Dr. Vincent Labatut. Since six years, you have helped me a lot on my research. When I was stuck at dead-ends, you gave me inspiration and hope to continue. Your character with endless patience and discipline has become a good role model for my further academic life. I am very lucky to have the chance to work with you.

I would like to express my sincere regards to Dr. Marc Plantevit. I have benefited a lot from our discussions. You gave me very valuable advices for this thesis. I would also like to extend my thanks to Prof. Hocine Cherifi. A part of this thesis was done under your supervision. My gratitude also goes to Prof. Atilla Başkurt. This thesis would not be started without your helps. I would like to express my gratefulness to Prof. François Jacquenet and Prof. Emmanuel Viennet for agreeing to review this thesis. I want to say a sincere thank to them for their time and thoughtful comments. And also, I would like to express my gratefulness to the members of jury for agreeing to participate to the presentation.

I would like to mention about my dear friends Pınar Uluer and Gözde Aytemur and my dear teacher Dr. Sultan Turhan. Our joyful discussions and exchanging ideas gave me the courage and belief to continue my research. Also, I want to cheerfully thank to Élise Desmier and Méhdi Kaytoue for their supports and saving time to improve my work. I am also grateful to my special friend Eleftherios Chatzidis. You always remind me to smile to the difficulties. I hope I could help you in your life as much as you do for me. I would like to thank you.

Last but not least my family. Words cannot express how grateful I am to them. My mother Ayşe Orman, my father Ramazan Orman and my sister Dr. Güneş Orman, you have always believed in me. You are always only and only supportive without questioning. I would like to thank you for all of the sacrifices that you have made on my behalf. I am very lucky to have a family so great and so unique.

Abstract

Complex Networks constitute a convenient tool to model real-world complex systems. For this reason, they have become very popular in the last decade. It is possible to encode various types of information in a complex network, depending on the studied system and modeling objectives and needs. In its most basic form, a complex network contains only nodes and links (plain network). But it is also possible to enrich it by setting link directions (directed network), associating attributes to nodes (attributed network) or links, or introducing a temporal dimension (dynamic networks).

Many tools exist to study complex networks. Among them, community detection is one of the most important. A community is roughly defined as a group of nodes more connected internally than to the rest of the network. In the literature, this intuitive definition has been formalized in many ways, leading to countless different methods and variants to detect communities. In the large majority of cases, the result of these methods is set of node groups in which each node group corresponds to a community. From the applicative point of view, the meaning of these groups is as important as their detection. However, although the task of detecting communities in itself took a lot of attraction, the problem of interpreting them has not been properly tackled until now.

In this thesis, we see the interpretation of communities as a problem independent from the community detection process, consisting in identifying the most characteristic features of communities. We break it down into two sub-problems: 1) finding an appropriate way to represent a community and 2) objectively selecting the most characteristic parts of this representation. To solve them, we take advantage of the information encoded in dynamic attributed networks. We propose a new representation of communities under the form of temporal sequences of topological measures and attribute values associated to individual nodes. We then look for emergent sequential patterns in this dataset, in order to identify the most characteristic community features.

Our solution to the interpretation problem takes the form of a five-stepped framework. First, we detect the communities and process the nodal topological measures. Second, we create a database of nodal sequences, thanks to these measures as well as the nodal attributes originating from the studied real-world system. Third, we identify the sequential patterns contained in this database. Fourth, we filter them using some objective criteria related to the notions of prevalence (support measure), emergence (growth rate) and closeness. Fifth, we finally select the most characteristic patterns covering each community. Our framework produces the characteristic sequential patterns and also allows identifying outlier nodes.

We perform a validation of our framework on artificially generated dynamic attributed networks. At this occasion, we study its behavior relatively to changes in the temporal evolution of the communities, and to the distribution and evolution of nodal features. We also apply our framework to real-world systems: a DBLP network of scientific collaborations, and a LastFM network of social and musical interactions. Our results show that the detected communities are not completely homogeneous, in the sense several node topic or interests can be identified for a given community. Some communities are composed of smaller groups of nodes which tend to evolve together as time goes by, be it in terms of individual (attributes, topological measures) or relational (community migration) features. The detected anomalies generally fit some generic profiles: nodes misplaced by the community detection tool, nodes

relatively similar to their communities, but also significantly different on certain features and/or not synchronized with their community evolution, and finally nodes with completely different interests.

Résumé

Les réseaux complexes constituent un outil pratique pour modéliser les systèmes complexes réels. Pour cette raison, ils sont devenus très populaires au cours de la dernière décennie. Il est possible d'encoder différents types d'informations dans un réseau complexe, en fonction du système étudié, ainsi que de l'objectif et des besoins de la modélisation. Dans sa forme la plus basique, un réseau complexe contient uniquement des nœuds et de liens (*réseau basique*). Mais il est également possible de l'enrichir en mettant des directions aux liens (*réseau orienté*), en associant des attributs aux nœuds du réseau (*réseau attribué*) ou des liens, ou en introduisant une dimension temporelle (*réseaux dynamiques*).

De nombreux outils existent pour étudier les réseaux complexes. Parmi ceux-ci, la détection de la communauté est l'un des plus importants. Une communauté est grossièrement définie comme un groupe de nœuds plus densément connectés entre eux qu'avec le reste du réseau. Dans la littérature, cette définition intuitive a été formalisée de plusieurs différentes façons, ce qui a conduit à d'innombrables méthodes et variantes permettant de les détecter. Dans la grande majorité des cas, le résultat de ces algorithmes prend la forme d'un ensemble de groupes de nœuds, comme par exemple une partition de l'ensemble des nœuds du réseau, dans lequel chaque groupe de nœuds correspond à une communauté. Du point de vue applicatif, le sens de ces groupes est aussi important que leur détection. Cependant, bien que la tâche de détection de communautés en elle-même ait attiré énormément d'attention, le problème de leur interprétation n'a pas été sérieusement abordé jusqu'à présent.

Dans cette thèse, nous voyons l'interprétation des communautés comme un problème indépendant du processus de leur détection, consistant à identifier les éléments leurs caractéristiques les plus typiques. Nous le décomposons en deux sous-problèmes : 1) trouver un moyen approprié pour représenter une communauté ; et 2) sélectionner de façon objective les parties les plus caractéristiques de cette représentation. Pour résoudre ces deux sous-problèmes, nous exploitons l'information encodée dans les réseaux dynamiques attribués. Nous proposons une nouvelle représentation des communautés sous la forme de séquences temporelles de descripteurs associés à chaque nœud individuellement. Ces descripteurs peuvent être des mesures topologiques et des attributs nodaux. Nous détectons ensuite les motifs séquentiels émergents dans cet ensemble de données, afin d'identifier les ceux qui sont les plus caractéristiques de la communauté.

Notre solution au problème d'interprétation prend la forme d'un procédé en cinq étapes. Tout d'abord, nous détectons les communautés et calculons les descripteurs . Deuxièmement, nous créons une base de données de séquences nodales, grâce à ces descripteurs. Troisièmement, nous identifions les motifs séquentiels fréquents contenus dans cette base de données. Quatrièmement, nous les filtrons en utilisant certains critères objectifs liés aux notions de prévalence (mesure de soutien), émergence (taux de croissance) et de la spécificité (fermeture). Cinquièmement et dernièrement, nous sélectionnons les motifs les plus caractéristiques couvrant chaque communauté. Notre procédé permet à la fois d'identifier les motifs séquentiels caractéristiques de la communauté et les nœuds déviants qui ne les respectent pas.

Nous effectuons une validation de notre procédé sur des réseaux attribués dynamiques générés artificiellement. A cette occasion, nous étudions son comportement relativement à des changements structurels de la structure de communautés, à des modifications des valeurs des attributs. Nous appliquons également notre procédé à deux systèmes du monde réel : un réseau de collaborations scientifiques issu

de DBLP, et un réseau d'interactions sociales et musicales tiré du service LastFM. Nos résultats montrent que les communautés détectées ne sont pas complètement homogènes, dans le sens où plusieurs sujets ou intérêts peuvent être identifiés pour une communauté donnée. Certaines communautés sont composées de petits groupes de nœuds qui ont tendance à évoluer ensemble au cours du temps, que ce soit en termes de propriétés individuelles (attributs, mesures topologiques) ou collectives (migration de la communauté). Les anomalies détectées correspondent généralement à des profils typiques : nœuds mal placés par l'outil de détection de communautés, ou nœuds différant des tendances de leur communautés sur certains points, et/ou non-synchrones avec l'évolution de leur communauté, ou encore nœuds complètement différents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

A *complex system* is a system possessing some emergent properties, due to the interactions of its constituting objects. They can be encountered in many different domains such as biology, physics, computer science, sociology, etc. *Network modeling* consists in representing such systems through graphs, using *nodes* to represent the objects and *links* for their interactions [94]. The use of complex networks eases the understanding of the emergent properties and thus helps analyzing the dynamics and functioning of the system of interest. For this reason, this modeling paradigm is now very popular in a wide range of areas [10, 94]. In its most basic form, such a complex network contains only nodes and links, and can be qualified of *plain*. However, one can introduce a richer information in this model, depending on the considered system and modeling needs and constraints, which makes it a very flexible tool. The network can be *directed* if the relations between objects are asymmetric, some *attributes* can be added to nodes or links in order to get a more complete model, time can be introduced in various ways, leading to *dynamic networks*, etc.

The research works conducted on complex networks modeling systems from different domains revealed the existence of common topological properties. One of the most famous is the *small-world* property, which states the average distance between two nodes increases logarithmically with the network number of nodes, due to the presence of shortcuts connecting different areas of the network [94]. Many networks are also *scale-free*, which means the degree distribution of their nodes follows a power law [9], which causes the presence of many nodes of very law degree and only a few nodes of very large degree. One of the most studied and important properties is the presence of a *community structure*, which reflects the existence of a modular structure in the interconnection of nodes [94]. A community is roughly defined as a group of nodes densely interconnected, compared to the rest of the network [32, 43]. In [137], the importance of the detection of community structures highlighted to discover functionally related objects, study interactions between modules, infer missing attribute values and predict unobserved connections in the network. Because finding the community structure facilitates the analysis of the network and also helps to summarize it, it has been studied widely by network science scholars. Hundreds of different algorithms were developed for the task of identifying the community structure of a network, an operation generally called *community detection* [43].

1.2 Problem

Community detection is an important task in the domain of complex network analysis. However, whatever the considered algorithm, its output can always be basically described as a list of node groups. From an applicative point of view, the question is then to make sense of these groups relatively to the studied

system. In other terms, for the community structure to be useful, it is necessary to *interpret* the detected communities. Yet, although the community interpretation problem is extremely important from the end user's perspective, only a very few works tried to tackle the problem of characterizing and interpreting the communities, up to now.

In the early stage, some authors interpreted the detected communities manually [111, 115, 116], which was possible only because the considered data were very small. Some authors then proposed to interpret the data based on some topological information only. For example, in [77], the authors conduct a visual analysis of the community size distributions, of how certain community-related measures evolve in function of the community size, and of the distribution of certain nodal measures. However, the topological information alone is not satisfying in terms of how the network can be comprehensively described. Some authors then proposed to take advantage of the attributes associated to nodes in certain networks, in order to improve this point. They used statistical tools such as logistic regression and linear discriminant analysis to characterize the communities in terms of their most distinctive attributes [73, 133]. However, those methods ignore all or most of the available topological information. Topological and attribute information are used together in attribute-based community detection methods [22, 137, 140], but the way they are combined is generally not explicitly described. To summarize, those works suffer from the fact they use a limited information compared to what the network can provide, or do not precisely identify which information they use. This leads to potentially inappropriate or incomplete results and interpretation.

In this thesis, we focus on the *community interpretation problem*. We view this problem as *independent* from the approach used for community detection. Separating them allow more flexibility in the way we can solve them. Consequently, the interpretation does not depend on the limits of the detection algorithm. For instance, we can use more information than that used during community detection. We see this problem as identifying *the most characteristic features* of the communities. To perform such task, we distinguish two sub-problems: first, *finding an appropriate way to represent a community*; and second, *defining an objective method to decide which parts of this representation are characteristic*.

We aim to use all possible information proposed by network models, in order to be able to retrieve the most characteristic features of the communities. For this reason, we want to take advantage not only of the structure of the network and values of its attributes, but also of its temporal dimension. We propose a new representation of communities for such attributed dynamic networks. It relies on temporal sequences of topological measures and attributes values associated to individual nodes. We then look for emergent sequential patterns in this dataset, in order to identify the most characteristic community features. To solve the problems we identified, we propose a five-stepped framework. At the first step, we detect the communities. It is indeed needed to have a community structure to interpret them. Nevertheless, one can use any community detection method at this first step. At the second step, we create a sequence database, in which each node sequence carries the topological, attributed- and community-related information describing the considered node. At the third step, we identify the sequential patterns present in the database, thanks to sequential pattern mining. At the fourth step, we filter these patterns thanks to some objective criteria related to the notions of prevalence (support measure), emergence (growth rate) and closeness. At the last step, we choose the most characteristic patterns covering each community among the ones selected at the previous step. As a result, our framework extracts the *characteristic patterns* of each community and the *outliers* nodes which are different from typical nodes of the same community.

1.3 Contribution

This thesis contributes to the research related to community structure in complex networks in several different ways. We order the contributions by their importance. The most prominent three contributions are:

- **Interpretation Problem.** We consider *community interpretation* as an independent problem and we state it formally. Two sub-problems arise from it: how to represent communities and how to extract the characteristic parts of this representation. To our knowledge, this definition is the first attempt in the domain.

- **Community Representation.** We represent communities as a set of sequences. It is a compact representation including all possible information related to a community (attributes, topology, evolution).

- **Community Characterization Framework.** We propose a five-stepped framework to solve the community interpretation problem. It relies on the extraction of sequential patterns from the representation of each community, while respecting certain criteria related to their frequency, emergence and closeness. Our framework also allows identifying outlier nodes. Among the output characteristic patterns, there are not only patterns of attributes and topological measures, but also the patterns including only community information. Thus, it provides a way to track the evolution of communities over time, in terms of community events.

Beside these prominent contributions, there are two secondary contributions of this thesis:

- **Validation of the Framework.** To study the behavior of our framework, we test it on some artificially generated networks. For this matter, we propose an extension of an existing generative model, able to produce attributed dynamic networks. We use the artificial data to see how our framework behaves depending on the nature and evolution of the attributes.

- **Application to Real-World Network.** We apply our framework to two real-world networks. The first one is a DBLP network reflecting the co-authorship relations of scientists. It is commonly used in network researches. The second network was specifically extracted from the LastFM web site for this thesis, by considering Jazz music listeners. It shows the connections between users listening to the same artist on the same time interval. We both detect the community evolution and interpret the communities for these two networks.

On the side, this thesis contributes also for :

- **Comparison of Detection Algorithms.** Our framework is independent from the community detection methods. Nevertheless, we still need a community structure to interpret. Thus, we present an experimental evaluation of four community detection methods developed for dynamic networks. We then select the most appropriate one to use on our further experiments.

1.4 Organization

This thesis is organized in 7 chapters. The *second one* is dedicated to explaining the fundamental concepts about complex networks and community structure. We also describe in detail the topological measures we use in our framework. In the *third chapter*, we propose a state-of-the-art of community detection methods. It is directed according to the theme of the thesis. We first describe different detection methods for plain network, before tackling attributed and then dynamic networks. We also indicate some of the existing methods for artificial network generation and performance assessments in the perspective of algorithm evaluation. We then concentrate on some existing works related to the community interpretation problem, by emphasizing their strong and weak points.

In the *fourth chapter*, we focus on the community interpretation problem itself. We first give some preliminary definitions in order to introduce formal concepts required to describe the framework. Then

we define formally the community interpretation problem and both related sub-problems. After this, we describe our framework, giving the details for each step. We illustrate how to use by applying it to a small example. Finally, we study the algorithmic complexity of our framework and give some precisions concerning its implementation.

In the *fifth chapter*, we study the behavior of our framework. We first describe the tools available for artificial network generation, and more specifically the extension we propose to take node attributes into account. We compare the performance of four community detection algorithm on some networks specifically generated to follow particular evolutions. We discuss how these considered type of evolution affects the algorithm performance. We use this occasion to give some examples of how to track major community evolution events thanks to the detection of sequential patterns. Finally, we study the behavior of our framework for some networks generated by using attributes of different natures. In the *sixth chapter*, we present the results obtained for the DBLP and LastFM networks. For each network, we first present the data we use in more details, and the major community evolution events we detected. Then, we describe the most supported patterns and their meanings for some prominent communities. Finally, we comment the characteristic patterns and interpret them relatively to the concerned communities.

Finally, in the *seventh chapter*, we summarize and criticize our work, propose some leads to solve the existing limitations, and identify our major perspectives.

Chapter 2

Complex Networks

Complex networks constitute a powerful modeling tool, able to represent most real-world systems. The objects composing the system are represented under the form of nodes while their relationships correspond to links. Thanks to this versatility, they have become very popular during the last decades, attracting the attention of scientists from many different domains. They are now used to model and analyze complex systems in areas like physics, biology, social sciences or computer science [10, 94]. In this chapter, we explain the notions of complex system and complex network. We describe some prominent topological properties of complex networks and define some of their topological measures, with a focus on the nodal ones.

2.1 A Modeling Tool

2.1.1 Notion of Complex System

The term of *system* is used for describing a set of interacting objects relatively isolated from their environment. *Complex system* refers to a system having at least one emergent property. A property is said to be emergent when it arises from the interactions between the system objects. Such a property appears only at the level of the system, and not when considering isolated objects. For example; let us consider society as a system constituted of communicating (the interactions) persons (the objects). Then, one can consider language as an emergent property, because it arises from the need for one person to communicate with others, in order to express his or her mind/feelings in a particular way. One can suppose language would not exist in the absence of interactions. Moreover, if enough speakers start using the same expressions, the language changes accordingly. Some examples of other complex systems, from different domains, include the brain including millions of neurons (objects) and axons (interactions), the immune system including different organs (object) and their coordination (interactions), biological cells which have many different functional organelles (objects) involved in various metabolic processes (interactions), ant colonies constituted of many ants (objects) communicating with each other (interactions), and the World Wide Web with its many web pages (objects) and hyperlinks (interactions).

Emergence is central in the notion of complex systems, because it can itself cause other, secondary, properties. Those are not necessary present simultaneously in a given system, but several of them are generally observed. One can mention *heterogeneity* (relationships between objects are not distributed homogeneously), *locality* (the system is not centralized), *feedback* (an object can affect itself indirectly at a later time, through a loop of other objects), *segmentation* (substructures can exist in the system), *fractal* (substructures can be complex themselves and they can include other complex substructure and so on), *hierarchy* (the system might be structured on different levels), *non-linear* (the behavior of the system is different than the sum of the individual behaviors of its objects). Certain authors put these secondary

properties at the core of the notion of complex system. For instance, in [16, 89, 94], a complex system must not only have emergent properties, but also have *no centralized control* and display a *non-linear* behavior.

It is also good to underline that *complex system* is not equal to *complicated system*. A system is generally considered to be complicated when it is huge, or because its behavior is chaotic or random. Here, complicated means *difficult to understand*: size can be an obstacle to a good understanding, as well as the fact a small change on an object has a huge effect at the whole system, or the fact its behavior cannot be predicted with certainty. By comparison, a complex system *can* also be complicated, but this is not a necessary condition to being complex. To summarize, both notions are independent: a system can be only complex, or only complicated, or both.

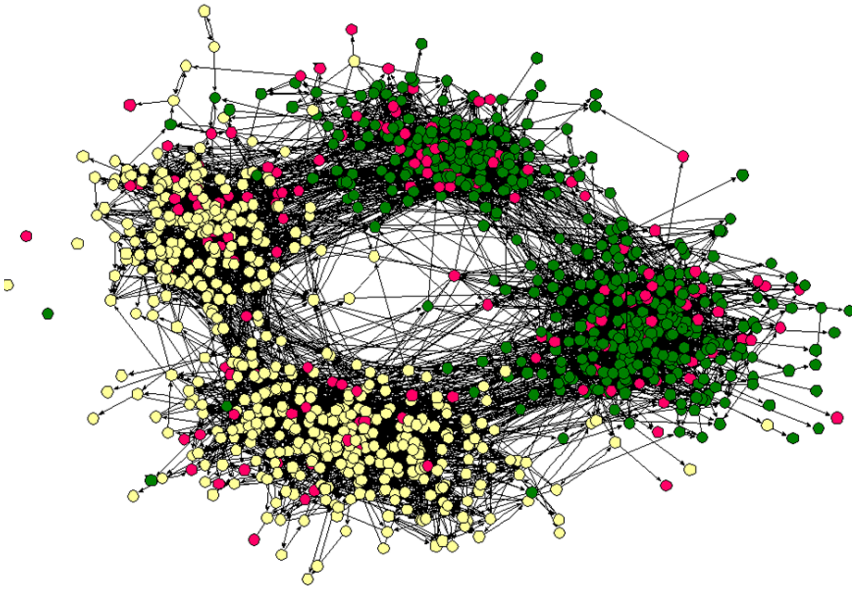

Fig. 2.1 Network Representation of high school friendship used in [91] (image courtesy of [75])

It is not possible to understand the dynamics of a complex system by reducing it to its elementary objects: because of the presence of some emergent properties, it is necessary to consider the relationships between them. For this purpose, graphs constitute a very efficient modeling tool, because they allow representing both object, using nodes, and relationships, using links. Such a representation of a complex system is called a *complex network* [10, 16, 94]. The existence of emergent properties in the modeled systems, as well as certain secondary properties mentioned earlier, cause the presence of non-trivial topological properties in the corresponding networks. Here, the term non-trivial means it is difficult to characterize the structure of the network using the traditional models of graph theory. On the one hand, a complex network is not a *regular* graph, since its links are heterogeneously distributed. It is therefore hardly comparable to a lattice [12], for instance. On the other hand, this heterogeneity is not completely random, since a complex network is characterized by certain substructures. So, it does not fit random graph models such as that of Erdos-Renyi [40]. The model proposed by Watts and Strogatz [38] is very illustrative of this situation. In the goal of replicating some property observed in real-world complex networks, they proposed a model whose randomness is controlled by a parameter. It can generate a continuum of graphs ranging from purely regular to completely random. It turns out mimicking the property of complex networks requires generating graph both partially regular and random.

Because complex networks are used in different areas, many different vocabularies exist in the literature to refer to the same concepts. For example, the objects in interaction are called *vertex*, *node* or *actor* and the relationships between them are called *edge*, *arc*, *link*, *tie*, *bond* or *connection*. In this work,

we use the terms of *node* for the objects and *link* for the relationship between them. One can visualize a graphical representation of a complex network modeling of the friendship of US high schools in Figure 2.1. In the network representation, besides the friendships, the ethnicity of the students is also shown by the different node colors. We formally define a network $G = (V, E)$ as a couple of the set of nodes (V) and the set of links (E). Let the total link number be l and total node number n for the rest of the document. In the next section, we describe briefly different types of networks.

2.1.2 Types of Complex Networks

The graph representation is so flexible that it can include different types of information, depending on the modeling needs. The simplest networks represent the system objects with nodes and their relationships with simple links, they are called *plain networks*. Plain networks are as simple as possible that they include only nodes and links but no additional information. The existence of a link between nodes represents only the existence of a relationship. For example; we can model friendship between the students of a university with plain networks where the nodes are the students and links are their friendship. Richer networks can be obtained by adding other forms of information.

First, it is possible to add loops and multiple links. A loop corresponds to a self-relationship, i.e. a relationship between an object and itself. Links are called multiple when there are several links between the same two nodes. The resulting networks are called *multiple networks*. As an example, the activity in a company could be modeled with such a network. Suppose the nodes are the employees and the links are the tasks that should be performed together. Two employees can work on more than one task together (multiple links) and one employee can work on a task individually (loop).

It is possible to add link direction to a network, in order to represent the asymmetry of a relationship, leading to so-called *directed networks* [94]. For example; emailing between the employees of a company can be modeled with such a network, in which the nodes are the people who send or receive emails, and the links symbolize the emailing. Links would be directed from the person who sends to the person who receives the mails.

In some systems, the relationships between objects have different intensities, i.e. some objects are connected more strongly than others. To model this feature, it is possible to use *link weights* [94], i.e. numeric values assigned to each link to model the power of the connection. These types of networks are called weighted networks. For instance; consider a collaboration network, in which nodes are authors and links are co-authorship: one can weigh the links to represent the number of common publications of the co-authors.

Besides, the various properties of the objects or connections can be also added to network under the form of node or link *attributes* [94]. For example; age, gender or as shown in Figure 2.1, ethnicity can be added as attributes of the nodes from any human interaction network. We can see the visualization of the different information that a network may hold in Figure 2.2. Here, from left to right; the consecutive networks correspond to plain, directed, multiple, weighted connections respectively. In the third network from the left, one node also has a self-connection. The last network from the left contains nodal attributes representing the gender of the person.

If the nodes all represent the same type of objects, i.e. the system is constituted of homogeneous objects, the network is called *unipartite*. This was the case for all the types of networks presented before this point. In some complex systems however, there are different types of objects, with possibly different types of interactions between them. To represent this kind of systems, we use *multipartite* networks. A network is said to be multipartite when it contains several different types of nodes, with links present only between nodes of different types. In other words, there can be no links between two nodes of the same type. One of the most common subcategory of multipartite networks is *bipartite networks* [56], i.e. networks with two types of nodes. For example, the Wikipedia edit system can be modeled as a

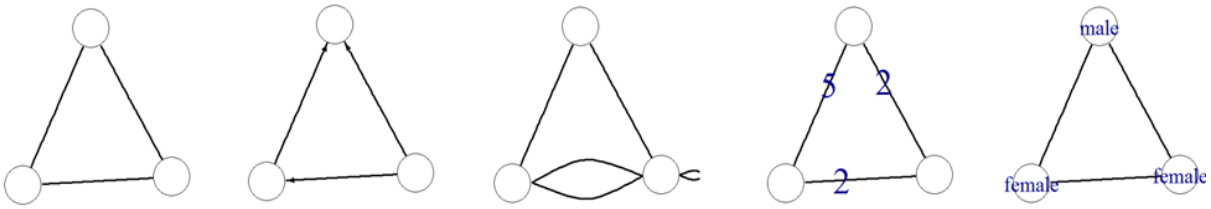

Fig. 2.2 Visual representation of different informations related to network representation. From left to right, the networks are plain, directed, multiple, weighted and attributed, respectively

bipartite network, in which some nodes represent users whereas the other are Wikipedia pages, and a link between a user and a page represents the fact the user edited this page [72]. In this representation, there is no link between two users or between two pages. Similarly, social tagging systems are often modeled with *tripartite* networks (i.e. multipartite networks with 3 types of nodes). For example, in the *del.icio.us* network, nodes can be either users, tags or URLs, and there can be links only between users and tags (a user utilizes a tag) and between tags and URLs (a URL is associated to a tag).

The heterogeneity described for nodes can alternatively be observed for links, i.e. the system contains different types of relationships, which is possible even if the nodes themselves are all of the same type. In order to represent such a system, one will use *multiplex*. For instance, let us consider a social network aiming at studying the effect of friendship on business. We can use a multiplex network with two types of links: one to represent friendship relationships, and the other to stand for trade ties.

A complex system is, by nature evolving, in the sense its objects and/or organization change over time. However, authors originally modeled them in a static way, for simplicity purposes. The resulting networks represent connections between objects and their properties, but they ignore the time-dependent changes over objects and connections. These types of networks which show a summary, or a single snapshot of the system, are called *static networks*. Some examples of static networks can be found in the resources of Newman [96]. Later, the researcher started modeling the evolving attributes of the modeled systems [19, 79, 104]. In [19], the authors point out different methods to study the evolution of a network. One of these methods is modeling the complex systems as a sequence of static networks such that consecutive static networks correspond to the representation of the consecutive snapshots of the same system. These types of networks are called *time evolving* or *dynamic networks* (in this document, we use the latter term).

As mentioned before, the various enhancements presented in this section can be combined depending on the modeling needs. In our study, we focus on a specific combination: *dynamic attributed networks*, i.e. sequence of snapshots, each one being a unipartite network without link weight or direction, but with nodal attributes. We describe them formally in Chapter 4.

2.2 Topological Properties

The modeling various systems like WWW, the Internet, telephone calls, Web-based social friendship sites, metabolic systems, protein-protein interactions, food webs, etc. from different disciplines like physics, biology, information technologies, gave birth to the science of complex networks. The study of these different systems revealed there exist some very widespread topological properties. In this subsection, we present the most prominent ones.

2.2.1 Small-Worldness, Scale-Freeness and Transitivity

The *small world* phenomenon is typical of many networks representing real-world systems, in which shortcuts connecting different areas of the networks allow reducing the average distance between any two nodes of the network. In the context of social networks, it has been popularized by the famous *six degrees of separation* experiment [132]. It refers to the idea that everyone is at most six steps away from any other person on earth, so that a chain of "a friend of a friend" relationships can be made to connect any two people in six steps or less. This property can be interpreted as propagation efficiency: spreading on the network remains relatively fast even if the network grows. Specifically, small world networks have a low average distance (i.e. the length of the shortest path between pair of nodes). Furthermore, in growing networks, it increases logarithmically with the number of nodes [94].

Most networks have a highly heterogeneous degree distribution, where the degree is the number of connections of a node. In such networks, a few nodes are linked to many other ones, and there is a large number of poorly connected nodes. Nodes with high degree are informally called hubs, because they are supposed to have a more central role in the network. Note the notion of hub can be defined in various other ways, such as the community-based one presented in section 2.3.7. This heterogeneous distribution is well described by a power-law. In other words, the probability for a node to have a degree k is $p_k \sim k^{-\gamma}$. Such networks are called *scale-free*, because the exponent of the distribution does not depend on their size. Experimental studies showed that the γ coefficient usually ranges from 2 to 3 [10, 16, 94]. In [9], an explanation for this property is proposed under the form of the *preferential attachment* principle: in a growing network, the new nodes are more likely to get connected to popular existing nodes, i.e. those already well-connected. This principle is also known as *Rich get Richer* [11]. So, the fact a system is scale-free reveals some relevant information regarding both its structure and its dynamics. In a network, the average and maximal degrees generally depend on the number of nodes it contains. For a scale-free network, it is estimated to be $\langle k \rangle \sim k_{max}^{-\gamma+2}$ [10, 16] and $k_{max} \sim n^{1/(\gamma-1)}$ [94], respectively.

Apart from these properties, some networks are also known as having a hierarchical structure: we observe less links between the nodes of different hierarchical levels, and we see denser links between the nodes of the same level [28]. Another common topological property of networks is having a high transitivity, compared to purely random networks with the same number of nodes and links. A high transitivity means the way the links are distributed over the network results in a large number of triangles. We explain in detail what the transitivity measures in section 2.3.3, including its formal definition. The fact real-world networks exhibit a high transitivity means nodes tend to form small tightly connected subgroups. For instance, in a social network, it is related to how well the friends of a person know each other. If none of them relates together, the transitivity is minimal, whereas if they are all friends, it is maximal. Social networks are known to be highly transitive [38].

The properties we mentioned here give an idea of the topology of complex systems. They come out of observations and statistical studies performed on static networks. Indeed, as mentioned before, early works tried to explain the evolving mechanism of complex network [9, 10, 108, 128] based only on static representations. The model proposed to explain the scale-free property highlighted the need for a dynamic approach to perform better analyses. Moreover, as highlighted in [42], knowing a network is scale-free is not enough for explaining its structure and dynamics: it might describe just a single aspect. Using richer representations of the complex systems could be helpful to improve their analysis, and therefore broaden the knowledge we have of them. Modeling the evolving systems with dynamic networks by also considering the individual attributes of the objects can constitute a more appropriate platform for further analyses. Of course considering time evolution and attributes leads to a specific type of network, for which it is necessary to develop appropriate analysis tools.

2.2.2 Community Structure

The notion of *community* originally is coming from social sciences. Traditionally, the two major uses of the term refer to groups of people sharing a common territory (neighborhood, town, city, etc.) or having common relationships (human relationship, family, etc.) [60]. However, more recently, the word community has been used to point out at groups of persons sharing emotions, having the feeling of belonging together [86]. In network science, a community roughly corresponds to a group of nodes more densely interconnected, relatively to the rest of the network [97]. The expression *community structure* refers to a network topological property, in which the heterogeneous distribution of links results in a segmentation of the network into a set of communities. Having a community structure is a common feature in biological and social networks [94]. In [137], the importance of the detection of community structure highlighted to discover functionally related objects, study interactions between modules, infer missing attribute values and predict unobserved connections in the network. When a community structure is present, the community size distribution seems to follow a power-law distribution $p_k \sim k^{-\beta}$ with β ranging from 1 to 2 [94]. Dense networks (where the number of links is significantly higher than the number of nodes) cannot have communities: only sparse networks (where the number of links is at the order of the number of nodes) can [43]. Indeed, the network must be sparse enough so that links are less numerous between communities.

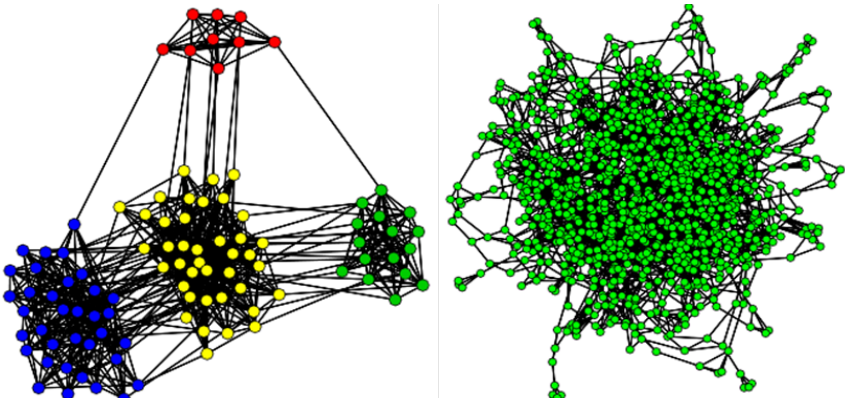

Fig. 2.3 Networks with (left) and without (right) a community structure

Community structure is one of the most studied network properties. There are several different terms referring to communities, like *modules*, *partitions*, *clusters* or *cohesive subgroups*. Naturally, several alternate definitions also exist. However, community detection-related works are often focused on designing new methods to detect communities, and less on what a community exactly is. For this reason, the notion of community is very frequently implicitly defined as the result of the considered community detection method. However, a few authors tried to give a proper definition of the concept.

Radicchi et al. [111] distinguish *strong* and *weak* communities. A strong community is a node set such that the internal degree of each node (i.e. number of neighbors inside its community) is greater than its external degree (number of neighbors outside the community), and a weak community is a structure such that its total internal degree exceeds its total external degree.

For Fortunato [43], there are three different families of definitions: *local*, *global* and *similarity-based*. The *local* definitions are based on the exploitation of an information concerning a node and its close neighborhood (the nodes to which it is connected relatively directly), and neglecting the rest of the network. One of the most common approach is clique-based. A clique is a group of nodes all connected together. A community can be defined as a maximal clique, or as a group of tightly connected cliques [34]. In a *global* definition, the network as a whole is considered. In this case, a community refers to a group of node with remarkable properties relatively to the rest of the network. This type of definition is

often expressed under the form of a measure, which is then optimized to detect the communities [43]. The most popular one is the modularity [93], which is described in chapter 3. Finally, a community can also be defined in terms of node *similarity*. Nodes belonging to the same community must be more structurally similar to each other than to other nodes. The basic way of estimating the structural similarity of two nodes consists in measuring how much their neighborhoods (direct neighbors) overlap. Once the notion of node similarity has been defined, a classic clustering method can be applied to identify the community structure [43]. It is worth noticing these three families of definitions are not mutually exclusive: for instance, the similarity used to define what a community can be based on global and/or local information. This highlights the difficulty of defining a relevant typology of community definitions.

Besides the information used to define what a community is, one can also consider the form a community take. Historically, the first type of community structure studied in the literature were disjoint, i.e. each node belongs to exactly one community [50]. The community structure is therefore a set *partition*, each community being a part. Later works allowed one node to belong to more than one community at same time, leading to *overlapping* communities [105]. In this case, the community structure can be seen as set *cover*. It is also possible to define the communities as *hierarchical*, such that every high level community can be split into several lower level ones. The nodes in the sub communities have denser inner links. This hierarchical structure can be represented under the form of a *dendrogram*, as in classic hierarchical clustering.

The notions of communities presented above focus on plain network. But, as mentioned before, network representation can be enhanced with more information. In this case, it is necessary to adapt the definition of what a community is. Here, we focus on the two specific enhancements we use in the rest of our work; dynamic networks and nodal attributes. For attributed networks, authors generally rely on the implicit assumption of *homophily*: they suppose that similar nodes tend to connect more frequently [87]. From this point of view, a community corresponds to a group of nodes which are not only densely interconnected, but also similar in terms of attributes. To the best of our knowledge, there is no systematic study showing a relation between attributes and topological features. So, the assumption that attributes and structure are dependent can be criticized, in the sense its veracity is certainly dependent on the modeled system.

In dynamic networks, the definition of a community structure is similar to what is used for static networks, and the focus is rather on the evolution of the communities. In [2, 52, 104], several community events are identified and studied: *birth*, *death*, *split*, *merge* and *continue*. Moreover, continuing communities can *expand* or *shrink*. In [104], the authors analyzed the community evolution in co-authorship and phone-call networks. They state that small communities, if they are stable (i.e. not expanding or shrinking very much), can survive for a long time. But instable ones have a very short lifespan. The opposite is observed for large communities: stable ones do not last a long time, whereas expanding ones do.

The task of defining the concept of community is highly related to that of detecting communities in networks. As mentioned before, many works in the literature do not define explicitly what a community is, but rather propose a way of detecting a structure, which is considered, *a posteriori*, as a community. In chapter 3, we describe community detection methods for plain networks, and then focus on our networks of interest for this work: dynamic attributed networks.

2.3 Topological Measures

In the literature, real-world networks have been characterized by their non-trivial topological properties. We described some of them in the previous section. Authors used *topological measures* to quantify these properties. In this subsection, we describe these measures in further details. We first propose a new typology to categorize them, and we then focus on those used later in this thesis. The measures are first

introduced theoretically, and the very last subsection is an example illustrating them.

2.3.1 Typology of Topological Measures

A topological measure allows to quantify a specific feature of network structure. The terminology and typology used in the literature to describe and classify them are not uniform, which is why we present our own here, in an effort of standardization. We distinguish two aspects of a measure, the type of entity and the information used to calculate the measure. First, a topological measure can be calculated for entities of different *scales*: node, substructure or whole network. Due to the topic of this work, we chose the community as the entity corresponding to the substructure level. Second, the information used for its calculation can characterize the entity with different *scopes*: local (a node and its direct neighborhood), intermediate (a substructure and the local information it contains), and the global (the whole network and the intermediate and local information it contains).

Let us consider a few examples to illustrate our typology. We consider *degree* as a measure defined at node scale with a local scope, because it describes a single node (scale) and relies on the number of its connections, which is local information (scope). We consider *hub dominance* to be a community scale and local scope measure. For a given community (scale), hub dominance is the ratio of the maximal internal degree (amongst the nodes it contains) to the size of the community (number of nodes it contains). Like the degree, the internal degree is based on local information (scope). For the sake of comprehensibility, we summarize our categorization for the topological measures in the table 2.1, with brief explanations and examples for each category. For the detailed explanation of each measure listed in the table, consult [48, 77, 94]. In this thesis, we focus on nodal measures, because we interpret the communities by their nodes topological and attribute evolution. One can find the detailed description of these measures in the next subsections.

2.3.2 Degrees

The *degree*, a.k.a *order*, of a node is the number of links attached to it. More formally, we note $N(v) = \{w \in V : (v,w) \in E\}$ the *neighborhood* of node v, i.e. the set of nodes connected to v in G. The degree $d(v) = |N(v)|$ of a node is the cardinality of its neighborhood, i.e. its number of neighbors. Similarly we can define the *internal neighborhood* of a node v as the subset of its neighborhood located in its community: $N^{int}(v) = N(v) \bigcap C(v)$. Then, the *internal degree* $d^{int}(v) = |N^{int}(v)|$ is defined as the cardinality of the internal neighborhood, i.e. the number of neighbors the node v has in its community. The calculation of total degree requires a local information, thus, its scope is local. However, the internal degree scope is intermediate, because we need to know the community structure.

Degree is a simple but important measure determining the popularity of a node. The higher its value, the more a node is important. As we explained in previous section, in most of the real-world networks, node degrees are distributed according to power-laws. Processing the degrees of all nodes in a network can be done by performing a single pass over its entire links. Suppose the degrees are represented by an integer array of size n, initially containing only zeros. For each link, we increment the values of the two cells corresponding to both nodes it connects. This process leads to a complexity in $O(l)$, where l is the number of links in the network.

2.3.3 Local Transitivity

The *local transitivity*, a.k.a. *clustering coefficient*, of a given node depends on how its neighbors are interconnected [38]. It is defined as the number of links present between these neighbors, divided by the number of links one would get if they were all interconnected. In other words, it represents the proportion of existing to possible links in the node neighborhood. We can formalize it as equation 2.1 , i.e. the ratio

Scale	Scope	Example	Description	Information Used
Node	Local	Degree	Number of connections of a node	Connections of the *node* of interest
Node	Intermediate	Embeddedness	Ratio of internal (community-wise) to total connections of the node	*Community* and its connections to other communities
Node	Global	Betweenness	Number of shortest paths in the network passing through that *node*	*Network* paths
Community	Local	Hub Dominance	Ratio of the maximal internal degree to the number of nodes in the community	Connections around each *node*
Community	Intermediate	Community Diameter	Longest distance between any two nodes of the community	*Community* paths
Community	Global	Average Betweenness	Mean betweenness over all the nodes in the community	*Network* paths
Network	Local	Average Degree	Mean degree over all the nodes in the network	Connections around the *node* of interest
Network	Intermediate	Modularity	Quality measure of community structure	*Communities* and their connections
Network	Global	Diameter	Longest distance between any two nodes in the network	*Network* paths

Table 2.1 Typology of Topological Measures according to Scope and Scale

of existing to possible triangles containing v in G. Transitivity scales between 0, for nodes which are not contained in any triangle, and 1, for nodes associated only to triangles. The scope of transitivity is local because one only needs the direct connections of the node and those of its direct neighbors.

$$T(v) = \frac{|\{(q,w) \in E : q \in N(v) \bigwedge w \in N(v)\}|}{d(v)(d(v)-1)/2} \tag{2.1}$$

The name transitivity comes from the fact this measure quantifies how transitive the modeled relation is. If the relation exists between the node of interest and two other nodes, it corresponds to the probability it also exists between these two nodes. Suppose we model friendship between persons, then this measure is the formalization of the saying: *"the friend of my friend is my friend"*. Thus, for a node, it measures if it belongs to close friendship groups. In real-world networks, the distribution of the local transitivity varies, as described in [71].

There are different methods to process the local transitivity: the best available exact method is in $O(l^{3/2})$ [121], to treat all nodes in the network. Note an approximate method exists too, which runs in $O(n)$ instead of $O(l^{3/2})$ [121].

2.3.4 Eccentricity

The *eccentricity* of a node is its furthest distance to any other node in the network [61]. More formally, we can define it as equation 2.2.

$$ecc(v) = max_{u \in V}(dist(v,u)) \tag{2.2}$$

Here $dist(v,u)$ is the geodesic distance between nodes u and v. The geodesic distance corresponds to the length of the shortest path between two nodes. Eccentricity measures the distance of a node to the rest of the network. Its scope is global, since we need to use paths spanning the whole network. In [129], it is stated the time complexity for calculating the eccentricity of a node is in $O(l)$ when using Dijkstra's algorithm. Hence, for the whole networks, the total complexity is in $O(nl)$.

2.3.5 Node Centralities

Centrality determines how influential a node is within a network. Among the various ways of defining formally such a characteristic, the most widely used are: *closeness*, *betweenness* and *eigenvector* centrality [10]. The scope of all the centralities described here is global, because they rely on the notion of path or distance.

Closeness centrality of a node v is defined as equation 2.3. Here $dist(v,u)$ is the distance between nodes u and v. It is the inverse of the sum of the distances between this node and all the other nodes [119]. The closeness centrality measures how close a node is to the all the other nodes. If a node has a high closeness centrality, it is able to get and to spread information efficiently [48]. The time complexity of the calculation of closeness centrality depends on that of shortest paths.

$$C_c(v) = \frac{1}{\sum_{u \in V} dist(v,u)} \tag{2.3}$$

For calculating a shortest path between two nodes, one generally uses a breath-first search, whose complexity is in $O(n+l)$, since every node and every link have to be scanned once. Since the same process must be repeated for each other node in the network, the complexity to process the closeness centrality of a node is in $O(n(n+l))$. Hence, for the whole network, the total time complexity is in $O(n^2(n+l))$.

Betweenness centrality is defined as equation 2.4. Here σ_{ij} is the total number of shortest paths from node i to node j, and $\sigma_{ij}(v)$ is the number of shortest paths from i to j running through node v, is a measure of accessibility [48].

$$C_b(v) = \sum_{i<j} \frac{\sigma_{ij}(v)}{\sigma_{ij}} \tag{2.4}$$

The betweenness centrality asserts the ability of a node to play a 'broker' role in the network by measuring how well it lies on the shortest paths connecting other nodes. A more central node has more importance in terms of information flow. Removing nodes with high betweenness centrality may corrupt the information flow and cause an important information loss or slow down its spread. Several algorithms have been proposed to calculate this centrality. The complexity of the method from [20] is the most efficient one which calculates in $O(nl)$ for the whole network.

Eigenvector centrality measures the influence of a node in the network based on its spectral properties. The eigenvector centrality of each node is proportional to the sum of the centrality of its neighbors [17]. We can formalize it as equation 2.5

$$C_e(v) = \frac{1}{\lambda} \sum_{i \in N(v)} C_e(i) \tag{2.5}$$

Here λ is a constant and $N(v)$ is the neighborhood of node v. Nodes with high eigenvector centrality are associated to many other nodes which are also associated to many other nodes, and so on. In the literature, there are different centrality measures related to neighborhoods of the first, second or higher orders, i.e. neighbors of neighbors (of neighbors, etc.), such as: Katz centrality [68], alpha centrality [18], hub and authority scores [70]. The time complexity to calculate the Eigenvector centrality depends on the number of neighbors of the node. In the worst case, a node may be associated to all the nodes in the network, leading to a complexity of $O(n)$. Hence, for the whole network, the time complexity is in $O(n^2)$ [70].

2.3.6 Embeddedness

The *embeddedness* measure assesses how much the neighbors of a node belong to its own community. It is defined as the ratio of the internal degree $d^{int}(v)$ to the total degree $d(v)$ of the considered node [77]. It can be formalized as equation 2.6

$$e(v) = \frac{d^{int}(v)}{d(v)} \tag{2.6}$$

This *internal degree* is the number of links the node has with other nodes from the same community, by opposition to its *external degree* $d^{ext}(v)$, which corresponds to connections with nodes located in other communities. The maximal embeddedness of 1 is reached when all the neighbors are in its community ($d^{int}(v) = d(v)$) whereas the minimal value of 0 corresponds to the case where all neighbors belong to different communities ($d^{int}(v) = 0$).

The scope of embeddedness is intermediate, because it requires both local and community-related information. In real-world networks, a majority of nodes, usually with low degree, have a very high embeddedness (so almost no links outside their community). For the remaining nodes, the embeddedness distribution depends on the considered system. Communication, Internet and biological networks exhibit a peak around $e = 0.5$, whereas social and information networks have a more uniform distribution. In all cases, the whole range of e is significantly represented, even small values [77]. Its processing only requires to combine the internal and total degrees, thus its time complexity is in $O(l)$, like for the degree.

2.3.7 Community Role Measures

Amaral and Guimerà [59] developed two measures, *within module degree* and *participation coefficient*, in order to characterize the role of nodes at the community level. Their scopes are intermediate, because they are based on community-related information. The *within module degree* is defined as the z-score of the internal degree, i.e. equation 2.7.

$$z(v) = \frac{d^{int}(v) - \mu(d^{int}(v), C(v))}{\sigma(d^{int}(v), C(v))} \tag{2.7}$$

Here $\mu(d^{int}(v), C(v))$ and $\sigma(d^{int}(v), C(v))$ denote the mean and standard deviation of d^{int} over all nodes belonging to $C(v)$, respectively. It expresses how much a node is well connected to other nodes in its community, relatively to this community. To process the z values, we need first to compute d^{int} for all nodes, which is in $O(l)$ as explained before. The mean and standard deviation of the internal degree must then be processed for each community, both operations being linear in the number of nodes of the considered community. Overall, both are consequently in $O(n)$. In total, when considering time, the complexity to process z is therefore in $O(l+2n) = O(l+n)$.

The *participation coefficient*, introduced in the same study [59], is based on the notion of community degree $d^c(v) = |N(v) \bigcap C_c|$, which represents the number of links a node v has with nodes belonging to community C_c. Incidentally, one can see the internal degree is a specific case of community degree, for which $C_c = C(v)$. This measure is formally defined as equation 2.8.

$$P(v) = 1 - \sum_{1 \leq c \leq \lambda} \left(\frac{d^c(v)}{d(v)}\right)^2 \tag{2.8}$$

Here λ is the number of communities in G. P characterizes the distribution of the neighbors of a node over all communities. More precisely, it measures the heterogeneity of this distribution. It gets close to 1 if all the neighbors are uniformly distributed among all the communities and 0 if they are all gathered in the same community. The processing of P requires first computing the total and community degrees of all nodes for all communities. As explained before, the total degree can be processed in $O(l)$. By proceeding like for the internal degree, one can process the community degree relatively to a given community also in $O(l)$. Therefore, dealing with all communities amounts to $O(l\lambda)$. The total complexity to process P is eventually in $O((\lambda+1)l) = O(\lambda l)$.

Amaral & Guimerà determined 7 distinct community node roles, based on a cartography of within module degree and participation coefficient [57]. The regions of these node roles are shown in Figure 2.4 on the within module degree and participation coefficient space.

The authors firstly distinguish nodes depending on whether their z is above or below a limit of 2.5. If $z \geq 2.5$, the node is considered to be a *community hub*, because it is significantly more connected to its community than the other members of the same community. If $z < 2.5$, the node is said to be a *community non-hub*. Note this notion of hub is different from the degree-based one introduced in section 2.2.1. A community-hub is relative to the structure of the community. A node may have high degree but all its neighbors in different communities, in which case it is not a community-hub. For community non-hub nodes, Amaral & Guimerà determined 4 roles by considering their participation coefficient values. The regions shown as $R1$, $R2$, $R3$ and $R4$ in figure 2.4 correspond to *ultra-peripheral* (i.e. all their links are in their community), *peripheral* (i.e. most of their links are in their community), *non-hub connector* (i.e. many links are in other community), and *non-hub kinless* (i.e. all their links are homogeneously distributed among all existing communities) nodes, respectively. For the community-hub nodes, they determined 3 roles shown as $R5$, $R6$ and $R7$ in the same figure. These roles correspond to *provincial hubs* (i.e. hub nodes with most of their links are in their community), *connector hubs* (i.e. hub nodes with many links to most of the other communities) and *kinless hubs* (i.e. hub nodes with all their links

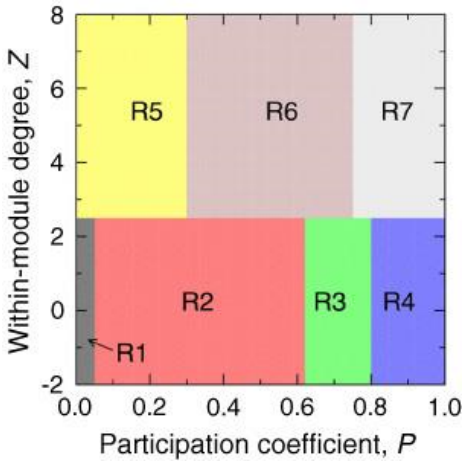

Fig. 2.4 Zones of Node Roles Inside of a community in $z-P$ space. (image courtesy of [57])

are homogeneously distributed among all existing communities) respectively.

2.3.8 Example

We explain the topological differences between the nodes of the same network with a small example. One can see a small network with 21 nodes separated into 3 disjoint communities and 41 links at Figure 2.5. The nodes A and C belong to first community; B and D belong to second community and finally node E belongs to third community. For this network, we listed the topological measures of nodes A, B, C, D and E in Table 2.2.

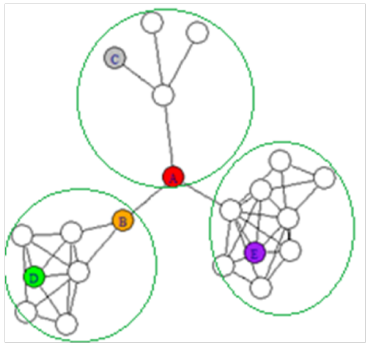

Fig. 2.5 Small network with 21 nodes and 3 communities

Node A is located at the center of the network, connecting three smaller components that can be seen distinctly on the figure. That is why, the betweenness centrality of this node is significantly higher than those of all the other nodes. Because node A does not belong to any triangle connection, its local transitivity is 0, as well as node C. The most transitive node is D. We can observe that node D and E both belong to well-connected neighborhood. The most popular node among the nodes we examine is E because its degree is the highest while node C is the least popular one. Regarding their eccentricity, we can say that node D is the furthest one. Although it is situated inside of a well-connected neighborhood, its access from other parts of the network is more difficult than the other nodes we analyze. As being the most in-between, node A is also the closest, as shown by having the highest score for closeness centrality. However, the highest score of Eigenvector centrality belongs to node E, because it is connected to many nodes that are connected to many others and so on.

	Node A	Node B	Node C	Node D	Node E
Degree	3.000	3.000	1.000	4.000	5.000
Internal Degree	1.000	2.000	1.000	4.000	5.000
Transitivity	0.000	0.300	0.000	0.830	0.700
Eccentricity	3.000	4.000	5.000	6.000	5.000
Closeness Centrality	0.023	0.020	0.013	0.013	0.016
Betweenness Centrality	127.000	84.000	0.0000	0.250	0.910
Eigenvector Centrality	0.219	0.055	0.009	0.018	0.786
Embeddedness	0.330	0.660	1. 000	1. 000	1. 000
Within Module Degree	-0.450	1.780	-1.540	0.000	0.250
Participation Coefficient	0.660	0.440	0.000	0.000	0.000

Table 2.2 Topological Measures Values for the Nodes

According to these measures, one can determine some node roles. For example, being a hub means having a degree above the network average. A *bottleneck* is a node which controls the flow from one part of the network to another part , and whose deletion increases the number of unconnected sub networks [33]. A *bridge* is a link which is connected to bottlenecks [33]. The removal of a bridge also causes to partition the network into several disconnected sub-networks. Here, node B is a hub and a bottleneck, node A is not a hub but is a bottleneck, node E is a hub but not a bottleneck and node C is neither a hub nor a bottleneck.

The embeddedness score of nodes C, D and E shows these nodes completely belong to their communities. They do not have connections with nodes from other communities. On the contrary, node A has 2 connections with other communities, and its embeddedness is consequently lower. Moreover, the fact this node A is connected to two different communities affects its participation coefficient. For node B, the participation coefficient is also larger than 0, meaning that it has connections to other communities. Regarding the within module degree scores, no node hold a community hub-role, according to the threshold determined by Amaral & Guimerà [59]. If we consider the Amaral-Guimerà roles, A is a non-hub connector, node B is peripheral and node C, D and E are ultra-peripheral.

Chapter 3

Community Detection

We now focus on the notion of community structure, which was introduced in the previous section. As mentioned before, a community can be intuitively defined as a group of nodes densely interconnected compared to the other nodes [32, 43]. We already highlighted the fact many real-world networks possess a community structure. Moreover, identifying the community structure of a system is crucial to improve its understanding. Indeed, it summarizes the complex structure of the network, and nodes belonging to the same community are often functionally related. The interactions between communities, the hierarchy inside a community, the central nodes of a community, or the nodes lying in-between communities, also give us important information about the dynamics of the system. For these reasons, the community structure constitutes an important topic, which is studied widely, in many very different domains.

Because of this popularity, hundreds of different algorithms were developed for the task consisting in identifying the community structure of a network, an operation generally called *community detection* [43]. However, although a lot of work has been performed regarding this detection problem, the task of interpreting the detected communities has been ignored almost completely until now. Yet, a community structure is not more than a partition of the node set: it has no meaning by itself. It must undergo an additional processing in order to reveal some information which would be meaningful relatively to the studied system. In other words, interpretation of community is complementary to their detection, and crucial to their exploitation.

In most of the community detection studies, the interpretation phase is skipped or manually done by some expert of the domain [50, 93, 111, 115, 116]. However, manual interpretation is somewhat subjective and does not scale well when considering larger networks. Moreover, it might be considered as a subjective approach. A few methods were defined to propose more objective tools [73, 77, 133], but they only partially handle the available information. For instance, some of them concentrate on nodal attributes [29, 137] while some others use only topological information [77]. As a result, such an interpretation is incomplete, and can lead to limited observations regarding the studied system.

In this chapter, we try to give a global view of the community detection task, by describing not only community detection methods, but also evaluation and interpretation methods. Our goal is not to be exhaustive: this description is driven by the objective of this work, which is to handle dynamic and attributed networks. For this reason, we first describe community detection in plain networks, in order to introduce some basic concepts. We then explain how they can be extended to handle the two network enhancements which interest us: attributed networks, then dynamic networks. For each network type, we try to distinguish several general families of methods, before describing one specific method in more details. We then describe a related problem, which is to evaluate community detection methods in terms of quality of the detected communities. Finally, in the last part of this chapter, we review the existing approaches for community interpretation.

3.1 Plain Networks

There are hundreds of different community detection methods using only the topological information present in plain networks. Although it is important to understand how community detection works, the point of this thesis is community interpretation, so we do not make an exhaustive review here. It is worth noticing more detailed reviews exist in the literature, see [25, 43, 110, 120]. In this subsection, we first summarize briefly some popular methods by explaining their detection strategies. We then focus on modularity, which is the most widely used quality measure for community structure, and the Louvain algorithm, which optimize modularity. We chose to focus on Louvain, because it is one of the most popular community detection methods, and extensions exist to deal with attributed or dynamic network. In our experiments, we also use Louvain and its variants.

3.1.1 Short Review of Existing Approaches

Centrality-based. In the early stage of community detection studies, the strategy for identifying the communities was to find the most central links, which are supposed to connect different dense node groups. We call this approach *link-centrality based*. The algorithms using link-centrality measures rely on a hierarchical divisive approach. Initially, the whole network is seen as a single community, i.e. all nodes are in the same community. The most central links are then repeatedly removed. The underlying assumption is that these particular links are located between the communities. After a few steps, the network is split into several components which can be considered as communities in the initial network. By iterating the process, one can split each discovered community again, resulting in a finer community structure. Algorithms of this category differ in the way they select the links to be removed. The first and most known algorithm using this approach was proposed by Newman [50], and relies on the *edge-betweenness* (a link-based variant of the betweenness centrality presented in Section 2.3.5). Radicchi et al. [111] propose a variant called *Radetal*, based on link transitivity (a link-based version of the local transitivity presented in Section 2.3.3), i.e. using a local measure instead of the global edge-betweenness.

Distance-based. Another category of approaches is based on *node distance* measures. Such a measure allows translating the topological information encoded in the network into a numerical value representing how far two nodes are. A community is then viewed as a group of nodes which are close to each other, but distant from the rest of the network. Once all node-to-node distances are known, detecting a community structure can be performed by applying a classic similarity-based cluster analysis algorithm [49]. There are many ways to express the distance between two nodes. The most basic one, for instance, is the geodesic distance, which considers the length of the shortest path between the nodes. In the popular *Walktrap* algorithm, the node distance is expressed in terms of random walks. Then, a hierarchical agglomerative clustering approach is applied [109] by using this measure.

Compression-based. The approaches based on *data compression* consider the community structure as a set of regularities in the network topology, which can be used to represent the network in a more compact way (than using the whole network itself). The best community structure is supposed to be the one maximizing compactness while minimizing information loss. The quality of the representation is assessed through measures derived from information theory. Algorithms essentially differ in the way they represent the community structure and how they assess the quality of this representation. Two popular approaches from this category are *InfoMod* [115] and *InfoMap* [116]. In the former, the simplified representation of the network is based on a community matrix representing the community structure and a vector representing the community membership of nodes. The latter uses a random walk as a proxy. Each node is assigned a unique code taking its community into account, and one tries to define communities so that the quantity of information needed to representation a walk is minimal.

Significance-based. A completely different approach consists in considering the *statistical significance* of the whole community structure, or of the individual communities. Indeed, in a complex net-

work, a community structure can be expected under certain circumstances, but groups of densely connected nodes can also appear only by chance. For instance, even a generative model not supposed to create any community structure can produce clusters of nodes due to random fluctuations. Statistical significance allows distinguish them from actual communities. An example from this category is *Order Statistics Local Optimization Method* (OSLOM), which is a local optimization method applied to a score measuring the statistical significance of individual communities [78].

Diffusion-based. Such approaches tackle the problem of community detection using a communication paradigm. They rely on the assumption that information is more efficiently exchanged between nodes of the same community. Therefore, communities can be detected by considering how information is propagated in the network. The most well-known method from this category is the *Label Propagation* algorithm [112]. The considered information takes the form of a label, and the propagation mechanism relies on a vote mechanism occurring between neighbors. The *Community Overlap Propagation Algorithm* (COPRA) is an extended version of Label Propagation, proposed by Gregory [53] to detect overlapping communities. *MarkovCluster* is also very popular among diffusion-based methods. By repeating a set of algebraic operations on the network transfer matrix, which describes the transition probabilities for random walker evolving in this network, it simulates a long-term diffusion process to detect communities [134].

Optimization-based. Finally, community detection can be seen as an *optimization problem*, provided one can define a measure of the quality of a community structure. There are several such measures [43], amongst which the most widespread is certainly the *modularity*. This chance-corrected measure assesses how much cohesive and separated the communities are, through their numbers of intra- and inter-community links [93]. The algorithms based on *modularity optimization* constitute the majority of the community detection algorithms. They differ in the way they perform this optimization. *FastGreedy* is the earliest method using this strategy. It applies a basic greedy approach [95]. The *Louvain* algorithm [15] can be considered as an improvement of Fast Greedy, relying on a hierarchical approach to handle much larger networks. It is very fast, and identifies highly modular community structures. For these reasons, it might be the most widespread algorithm for community detection, not only for pure applications, both also as a basis to build methods able to handle enhanced networks. We consequently decided to use this algorithm for our work. In the next subsection, we describe the modularity measure in further details, and then explain the functioning of Louvain.

3.1.2 Modularity Measure

The modularity measure has been introduced by Newman and Girvan [97] to assess the quality of a network partition. They first define what could be called a community contingency matrix, whose elements p_{ij} represent the fraction of total links from nodes in community i towards nodes in community j. The fraction of links inside community i is therefore p_{ii}. In the case of undirected networks, we have $p_{ij} = p_{ji}$ and the matrix is symmetric; however, note that for normalization purposes, the off-diagonal terms correspond to half the fraction of links between communities i and j. In this matrix, let p_{i+} and p_{+j} be the sums over row i and column j, respectively. If the network has no community structure, or if the considered communities are not defined accordingly to the network structure, then one can suppose the links are randomly distributed. Under this hypothesis, the expected fraction of links inside community i can be estimated as the probability for a link to start in community i, which is p_{i+}, multiplied by the probability to end in community i, which is p_{+i}. The matrix being symmetric, we have $p_{i+}p_{+i} = (p_{i+})^2$. The modularity measure is defined as the difference between the observed and expected fractions of links in each community, summed over all communities:

$$Q = \sum_i p_{ii} - \sum_i (p_{i+})^2 \tag{3.1}$$

When the communities are not better than a random partition, or when the network does not exhibit any community structure, Q is negative or zero. Its upper bound is 1, but it can be approached only if the network has a strong community structure and if the communities have been perfectly detected. A network with a well-separated community structure is known to have a modularity between 0.3 and 0.7 [93, 131]. In Figure 3.1, one can distinguish two networks with different modularity values.

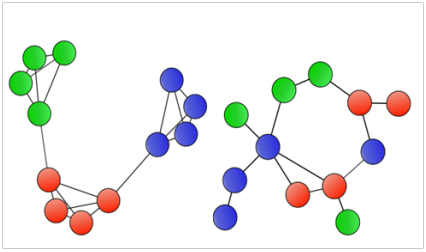

Fig. 3.1 Two plain networks with different modularity values. For the left one, $Q = 0.8499$, and for the right one, $Q = 0.0545$ (image courtesy of [141]).

The modularity measure is known to have some flaws. First, it has a so-called *resolution limit* [44], which affects the size of the communities it allows detecting. This is due to the fact the modularity compares the distribution of links over communities with its expected counterpart for a random network possessing the same number of nodes and degree distribution (but randomly attached links). This assumption implies the expected number of links between communities decreases when the network size increases. Thus, if the network is large enough, the expected number of links between communities in this null model may be smaller than one. In this case, a single link between two otherwise well-separated communities is interpreted as a sign of strong inter-community connection by the modularity. Thus, maximal modularity is obtained for a community structure including less communities than in the actual community structure. Second, there is a *relevance* problem, since it is possible to find partitions with relatively high modularity values even in completely random networks, not supposed to possess this property, such as Poisson random networks [58]. Finally, the modularity is also known to lead to *degenerated* results [51]. For one network, one can identify an exponential number of significantly different community structures with a near-maximal modularity. This means even an excellent optimization algorithm can lead to very different communities when applied several times (provided it is at least partially random, which is generally the case), which in turns causes an instability. However, despite those limitations, an overwhelming majority of community detection algorithms use modularity as an optimization criterion, because it proved to be very efficient in practice. For this reason, in this work we use modularity-based approaches.

3.1.3 Louvain Algorithm

Louvain is a modularity optimization algorithm proposed by Blondel et al. [15]. It is an improvement of Fast Greedy, introducing a two-phase hierarchical agglomerative approach. During the first phase, the algorithm applies a greedy optimization to identify the communities. In the initial state, a community corresponds to a single node. In other words, there are as many communities as nodes. The algorithm then applies the same process to each node: it considers all possible moves from its current community to those of its neighbors. The move leading to the highest modularity gain is performed, provided this gain is positive. If no positive gain is possible, the node stays in its original community. Louvain repeats this procedure until no further improvement can be achieved.

During the second phase, the algorithm builds a new network whose nodes are the communities found during the first phase. The links located inside the original communities are represented by loops in the

new network, and the links located between them become links between the corresponding nodes of the new network. These new links are weighted in order to represent the number of original links. Once the new network has been created, the first phase is applied again, this time on this new network. The time consuming phase is clearly the first one. The authors call the combination of these two phases a *pass*. Passes are repeated until there are no more changes during the first phase (i.e. the modularity cannot be increased by moving a node), and a (local) maximum is reached. Louvain naturally incorporates a notion of hierarchy, as communities of communities are built during the process. The pseudo-code of this algorithm is given below.

Algorithm 1 Louvain

Require: $G = (V, E)$
Ensure: $\mathscr{C}$
  repeat
    $n = |V|$
    $\forall v_i \in V, C(v_i) \leftarrow$i, *with* i$\in \{1, \cdots, n\}$
    $\mathscr{C} = \{C_1, \cdots, C_n\}$
    $Q_{init} \leftarrow modularity(\mathscr{C})$
    for $v_i \in V$ **do**
      $(\mathscr{C}, Q) \leftarrow rebuildCommStr(v_i, \mathscr{C}, Q)$
    end for
    while some nodes are moved **do**
      if $Q > Q_{init}$ **then**
        $G \leftarrow rebuildGraph(G, \mathscr{C})$
      end if
    end while
  until $\mathscr{C}$and Q do not change

Here, the function `rebuildCommStr(···)` moves the node v_i, if necessary, to that of the neighbors communities which maximizes the modularity gain. The function `rebuildGraph(···)` updates the graph by taking each community as a node and total weight of the links between communities as the link weight. Although the exact computational complexity of the method is not known, in [14], it is indicated that the method seems to run in time $O(nlogn)$ with most of the computational effort spent on the optimization at the `rebuildCommStr(···)`. In fact, the algorithm is limited not by the computational time but by the storage capacity for very large networks. The authors tested it on 118 million nodes network and observed it worked very fast (compared to alternative approaches).

3.2 Attributed Networks

The works taking advantage of attributes during the community detection are relatively recent, and therefore much less numerous than for plain networks. This sub domain can therefore be considered as an emerging topic. Indeed, until recently, there were only a few publicly available datasets containing both structure and attributes. There are several reasons for that. First, storing these extra data (attributes) leads to technical difficulties. Second, many systems are composed of very simple objects which do not require to be individually described. However, with the apparition and growing popularity of Internet-based social networking services, it became easier to access this type of data. In this section, we try to categorize and summarize the main existing approaches, by describing their key points. We identified four categories: *Model-Based, Integration-Based, Similarity-Based* and *Optimization-Based* methods.

After having presented the main approaches, we expose their main limitation, at least regarding the topic of this work: they all rely on the assumption of homophily.

3.2.1 Model-Based Methods

We regroup under the name *model-based* approaches a set of probabilistic methods using some generative model to represent the relationships between the network structure (including its communities) and the node attributes. It allows to explicitly model the dependence between link distribution and nodal attributes. Their basic idea is to describe the communities as latent objects depending on the random mixture of link and attributes distribution in the network. The links and nodal attributes are then treated by a single statistical model. Communities are found by fitting the parameters of this statistical model. The model-based methods are usually inspired by *Latent Dirichlet Allocation* (LDA) [13]. LDA is a generative model for collections of discrete data, originally used for identifying topics in collections of documents. In that context, each document is supposed to be related to a mixture of various predefined topics. A document is considered as a set of words, each one characteristic of a specific topic. The name of the model comes from the assumption the topics follow a Dirichlet distribution over the documents. For this model, the parameters are estimated using statistical inference methods such as Gibbs sampling, Expectation Maximization, etc. In the case of community detection, the topics (latent variable) correspond to the communities. Most of those community detection algorithms from this category have originally been developed to handle text-related networks, such as networks of blogs where the nodes are the authors, the links are their friendship and the attributes describe the documents the authors wrote.

In [8], the authors propose a method named *Block-LDA*, which is a joint model of latent topics and stochastic block models. In this model, link structure helps to improve attribute inference and vice-versa. Similarly, in [22], the membership of the nodes to the communities is modeled with a mixture model based on LDA. This algorithm is called *relational topic model*. The probability of existence of a link between two nodes is determined by a binomial distribution whose parameters depend on both topic similarity and community. In [83], the algorithm *Topic-Link LDA* is proposed as a relational topic model based on LDA, such that the links are modeled as a binary random variable conditioned to the attributes of nodes. Yang et al. [137] propose the algorithm *Community Detection in Networks with Node attributes* (*CESNA*), also inspired by LDA. Unlike the other methods, CESNA is capable of finding overlapping, non-overlapping or hierarchical communities if they appear in the same network.

3.2.2 Integration-Based Methods

In the second category, we gathered algorithms which modify the network, in order to embed the attribute information *inside* the structure. The advantage of this method is that a classic community detection method (i.e. one developed for plain networks) can then be applied to the augmented network. In other words, these algorithms apply a pre-processing to generate a new network including attribute information embedded in structure, and then only perform the actual community detection.

SA-Cluster, proposed by Zhou et al. [140], is particularly representative of this category. The authors define an *augmented network* based on the original one, by adding new nodes and links in an attempt to re-encode attribute values under a purely structural form. In this augmented network, an *attribute node* corresponds to one of the possible original attribute values while an *original node* represents a structural node. An *attribute link* is then placed between an original node and an attribute node if the original node takes the corresponding value for the considered attribute in the original network. An *original link* is representing a topological link of the original network. Different weights are assigned to both attribute and original links, in order to discriminate their importance. The value is constant for original links (1.0), whereas for attribute links, it is adjusted during the optimization process. This optimization process is based on using a random-walk distance between nodes as the distance measure. To find the communities,

the *k-medoid* clustering method is applied on the augmented network. At each iteration, the weights of the attribute links are adjusted until the clustering method converges.

This *CODICIL* algorithm presented by Ruan et al. [118] uses a similar approach. First, it creates attribute links such that, for each node, there is a new attribute link between this node and its neighbors having the most similar attributes. Attribute similarity is measured using the cosine method. Second, CODICIL samples the most meaningful links among the union of content and topological links, in order to reduce the network. The criterion of retaining the links is the relevancy of links properties with its neighbors. Then the algorithm applies a classic community detection method on this new network.

3.2.3 Similarity-Based Methods

The third category contains methods relying on *classic clustering techniques*, through the use of similarity measures combining attributes and structure. By classic, we mean here that those algorithms were designed for tabular data, not networks. Many of these tools rely on distance (or similarity) matrices, in which objects are compared pairwise. The idea is that if one can define such a measure to compare nodes, then he can directly apply a classic clustering method on the obtained similarity matrix.

Combe et al. [24] propose a method handling networks whose nodes represent texts and links represent connection between them. It is based on hierarchical clustering and uses a hybrid distance measure, corresponding to the weighted sum of a structural and an attribute-related distance. For the structural part, the authors use the geodesic distance (length of the shortest path between two nodes). For textual attributes, each text is described by a vector whose members correspond to the weight of the terms of that text. The attribute distance is described as the Euclid distance between the two vectors expressing the attributes of each node. The method is specific to this type of network, in the sense not all nodal attributes take this form, depending on the considered system.

Steinhaeuser and Chawla [127] propose three different link weights assessing the similarity of two nodes. The first weight is the difference of local transitivity between the two nodes. However, there is a hint here. The difference is calculated twice: once normally, and once by considering the two nodes are not connected. Then those two differences are summed to get the final weight. The second weight is the Jaccard's coefficient [67] processed over the nodes neighborhoods, i.e. ratio of the number of common neighbor to the total number of neighbors. The third link weight is a node attribute similarity. For two neighbor nodes, in the case of a nominal attribute, the weight of the link between them increases by one if they have the same attribute value. For continuous attributes, the attribute is normalized to range from 0 to 1 and then the arithmetic difference between them is used as the weight. Once the weights are calculated for all attributes, they are normalized to range from 0 to 1. The community detection criterion is then very simple: if a link weight is above a certain threshold, then the two connected nodes are put in the same community. The authors find the communities according to each weight separately.

Finally, Moser et al. [92] proposed the algorithm *NetScan*, as a modified version of the k-means algorithm. They apply k-mean with two additional criteria while assigning the nodes to the communities. At each iteration of k-mean, apart from the rules of k-mean: (1) the nodes of the same community must be mutually reachable and (2) there must be a central node such that the attribute distance of each other nodes in the same community is less or equal to a predefined radius. Thus, they consider link structure and attribute similarity as two independent criteria.

3.2.4 Optimization-Based Methods

The forth category includes the algorithms using *optimization techniques*. They rely on a fitness function combining both structural and attribute-related information. This function represents the quality of the partition, in terms of connections and attributes. Many different methods can then be applied to optimize it. In the literature, most fitness functions are variants of the modularity, or use it in some way.

Cruz et al. [27] use on the one hand the modularity to measure the quality of a node partition from a structural perspective, and on the other hand data entropy to consider if from the attribute point of view. Their optimization method is an extension of Louvain (described in section 3.1.3), which tries to maximize the modularity and minimize the entropy. At first, an initial partition is detected structurally, by a single modularity optimization step. Then, new partitions are defined according to entropy minimization, by using a Monte-Carlo method. Finally, the community aggregation step of Louvain is performed. After the aggregation, another step of modularity optimization is performed. This process is repeated iteratively until no modularity improvement can be achieved.

Dang and Viennet [29] propose an algorithm which is also directly inspired by Louvain. However, unlike Cruz et al. [27], they introduce a composite modularity as a weighted combination of a structure modularity (classic modularity) and an attribute modularity (specifically defined to handle attributes). The equation of this modularity is given in Eq. 3.2.

$$Q = \sum_{C} \sum_{i,i \in C} (\alpha.S(i,j) + (1-\alpha).simA(i,j)) \tag{3.2}$$

Here $\sum_C \sum_{i,i \in C} S(i,j)$ corresponds to the structure modularity and $\sum_C \sum_{i,i \in C} simA(i,j)$ is attribute modularity and α is a user defined coefficient. The authors used the Euclidean distance as the similarity measure in their attribute modularity.

Akoglu et al. [1] use a matrix-based approach. They consider the structure through the network binary $n \times n$ adjacency matrix A, and the attributes through a binary $n \times f$ feature matrix F. Here n is node number and f is the feature number. If there is a link between two nodes i and j, then $A[i,j] = 1$. If node i possesses feature k, then $F[i,k] = 1$. For other cases, both matrix takes value 0. The main idea is to find clusters in these two matrices simultaneously, by applying the Minimum Description Length (MDL) principle. They apply a greedy iterative heuristic in order to minimize their information theory-based fitness function.

3.2.5 Assumption of Homophily

It is important to note all the attribute-based community detection methods rely on the assumption of *homophily* at the community level. A node set has the property of homophily, also called attribute *assortativity*, when similar nodes (in terms of the considered attribute) tend to connect more [87]. The opposite property is called *heterophily* (different nodes tend to connect more), or attribute *disassortativity*. The reason behind this assumption is that those community detection methods are designed to solve specific problems, and handle certain types of networks. They correspond to cases were the user is specifically looking for communities whose nodes are both simultaneously tightly connected and similar in terms of attributes.

This is hardly generalizable to all systems, though. Homophily is a common property, observed at the global level in many networks [94]. However, heterophily, although rare, is also observed. Especially in the case where several attributes are defined, it is not uncommon to observe homophily for only some of them. Moreover, to the best of our knowledge, there is no study showing that, in real-world networks, nodes located *in the same community* are similar in terms of attributes. Even if this could be observed in certain networks, it is also possible to find, in the same network, certain communities which do not respect this rule. The study from [74] is a good example of this situation. While studying an attributed social network of university students, they partition the node set first by using the structure only, second by using the attributes only, and then compare the resulting groups. For these data, it turns out both types of information (relational vs. individual) partially overlap, but are still significantly different. Moreover, attribute homogeneity varies from one community to the other.

In conclusion, our point here is that community detection methods using attributes are appropriate

under certain, non-general, conditions only. Moreover, contrary to what one could intuitively believe, they do not necessarily make the task of interpreting the identified communities easier, as we will see in section 3.5.

3.3 Dynamic Networks

Community detection in dynamic networks is also an emerging topic, for the same reasons than attributed networks. However, in the case of dynamic networks, community detection is more complicated than the one for static networks because of maintaining time information as well. Although the domain is quite new, an explicative summary of the existing algorithms is given in [3, 130]. Here, we will give an overview of the existing methods by distinguishing two different categories. First, we have *two-stage* approaches, also called static approaches, which consist in performing a static detection on each time slice, and then matching the communities from one time slice to the next. Second, there are the *evolutionary* approaches, which directly use the available temporal information to find the community structure. They can be separated into two categories again: *online* and *offline* methods. The former are also called incremental methods, and they consist in using the temporal information of the previous time slice to find the community structure of the next one. The latter include the methods using the complete available temporal information (all time slices together) to find one community structure supposed to be relevant for the while time line.

Figure 3.2 illustrates the difference between the two main categories, i.e. using the temporal information, or not, for the detection. Its left side explains the idea of two-stage detection. The vertical solid arrows show how the network structure of each time slice is used separately, to perform static community detection. The dashed horizontal arrows between the time slices indicate the use the temporal information to match the communities detected in consecutive time slices. The right side of the figure represents the evolutionary approach. The lower horizontal arrows represent the dynamic update of the community structure based on the previous time slice(s), and the vertical dashed arrows show how the evolution of the network structure is taken into account during this update. With this method, the detection at a time slice is done by using the community structure of the previous time slice. Unlike two-stage methods, evolutionary methods do not treat each time slice separately, but instead use overall temporal information together.

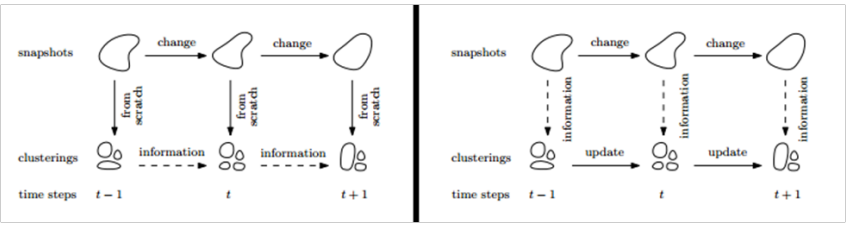

Fig. 3.2 Illustration of two different community detection strategies for dynamic networks. (Image courtesy of Tanja et al. [130]).

3.3.1 Two-Stage Approaches

As indicated by their name, the process performed by these approaches is composed of two distinct steps: first communities are detected in each time slice independently, and second they are matched from one time slice to the other, in order to track their evolution. For the first step, static community detection methods can be applied, therefore the second stage is more interesting, here. One way of finding such a match is using set theory. For example, Hopcroft et al. [64] look at the maximum common node numbers

in the communities from consecutive time slices. However; because of the instability of the community detection method used, many communities can disappear or change between consecutive time slices. It is not possible to decide if these communities are really dying (or changing too much) or if they do not appear in the following time slice because of the community detection method. So, the match is done only between stable communities, which are very few in reality.

Spiliopoulou et al. [124] propose a framework called *MONIC*. In MONIC, the match of a community C in the following time slice is the community whose intersection with C is the largest, provided this intersection passes a given threshold. If there is no such community, then there is no match at all. In MONIC, the community is tracked based on some rules to determine merge, split, disappear or appear. If there is one match for C, the community C is a *natural* community. For split, the match is searched as more than one community whose union is a subset of C. A merge event is detected by looking if there are several matches for C in the same time slice. An appearing event is detected if there is no match at the previous time slice. If none of these events occurs, it means C disappears. The matching method proposed by Asur et al. [2] is quite similar to MONIC. However, for each event, they use a threshold for a more stable matching. Greene et al. [52] also use a similar approach but they use Jaccard's index [67] as the matching criterion, rather than the simple intersection size.

Another way to find the match is using a community detection algorithm to match the communities between two time slices. In [104], the communities at each time slice are detected by *clique percolation* [34] (a method for finding overlapping communities by regarding adjacent k-cliques sharing $k-1$ nodes). Then, for each pair of consecutive time slices, a union network including all the links of time slice t and $t+1$ is created. Finally, the clique percolation method is applied again on this union network. The communities of time slice $t+1$ that are grouped with communities at time t of the union network are considered as their evolution. The limitation of this approach is that it can miss merge, birth or split events if the modifications (appearing or disappearing links) are not radical enough.

Another way of using community detection is searching the matches of the community ID's of consecutive time slices. For such a matching, of course specified algorithms should be invoked. In [122], authors proposes a method as an extension of *LabelPropagation* [112] which finds the communities of different time slices with relevant community ID's. This algorithm changes the labels locally if a new node added or removed at the incremental time slice.

3.3.2 Evolutionary Approaches

Chakrabarti et al. [21] introduced the concept of *evolutionary clustering*, which was initially defined for non-network data. However, the idea can be adapted to network science as well. It consists in considering a part (or all) of the sequence of networks to detect communities, by opposition to using each time slice separately. A method based on the evolutionary clustering principle should respect two criteria. First, the community structure of any time slice should be as good as possible for that time slice. Second, the community structure should not shift dramatically from one time slice to another. In the same article, Chakrabati et al. introduced the notion of *temporal smoothness* for their evolutionary clustering. It expresses the assumption that when a real community shifts over time, this should be view as a smooth of transition. Thus it is necessary to update the existing information from the previous time slice(s) to get the community structure of the current one. They define a general quality function as $Q = Q_{snapshot} + \alpha Q_{stability}$, where $Q_{snapshot}$ is the fitness value of the considered time slice, $Q_{stability}$ is the difference between the previous and considered time slices, and α is a user-defined constant to regulate the fitness value. There are two ways of applying the evolutionary principle: in *online methods*, only the last, of the few last time slices are considered; whereas in *offline methods*, the whole time line is used.

3.3.2.1 Online Methods

These methods are called online because they allow detecting communities on the fly, since they only use the past state of the network. They are also sometimes called *incremental methods*, because they build the community structure of the current time slice upon that of the previous one. Community detection is performed by analyzing at least two consecutive time slices. Because they use less information than the whole dynamic network, these methods are faster and useful for real-time monitoring.

Görke et al. [54] propose two modularity-based methods: a modification of Louvain and a modification of FastGreedy [95] , obtained by adapting the Modularity to take temporal smoothness into account. The authors apply similar heuristics with the original methods of FastGreedy and Louvain on the previously found communities, in a way such that nodes will not be affected by the changes of the network in their previous community.

Aynaud and Guillaume [5] propose *Incremental Louvain* which is also a modified version of Louvain taking into account temporal smoothness. The first phase of the Louvain algorithm (as explained in section 3.1.3) is modified: rather than initially putting each node in a separate community, it starts with the community structure of the previous time slice, and optimizes the modularity according to the current time slice.

3.3.2.2 Offline Methods

Some methods use the whole time line of the dynamic network rather than a subset of its time slices. They are consequently called offline, since they can be applied only after the studied process has been completely measured. The idea is finding one community structure which would be relevant for all time slices. In [4, 6], the authors propose a modified version of the Louvain algorithm which optimizes the average quality over all time slices, inspired by the modularity. The first phase of the classic Louvain method consists in repeatedly performing the best node move (i.e. moving a node from one community to the other) in terms of modularity gain. Here, a move is considered for each time separately, and the corresponding gains are averaged. The move effectively performed corresponds to the one with the highest average gain. The community structure is therefore always the same for all time slices: only the network structure can be different. During the second Louvain phase, a network of higher hierarchical level is created by considering each community as a node in the new network, and links between communities as weighted links in the new network. Here, the same process is applied to each time slice. As in the original Louvain algorithm, these 2 steps are repeated until modularity cannot be improved anymore.

3.3.3 Conclusion

Researches related to community structure in dynamic networks are quite new when we compare to static networks. That is why, there are many open questions as how to validate the algorithms, which type of matching is more pertinent, how to model the network for community detection; considering the new comers as added nodes may cause problems for detection methods, and so on. Only in few works, authors give some experimental results for validation of their methods. For example, Aynaud and Guillaume [5] find that using the Incremental Louvain method give a more stable result for dynamic networks than applying classic Louvain. However, we should underline that the authors do not use a ground truth for their comparison but the arXiv network dataset of Newman [96], which is originally a static network. Aynaud and Guillaume randomly remove one node for creating an evolution event. Their assumption is that removing one node should not affect the community structure, thus, result community structures on each time slice should be rather stable. They measured the stability of some static community detection algorithms (Static Louvain, Walktrap and Fastgreedy) and their Incremental

Louvain algorithm by regarding mutual information and number of necessary transformation to have the community structure before removing a node. Obviously, there are many paths to walk at this domain. One of them is certainly taking into account the attribute information for dynamic community detection. To the best of our knowledge, none of the existing methods of dynamic community detection uses the attribute information and their evolving with the topological structure.

3.4 Evaluation of the Algorithms

Community detection algorithms can be evaluated on various criteria such as time or spatial complexity. However, the most important aspect is the quality of the detected community structure. Authors traditionally validate their community detection algorithms on real-world networks [41, 50, 63, 84, 95, 112]. But in real-world networks, actual communities are not always defined objectively, or even known. Additionally, it is not possible to control the topological properties of the networks to see the limits of the algorithms. Consequently, the algorithms are tested on a very specific and limited set of features. Artificial networks overcome these limitations, by allowing to control the network structure, to define the communities *a priori* and to generate many networks randomly. This is the reason why many authors also use them to validate their algorithms [50, 95, 97, 109, 114].

The classic method for evaluating algorithms with artificial networks consists in creating a benchmark by using an artificial generative network model with a community structure, applying the algorithm on this benchmark and assessing its performance. This leads us to the second point of interest in the evaluation of community detection algorithms: which tool to use to measure the performance of an algorithm. This task is performed by comparing the community structure estimated by the algorithm with the predefined one. In this section, we first review the main methods to generate artificial networks possessing a community structure, and then describe the measure used for performance assessment.

3.4.1 Network Generation

As the rising interest on community detection algorithms, the interest to develop models generating artificial networks with community structure also increased. In the early stage, a simple model proposed by Newman [50] was used. It relies on the principle of the Erdős-Rényi model, the model generating networks with Poisson degree distribution [40]. It starts with 128 unconnected nodes, which are assigned to 4 communities such that each of them has same the number of nodes. Then, links are placed between the nodes with a probability p_{in} if the nodes are in same community, and p_{out} otherwise. These probabilities are chosen in a way which keeps an average degree close to 16 and $p_{out} < p_{in}$. This model was used by many authors to validate their community detection algorithms [23, 36, 50, 115]. However, the model has some drawback as all nodes have roughly the same degree, the community sizes are all the same, and the network is very small. Danon et al. [30] proposed a modified version generating networks with heterogeneous community sizes, which is more realistic. They also showed that the performance of community detection algorithms is affected when applied to those more realistic artificial networks.

Bagrow [7] propose a model which at first creates networks by using the Barabási-Albert preferential attachment model [9] which generates network with power-law degree distribution, and containing 512 nodes. Then, the network is split into 4 equal sized communities. The amount of links between the communities is controlled by a *rewiring* process which consists in moving the extra links lying between the communities into the communities, in order to reach the desired degree distribution. This model allows generating networks with a more realistic degree distribution, and to control the level of separation of the communities.

A more realistic model was proposed by Lancichinetti et al. [76], allowing to control the number of nodes, degree distribution, maximal and average degrees, community size distribution and separation of

the communities. This model at first generates networks with power-law degree distribution by using configuration model [90]. Then, it determines the communities whose sizes are distributed in power-law. At last, it rewires the links between the communities by respecting the degree distribution and a separation threshold specified under the form of a parameter. This model creates relatively more realistic networks. However, we have shown [101] that certain important topological properties depend on the value of the separation parameter. As a side effect, when one wants to change the difficulty of the problem by making the communities less clearly separated, he also affects the network realism, and therefore the community detection algorithms. To overcome this drawback, we proposed a modification to LFR by using the Barabási-Albert preferential attachment model [9] and one of its variants called Evolutionary preferential attachment [108], rather than configuration model, in the first step of the generating process. The results of this study showed that using Barabási-Albert model in LFR produces more stable networks, in terms of level of realism relatively to changes in the separation parameter. Our experiments showed changes in the level of realism affects the performance of certain community detection algorithms, which confirms previous studies such as [30].

For the algorithms using attribute information or dynamic networks, there is not yet a mature validation platform. In [69], the authors propose a dynamic network generator model to test their algorithm. This model is a modified version of Newman's model that we explain in the beginning of this section. They generate 10 networks to represent 10 different time slices. Then, to introduce the dynamics of time evolution, for every community, they randomly choose three nodes at a given time slice and change their position at the following time slice. This model indeed carries the same drawbacks than Newman's. Moreover, its time evolution dynamic is not realistic either. Greene et al. [52] propose another dynamic network generator model which is an inspiration of Lancichinetti et al.'s model. This model is more realistic than the previously explained one. We can directly adjust the number of time slices, the number of nodes whose position change, the number of communities undergoing events, and the nature of these events (birth, death, merge, split, expand, contract, hide, switch) thanks to model the parameters. We use this model in our experiments thus, we explain it in more details in Chapter 5. Although these models are proposed for dynamic networks, they are not used widely yet in the domain.

In fact, showing the consistency of an algorithm is rather subjective. That is why, it is important to compare the algorithms on an objective platform. The more the generative model is realistic, the more relevant the comparison. Furthermore, one should consider the widest information related to result set for the validation of the algorithms. Thus, the most appropriate validation method should take into account topological, attribute and evolutionary information of the found community structure.

3.4.2 Performance Assessment

The traditional method to assess the performance of community detection methods consists in considering a community structure as a partition of the node set. From this, the performance can be quantified by measuring the distance between the partition estimated by the considered algorithm and the reference partition (the one defined bu the generative model, if we consider artificial data, or the ground truth if we consider a real-world network). Several different measures exist for this purpose, most of them already used in the context of clustering on classic, tabular data (i.e. not graphs). The most commonly used are the *Fraction of Correctly Classified* [50], the *Rand Index* [113] and its chance-corrected version, the *Adjusted Rand Index* [65] , and the *Normalized Mutual Information* [31, 47].

The *Fraction of Correctly Classified* nodes (FCC) has been used by several authors in the context of community detection, the first seeming to be Girvan and Newman [50]. According to this measure, a node is correctly classified if its estimated community is the same than for the majority of nodes present in its reference community. Moreover, if an estimated community corresponds to a fusion of several reference communities, all the concerned nodes are considered as misclassified [31, 95]. The

total number of correctly classified nodes is divided by n to normalize the measure, which results in a value between 0 and 1.

The *Rand Index* (RI) [113] corresponds to the proportion of node pairs for which both the estimated and reference community structures agree. For a given pair, there is agreement when both nodes belong to the same community, or to different communities, for both community structures. Consequently, there is disagreement if the nodes are in the same community for one community structure, whereas they belong to two different ones for the other. The Rand Index ranges from 0 (the algorithm completely failed to estimate the community structure), to 1 (the algorithm perfectly estimated the community structure). The *Adjusted Rand Index* (ARI) is a corrected for chance version of the RI [65] ranging from -1 (less than chance agreement) to 1 (complete agreement). Zero corresponds to a pure chance agreement.

The *Normalized Mutual Information* measure (NMI) was defined in the context of classic clustering [47] to compare two different partitions of one data set, by measuring how much information they have in common.Danon et al. [31] used it to assess the performance of community detection algorithms, and the measure was subsequently used by various other authors [76]. If the estimated communities correspond perfectly to the reference ones, the measure takes the value 1 , whereas it is 0 when they are independent. NMI is the most widely used measure in performance evaluation in community detection algorithms. We give the formula of this measure in equation 3.3.

$$I = \frac{-2\sum_i \sum_j m_{ij} \log(n m_{ij}/m_{i+}m_{+j})}{\sum_i m_{i+} \log(m_{i+}/n) + \sum_j m_{+j} \log(m_{+j}/n)} \tag{3.3}$$

Let us now consider the case of richer networks. Works dealing with attributed networks, such as [27], use the same measures than those presented above, especially NMI and RI. Indeed, even when using attributes, it is possible to consider community structures as partitions, so the same principle applies. In the case of dynamic networks, the temporal dimension introduces an additional difficulty: communities evolve, so the performance of the algorithm might also evolve through time. The standard approach to deal with this case consists in considering the evolution of one of the cited measures. In [69, 138], for instance, each the community structure detected at each time slice is compared with the corresponding reference one by using the NMI. On the contrary, Aynaud and Guillaume [5] define a sort of edit distance they use as an alternative to the NMI . It corresponds to the minimum number of nodes whose community must be changed in order to convert one community structure into the other.

It is worth noticing all those measures have a limitation, though. Indeed, by considering community structures as partitions, they completely ignore the topological nature of the communities. As a consequence, as we showed empirically in our previous studies [102, 103], two community detection tools can reach the exact same level of performance, but still estimate community structures with very different link distributions. To solve this problem, we proposed to compare the estimated and reference community structures not only in terms of partitions, but also in terms of the topological properties of the communities. For this matter, we used some community-oriented topological measures recently defined in the literature [77, 80]: community-wise density, average distance and transitivity, hub dominance, and embeddedness. Our experiments showed using this approach in addition to a traditional performance measure allows to get a more relevant view of the relative quality of the considered algorithms. However, considering several measures also introduces more complexity in the evaluation process, and difficulties when interpreting the results and ranking the algorithms. In this thesis, we will need to perform such a comparison later, however it is not our main purpose. For this reason, we decided to stick to the standard approach, i.e. a single partition-oriented measure, for matters of simplicity. We selected the NMI, which is now the most widespread such tool.

3.5 Interpretation of the Communities

As mentioned in the previous sections, there are many community detection algorithms, which differ in terms of type of the processed network, nature of the detected communities, algorithmic complexity, result quality and other aspects [43]. Whatever the algorithm, though, its output can always be basically described as a list of node groups. More specifically, in the case of disjoint communities, it is a partition of the set of nodes. From an applicative point of view, the question is then to make sense of these groups relatively to the studied system. In other terms, for the community structure to be useful, it is necessary to interpret the detected communities. In this thesis, we view this problem as *independent* from the approach used for community detection: we suppose a partition of the node set has been identified, and one wants to take advantage of this information to improve his understanding of the system. Considering the interpretation problem is independent from detection problem lets us treat any community structure, which can be outputted by any detection algorithm. This is an important, point because if interpretation occurs during detection, the characterization of the communities might be limited by the nature of the detection algorithm. Moreover, the interpretation process does not necessarily have to use the same information than the detection process, nor use it in the same way. After all, the interpretation is supposed to be complementary of the detection itself.

This interpretation problem is extremely important from the end user's perspective. And yet, almost all works in the field of community detection concern the definition of detection tools, and their evaluation in terms of performance. Only a very few works try to tackle the problem of characterizing and interpreting the communities. In the rest of this section, we review and discuss the existing approaches. We distinguish five categories: manual methods, which are historically the first ones, methods based only on the network structure, methods relying on attributes and standard statistical tools, methods coupled with an attribute-based community detection process, and finally methods relying on frequent pattern mining principles.

3.5.1 Manual Interpretation

In early community detection works, the studied networks were very small compared to today's standard. The interpretation of the communities could therefore be done manually, i.e. by studying subjectively the individuals composing some community of interest, and trying to identify some relevant patterns or regularities in order to reach some observations considered useful to understand the studied system [50, 93]. When the scale of the networks increased from tens to hundreds, it was still possible to consult domain experts to perform interpretation in a similar fashion [111, 115, 116].

However, this method shows its limit on larger networks. Blondel et al. [15] applied their Louvain algorithm on a network of Belgian mobile phone communications, including 2.6 million nodes representing persons. Interpreting such a large network obviously requires a more automatic approach. Blondel et al. verified the accuracy of the top level of their hierarchical community structure by considering the homogeneity of the nodes on an attribute representing the language people speak. This highlights the difficulty of interpreting communities in networks of this size, and also shows one solution can be to consider some additional information, such as nodal attributes.

3.5.2 Community Topology

Sometimes, the only available data is the network structure, and a purely topological interpretation method is required, such as the one proposed by Lancichinetti et al. [77]. The authors made a valuable empirical analysis on a wide range of real-world networks from several domains, including information, communication, technological, biological, and social networks. Their approach is based on a visual analysis of the community size distributions, of how certain community-related measures evolve in function

of the community size, and of the distribution of certain nodal measures. Note those measures are the same we use in our previously mentioned work (cf. section 3.4), when evaluating community detection methods from a topological perspective: scaled link density, community-wise average distance, hub dominance and embeddedness. They take advantage of these observations to comment the size, density and possible shapes of the communities, e.g. star-like, tree-like, clique-like, etc. They conclude that networks from the same domain have similar community structure. For social networks, some important topological properties are that *"they are denser than trees but sparser than cliques, with the exception of small communities which appear more tree-like, hubs play the least dominant role in social networks and there are few or no dominant hubs in large communities"*.

Leskovec et al. [80] proposed another purely topological method, but their focus is more on the community structure in general, than on the communities taken individually. It is based on the notion of *network community plot*, which requires a specific process regarding how communities are detected, unlike the approach of Lancichinetti et al. Leskovec et al. use the *conductance* measure [123] to assess the quality of a single community. They select a community detection algorithm able of identifying the best communities (in terms of conductance) of a given size. They apply this tool repeatedly, for various community sizes. They then plot the obtained conductance in function of the community size, to get the network community plot. This plot is then visually examined to characterize the community structure of the network. Using this method, they made experimental analysis on 70 social and information networks. As a result, they observed that, as the communities grow in size, they *blend in* the network.

These two studies are important, because they allow studying the community structure of plain networks. However, it should be noted the resulting observations are quite general, in the sense communities are studied and characterized collectively, in order to identify trends in a network, or even a collection of networks.

3.5.3 Classic Statistical Tools

Certain works try to take advantage of the extra information available in attributed networks to help understanding the communities. Let us focus first in those where communities are detected on a purely topological basis, and attributes are used later to characterize them, using various statistical tools. Labatut and Balasque [73] study an attributed social network of university students. They first detect communities using FastGreedy [95], and then use some relatively standard purely topological measures such as embeddedness and *Amaral&Guimerà*'s roles [59], but also attribute-based measures such as homophily. More interestingly, they also use statistical tools like regression and linear discriminant analysis to characterize the communities by their most distinctive attributes. They use their results to describe communities in general, but also to identify outlier students of particular interest. However, they make the assumption that the most discriminant attributes are the same for all communities.

Tumminello et al. [133] introduced a characterization method based on the identification of the communities most significant attributes. They use a quantitative statistical method to detect attributes which are not statistically consistent with a null hypothesis of random occurrence over all the elements of the system. They called them *over-expressed* attributes. According to this method, over-expressed attributes are not necessarily the most frequent ones, but their frequency in the community should be significantly larger than the frequency over the whole network. They experiment their analysis on two networks. The first one is the IMDB network, where nodes are movies, with attributes production country and language, and links connect two nodes if at least one actor played in both corresponding movies. The second network is a co-authorship network, where nodes are academic articles, with attributes journal and topic, and links connect two nodes if at least one author is an author for both of them. Communities are detected with the InfoMap (c.f. Section 3.1) algorithm, which uses only the topological information. The results show that both medium and large communities can be characterized by over-expressed attributes.

3.5.4 Attribute-Based Community Detection

As explained in section 3.2, certain community detection methods are specifically designed to use nodal attributes (and topological information), so it seems natural to suppose some by-product of their processing can be used in some way to easy the interpretation of communities.

With certain approaches, it is possible to obtain the most characteristic attribute values of a community. For instance, the works of [8, 22, 83] are dedicated to find out the topics of documents. This methods output not only a community structure, but also some keywords corresponding to the main topics of the communities. Zhou et al. [140] find communities in a DBLP network of co-authors, using two attributes representing how prolific an author is and his primary topic of interest. The interpretation is then done using the same two attributes, in particular with the most common primary topic of the community. It is worth noticing that the methods presented in the previous subsection are aiming at identifying the most relevant attributes to describe communities, whereas here the focus is on characteristic attribute values. Indeed these works, there are only a few attributes, which makes the attribute selection question irrelevant. They would not apply on datasets containing many attributes, though.

On the contrary, in [137], the focus is on attribute selection. The authors define a model in which a weight is assigned to each attribute. During the detection process, a weight is increased if the corresponding attribute allows to improve the detection. Thus, for each community, their tool measures the most important attributes.

Attribute based methods propose more advantages than purely topological methods for the interpretation. But, the drawback of this type of interpretation is the lack of coordination of found thematic attributes with the topology of the communities. Only few authors make effort to comment the significant attributes of each community with very restricted information related to topology. For example; in [1], authors contribute to the interpretation by commenting the found attributes with the sizes of the communities.

3.5.5 Frequent Patterns

A method was recently defined based on frequent pattern mining, which can be used for community interpretation. Stattner and Collard [125] introduced the notion of *frequent conceptual link*. A conceptual link corresponds to set of links connecting nodes sharing similar attributes. A frequent conceptual link is a conceptual link whose frequency among all links of the network is above a given threshold. This method can be seen as a generalization of the notion of homophily, and was initially used to simplify the network and help understanding it. Finding frequent conceptual links corresponds to detect groups of nodes sharing common attributes, with a pattern mining point of view. Stattner and Collard [126] detect communities by applying the Louvain algorithm (cf. section 3.1.3), and independently mine frequent conceptual links in the same network. They then propose a set of homogeneity measures to quantify the similarity between these two node groupings. The approach is not unlike that adopted in [74] to compare communities and clusters. They interpret the structural communities by studying how the conceptual links are distributed among them. Their results show that some communities are homogeneous in their attributes while some others are not, which is consistent with the finding from [73]. It also supports the argument we made against attribute-based community detection methods, regarding their homophily assumption not being valid for all real-world systems.

3.5.6 Conclusion

As mentioned before, although the interpretation of communities is an important problem and a task complementary to community detection itself, only a few works tried to address it. Authors historically interpreted their data manually, but this subjective approach cannot be applied on large networks. Some

methods focus on the structural information only, but they are limited in the observations they allow to make. When nodal attributes are available in the considered data, using them seems to improve the quality of the interpretation. Some authors used classic statistical tools such as discriminant analysis to characterize the communities, but they considered only the attributes: the only topological information used is community membership itself. Certain attribute-based community detection tools output some results one can use to characterize communities. Since both the network structure and the node attributes are used when detecting such communities, this interpretation method can be considered as using both types of information. However, it is not always clear which topological information is used exactly, because the notion of community is not clearly defined in this context. Moreover, we consider the interpretation problem as separated from the community detection task. Finally, some authors proposed to use data mining methods and characterize communities in terms of frequent patterns, considering both links and attributes. However, the fact they use only direct connections can be considered as a limitation.

In the rest of this thesis, we propose a method to improve community interpretation by focusing on two points. First, we consider *time*. This means our method will be appropriate for static networks. However, we hope the additional information corresponding to this temporal dimension will allow us to enhance the description of the communities the same way attributes did when compared to purely topological methods. Moreover, a complex network represent by definition an evolving system, so it is relevant to take into consideration the evolution of both topology and attributes in a systematic way. Second, we take advantage of *both* individual (attributes) and relational (structure) information in our method. Moreover, we want to allow considering more general structural data than the plain direct connections, and more advanced measures than the simple homophily. We also want the used topological information to be clearly and explicitly identified, in order to ease interpretation. In the next chapter, we describe in details how we combined those different elements to define our community interpretation method.

Chapter 4

Community Interpretation

As explained in the previous chapter, a community structure by itself is just a partition of the node set. One cannot take advantage of it without first interpreting its communities. This allows, for instance, identifying differences or similarities between them, or understanding their role in the network. Some authors tried to deal with the problem, but their approaches suffer from various limitations: some are subjective, others focus on nodal attributes only, or on topological properties only, or mix these data in an unclear way. Moreover, in most of these works, the problem of community interpretation is not defined as a problem in itself. We dedicate this chapter to the definition and formalization of the problem of community interpretation, and to the proposition of a solution for this problem.

We see this problem as, on the one hand, representing a community in an appropriate way, and on the other hand, finding the most characterizing elements from this representation. We propose to represent the communities under the form of sequences of nodal attributes and topological measures, therefore using the richest information available. It is then possible to detect common changes in attributes and topological features over time periods, through the identification of frequent and emerging sequential patterns of node attribute values and topological measures. These patterns can be used for both interpreting the communities, and identify their outliers (i.e. nodes with non-standard behavior). The frequent/emerging patterns represent the general trend of nodes in the considered community, whereas the outliers correspond to nodes with a specific role in the community, or located at the fringe of the community.

In the following section, we first define the fundamental concepts required to give our explanations. Second, we state and formalize the problem. Third, we describe a framework implementing our solution. It is constituted of different steps related to constituting the community structure representation, and taking advantage of it to extract valuable knowledge. Fourth, we explain how the result characteristic patterns are interpreted for the communities. Fifth, we give an example of how our framework is working on a toy problem, and explain the obtained results. Finally, we study the algorithmic complexity of our method.

4.1 Definitions

We dedicate this section to explaining the preliminary concepts necessary for the statement of our problem and the description of our framework. The main idea we use is to represent the community-related information under the form of sequences. Thus, we separate the definitions in two parts. The first one gathers all network-related concepts, whereas the second part includes all the notions we need concerning pattern mining.

4.1.1 Network-Related Concepts

We formally define a ***dynamic attributed network*** as $G = \langle G_1, \ldots, G_\theta \rangle$, i.e. a sequence of chronologically ordered graphs $G_t (1 \leq t \leq \theta)$, which we call time slices. A time slice $G_t = (V, E_t, A)$ is a triple such that V is the set of nodes, $E_t \subseteq V \times V$is the set of links and A is the set of node attributes. The nodes are the same for all time slices, and we can consequently note $\mid V \mid = n$ the size of each time slice, as well as the dynamic network. Let us define $f_t(w,v) = \begin{cases} 1, & \text{if } (w,v) \in E_t \\ 0, & \text{if } (w,v) \notin E_t \end{cases}$ the adjacency function, which determines if there is a link between nodes w and v at time t. We only deal with undirected graphs, so we consider all pairs (w,v) to be unordered.

What we call the ***integrated weighted network*** $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ associated to a dynamic network G, is a pair such that $\mathscr{E} = \bigcup_{t=1}^{\theta} E_t$. Moreover, the weight of a link (w,v) of $\mathscr{E}$ is defined with the following weight function: $f(w,v) = \sum_{1 \leq t \leq \theta} f_t(w,v)$ Suppose the links of each time slice are represented under the form of lists, as it is often the case, and the links of the integrated network are represented using a hash table associating an integer to a link (representing its weight). The hash table is originally empty. Then, for a given time slice t, we update the hash table by considering iteratively all l links of G_t: if the link is already in the hash table, then its associated integer is incremented, otherwise it is inserted with value 1. The same process must be repeated for all θ time slices. In the worst case, the access time for a list or hash table is linear to the number of elements it contains. So, the total complexity for the processing of $\mathscr{G}$ is in $O(l^2\theta)$.

A ***topological measure*** quantifies the structural properties of the network or its components. Here, we focus on nine nodal measures: internal degree, local transitivity, eccentricity, betweenness centrality, closeness centrality, eigenvector centrality, within module degree, participation coefficient and embeddedness. The detailed description of these measures is given in chapter 2. Each one is processed for each node, at each time slice.

A ***node descriptor*** is either a topological measure or a node attribute from A. Let $D = \{D_1, D_2, \cdots, D_k\}$ be the set of all descriptors. Each descriptor from D can take one of several discrete values, defined in its domain $\mathscr{D}_i (1 \leq i \leq k)$. All our topological measures are real-valued, so we have to discrete them to fit this definition. Moreover, the same apply to real-valued attributes. This case is considered in section 4.3.3.

We consider a ***consensual community structure*** $\mathscr{C}$ of a dynamic attributed network G to be a single partition of its node set. We call it consensual, because the same community structure is supposed to be (relatively) relevant for all time slices of the dynamic network. A community corresponds to a part of this partition. We note the communities $C^c (1 \leq c \leq \lambda)$, for λ distinct communities. Let us define $C(v)$, a function associating a node v to its community in $\mathscr{C}$. The ***community size*** of a given community C^c is $|C^c|$, i.e. the number of nodes it contains.

An ***evolving community structure*** $\mathscr{C}^s = \langle \mathscr{C}_1, \cdots, \mathscr{C}_\theta \rangle$ of a dynamic attributed network G is a sequence of chronologically ordered community structures $\mathscr{C}_t (1 \leq t \leq \theta)$, where each $\mathscr{C}_t$ corresponds to the community structure of time slice G_t. In this context, $C_t(v)$ is the function associating a node v to its community in $\mathscr{C}_t$; and the size of a given community C_t^c is its number of nodes $|C_t^c|$.

4.1.2 Pattern-Related Concepts

An ***item*** $(D_i, x) \in D \times \mathscr{D}_i$ is a couple constituted of a descriptor D_i and a value x from its domain $\mathscr{D}_i$. The set of all items is noted I. An ***itemset*** h is any subset of I. Although an itemset is a set, in the rest of this thesis we represent them between parentheses, e.g. (l_1, l_3, l_4) because it is the standard notation in the literature.

A ***sequence*** $s = \langle h_1, \cdots, h_m \rangle$ is a chronologically ordered list of itemsets. Two itemsets can be consecutive in the sequence while not correspond to consecutive time slices: the important point is that

the first to appear must be associated to a time slice preceding that of the second one. In other words, h_t occurs before h_{t+1} and after h_{t-1}. The ***size*** of a sequence is the number of itemsets it contains.

A sequence $\alpha = \langle a_1, \ldots, a_\mu \rangle$ is a ***sub-sequence*** of another sequence $\beta = \langle b_1, \ldots, b_\nu \rangle$ iff $\exists i_1, i_2, \ldots, i_\mu$ such that $1 \leq i_1 < i_2 < \ldots < i_\mu \leq \nu$ and $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_\mu \subseteq b_{i_\mu}$. This is noted $\alpha \sqsubseteq \beta$. It is also said that β is a ***super-sequence*** of α,which is noted $\beta \sqsupseteq \alpha$.

The ***node sequence*** for a node v is a specific type of sequence of size θ. We use the notation $u(v)$ to represent node sequence of node v. $u(v) = \langle (l_{11}, \cdots, l_{k1}) \cdots (l_{1\theta}, \cdots, l_{k\theta}) \rangle$ where l_{it} is the item containing the value of descriptor D_i for v at time t. A node sequence $u(v)$ includes θ itemsets, i.e. it represents all time slices. Each one of these itemsets contains all k descriptor values for the considered node at the considered time. In other words, $u(v)$ contains all the available descriptor-related data for node v.

The ***extended node sequence*** $u_{ext}(v)$ of a node v is its node sequence including also the ID of the community or communities containing v. We define two different such extensions, to match the cases of consensual and evolving community structures. Let us consider first the case of a consensual community structure $\mathscr{C}$. We use the approach described in [107] to handle the case where classes are assigned to item sequences. Indeed, in our case, we can consider communities as classes. Let us note $\alpha \bullet F$ the *concatenation* of a sequence $\alpha = \langle a_1, \ldots, a_\mu \rangle$ and a symbol F, such that $\alpha \bullet F = \langle a_1, \ldots, a_\mu, \{F\} \rangle$. We define the ***concatenated node sequence*** of a node v as the concatenation of its node sequence and of its community ID, i.e. $u_{con}(v) = u(v) \bullet C(v) = \langle (l_{11}, \cdots, l_{k1}) \cdots (l_{1\theta}, \cdots, l_{k\theta}), C(v) \rangle$, where h_t is the itemset of $u(v)$ at time t. By abusing the notation, the $C(v)$ represent here the communities ID, and not the communities themselves. This remark holds for the rest of this section. Let us now consider the case of evolving community structures, i.e. community structures represented as sequences $\mathscr{C}^s$. This time, we add the community IDs of a node v, for each time t, to the itemset h_t describing v.However, depending on how the community structure was detected, the same ID can appear at different time slices to refer to different communities. For instance, community #1 at t=1 and community #1 at t=2 might be the same. Yet, it is important to be able to distinguish them. To overcome this problem, we concatenate the corresponding time slice index to the community ID, using @ as a separator. For example, community #2 found at t=3 will be represented by $C2@3$. Thus, the extended sequence of v is $u_{enl}(v) = \langle (l_{11}, \cdots, l_{k1}, C_1(v)@1) \cdots (l_{1\theta}, \cdots, l_{k\theta}, C_\theta(v)@\theta) \rangle$. We call this type of extension ***enlarged node sequence***. In the rest of this thesis, $u_{con}(v)$ will be used to denote the ***concatenated node sequence*** of node v and $u_{enl}(v)$ will be used to denote its ***enlarged node sequence***.

What we call ***raw database*** M is the collection of the node sequences $u(v)$ of all the nodes in the considered network. An ***extended database*** M_{ext}, for a dynamic attributed network G, is the collection of extended node sequences (either concatenated node sequences or enlarged node sequence, depending on the considered community structure). We note M_{con} the collection of concatenated node sequences, and M_{enl} that of enlarged node sequences.

The set of ***supporting nodes*** $S(s)$ of a sequence s is defined as $S(s) = \{v \in V : u(v) \sqsupseteq s\}$. The support of a sequence s, $Sup(s) = \frac{|S(s)|}{n}$, is the proportion of nodes, in G, whose node sequences are equal to s, or super-sequences of s.

The set of ***supporting nodes*** of a sequence s ***relatively*** to a node group X is defined as $S(s,X) = \{v \in X : u(v) \sqsupseteq s\}$. Its support relatively to the same node group, $Sup(s,X) = \frac{|S(s,X)|}{|X|}$, is the proportion of nodes, in X, whose node sequences are equal to s, or super-sequence of s.

The growth rate of a pattern s relatively to a node group X is $Gr(s,X) = \frac{Sup(s,X)}{Sup(s,\overline{X})}$. Here $\overline{X}$ is the complement of X in V, i.e. $X = V \setminus \overline{X}$. The growth rate measures the emergence of s. It is the ratio of the support of s in X to the support of s in $\overline{X}$. Thus, a value larger than 1 means s is particularly frequent (i.e. emerging) in X, when compared to the rest of the network. Note that here, a node group might directly correspond to a community (in the case of a consensual community structure), or to the nodes belonging to the same community over several time slices (in the case of an evolving community structure).

We say a sequence is ***community-related*** if there is at least one community ID among its items.

If its items contain only community IDs, then we say it is a ***community sequence*** $\langle C_{t_1}^{c_1}, C_{t_2}^{c_2}, \cdots, C_{t_m}^{c_m} \rangle$. On the contrary, we call it ***community-independent*** if there is no community ID at all among its items. For a community-related sequence s, we define its ***community-wise sub-sequence*** s_{wise} as its maximal community subsequence. In other words, it is a sequence $s_{wise} \sqsubseteq s$ such that s_{wise} is a community sequence and there is no other community sequence b fulfilling both conditions $b \sqsubseteq s$ and $s_{wise} \sqsubset b$. Similarly, for a community-related sequence s, we define its ***community-less sub-sequence*** s_{less} as its maximal community-independent subsequence. In other words, it is a sequence $s_{less} \sqsubseteq s$ such that s_{less} is a community-independent sequence and there is no other community-independent sequence b fulfilling both conditions $b \sqsubseteq s$ and $s_{less} \sqsubset b$.

Given a minimum support threshold noted min_{sup} , a ***frequent sequential pattern (FS)*** is a sequence whose support is greater or equal to min_{sup}. A ***closed frequent sequential pattern (CFS)*** is a FS which has no super-sequence possessing the same support.

4.2 Problem Statement

In this section, we state our problem in details and give its formalization. We see the problem of *community interpretation* as the operation consisting in identifying the most *characteristic features* of certain groups of nodes. But *what* is a feature? And *how* can we know if it is a characteristic one? To be able to solve the interpretation problem, we need first to answer these two questions. In other words, our problem of interest can be broken down to two sub-problems:

1. *Finding an appropriate way to represent a community;*
2. *Defining an objective method to decide which parts of this representation are characteristic.*

In this section, we consider separately these two sub-problems and formalize them.

4.2.1 Appropriate Community Representation

It is obviously not possible to know in advance which pieces of the information describing the considered community will be the most characteristic. Therefore, we need to be able to represent all the available information, in a computationally efficient way. A community can be described only in terms of its constituting elements, since it is by definition a set, so its representation must be defined at the nodal level. Moreover, as mentioned in chapter 3, a community can be defined in terms of node similarity, interconnection, or coevolution, so it is necessary to use a representation able to handle these different types of information. More formally, this means a community must be represented through its nodes attributes, topological properties and temporal evolution. The attributes correspond to the individual characteristic of the objects composing the modeled complex system; the topological properties describe how these objects interact; and the temporal evolution is the consequence of the system dynamics.

> *In the context of community interpretation, the problem of* ***finding an appropriate community representation****, for a dynamic attributed network and its community structure, is equivalent to that of encoding all information describing the evolution of each node from each community. This encoding should be compact enough to avoid redundancies, but complete enough to describe the evolution of the community.*

To fulfill these constraints, we propose to represent each node using the *sequence* of its attribute and topological measures, taken at different discrete times of the system evolution. We could use the extended database M_{ext} described in the previous section, which is the collection of extended node sequences $u_{ext}(v) \forall v \in V$. However, this representation would not be very compact, because several nodes could be

described (totally or partially) by similar sequences. To avoid this, we instead represent a community through the sequential patterns present among the sequences describing the nodes it contains.

For a node v, we note P_v the set of all possible sub-sequences of its extended node sequence, i.e. $P_v = \{s \sqsubseteq u_{ext}(v)\}$. Let P be the union of all the P_v over all nodes, i.e. $P = \bigcup_{v \in V} P_v$, and B be the subset of its community sequences. Let m_i denote such a community sequence and μ be the cardinality of B, then we have $B = \{m_1, \cdots, m_\mu\}$. We state the problem of community representation is finding a set $\Gamma = \{\gamma_1, \cdots, \gamma_\mu\}$ such that $\gamma_i = \{s \in P : s \sqsupseteq m_i\}$ with $i \in [1, \mu]$. In other words, for each community sequence found in the extended database, we want to identify the set of all its super-sequences present in the same database. Those, by definition, are themselves community-related sequences. Note that, when using the concatenated node sequences $u_{con}(v)$, the m_i terms are singletons $\{C^i\}$, and the cardinality of Γ therefore corresponds to the number of communities in the consensual community structure, i.e. $|\Gamma| = \lambda$.

The set Γ includes all the necessary information related to each community sequence. It represents the sub-sequences common to more than one node only once, and therefore eliminates the redundancies. Consequently, it fulfills our requirements of being both a complete and compact representation.

4.2.2 Identifying Characteristic Features

Let us now go back to the second problem, which is to find, in an objective way, which parts of the community description are characteristic. The criteria used to identify this relevance must be compatible with our representation of a community, which takes the form of sets of sequential patterns.

> *In the context of community interpretation, the problem of* ***identifying characteristic features*** *amongst the sequences representing a community consists in selecting some objective criteria to assess the representative power of these sequences, and a method to select the most representative ones.*

The fact our data takes the form of sequential patterns is pointing us to the task of mining sequential patterns, which is a specialization of pattern mining. Such approach is generally used to identify the most frequent and/or emergent behaviors in a whole dataset, and interpret them as domain knowledge. It has been used for the analysis of non-network data before [85], but not for networks, at least to our knowledge. However, it is appropriate for our representation and goals, provided we apply it on single communities (by opposition to the whole network).

The representation defined in the previous section takes the form of a set $\Gamma = \{\gamma_1, \cdots, \gamma_\mu\}$, where each γ_i represents the set of community-related super-sequences of a community sequence m_i (as of the considered extended database). We put one condition and two criteria for a member of γ_i to be characteristic of m_i. The condition is that it must be *informative*. We consider a sequence to be informative if it is closed. The two criteria are that the sequence must be both *prevalent* and *distinctive*. We measure the prevalence of a sequence s in γ_i with its support $Sup(s, S(m_i))$, where $S(m_i)$ is the supporting node set of the reference community sequence m_i. We measure the distinctiveness through the growth rate $Gr(s, S(m_i))$.

Then, the problem of identifying the characteristic features of a community sequence m_i consists in selecting a subset $\gamma_i' \subseteq \gamma_i$ such that $\gamma_i' = \{s \in \gamma_i : s \text{ is closed} \wedge Sup(s, S(m_i)) \geq min_{sup} \wedge Gr(s, S(m_i)) \geq min_{gr}\}$, where min_{sup} and min_{gr} are lower limits for the support and growth rate, respectively. Note that in the case of a consensual community structure, $Sup(s, S(m_i)) = Sup(s, C^i)$ and $Gr(s, S(m_i)) = Gr(s, C^i)$. We note $\Gamma' = \{\gamma_1', \cdots, \gamma_\mu'\}$ the set containing the sets of characteristic patterns for each community sequence in the network.

4.3 Framework

We propose a framework to solve the problems we defined in section 4.2. We start describing our framework by first giving an overall description of its structure, and then we detail its constituting steps. For clarity matters, we assigned a number (1, 5, 5-A, etc.) to the different steps and sub-steps of the process. In the subsequent parts, we will use these codes when referring to a specific part of the process.

4.3.1 General Description

Our framework includes five steps to interpret the communities. We give a schematic description of the framework in Figure 4.1. In this diagram, we can notice the relation between the different steps: each step uses the output of the previous one(s).

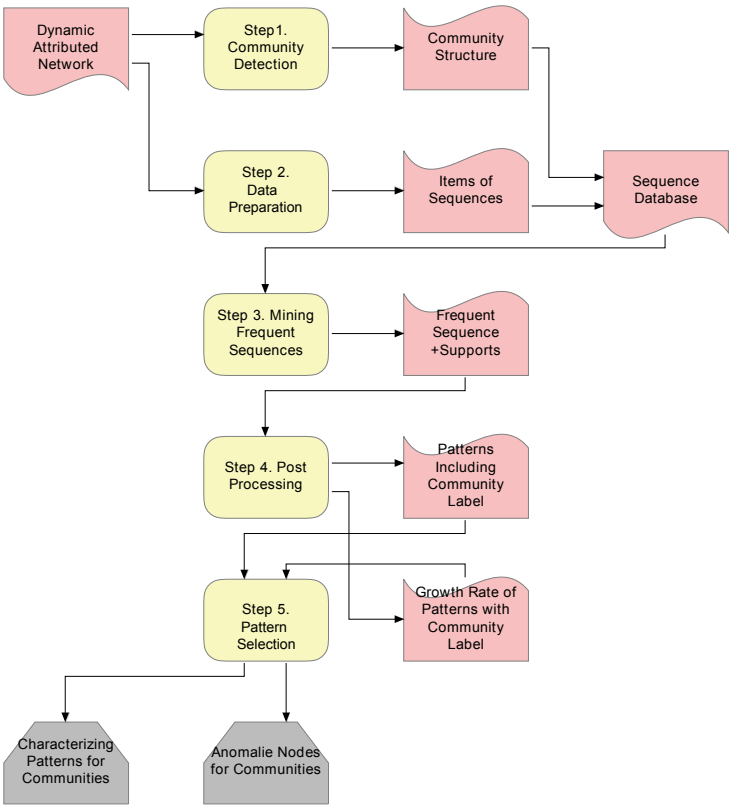

Fig. 4.1 Flow Diagram of the Community Characterization Framework

In the *first step*; we detect the communities of the dynamic attributed network. Our method is flexible enough to be applied to any community structure taking the form of a partition (consensual community structure), or a series of partitions (evolving community structure) of the node set. In other words, it is possible to use classic methods presented in section 3.1, attribute-based methods from section 3.2 and dynamic methods from section 3.3. In this work, we decided to use four different modularity-based algorithms differing by the way they consider the network dynamics.

In the *second step*, we prepare the data that we use for further analysis and for the creation of the sequence database. This data includes the attributes, topological measures and community ID of all the nodes.

In the *third step*, we find the frequent sequential patterns in the network. For this purpose, we use the ***CloSpan*** algorithm [136], which was originally designed for raw data (i.e. not for networks). We use the

community sequences and community-related sequences for further analysis.

In the *fourth step*, we measure the relevance of the community-related patterns. The most relevant patterns are the ones we use to interpret the community sequence. This step requires a post-processing in order to measure the relevance of each pattern.

In the *fifth (and last) step*, we look at the cover of the significant pattern(s) in terms of their supporting nodes, in order to cover a maximum number of the community nodes. Thus, we choose the patterns best covering the community. Moreover, we also identify anomaly nodes that are significantly different from their communities.

The first two steps can be considered as pre-processing, relatively to our two sub-problems (community representation and characteristic features identification). The last three steps are directly related to solving these problems. It is important to highlight the fact the method we use in step 3 (CloSpan) allows us to bypass several of the intermediary points we developed in the problem statement. Indeed, this tool takes the extended database M_{ext} as an input, and directly outputs the frequent and closed patterns. Therefore, it is not necessary to explicitly handle the compact representation Γ.

4.3.2 Step 1: Detecting Communities

We interpret the communities independently from the method used for their detection. But of course, we need a reference community structure to work with it. Here, we propose to use four different community detection methods to deal with the dynamic network. They are all different versions of the Louvain algorithm [15]. Two of these methods result in a consensual community structure defined for the whole time range, whereas the two others identify an evolving community structure, i.e. a sequence of community structures, each one describing a single time slice in a chronological order.

We call the first method *Classic Louvain* [15]. We apply the regular Louvain method, without any change, to each time slice independently. This approach produces an evolving community structure, but it does not consider temporal smoothness (cf. section 3.3). Note this naïve approach has been used before as a baseline in [5].

The second method is *Incremental Louvain* [5]. It takes into account temporal smoothness for time evolution. For the first time slice, the original version of Louvain is applied. For the following time slices, the algorithm starts with the community structure found at the previous time slice (rather than putting each node in its own community), and the rest of the detection is the same than with the original Louvain.

The third method is *Multistep Louvain* [6]. It considers temporal smoothness by trying to maximizing the modularity averaged over all time slices. As a result, it produces a consensual community structure which is supposed to be relevant for all time slices.

Finally, the last method is *Integrated Louvain*. This time, we apply the original Louvain method on the integrated weighted network $\mathscr{G}$ created from the dynamic network $G = \langle G_1, \ldots, G_\theta \rangle$. Here, we use the weight of the links as the power of the connection depending on time. The longer two nodes are linked, the stronger the connection between them in the integrated network. Thus, they should more probably be in the same community than the ones which do not have such powerful connections. Integrated Louvain produces one community structure, supposedly relevant for all time slices. It was our hypothesis to use an integrated network. However, when we surveyed the literature, we realized Aynaud and Guillaume [6] also used the same approach (but calling it *sum-method*).

4.3.3 Step 2: Data Preparation

As mentioned before, the descriptors used to fill our sequence database are nodal attributes and topological measures. Topological measures are numerical by definition (either real like local transitivity, or

integer like degree). Attributes can be of any data type, including numerical values as well. It is not possible to directly use real values for sequential pattern mining, because this method handles only discrete data. So, it is necessary to *discretize* this type of descriptor. Moreover, even for integer descriptors, it can be interesting to *bin* their domains (i.e. to decrease the cardinality of the domain), for several reasons. First, having too many distinct values in the domain of a descriptor tends to significantly increase the number of detected patterns while decreasing their support, thereby preventing to uncover results which would be sufficiently general to be informative. Second, due to the increase in the number of patterns, both processing time and memory occupation also significantly increase during pattern mining. Three, two distinct descriptor values can have meaning so close that they can be considered as similar without significant information loss. Finally, also for computational reasons, it can be appropriate to go beyond discretization and binning, and to *combine* several descriptors into one. This is particularly true for descriptors representing distinct aspects of the same features, such as centrality measures.

Whatever the preparation method (discretizing or binning), it is necessary to define thresholds. There are two ways of determining these thresholds: either using some expertise regarding the system or measures, or using an automatic approach, for instance when no expertise is available, or the expertise is not reliable enough. Various automatic approaches exist. Let us first consider the case where we want to keep each descriptor separated (i.e. we do not want to combine them or reduce their number). One method then consists in studying the distribution of the values for the descriptor of interest. The thresholds for discretization or binning can then be decided by considering the zones of the domain where the values are denser. If the values fit a known distribution, such as power-law or normal, the thresholds can also be selected depending on the properties of this distribution.

Another method consists in applying a clustering algorithm. This approach allows determining automatically the thresholds, but also combining descriptors; which in turn might decrease the computational load, as explained before. This approach was used before in the context of complex networks analysis by Dugué et al. [37], to study the topological roles of nodes. *Duguè et al*. defined some variants of Amaral & Guimera's measures (the original versions are described in section 2.3. They then applied the k-means algorithm [62] to group nodes and automatically identify the different community roles present in the network. We propose to generalize this idea to all descriptors, if necessary, including attributes. Of course, it is possible to combine certain descriptors while keeping the other as they are, or to consider several groups of measures to be combined. Therefore, the first step of this preparation consists in determining which groups of measures one wants to combine. The clustering method can then be applied on each group separately. The descriptors constituting a group are then replaced by a single, new, composite descriptor, whose values correspond to the identified clusters.

In our case, we selected the k-means algorithm too (like *Duguè et al*.), because it is a well-known and fast method. It partitions the considered data into k groups, such that the sum of the distances between each instance and the closest cluster center is minimized. In order to decide the number of clusters (parameter k), several measures exist to quantify the quality of the clusters, in terms of internal cohesion and external separation. We chose to use average Silhouette width [117], whose interpretation is clearly defined. It ranges from -1 to 1: the higher the average Silhouette width, the better the clustering. Moreover, this measure offers a visual support helping the decision making process. In this graphical representation, each cluster is represented in a way to show which instances are located between the clusters or clearly inside a cluster.

Some general remarks can be drawn regarding the topological measures we selected for this work. It is possible to distinguish three groups of thematically related measures. First, we propose to cluster together all three measures aiming at describing the position of nodes relatively to their community: embeddedness, z-score and participation coefficient. This means one single descriptor will represent all three measures at once. Here, the clustering allows both discretizing those real-valued measures, but also combining them. Second, we consider all three centrality measures (c.f. section 2.3) as well as the

eccentricity to be different aspects of the same concept regarding the position of the node in the whole network, so it seems relevant to also group them to apply the clustering process. Third, if necessary, we propose to group the two remaining measures (degree and local transitivity) together, since they both describe the local connectivity of the nodes.

4.3.4 Step 3: Mining Frequent Sequences

In this study, we used the algorithm ***CloSpan*** (Closed Sequential Pattern Mining), developed by Yan et al. [136], to find out all possible CFS for a given min_{sup}. CloSpan is an efficient algorithm which is able to mine long sequences in practical time for real-world data. It relies on the mining strategy introduced in ***PrefixSpan*** (Prefix-Projected Sequential pattern mining) [106]. However, unlike PrefixSpan, CloSpan mines closed sequences and it applies some extra pruning techniques to decrease the size of the search space. In the beginning, the algorithm preprocesses the sequence database by sorting every itemset and removing infrequent items or empty sequences. Then, a two-stepped main procedure is recursively applied to perform depth-first search, starting from the sequences with one item.

At the first step, the algorithm creates the candidate set, which is a super-set of closed frequent sequences. These candidates are stored into the so-called *prefix sequence lattice*. This lattice is a data structure designed to reduce the pruning space. It is based on the notion of *prefix sequence tree*: a tree representing the sequences in incremental lexicographic order, from its root to its leaves. The prefix sequence lattice is a modified version of this tree, which has shortcut links between different nodes of the tree Here, for every sequence s to be treated, it checks if there is any sequence s' already treated which is a sub- or a super-sequence of s with the same support. If so, it is not necessary to create a new node in the prefix sequence lattice for s. Only the links of the lattice are modified. If not, s is integrated to the lattice. As the output of first step, all possible candidates are produced and registered in the prefix sequence lattice.

At the second step, the non-closed sequences are eliminated. A naïve approach consists in checking, for all candidate sequences in the prefix sequence lattice, if there is any super-sequence with the same support. But it is a costly operation. Thus, Yan et al. adopt the fast subsumption checking algorithm introduced by Zaki and Hsiao [139]. It is designed to manage a hash table in which, for each sequence s, the associated hash key is the sum of the corresponding sequences ids. Here, the corresponding sequences of a sequence s refer to all sequences with prefix s, i.e. the members of its *projected database* (c.f. [106, 136] for explication and formalization of *projected database* of a sequence).

In [82], the authors claim the time complexity of the last step of CloSpan (pruning prefix sequence lattice) is in $O(n^2)$ where n is the size of the data (in our case, it is the number of nodes), if the maximum length of the frequent sequences is constrained by a constant. It outputs both the sequences and their supports, but not the supporting node sets, so an additional processing is required to identify them.

4.3.5 Step 4: Emergence and Post-Processing

In our case, we want to characterize communities in terms of CFS. Thus, we need to identify the most distinctive sequential pattern(s). For this purpose, we turn to the notion of emerging pattern, i.e. a pattern more frequent in a part of the network than in the rest of it. We measure the emergence of a pattern with its Growth Rate. The higher the growth rate, and the more representative a sequence for a community.

In order to calculate the growth rate, it would be necessary to search CFS for all community sequences separately, which can be a costly operation. But a more efficient way exist, which we will describe in this section. As mentioned before (section 4.3.2), we can detect two kinds of community structure in a dynamic network: a consensual community structure (using the Integrated and Multistep versions of Louvain) or an evolving community structure (Classic and Incremental versions of Louvain). For a consensual community structure, it is possible to associate a node to a single community for all

time slices; however, it is not the case for an evolving community structure. Thus, this leads to two different processes. We consider the case of consensual community structures first, as it is simpler. Then, we address the case of the evolving ones.

4.3.5.1 Step 4.A: Consensual Community Structure

As mentioned before, we use the method originally designed to process classes of non-network data in [107]. It relies on the extended database presented in section 4.1.2. After having identified the frequent sequences by applying CloSpan to M_{con}, the patterns concerning a community of interest C^c can be obtained simply by selecting all the CFS ending with C^c. The support of such CFS (of the form $\bullet C^c$) in M_{con} corresponds to the support one would have obtained on the original database M. Thus, applying CloSpan only once on M_{con} is enough to find the patterns related to each community. We do not need to search CFS for each community separately. Once we find out the patterns related to each community, we can process their support on the whole network through a simple request. Hence, all the information needed to calculate Gr is provided when applying CloSpan to M_{con}.

Processing the growth rate of all CFS relatively to the communities then requires separating the CFS depending on how they end: with or without a community ID. In other words, for a given *community-independent pattern* (i.e. it contains only descriptors), we can have several *community-related patterns* (i.e. containing at least a community ID) corresponding to the same sequence, with an additional community ID at the end. The support of a community-related pattern gives us the support of the corresponding community-independent pattern in the considered community. The support of the community-independent pattern gives us its support for the whole network. The network and community sizes are already known. It is possible to the CFS do not contain the community-independent pattern associated to certain community-related patterns. But in this case, this means the community-independent pattern is not closed, and has the same support than the community-related ones. Thus, we have all the necessary information to process the growth rate. The pseudo code of this procedure is given in 2. In the 6th line, the function `processSupport(···)` searches again the CFS to find the support of the community-independent patterns.

Algorithm 2 Extraction of Emerging Sequences from M_{con}

Require: $CFSset$, n, $communitySizes[\cdots]$
Ensure: $ConcatenatedCFS$, $theirSup$, $theirCommunity$, $theirGr$
  for all $s \in CFSset$ **do**
    if $isCommunityRelated(s)$ **then**
      $(s_{less},\ s_{wise}) \leftarrow separate(s)$
      $Sup[s_{less}] \leftarrow processSupport(s_{less})$
      $Gr \leftarrow processGrowthRate(s,\ Sup[s],\ Sup[s_{less}],\ n,\ communitySizes[s_{wise}])$
    end if
  end for

Let us note r the number of CFS found for the whole network. We process each one of the r patterns to keep only the community-related patterns and retrieve their supports. First, we identify the community-related patterns by the help of the function `isCommunityRelated(···)`. This requires processing each one of the r CFS and examining its very last item. Indeed, in the case of *concatenated* node sequences, if a community ID is present in a CFS, it can only be at the end of the sequence. This test is done in constant time. Second, the function `separate(···)` processes the CFS to identify its community ID, which is given directly by its community-wise subsequence and community-less subsequence. This operation can also be done in constant time, for the same reason than before. Third, we obtain the support of this subsequence thanks to the function `processSupport(···)`. For this purpose, we scan once

more the detected CFS: two cases are possible. First case: the community-less subsequence appears among the CFS as a community-independent pattern. In that case, its supports was already processed by CloSpan and we just need to retrieve it. Second case: the sequence is not one of the detected CFS, which means its not closed, and the corresponding community-independent pattern consequently has the same support than its community-related counterpart. Thus, in this case too, we can retrieve the support instead of calculating it. Comparing two sequences can be done in $O(\theta)$, where θ is the size of the longest possible sequence. In the worst case, we need to do this comparison once for each of the r CFS, so `processSupport(···)` has a complexity in $O(r\theta)$. Fourth, once the supports are known, processing the growth rate with `processGrowthRate(···)` can be done in constant time. For a single CFS, we get a total processing in $O(r\theta)$ for all these 4 steps. Finally, we need to repeat this process for all r CFS, leading to a grand total of $O(r^2\theta)$.

4.3.5.2 Step 4.B: Evolving Community Structure

Let us now consider the case of the evolving community structures, where the community detection method outputs a sequence of community structures (instead of a single one for the whole time range). We use the enlarged node sequence and create the database M_{enl}. We then apply CloSpan on M_{enl}. If enough nodes undergo the same evolution while staying in the same *relative* community (i.e. they all keep on belonging to a single community, even if this community can be different at each time slice), then we can catch their common evolution under the form of a frequent sequential pattern. However, small sets of nodes migrating frequently from one community to another will stay unnoticed with this method, since it relies on the constraint of being frequent. If the nodes of a community appearing in a specific time slice support a sequential pattern, we can also concentrate to this community.

In order to process the growth rate of all CFS relatively to all communities, we do not separate CFS depending on how they end, like we did for the consensual community structure case Instead, we separate the CFS community-related patterns from the community-independent patterns. For the community-related patterns, we apply a parsing procedure similar to the one we explained for the case of consensual community structures (previous subsection). Like before, we parse the pattern to find its community-wise subsequence; however, this time we additionally also look for its community-less subsequence. The reason for retrieving this community-wise subsequence is to determine the size of that community (or community sequence): in order to calculate the growth rate, we need that information. In case of evolving communities, it is not possible to have this information directly from the community structure. We give the pseudo-code of this procedure in 3. The function `processSupport(···)` appearing in lines 6 and 7 at first scans the CFS to find out if the parameter pattern was already identified as a direct output of CloSpan. If it is not, the function queries the sequence database to find its support.

Algorithm 3 Extraction of Emerging Sequences from M_{enl}

Require: $CFSset$, M_{enl}
Ensure: $CommunityRelatedCFS$, $theirSup$, $theirGr$
  for *all* $s \in CFSset$ **do**
    if $isCommunityRelated(s)$ **then**
      $(s_{less},\ s_{wise}) \leftarrow separate(s)$
      $Sup[s_{wise}] \leftarrow processSupport(s_{wise}, M_{enl})$
      $Sup[s_{less}] \leftarrow processSupport(s_{less}, M_{enl})$
      $Gr \leftarrow processGrowthRate(s,\ Sup[s],\ Sup[s_{wise}],\ Sup[s_{less}],\ n)$
    end if
  end for

Here, the processing is less straightforward than for the consensual community structures, because

the community ID is not necessarily at the end of the sequence. We first use `isCommunityRelated(···)` to check if the sequence contains at least one community ID, which has a complexity in $O(\theta)$, where θ is the size of the longest possible sequence. Second, and like before, the function `separate( ···)` is applied to split the sequence into its corresponding community-less and community-wise subsequences. This operation also has a complexity in $O(\theta)$. Third, we need to process the supports of these subsequences, thanks to the function `processSupport(···)`. However, unlike for consensual community structures, this time it is possible that these subsequences are not in the CFS, because they are not necessarily closed. Finding their supports requires to scan the whole extended database M_{enl}. In the worst case, we need to compare both community-less and community-wise subsequences with each of the n sequences M_{enl} contains. Each time, it is necessary to check if the considered node sequence is a supersequence of the pattern of interest. In the worst case, such an operation has a complexity in $O(\theta^2)$. So, this third step has a total complexity in $O(\theta^2 n)$. Fourth, like before, the growth rate is processed in constant time thanks to the function `processGrowthRate( ···)`. The total processing for these 4 steps is in $O(\theta^2 n)$. It must be repeated for all r CFS, so we get a grand total in $O(r\theta^2 n)$.

The last point is common to both types of community structures (consensual and evolving). Once the emerging CFS have been identified, we need to extract their supporting nodes, which are not directly outputted by CloSpan. Doing so through a post-processing would be computationally very costly. Instead, to accelerate our post-processing, we modified the original source code of CloSpan, so that it now extracts the supporting nodes and outputs them directly while mining the CFS.

4.3.6 Step 5: Selecting the Characteristic Patterns

After the fourth step, we have the community-related closed patterns related to communities whose support is greater than a given min_{sup}. We also calculated their growth rate for the concerned community (or community sequence). In this fifth step, we select the most characteristic patterns according to their growth rates and coverage of the community, in order to obtain the $\gamma_i^{'}(1 \leq i \leq \mu)$ described in the problem statement, i.e. the sets of characteristic patterns for each community or community sequence m_i. In this section, we first explain our pattern selection method, then we describe how we also detect anomalies simultaneously.

4.3.6.1 Step 5.A: Pattern Selection

At this point, all the considered patterns have a support larger than the threshold min_{sup} and are all CFS. To complete the process, we must eliminate all patterns whose growth rate is smaller than the min_{gr} threshold. Recall that for $Gr > 1$, the pattern is considered as emerging for the given set. The higher the growth rate, the more emerging the pattern. So, $min_{gr} = 1$ is a minimal value to use as a threshold. Of course, depending on the data and objectives, it is also possible to use a higher threshold, in order to make a stricter selection.

After this elimination of non-emerging patterns, we have our characteristic patterns, taking the form of the $\gamma_i^{'}$ sets. However, there can still remain too many of them to perform a relevant interpretation. In this case, we propose an additional post-processing allowing to select the most interesting and characteristic ones to ease the interpretation. We note $\gamma_i^{''} \subseteq \gamma_i^{'}$ the resulting subset of patterns. First, we propose to either consider the pattern with highest growth rate, or with highest support. The complementary selection procedure we propose is generic, and can be applied in both case. In the rest of our explanations, we refer to the criterion of interest (growth rate or support) as the pattern score. Once the pattern with highest score has been selected, there is no guarantee for it to cover a sufficient part of the community. And indeed, in practice it appears to be the opposite (especially for the growth rate). It is thus needed to identify other complementary patterns, allowing us to obtain a more complete coverage of the community. Intuitively, we want to find a small number of patterns, such that they cover a significant part

of the community, and are different in terms of supporting nodes. Note that, in any case, we consider the pattern with the highest score as the most relevant, and search for additional ones to complete the coverage.

Let us define$\gamma_i^{''}$ by refining our criteria, for the reasons we explained earlier. For a community sequence m_i, the set of characteristic patterns $\gamma_i^{'}$ is already known. Note that $\forall s \in \gamma_i^{'}$, s is closed, $Sup(s) > min_{sup}$ and $Gr(s,i) > min_{gr}$ The characteristics of s are also given as Sup and Gr. We also know the supporting nodes of s in m_i, which is given by $S(s,m_i)$. Then, we define $\gamma_i^{''} \subseteq \gamma_i^{'}$ as the subset of sequences from $\gamma_i^{'}$ such that $\forall s \in \gamma_i^{''}$, the following criteria should be ensured:

- The cardinality of $\bigcap_{s\in\gamma_i^{''}} S(s,i)$ must be minimal;
- The cardinality of $\bigcup_{s\in\gamma_i^{''}} S(s,i)$ must be maximal (if possible: the whole community);
- The cardinality of $|\gamma_i^{''}|$ must be minimal.

This problem is equivalent to a specific case of the *set covering problem*. In this classic problem, one considers a set called *universe*, as well as a finite number of subsets whose union is the universe. The goal is to find the minimum number of subsets such that their union is the universe. In our case, the problem of community interpretation is to cover a significant part of the community (universe) with a minimum number of supporting node sets (subsets). Additionally, we also want the intersection of the selected supporting node sets to be as small as possible. Thus, among the constraint we listed above, (2) and (3) correspond to conditions of the set cover problem, whereas (1) is an extra condition.

In order to solve our problem, we perform an iterative procedure. At each iteration, we select the pattern whose supporting nodes are the most different possible (in terms of set intersection) from those already chosen. At the same time, we want the nodes supporting the chosen pattern to be the most similar possible to the part of the community not yet covered. Here, we consider Jaccard's coefficient as the similarity measure between the supporting node sets of the patterns. In case of equality, the pattern score is considered as a secondary criterion. We perform the choice based on two Jaccard's coefficient values noted sim_1 and sim_2 , which measure the similarity between a pattern s and the uncovered part U and the covered part E of the community, respectively. The overall measure for the pattern is then $sim = sim_1 - sim_2$. At each iteration, we choose a pattern which maximizes sim. This iteration continues until convergence. After convergence, if there are still some uncovered nodes, we mark them as having no patterns. We give the pseudo code of the pattern selection procedure in 4.

Algorithm 4 Choosing Patterns for all Community Sequences

Require: $CFSset$, $Gr[\cdots]$,$S[\cdots]$,B,min_{gr}
Ensure: $\Gamma^{''}$
  $CFS^{'} \leftarrow filter(CFS, Gr[\cdots], min_{gr})$
  $\Gamma^{'} \leftarrow group(CFS^{'}, B)$
  for i $in\{1\cdots\mu\}$ **do**
    $remainingPatterns \leftarrow \gamma_i^{'}$
    $U \leftarrow S(m_i)$
    $E \leftarrow \varnothing$
    repeat
      $\gamma_i^{''} \leftarrow choose(remainingPatterns, U, E)$
      $update(U)$
      $update(E)$
    until $U does not change$
  end for

First, the function `filter(···)` just removes the patterns whose Gr is smaller than given min_{gr} it requires to scan all the remaining CFS once, so it can be done in $O(r)$, in the worst case. Second, the function `group(···)` gathers the patterns according to their community-wise subsequences. Thus, for each community sequence m_i, $(1 \leq i \leq \mu)$ we can later treat $\gamma_i^{'}$. In the worst case, this grouping requires checking if each one of the r patterns is a super-sequence of each one of the μ community sequences. Such a sequence comparison can be performed in $O(\theta^2)$, so because of the repetitions, we get a time complexity in $O(\theta^2 \mu r)$ for this first operation, where r is the total number of chosen CFS and μ the number of community sequences. Then, for each community sequence m_i, we at first access all its related patterns, which are contained in $\gamma_i^{'}$. We then look for the most characteristic ones. Note that we do not treat every community sequence appearing in the community structure, but only those outputted by CloSpan. There are several reasons for this. First, in the case of a consensual community structure, it is not necessary to characterize very small communities because the notion of being frequent is not relevant, and it is possible to interpret these communities manually. Second, in the case of an evolving community structure, it does not make any sense to interpret every possible community sequence. We already catch the most prominent events and remarkable community sequences through sequence mining. Thus, the points of interest are limited by the communities (or their sequences) which appear as patterns.

For each community sequence, we at first define the uncovered and covered nodes sets with the patterns that we treat for the initial iteration. Then, the repeat loop continues until the uncovered node set does not change. We process the function `choose(···)` at each iteration. It calculates the similarity $sim = sim_1 - sim_2$, and returns the most suitable pattern, according to the criteria we previously described. For this matter, it must process Jaccard's coefficient twice, to compare on the one side the supporting set of the pattern, and on the other side, successively, the covered node set then the uncovered node set. The covered node set corresponds to nodes from the community sequence which are already covered by the previously selected patterns. On the contrary, the uncovered node set contains nodes not already covered by these patterns. If the sets are presented by ordered lists, processing Jaccard's coefficient can be done in a time linear to the cardinality of the smallest set. For us, the worst case happens when the community sequence and the supporting set of the pattern we treat include all the n nodes of the network. In practice, the sizes of these sets are much smaller though, since the communities are significantly smaller than the whole network. Then, at the first iteration, the uncovered node sets contains all the n nodes from the community sequence, and it takes $O(n)$ to process Jaccard's coefficient. Therefore, it also takes $O(n)$ to process *sim*, which is the difference between both calculated Jaccard's coefficient values. The same operation must be repeated for all the patterns associated to the considered community sequence, in order to identify the most appropriate one depending on its *sim* value. The number of patterns in a community sequence is approximately r/μ, so processing all of them is in $O(nr/\mu)$. This includes identifying the pattern with optimal *sim* (this process can be conducted at the same time).

Once the `choose(···)` function has returned its results, all three working sets (currently selected patterns, covered nodes and uncovered nodes) must be updated. If a set is represented as an ordered list, this operation can be done in a time linear to its cardinality. So in our situation, $O(n)$ in the worst case. The repeat loop is processed again until all the nodes concerned by the considered community sequence are covered. In the worst case, all r/μ patterns must be used. However, in practice, only a few iterations are required. Moreover, in the case where many patterns are required, the whole process loses its interest, since its goal is to select only a small number of patterns in order to ease human interpretation. So, when the number patterns needed to complete the coverage is too large, it is not worth continuing the process. For these reasons, in the rest of our explanations, we consider the number of repetitions of this loop of constant order. So, the total complexity for the repeat loop, and therefore function `choose( ···)` , is in $O(nr/\mu)$. Since the for loop of must be repeated μ times in the worst case, we get a complexity of $O(nr)$. Thus, when considering functions `group(···)` and `filter(···)`, the grand total complexity for the whole algorithm is in $O(r(\theta^2\mu + n + 1))$. It can be simplified in $O(r(\theta^2\mu + n))$.

Note that in certain situations, some problems might arise when looking for the characteristic patterns. First, it is possible that CloSpan does not find any frequent pattern for a given community, because the minimum support is too high. In this case, the user must either conclude there is no characteristic pattern for the community (since none of them manage to gather a sufficient support), or adjust the value of this parameter. Second, there cannot be any pattern having a growth rate larger than giving minimum growth rate threshold. In this case, there is no emerging pattern for the communities. Thus our framework cannot find any of them. Third, for a given community, it is possible that our selection procedure outputs too many supplementary patterns to cover the whole community. In this case, the user can either focus on the first few best patterns while keeping their support in mind, or conclude that the community cannot be characterized by some specific patterns, because of its heterogeneity.

4.3.6.2 Step 5.B: Anomalies

Once the most characteristic patterns of a community sequences have been identified (the most emerging one, the one with highest support and the supplementary patterns), it is possible to use them to detect anomalies, i.e. nodes not following those patterns. Let m_i be a community sequence, then $S(m_i)$ is its supporting node set, and $S(s,S(m_i))$ is the supporting node set of pattern s in community sequence m_i. We define the *characterized supporting node set* of m_i as $S^+(m_i) = \bigcup_{s\in\gamma_i''} S(s,S(m_i))$ as the nodes described by characteristic patterns. It is the union of the supporting node sets for all the characteristic patterns of the considered community sequence. Then, we define the *anomaly node set* $S^-(m_i) = S(m_i) \setminus S^+(m_i)$ as the set of nodes not following any characteristic pattern. These nodes are different in the sense they do not follow the general trends observed in their communities. We detect anomalies automatically when finding representative patterns.

4.4 Interpretation

Our framework outputs the characteristic patterns of communities or community sequences. Additionally, in the case of evolving communities, it is also interesting to study community sequence patterns (i.e. patterns containing only community IDs), in order to have an idea of the way the community structure evolves, and more particularly of the changes occurring in the communities. In this section, we explain how we can detect and understand such community evolution events, and how the patterns can be interpreted to understand the communities.

4.4.1 Community Evolution Events

When using a consensual community structure, the community sequences that CloSpan outputs are all of size 1: each one represents a particular community, valid for all time slices. Interpreting them is not relevant, because these is no related evolutionary events. However, when using an evolving community structure, sequential pattern mining gives us the opportunity of following the major events regarding community evolution, by focusing on the patterns outputted by CloSpan and including only community IDs. The interpretation of these patterns gives us some information about the major events in the community life, like: birth, death, merge, split, hide, switch, expansion, contraction or stability. The patterns including only community IDs and whose size is equal to k points to group of nodes which are always together in the same community during k time slices. Specifically, if $k = 1$, then by definition the node group corresponds to a while community, considered at a certain time. Here, we present an interpretation method taking advantage of this type of patterns to detect community events.

We first find out all patterns including the ID of a community of interest C , among the community sequences outputted by CloSpan. We then order these patterns chronologically. This is possible thanks

to the encoding we defined to represent the community IDs in the sequence database. If there is a pattern of size 1 including only C, and not any other community ID, at time slice t, then its support shows the size of community C at time t. The form of this pattern is $< (C@t) >$. This pattern is our reference in terms of temporal order: we will now consider other patterns relatively to this time. The patterns starting with $(C@t)$, and whose size is greater than 1, show the future of C, i.e. how it is going to evolve later on. Their form is $< (C@t)(\cdots)\cdots(\cdots) >$. Each such pattern represents a node group of $C@t$ following a specific evolution. Similarly, the patterns ending with $(C@t)$, and whose size is greater than 1, show the past of C, i.e. how it reached its state at time t. Their form is $< (\cdots)\cdots(\cdots)(C@t) >$. The third type of pattern shows both future and past together: their size is greater than 2 and they contain $(C@t)$, but neither as the starting nor ending item. Their form is therefore $< (\cdots)\cdots(C@t)\cdots(\cdots) >$.

Once the patterns related to the community of interest have been ordered, we evaluate them together. If there are more than one future-related patterns, they show how different node groups constituting C leave the community to either join different communities, or start a new community, after t. Similarly, if there are more than one past-related patterns, they represent different node groups leaving different communities before t and joining C at time t. We must underline here that the patterns of size greater than 1 represent *node groups* which are together at different time slices. Most of the time, they might not represent the whole community to which they belong, but only a part of it. In order to determine if a pattern of size longer than 1 describes an event related to a whole community, one needs to compare the support of the node groups of each community ID appearing in the pattern, with that of the pattern itself. This matching operation is not really in the scope of this thesis, so we do not develop this point further. Here, we only want to reveal how different node groups get separated or merged around the community of interest. Our aim is to direct the interpretation process by finding out first the behavior of different node groups in the community we study. This analysis allows us to know if the trend emerging in a community covers most of its nodes over different time slices.

4.4.2 Community Interpretation

For a consensual community structure, the output of our framework directly leads us to the emerging trends inside the communities. However, in the case of an evolving community structure, the interpretation of the final characteristic patterns requires some additional work. The main reason for this is that the focus shifts from the community as a whole to groups of nodes generally smaller than the community. For this reason, the type of preliminary study based on community sequences described in the previous subsection can help identifying important or interesting communities on which one can then focus.

Suppose we want to study some community of interest $C@t$. Our first step consists in identifying all patterns containing $C@t$. At this point, two types of patterns can be found: those containing only one community ID $(C@t)$ and those containing several of them ($C@t$ and some other ones). Let us consider the former case first, as it is simpler. Such a pattern can be interpreted as the description of a group of nodes which, at a given time t, happens to all belong to community $C@t$. For interpretation purposes, we consider the community-less subsequence of the pattern. It indicates how some characteristic descriptors of this node group evolve over time. Here, only the relative order of the descriptor-based items is important to us: we ignore their position relatively to the community ID in the sequence.

In the case of a pattern containing several community IDs, we do not consider it as a whole community but we consider as a node group which supports the community sequences of interest. The interpretation of the community-less subsequence of the pattern is same with previous case. However, this interpretation is valid only for the nodes which are together at several communities (represented by different community ID's in community sequence) of different time slices. When we comment the meaning of the pattern, we mention that the trend represented in pattern is valid for specific node group but not the whole communities.

4.5 Example

In order to better understand how our framework runs on a dynamic attributed network, we give an example on the tiny network shown in Figure 4.2. It contains 7 nodes whose interconnections change over three time slices. The node is $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ (the figure shows only their indices). There is only one attribute a assigned to these nodes. Their values are shown in the figure, and they can change from one time slice to the other. We consider the degree as the only topological measure, for the sake of simplicity (rather than using all nine measures previously presented). In the rest of the section, we at first consider consensual community structure, and then evolving community structures. In our explanation, we refer to the code of each step of our framework as defined in its description.

4.5.1 First Case: Consensual Community Structure

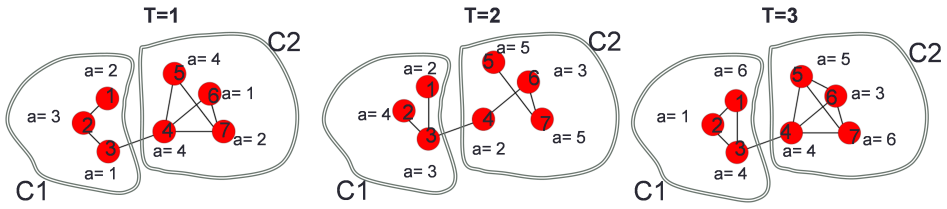

Fig. 4.2 A small dynamic attributed network with consensual community structure fitting all time slices

For this example, we suppose the community structure has already been detected, i.e., Step 1 has already been performed. There are two communities $C1$ and $C2$, shown with the grey borders in Figure 4.2. The domain of attribute a is $\{1,2,3,4,5,6\}$, while the possible degree values (for this network) are $\{1,2,3,4\}$. In this example, we skip the Step 2 of our framework, and do not bin the values of these integer-valued descriptors. Thus, the set of all possible items over all time slices is: $\{a=1, a=2, a=3, a=4, a=5, a=6, d=1, d=2, d=3, d=4\}$, where d is the degree. Let us consider v_3 in Figure 4.2: the itemset describing this node at $t=1$ is: $(a=1, d=4)$. The sequence describing the same node for $t=1$ and $t=2$ is: $<(a=1,d=2)(a=3,d=3)>$. The node sequence of the same node is: $u(v_3) =< (a=1,d=2)(a=3,d=3)(a=4,d=3)>$. We perform the concatenation operation as indicated in Step 4.A, to create the concatenated sequence database M^{con} for further analysis. For the same node v_3, we obtain the concatenated node sequence: $u(v_3) \bullet C(v_3) =< (a=1,d=2)(a=3,d=3)(a=4,d=3)(C1)>$. The database M^{con}, which will be used in Step 3, is given in the table 4.1.

Node ID	u_{con}
1	<(a=2,d=1)(a=2,d=1)(a=6,d=2)(C1)>
2	<(a=3,d=2)(a=4,d=1)(a=1,d=2)(C1)>
3	<(a=1,d=2)(a=3,d=3)(a=4,d=3)(C1)>
4	<(a=4,d=4)(a=2,d=4)(a=4,d=4)(C2)>
5	<(a=2,d=2)(a=5,d=1)(a=6,d=3)(C2)>
6	<(a=1,d=2)(a=3,d=2)(a=3,d=3)(C2)>
7	<(a=4,d=3)(a=5,d=2)(a=5,d=3)(C2)>

Table 4.1 Concatenated Node Sequence Database for Network of Figure 4.2

During Step 3, we apply CloSpan with $min_{sup} = 0.1$, and find all the CFS listed in table 4.2. After

having found the CFS, we post-process these patterns as explained in Step 4.A, to calculate the growth rate. In table 4.3, we display the Gr of the community-related patterns from table 4.2.

CFS	$\|S\|$	sup
<(a=1 d=2)(C1)>	2	0.29
<(a=1 d=2)(a=3 d=2)(a=3 d=3)(C2)>	1	0.14
<(a=1 d=2)(a=3 d=3)(a=4 d=3)(C1)>	1	0.14
<(a=1 d=2)(a=3 d=3)>	2	0.29
<(a=1 d=2)(a=3)(d=3)>	2	0.29
<(a=1 d=2)>	3	0.43
<(a=2 d=1)(a=2 d=1)(a=6 d=2)(C1)>	1	0.14
<(a=2 d=3)(a=5 d=2)(a=6 d=3)(C2)>	1	0.14
<(a=2)(a=6)>	2	0.29
<(a=2)(C2)>	2	0.29
<(a=2)(d=2)>	2	0.29
<(a=2)>	3	0.43
<(a=3 d=2)(a=4 d=1)(a=1 d=2)(C1)>	1	0.14
<(a=3 d=2)>	2	0.29
<(a=3)(a=4)(C1)>	2	0.29
<(a=3)>	3	0.43
<(a=4 d=2)(a=5 d=1)(a=5 d=3)(C2)>	1	0.14
<(a=4 d=4)(a=2 d=2)(a=4 d=4)(C2)>	1	0.14
<(a=4)(C2)>	2	0.29
<(a=4)(d=2)>	2	0.29
<(a=4)>	4	0.57
<(a=5)(d=3)(C2)>	2	0.29
<(d=1)(d=2)(C1)>	2	0.29
<(d=1)>	3	0.43
<(d=2)(C1)>	3	0.43
<(d=2)(a=4)(C1)>	2	0.29
<(d=2)(a=4)>	3	0.43
<(d=2)(C2)>	4	0.57
<(d=2)(d=1)>	2	0.29
<(d=2)(d=2)>	2	0.29
<(d=2)(d=3)(C2)>	3	0.43
<(d=2)(d=3)>	4	0.57
<(d=2)>	7	1.00
<(d=3)(d=3)>	2	0.29

Table 4.2 Closed Frequent Patterns for minimum support 0.1 of Network from Figure 4.2

Most of the patterns growth rates are infinite. It means those patterns appear only in the indicated communities but never in the rest of the network. In the rest of the procedure, we consider these patterns as the most significant and characteristic of the community in which they appear. However, there are still too many of them for human interpretation. Thus, we need to apply our pattern selection method, as explained in Step 5.A.

The input data for Step 5.A is given in table 4.4. In this table, the patterns are ordered by community ID. We apply our selection method first on $C1$. At the first iteration, the uncovered node set is $\{v_1, v_2, v_3\}$

and the covered node set is $\emptyset$. The similarity results of the patterns #1, 2, 3, 4 and 5 are 0.33, 0.33, 0.33, 0.66 and 0.66 respectively. We choose randomly one of the patterns among #4 and 5 because their similarity all reaches the maximum. Say we pick pattern #4. At the second iteration, the updated uncovered node set is $\{v_1\}$, and the updated covered node set is $\{v_2, v_3\}$. The similarity results of patterns #1, 2, 3 and 5 are -0.5, 1, -0.5 and 0, respectively. The pattern the most similar to the uncovered set, and at same time the most dissimilar to the covered set is pattern #2. So, in two iterations, we covered all the nodes of this community. The characteristic patterns of $C1$ therefore are <(a=2 d=1)(a=2 d=1)(a=6 d=2)> and <(a=3)(a=4)>. Note that there is no anomaly in this community.

s_{less}	**C**	$\lvert S(s_{less}, C)\rvert$	$\lvert S(s_{less})\rvert$	$Gr(s_{less}, C)$
<(a=1 d=2)>	C1	2	3	2.67
<(a=1 d=2)(a=3 d=2)(a=3 d=3)>	C2	1	1	∞
<(a=1 d=2)(a=3 d=3)(a=4 d=3)>	C1	1	1	∞
<(a=2 d=1)(a=2 d=1)(a=6 d=2)>	C1	1	1	∞
<(a=2 d=3)(a=5 d=2)(a=6 d=3)>	C2	1	1	∞
<(a=2)>	C2	2	3	1.50
<(a=3 d=2)(a=4 d=1)(a=1 d=2)>	C1	1	1	∞
<(a=3)(a=4)>	C1	2	2	∞
<(a=4 d=2)(a=5 d=1)(a=5 d=3)>	C2	1	1	∞
<(a=4 d=4)(a=2 d=2)(a=4 d=4)>	C2	1	1	∞
<(a=4)>	C2	2	4	0.75
<(a=5)(d=3)>	C2	2	2	∞
<(d=1)(d=2)>	C1	2	2	∞
<(d=2)>	C1	3	7	1.00
<(d=2)(a=4)>	C1	2	3	2.67
<(d=2)>	C2	4	7	1.00
<(d=2)(d=3)>	C2	3	4	2.25

Table 4.3 Community-less subsequences of Closed Frequent Patterns with Communities, the length of the Supporting Nodes in their communities, the length of the Supporting nodes in the Network and their Growth Rates

Pattern ID	s_{less}	**C**	$\lvert S(s_{less}, C)\rvert$
1	<(a=1 d=2)(a=3 d=3)(a=4 d=3)>	C1	$\{ v_3 \}$
2	<(a=2 d=1)(a=2 d=1)(a=6 d=2)>	C1	$\{v_1 \}$
3	<(a=3 d=2)(a=4 d=1)(a=1 d=2)>	C1	$\{v_2 \}$
4	<(a=3)(a=4)>	C1	$\{v_2, v_3 \}$
5	<(d=1)(d=2)>	C1	$\{v_1, v_2 \}$
6	<(a=1 d=2)(a=3 d=2)(a=3 d=3)>	C2	$\{v_6 \}$
7	<(a=2 d=3)(a=5 d=2)(a=6 d=3)>	C2	$\{v_7 \}$
8	<(a=4 d=2)(a=5 d=1)(a=5 d=3)>	C2	$\{v_5 \}$
9	<(a=4 d=4)(a=2 d=2)(a=4 d=4)>	C2	$\{v_4 \}$
10	<(a=5)(d=3)>	C2	$\{v_5, v_7 \}$

Table 4.4 Chosen community-less subsequences of CFS, their communities and their supporting nodes

For $C2$, at the first iteration, the uncovered node set is $\{v_4, v_5, v_6, v_7\}$ and the covered node set is $\emptyset$. The similarities for patterns #6,7, 8, 9 and 10 are 0.25, 0.25, 0.25, 0.25 and 0.50, respectively. We choose pattern #10 at the end of the first iteration. Then, the updated uncovered set is $\{v_4, v_6\}$ and the updated covered set is $\{v_5, v_7\}$. The similarities at the second iteration for patterns #6,7, 8 and 9 are 0.50, -0.50, -0.50 and 0.50 respectively. We choose one of the patterns among #6 and 9. Say we pick pattern #6. Then, at the third iteration, the updated uncovered set is $\{v_4\}$ and the updated covered set is $\{v_5, v_6, v_7\}$. The similarities at the third iteration, for patterns #7, 8 and 9 are -0.66, -0.66 and 1 respectively. Say we then select pattern #9. Thus, this community is characterized by three patterns : <(a=5)(d=3)>, <(a=1 d=2)(a=3 d=2)(a=3 d=3)> and <(a=4 d=4)(a=2 d=2)(a=4 d=4)> and has no anomaly.

4.5.2 Second Case: Evolving Community Structure

Let us now consider evolving community structures. In this case, we do not need performing the concatenation operation to calculate the growth rates. However, we take into account the community IDs of each time slice as a descriptor. We use the same network than in the previous example, but with an evolving community structure, in which v_4 changes its community from $C2$ to community $C1$ at t=2, before going back to $C1$ at t=3.

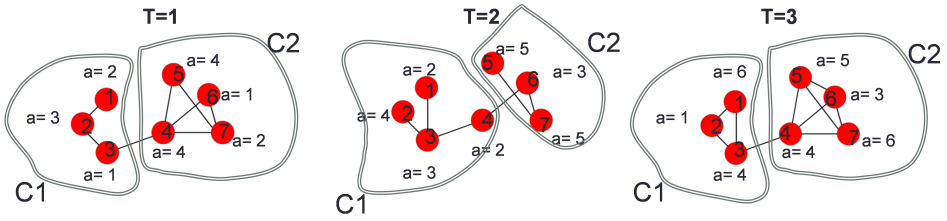

Fig. 4.3 A small dynamic attributed network with evolving community structure

The enlarged database M_{enl} is given in table 4.5. We apply at first CloSpan as described in Step 3, with $min_{sup} = 0.1$ and find the CFS. Then, we process Step 4.B and extract all community-related patterns.

ID	u_{enl}
1	<(a=2,d=1, C1@1)(a=2,d=1,C1@2)(a=6,d=2,C1@3)>
2	<(a=3,d=2,C1@1)(a=4,d=1,C1@2)(a=1,d=2,C1@3)>
3	<(a=1,d=2,C1@1)(a=3,d=3,C1@2)(a=4,d=3,C1@3)>
4	<(a=4,d=4,C2@1)(a=2,d=2,C1@2)(a=4,d=4,C2@3)>
5	<(a=4,d=2,C2@1)(a=5,d=1,C2@2)(a=5,d=3,C2@3)>
6	<(a=1,d=2,C2@1)(a=3,d=2,C2@2)(a=3,d=3,C2@3)>
7	<(a=2,d=3,C2@1)(a=5,d=2,C2@2)(a=6,d=3,C2@3)>

Table 4.5 Enlarged Node Sequence Database for Network of Figure 4.3

We also parse these patterns to find their community-wise and community-less subsequences, as explained at Step 4.B. The patterns resulting from this step are listed in table 4.6. Each row corresponds to a community-related pattern, its community-less and its community-wise subsequences, and finally the growth rate of the considered community-related pattern. The table is sorted according to the community-wise subsequences. Note that the growth rates correspond here to the ratio of the support of

the community-less subsequence in the supporting nodes of community-wise subsequence to the same community-less subsequence in the rest of the network.

s	s_{less}	s_{wise}	$Gr(s_{less}, S(s_{wise}))$
<(a=1 d=2 C1@1)(a=3 d=3 C1@2)(a=4 d=3 C1@3)>	<(C1@1)(C1@2)(C1@3)>	<(a=1 d=2)(a=3 d=3)(a=4 d=3)>	∞
<(a=2 d=1 C1@1)(a=2 d=1 C1@2)(a=6 d=2 C1@3)>	<(C1@1)(C1@2)(C1@3)>	<(a=2 d=1)(a=2 d=1)(a=6 d=2)>	∞
<(a=2 d=3 C2@1)(a=5 d=2 C2@2)(a=6 d=3 C2@3)>	<(C2@1)(C2@2)(C2@3)>	<(a=2 d=3)(a=5 d=2)(a=6 d=3)>	∞
<(a=2 C1@2)>	<(C1@2)>	<(a=2)>	1.50
<(a=2)(C2@3)>	<(C2@3)>	<(a=2)>	1.50
<(a=3 d=2 C1@1)(a=4 d=1 C1@2)(a=1 d=2 C1@3)>	<(C1@1)(C1@2)(C1@3)>	<(a=3 d=2)(a=4 d=1)(a=1 d=2)>	∞
<(a=3)(C1@3)>	<(C1@3)>	<(a=3)>	2.67
<(a=4 d=2 C2@1)(a=5 d=1 C2@2)(a=5 d=3 C2@3)>	<(C2@1)(C2@2)(C2@3)>	<(a=4 d=2)(a=5 d=1)(a=5 d=3)>	∞
<(a=4 d=4 C2@1)(a=2 d=2 C1@2)(a=4 d=4 C2@3)>	<(C2@1)(C1@2)(C2@3)>	<(a=4 d=4)(a=2 d=2)(a=4 d=4)>	∞
<(a=4 C2@1)(C2@3)>	<(C2@1)(C2@3)>	<(a=4)>	0.75
<(d=2 C1@1)(C1@2)(C1@3)>	<(C1@1)(C1@2)(C1@3)>	<(d=2)>	0.53
<(d=2 C2@1)(C2@2)(d=3 C2@3)>	<(C2@1)(C2@2)(C2@3)>	<(d=2)(d=3)>	1.33
<(d=2 C1@1)(a=4)>	<(C1@1)>	<(d=2)(a=4)>	2.67
<(d=2)(d=3 C2@3)>	<(C2@3)>	<(d=2)(d=3)>	2.25
<(d=2)(C2@3)>	<(C2@3)>	<(d=2)>	1.00
<(C1@1)(d=1 C1@2)(d=2 C1@3)>	<(C1@1)(C1@2)(C1@3)>	<(d=1)(d=2)>	∞
<(C1@1)(C1@2)(C1@3)>	<(C1@1)(C1@2)(C1@3)>	<>	1.00
<(C2@1)(a=5 C2@2)(d=3 C2@3)>	<(C2@1)(C2@2)(C2@3)>	<(a=5)(d=3)>	∞
<(C2@1)(d=2 C2@2)(d=3 C2@3)>	<(C2@1)(C2@2)(C2@3)>	<(d=2)(d=3)>	1.33
<(C2@1)(d=2)(C2@3)>	<(C2@1)(C2@3)>	<(d=2)>	0.56
<(C2@1)(C2@2)(d=3 C2@3)>	<(C2@1)(C2@2)(C2@3)>	<(d=3)>	4.00
<(C2@1)(C2@3)>	<(C2@1)(C2@3)>	<>	1.00
<(C1@2)(a=4)>	<(C1@2)>	<(a=4)>	0.75
<(C1@2)>	<(C1@2)>	<>	1.00

Table 4.6 Community Related CFS, their community-less and community-wise subsequence and their Growth Rate

Before explaining the selection phase, let us look at the community sequences describing the community evolution. All possible community-sequences showing community evolution are given in table

4.7. Among them, the size of patterns #1, 2 and 2 are all 3. Thus they explain a trend spanning all time slices. Observing them in detail reveals events in the evolution of the communities. Patterns #1 and 3 show there are 2 groups of 3 nodes staying in $C1$ and $C2$, respectively, for all time slices. Pattern #2 highlights the behavior of single node, which moved from $C2$ to $C1$, and then goes back to $C2$. Thus, $C1$ expands at time slice 2, but then shrinks at time slice 3. Patterns #4 and 5 are showing the trend of two different node groups including 4 nodes. The first group (#4) is in $C2$ at first and last time slices. The second group (#5) represents the nodes which are at $C1$ at second time slice.

Pattern ID	s	**|S(s)|**	**Size**
1	<(C1@1)(C1@2)(C1@3)>	3	3
2	<(C2@1)(C1@2)(C2@3)>	1	3
3	<(C2@1)(C2@2)(C2@3)>	3	3
4	<(C2@1)(C2@3)>	4	2
5	<(C1@2)>	4	1

Table 4.7 All possible community sequences for minimum support 0.1

	s	s_{wise}	$S(s_{less}, S(s_{wise}))$	Gr
1	<(a=1 d=2)(a=3 d=3)(a=4 d=3)>	<(C1@1)(C1@2)(C1@3)>	$\{v3\}$	∞
2	<(a=2 d=1)(a=2 d=1)(a=6 d=2)>	<(C1@1)(C1@2)(C1@3)>	$\{v1\}$	∞
3	<(a=3 d=2)(a=4 d=1)(a=1 d=2)>	<(C1@1)(C1@2)(C1@3)>	$\{v2\}$	∞
4	<(d=1)(d=2)>	<(C1@1)(C1@2)(C1@3)>	$\{v1, v2\}$	1.5
5	<(d=2)(a=4)>	<(C1@1)>	$\{v2, v3\}$	1.5
6	<(a=2)>	<(C1@2)>	$\{v1, v4\}$	∞
7	<(a=3)>	<(C1@3)>	$\{v2, v3\}$	2.67
8	<(a=2 d=3)(a=5 d=2)(a=6 d=3)>	<(C2@1)(C2@2)(C2@3)>	$\{v7\}$	∞
9	<(a=4 d=2)(a=5 d=1)(a=5 d=3)>	<(C2@1)(C2@2)(C2@3)>	$\{v5\}$	∞
10	<(a=4 d=4)(a=2 d=2)(a=4 d=4)>	<(C2@1)(C1@2)(C2@3)>	$\{v4\}$	∞
11	<(d=2)(d=3)>	<(C2@1)(C2@2)(C2@3)>	$\{v6, v7\}$	2.67
12	<(a=5)(d=3)>	<(C2@1)(C2@2)(C2@3)>	$\{v5, v6, v7\}$	2.25
13	<(d=2)(d=3)>	<(C2@1)(C2@2)(C2@3)>	$\{v5, v6, v7\}$	1
14	<(d=3)>	<(C2@1)(C2@2)(C2@3)>	$\{v5, v6, v7\}$	∞
15	<(a=2)>	<(C2@3)>	$\{v4, v7\}$	∞
16	<(d=2)(d=3)>	<(C2@3)>	$\{v5, v6, v7\}$	1.33
17	<(d=2)>	<(C2@3)>	$\{v5, v6, v7\}$	4

Table 4.8 Patterns (s), their community-wise sequences (s_{wise}), supporting nodes ($S(s_{less}, S(s_{wise}))$) and growth rates (Gr)

The input data of the pattern selection process performed in Step 5.A is given in table 4.8. We make the selection here manually, as we did for previous example. But this time, rather than searching characteristic sequences for communities, we must do it for each community sequence. In table 4.8, the patterns are ordered according to their community sequences. Let us start with sequence $< (C1)(C1)(C1) >$. At the first iteration, the uncovered node set is $\{v_1, v_2, v_3\}$, and the covered node set is $\emptyset$. We start with the most emerging pattern. However, here, there are three patterns with $Gr = \infty$. Thus, we make the selection according to their similarity. The similarity results of patterns #1, 2 and 3 are 0.33, 0.33 and 0.33,

respectively. Remark that we did not consider pattern #4 yet because its growth rate is not the highest one. For the first iteration, we favor the most emerging ones, but not for the following iterations. We choose pattern #1 for the first iteration. At the second iteration, the uncovered node set is $\{v_1, v_2\}$, and the covered node set is $\{v_3\}$. The similarity of patterns #2, 3 and 4 are 0.5, 0.5 and 1, respectively. We select pattern #4 and the coverage is complete. Community sequence $< (C1)(C1)(C1) >$ is characterized by patterns <(a=1 d=2)(a=3 d=3)(a=4 d=3)> and <(d=1)(d=2)>. There is no anomaly for this community sequence. We do not give the details of the procedure for the rest of the community sequences, because they do not illustrate any particular new point.

4.6 Algorithmic Complexity

The overall complexity of our method includes calculating all topological measures, applying the Louvain algorithm, then the CloSpan algorithm, processing the growth rates, and finally selecting the most representative patterns.

Topological Measures. In section 2.3, we described the complexity of all topological measures:

- Degree : $O(l)$
- Local transitivity : $O(l^{3/2})$
- Eccentricity : $O(nl)$
- Closeness centrality : $O(n^2(n+l))$
- Betweenness centrality : $O(nl)$
- Eigenvector centrality : $O(n^2)$
- Embeddeness : $O(l)$
- z-score : $O(l+n)$
- Participation Coefficient : $O(\lambda l)$

The total complexity is therefore in $O(l)+O(l^{3/2})+O(nl)+O(n^2(n+l))+O(nl)+O(n^2)+O(l)+O(l+n)+O(\lambda l)$. In the case of real-world complex networks, it is common to have sparse graphs, in which case the number of links l is considered to be of the same order than the number of nodes n (for a large n) [95]. So, these complexities can be simplified in $O(n)+O(n^{3/2})+O(n^2)+O(n^3)+O(n^2)+O(n^2)+O(n)+O(n)+O(\lambda n)$. Furthermore, we can consider the number of communities is negligible compared to the number of nodes. Thus, after simplification, the overall complexity for processing the topological measures is in $O(n^3)$. Since we need to process each one of the θ time slices, we get $O(n^3\theta)$ in total. Obviously, this complexity can change if one use other measures instead of those selected for this work.

Community Detection. The complexity of Incremental and Classic Louvain, as explained in Step 1, is in $O(n\log n)$ [14] for one time slice. It is applied for θ time slices, so the total complexity of these methods is in $O(\theta n\log n)$. For Integrated Louvain, we at first create the integrated network, which requires a time in $O(l^2\theta)$. Then, we apply Louvain on this network once. Thus, the total complexity is in $O(l^2\theta+n\log n)$. For Multistep Louvain, although the exact complexity is not given in the original reference, we assume that it is in $O(\theta n\log n)$ because the optimization step of the original Louvain is applied to each time slice. Finally, comparing $O(l^2\theta+n\log n)$ and $O(\theta n\log n)$, the total complexity of the community detection step is in $O(l^2\theta+n\log n)$ because $l^2 \gg n\log n$. Of course, it would be possible to use other community detection algorithms, with possibly different complexities.

Frequent Sequence Mining. As we explained in Step 3, the last step of CloSpan is in $O(n^2)$, according to [82].

Post-Processing. As explained in Step 4, the processing of the growth rate is in $O(r^2\theta)$ for consensual community structures and $O(nr\theta^2)$ for evolving community structures, where r is the number of patterns found in the network, θ is the number of time slices, and n is the size of network (number of nodes).

Pattern Selection. The selection of the most representative sequence is in $O(r(\theta^2\mu + n))$, as explained in Step 5, where μ is the number of communities (or community sequences) to be treated.

Overall Complexity. The overall complexity of our method is then in $O((n^3 + l^2 + nr\theta + r\mu\theta)\theta + n^2 + nr + n\log n)$ for evolving community structures, and $O((n^3 + l^2 + r^2 + r\mu\theta)\theta + n^2 + nr + n\log n)$ for consensual community structures. We can simplify both expressions by simply removing the terms of lower order, i.e. the 3 last ones, which leads to $(n^3 + l^2 + nr\theta + r\mu\theta)\theta$ and $O((n^3 + l^2 + r^2 + r\mu\theta)\theta$, respectively. As mentioned before, for sparse networks we consider $l \sim n$, so we can remove both l^2 terms, since $n^3 > l^2$. By definition, $r > \mu$, since the the considered community sequences (cardinality μ) constitute a subset of the detected CFS (cardinality r), so we have $r^2\theta > r\mu\theta$. In practice, CloSpan detects many more CFS (cardinality r) than there are nodes in the network (cardinality n), so $r > n$, and consequently $r^2\theta > nr\theta$. After these simplifications, the overall complexity for both consensual and evolving community structures is in $O(n^3\theta + r^2\theta^2)$. It relies on the number of nodes n in the network, the number of patterns r outputted by CloSpan and the number of time slices θ we study. The value of θ depends on the application. In the data we study in the rest of the thesis, it is negligible compared to both other variables. However, in general, it can be very high, like for example if the studied system is observed every minute for one year.

4.7 Implementation Platforms

We perform the implementation of our framework by using different platforms. We work on a server with 32 GB RAM, 16 cores 2.67 GHz Intel(R) Xeon(R) CPU, and Ubuntu Operating system version 10.04. We used the updated C++ source code of the Classic Louvain algorithm given in [55]. We took advantage of a feature of this implementation, which allows to start the process on an existing community structure. This way, we could base the community structure estimated at a given time on that of the previous time, which is the principle of Incremental Louvain algorithm. We used all 9 topological measures defined in 2.3. The topological measures are calculated for each node at each time slice by using the iGraph library [66] version 0.6.5 hosted on CRAN (Comprehensive R Archive Network). Most of the measures are already predefined in the iGraph library, as parametric functions. For the embeddedness, within module degree and participation coefficient, we implemented the functions using the R language. We discretized the topological measures, and grouped them using a k-means. We implemented this part in R, too, because we took advantage of some existing R libraries: cluster, stats. Regarding CloSpan, the original C++ source code is publicly available by Yan [135]. However, we decided to use an open-source implementation written in Java by Fournier-Viger [46], because it allows to modify the source code in a more convenient way. This modification was required to extract the supporting node set while mining the closed patterns. All the other procedures related to finding the Growth Rate and selecting the characteristic patterns of a community for interpretation were written in R and bash script .

Chapter 5

Behavior of Community Interpretation Framework

In the previous chapter, we proposed a framework to interpret the communities in dynamic attributed networks. Here, we study how this framework behaves on some controlled data, representing different types of networks. More precisely, we want to answer the following questions:

- How does the community detection method affect the results?
- Do community sequences efficiently capture major community events?
- What are the limits of our framework in terms of applicability? Can we find characteristic patterns for the communities in a reasonable time?
- How does the descriptors distribution affect the results?

To answer these questions, we take advantage of random models allowing to generate artificially networks, which we already mentioned in section 3.4. At first, in section 5.1, we describe the models we used in our experiments, including some modifications we proposed. In section 5.2, we compare the results of the four community detection algorithms selected in the previous chapter for our Step 1, in order to identify the most appropriate one. In section 5.3, we explain how one can interpret and detect the community evolution events and track a community thanks to the community patterns. In the last part (section 5.4), we present the experiments we conducted to study the behavior of our framework when applied to different types of nodal attributes.

5.1 Generation of Artificial Networks

As already explained in chapter 3, using artificially generated networks is a common approach to compare different community detection algorithms designed for static networks [43, 76, 98, 101–103]. They rely on some models allowing to generate networks possessing a community structure, in a partially random way. It was shown that the level of realism of the generated topology has an effect on the performance of certain community detection algorithms [101]. In chapter 3, we briefly presented the main models, including the most realistic one to date, which was proposed by Lancichinetti et al. [76] (**LFR**). In this section, we explain in further details how the LFR model works (subsection 5.1.1). However, this model was developed for generating static networks only. Recently, an extension to this LFR model was proposed by Greene et al. [52] to generate *dynamic* networks with predefined evolving community structure. In subsection 5.1.2, we describe how Greene et al. modified LFR to handle time. But for this

work, we also need the network nodes to possess attributes. So, in subsection 5.1.3, we present how to extend the same model even further, in order to generate *attributed dynamic* networks, with both a predefined evolving community structure and nodal attributes.

5.1.1 Original LFR Model

The method proposed by Lancichinetti et al. [76] (LFR) randomly generates plain networks with mutually exclusive communities. Nodes degrees and community sizes are both distributed according to a power law. The model was subsequently extended to generate weighted and/or directed networks, with possibly overlapping communities [45]. This model allows to control directly the following parameters: number of nodes n, desired average $\langle k \rangle$ and maximum k_{max} degrees, exponent γ for the degree distribution, exponent β for the community size distribution, min_c and max_c for minimum and maximum community sizes, respectively, and mixing coefficient μ representing the desired average proportion of links between a node and nodes located outside its community, called inter-community links.

There are two steps in LFR. First, a scale-free network is created by using a random model. Second, a community structure is randomly drawn, and the network is rewired to control the proportion of inter-community links. In the first step, the configuration model [90] is used in order to generate a network containing n nodes, with average degree $\langle k \rangle$, maximum degree k_{max} and a power law-distributed degree, with exponent γ. Then, the second step is applied in two phases. First, the communities are randomly drawn, so that their distribution size follows a power law with exponent β. These are just virtual communities, i.e. groups of nodes, and the topology of the network does not reflect them for now. Second, an iterative process takes place to rewire certain links without changing the node degrees. The rewiring aims to control the proportion of inter-community links of each node according to a user defined parameter called the mixing coefficient μ. It is generally not possible to meet this constraint exactly, and the mixing coefficient is therefore only approximated in practice. Its value determines how clearly the communities are defined. For small μ values, the communities are distinctly separated because they share only a few links, whereas when μ increases, the proportion of inter-community links becomes higher, making community identification a difficult task. The network has no community structure for a limit value of the mixing coefficient given by Equation 5.1:

$$\mu < \frac{(n - n_c^{max})}{n} \tag{5.1}$$

Here n and n_c^{max} are the number of nodes in the network and in the biggest community, respectively [45]. By construction, the LFR method guaranties to obtain values considered as realistic [26, 94] for several properties: size of the network, power law distributed degrees and community sizes. Other properties are not directly controlled, so we studied them empirically in our previous work [98]. It turns out LFR generates small-world networks, with relatively high transitivity. This is realistic [94], but holds only under certain circumstances. In particular, transitivity is dramatically affected by changes in μ, and becomes clearly unrealistic the closer it gets to 1. These properties are directly related to the network structure, which is an important information used by community detection algorithms. Moreover, the sensitivity to μ is also a concern: by increasing its value, not only do the communities become less separated, which is the desired behavior, but the network additionally becomes less realistic.

5.1.2 Extension for Dynamic Networks

The LFR model is designed to generate static networks. Greene et al. [52] propose an extension allowing to generate dynamic artificial networks, with evolving community structures. We call this model **LFR-D** in the rest of this work. The number of time slices is directly controlled through a parameter s. The method starts with the generation of a static network for the first time slice, using the original LFR model,

with the parameters we previously described. For the following time slices, LFR-D alters the community structure by modifiying the network topology through 5 different types of community-related events.

We can separate these events into two categories. First, the events causing direct changes in the community structure, which we call *large-scale events*. They include the birth of a new community, the death of an existing one, the split of a community into several others, the merge of several existing communities into a single one, and the temporary disappearance of a community. Second, the events corresponding to local modifications, not likely to cause important changes in the community structure; which we call *small-scale events*. These include the expansion or contraction of an existing community, and the switch of a few nodes from one community to another. Depending on the type of event, one needs to set up different parameters. Let us list and explain the event types and their parameters, in more details:

- **Switch**: For each time slice, a set of randomly selected nodes are moved into other communities. A parameter p allows to control the proportion of nodes concerned by the switch at each time slice: 0 for no nodes at all, and 1 to switch all nodes in the network. It is possible to combine switch events with other events.

- **Birth-Death**: These events allow to create new communities (birth), or to remove some existing ones (death). They are controlled by two parameters: **birth** and **death** determine the number of communities to be created and to be removed, respectively, at each time slice. For a creation, a group of randomly selected nodes from existing communities are moved to the newly created one. For a removal, the nodes of the concerned community are randomly distributed over the existing ones.

- **Merge-Split**: This event combines two communities into a single one (merge) or separate one community into two smaller ones (split). Two parameters **merge** and **split** allow determining the number of communities to combine and to separate, respectively at each time slice. The communities to merge or split are selected randomly.

- **Expansion-Contraction**: These events increase (expansion) or decrease (contraction) the size of some communities. There are three parameters r, **expand** and **contract**, controlling the rate of expansion/contraction (proportion of nodes joining/leaving the community) and the numbers of communities whose size must be increased/decreased, respectively at each time slice.

- **Hide-Appear**: This event makes a community disappear by turning each of its nodes into a new, separate community. Then, at the following time slice, these nodes gather again to form the original community. The number of communities to hide is determined by a parameter **hide**. Unlike other events, hide events do not occurs at each time slice, but only at a limited number. Both the number of events and the concerned slices are drawn randomly.

Except for the switch type, all those events are considered independent. For instance, LFR-D does not allow to generate a network using both birth-death and merge-split events. Each event, except hide-appear, occurs at each time slice.

5.1.3 Extension for Nodal Attributes

Neither the original LFR model nor extension LFR-D are able to generate networks with nodal attributes. Moreover, we could not find any study focusing on the generation of attributed networks in the literature. This might be due to the fact that the number of attributes, their domains and distribution over the network and the communities could be very system-specific. In order to fill this absence, we propose a relatively simple yet flexible model extending that of Greene et al., able to associate attributes to nodes.

Our goal here is not to deliver a realistic model, but rather to produce some controlled data which we will use to evaluate the performances of our framework. In the rest of the thesis, we call our model **LFR-DA**, and this subsection is dedicated to its description. We control the number of attributes and their values through different parameters. Let us describe our model by explaining their role in the generative process. We note A the attribute set we want to generate. We have five parameters:

- **Number of attributes** $|A|$. An integer value stating the number of attributes to be generated for each node.

- **Attributes Domain** $\mathscr{D}_a$.The domain corresponds to the different values that an attribute can take. We state the domain through two integer values representing its *upper* and *lower* bounds, noted min_a and max_a, respectively. For instance, if $min_a = 2$ and $max_a = 5$, the attribute a can take the values $\mathscr{D}_a = \{2,3,4,5\}$.

- **Distribution type** h. A categorical value controlling how the attribute is distributed over nodes of the same community. We propose 4 different distributions:

  - *Homogeneous*: For a given attribute, all nodes of the same community take the same value from its domain. In other term, the attribute follows a *degenerate distribution* over the community, for a value randomly selected in its domain. In a given communities, the selected value can therefore be different for each one of the considered attributes.
  - *Binomial*: For a given attribute, we first randomly pick a value in its domain, called characteristic value. The values assigned to the nodes are then randomly drawn following a *binomial distribution* centered on the characteristic value. In other words, a significant proportion of the nodes from the same community take the characteristic value, but there also some nodes taking slightly different values.
  - *Power*: For a given attribute, there is a significant number of nodes taking small values, but there are also a non-negligible number of nodes taking high values. The values are distributed over the community nodes according to a *power law*.
  - *Uniform*: For a given attribute, all values of its domain are evenly represented in the considered community. The attribute values are drawn according to the *uniform distribution*.

- **Evolution Percentage** q. This percentage represents the proportion of nodes whose attribute values will change at each time slice.

The procedure to generate the attributes is very simple and flexible. For each attribute a, we first generate its domain by respecting the specified bounds min_a and max_a. Then, for each community of the first time slice, we generate the specified number $|A|$ of attribute values for each node, by respecting the specified distribution h. For the following time slices, for each community, we randomly select q nodes and change all their attribute values randomly, by respecting the domains and distribution of each attribute.

5.1.4 Summary of the Parameters

The LFR model and the two successive extensions we use in this work end up relying on a number of parameters. In order to be more comprehensible, we list and group these parameters here in table 5.1. In the dynamic extension, there are many different parameters, but most of them have the same role for different types of events. Thus, we summarize them in a more comprehensible way.

Parameter Symbol	Meaning	Generation Role
n	Node number	Static Network Generation
$\langle k \rangle$	Average Degree	Static Network Generation
k_{max}	Maximum Degree	Static Network Generation
γ	Degree Distribution Exponent	Static Network Generation
$[min_c, max_c]$	Community Size Range	Community Structure Generation
β	Community Size Distribution Exponent	Community Structure Generation
μ	Inter-Community Links Proportion	Community Structure Generation
s	Number of Time Slice	Dynamic Network Generation
e	Community Evolution Type	Community Evolution Event
C	Number of Communities to Evolve	Community Evolution Event
p	Fraction of Nodes in the Network to Change Position	Community Evolution Event
r	Fraction of Expansion-Contraction	Community Evolution Event
$\|A\|$	Attribute Number	Attribute Generation
$[min_a, max_a]$	Upper and Lower Bounds of the Attribute Domain	Attribute Generation
h	Distribution type	Attribute Generation
q	Percentage of Nodes in Each Community whose Attributes will evolve	Attribute Generation

Table 5.1 Summary of Generation Parameters

5.2 Comparing Community Detection Algorithms in Dynamic Networks

Although our community interpretation framework is independent from the community detection method, it is still necessary to have a reliable community structure for the relevancy of the analysis. And, as mentioned in chapter 3 there are several community detection algorithms for dynamic networks, differing in various ways. Noticeably, some produce consensual community structures (relevant for all time slices) while some others produce different community structures for each time slice. Some authors [69, 138] used artificial networks to test their own algorithms, but the generative models they use are rather unrealistic compared to LFR-D, i.e. the dynamic extension of the LFR model presented in the previous section. Moreover, none of these works compare several algorithms: instead, they are dedicated to analyzing the performance of the sole proposed algorithm. In chapter 4, we selected four methods for community detection in dynamic networks: *Classic Louvain*, *Incremental Louvain*, *MultiStep Louvain* and *Integrated Louvain*. Here, we want to analyze their partitioning performance on some data generated through LFR-D.

The method of our analysis is simple: we generate networks with LFR-D (so without any attributes, since they are not needed by the considered community detection algorithms). Thus, we have an evolving community structure considered as the ground truth. We then apply these four algorithms on the generated dynamic networks and evaluate their partitioning quality by using the NMI as a performance measure. *Classic Louvain* and *Incremental Louvain* produce evolving community structure for each time slice. We compare their result of each time slice with the reference community structure at each time slice. However, *MultiStep Louvain* and *Integrated Louvain* estimate consensual community structures over all time slices. We compare their result with each time slice reference structure. In the following part, we at first explain which parameter values we use for network generation in section 5.2.1, and then we describe and comment our results in section 5.2.2.

5.2.1 Parameter Values

Due to the high number of parameters and the size of their domains, it would not be possible to generate data for all possible combinations of parameters, so we had to make some choices. We decided to examine the behavior of the four considered algorithms on two scenarios: first, when changing the community mixing (controlled by the mixing parameter μ of LFR), i.e. when considering more or less clearly separated communities; and second, when changing the nature and magnitude of the community evolution (controlled through the parameters of LFR-D). For both scenarios, some of the parameters are fixed: those are listed in Table 5.2. We decided the values of those related to static network and community structure generation in the light of our previous experiments [98, 101–103]. For the number of time slices, we fixed the value $s = 10$.

Parameter Symbol	Value
n	10000
$\langle k \rangle$	10
k_{max}	1000
γ	3
$[min_c, max_c]$	$[3, 1000]$
β	2
s	10
e	$\{$*switch, birth-death, merge-split, expansion-contraction, hide-appear*$\}$

Table 5.2 Common Parameter Values for All Network Generations

For the first scenario, we use the parameter values given in table 5.2 with 3 different values of μ corresponding to networks with 3 levels of community structure separation: ***separated*** ($\mu = 0.3$), ***medium*** ($\mu = 0.6$) and ***mixed*** ($\mu = 0.9$). The value 0.9 corresponds to the limit of existence of a community structure, according to equation 5.1. Thus, we do not expect good performances for this value of μ. Considering the evolution of communities, we use the default values of LFR-D parameters, which are $C = 1$, $p = 0.1$ and $r = 0.1$.

For the second scenario, we use again the values from table 5.2. We additionally fix $\mu = 0.3$ (equivalent to the separated level) and adjust the parameters related to community evolution events. Our aim is to observe the behavior of the four algorithms at four different levels of evolution: ***low***, ***medium***, ***medium-high*** and ***high***. The parameter values for these 4 types of networks are given in table 5.3. We should indicate that the number of evolving communities, given in table 5.3, is chosen according to the total number of communities in the generated networks, according to the parameters of table 5.2, which is 554. Note that the evolution of the attributes in this experiment is at the ***low*** level. For this reason, the results we present in this subsection also are used in the next section 5.4 when comparing evolution levels.

	low	**medium**	**medium-high**	**high**
C	1	20	100	400
p	0.1	0.2	0.5	0.9
r	0.1	0.1	0.1	0.1

Table 5.3 Community Evolution Parameter Values for Different Network Types

5.2.2 Results

We evaluate first the algorithms performance according to the scenario focusing on community separation, as described in subsection 5.2.2.1. We then discuss the performance obtained with the second scenario, when we change the nature of community evolution, as explained in section 5.2.2.2. For all experiments, in order to achieve a certain statistical stability, we generate 10 networks for considered set of parameter values. We then apply the four algorithms on each generated network. We process the NMI values to compare the reference and estimated community structures at each time slice, for each algorithm, and for all 10 iterations. In order to get an overall performance score, we average the NMI over the 10 iterations. Plots of the obtained results show the NMI means and standard deviations for each algorithm.

5.2.2.1 Effect of Community Separation

Figure 5.1 shows the NMI results on the networks generated with five different community evolution events, when the communities are *separated*. In the plot, the x axis represents time, the y axis represents the mean NMI, and the colors represent the algorithms.

We can notice that for all evolutionary events except *hide-appear*, Classic Louvain and Incremental Louvain both exhibit a stable performance to the community evolutions, with NMI values ranging from 0.85 to 0.90. These results are high enough to conclude these two algorithms find community structures similar to the reference ones for each time slice. However, for the *hide-appear*, we see the performances of both algorithms are less stable than for other events. Moreover, Classic Louvain undergoes a performance drop at time $t = 8$, associated to a high standard deviation, which highlights the weakness of this algorithm. Let us remind the difference between Classic Louvain and Incremental Louvain: the latter

applies the *temporal smoothness* principle, and uses the community structure of the previous time slice as the base to find the community structure of the following time slice. But the former, finds the community structure from scratch, independently at each time slice. It seems when the Louvain algorithm does not use the evolutionary information, it cannot catch *hide* type communities, even if these communities are *separated* in terms of their separation. In contrast, when the algorithm uses the information of previous time slices, it is more stable.

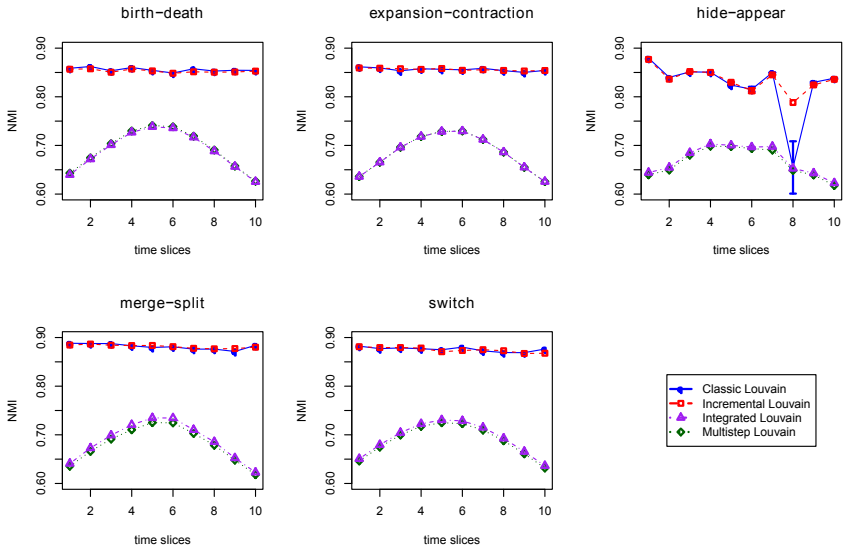

Fig. 5.1 NMI results on networks generated with different community evolution events, when $\mu = 0.3$.

As it can be seen in Figure 5.1, the NMI results of Multistep and Integrated Louvain appears as curved lines, which at first increase until $t = 5$ and then decrease with time. The values range from 0.65 to 0.75. These NMI values are still good, but show that estimated communities are not as close from the reference ones as with the other algorithms. These two algorithms find a consensual community structure, which is supposedly relevant for all time slices. Multistep Louvain tries to optimize average modularity while Integrated Louvain is original Louvain applied to the integrated network. It seems that, in terms of partitioning performance, these two approaches do not differ very much. While the community structure they find is less similar to the reference structure at beginning and end of the considered time period, it is more similar in the middle of this period. Due to the way the generative model works, the community structure at time t is likely to be similar to those of the previous $(t-1)$ and following times $(t+1)$. Thus, those located in the middle of the time range are similar to more of them, which means they have more weight in the temporal integration processes these algorithms rely upon. Because of this higher weight, they affect more the consensual community structures outputted by these tools.

In Figure 5.2, we represent the NMI results when the community separation level is *medium*, i.e. $\mu = 0.6$. This means the proportion of links inside communities is approximately 0.4. The communities have more links outside, but they can still be detected, in theory. In terms of algorithm results, the general trend is the same than what was observed for *separated* community structures. Classic Louvain is stable except for *hide-appear* events. Incremental Louvain stays stable for all event types. Multistep and Integrated Louvain exhibit the same curved type of performance than before. However, this time, the NMI values for Classic and Incremental Louvain are much lower, ranging from 0.45 to 0.50 for the different event types. These values tell us these two algorithms cannot find a good community structure when they are not well separated.

When the community separation level is *mixed*, in general, the performance trends of the algorithms do not change. Figure 5.3 shows the NMI values obtained for these data. However, we remark that

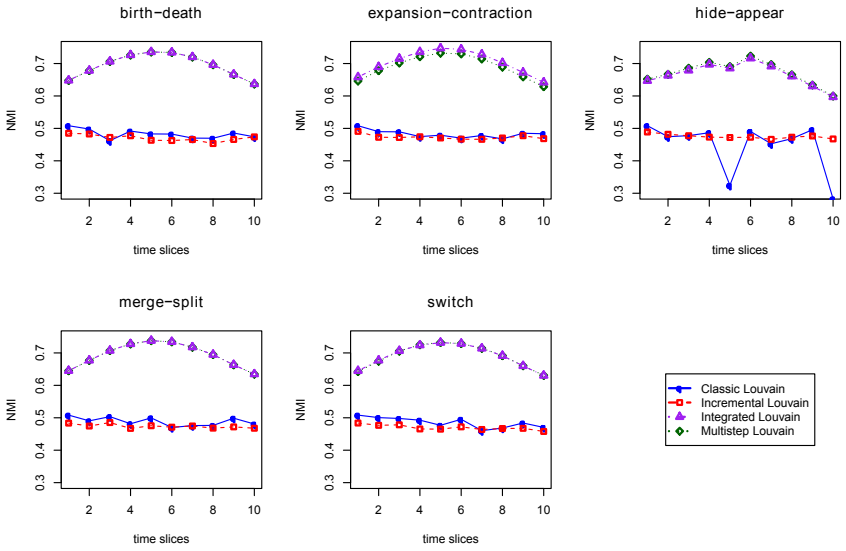

Fig. 5.2 NMI results on networks generated with different community evolution events, when $\mu = 0.6$.

Classic Louvain becomes very unstable for all event types. Incremental Louvain is also very unstable for the *hide-appear*. The differences between Integrated and Multistep Louvain become more separated. But, here, there is a more important point to comment: the NMI values for all algorithms are much lower than before: smaller than 0.20. Thus, the estimated community structures are very different from the reference ones. Note that, in these networks, the community separation level is *mixed*. We expect the algorithms to perform reasonably well. However, it is still interesting to see that the algorithms keep their general trend.

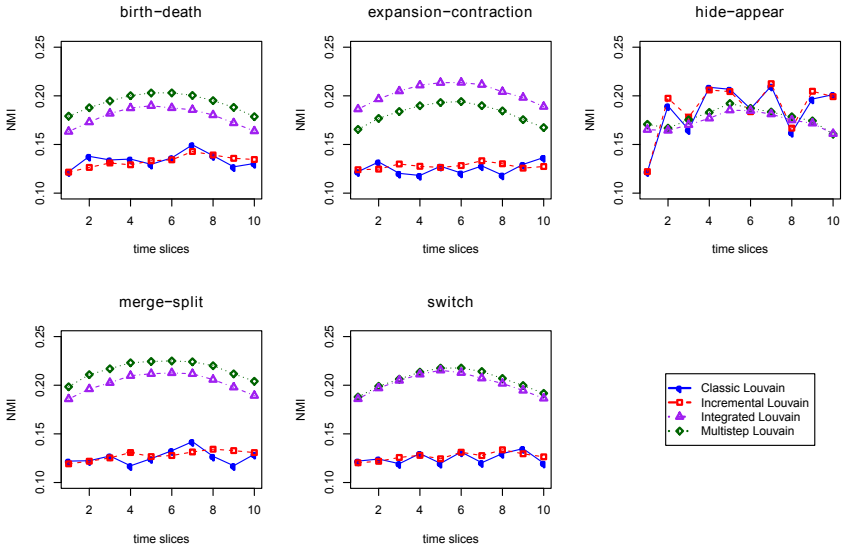

Fig. 5.3 NMI results on networks generated with different community evolution events, when $\mu = 0.9$.

5.2.2.2 Effect of Community Evolution

Here, we present the performance of the four algorithms when we change the community evolution parameters. As declared in section 5.2.1 we defined four levels of community evolution: ***low***, ***medium***,

medium-high and ***high***. The first value was already illustrated by the plots from the previous section (low evolution level). For the other three values, we present the performance of the four algorithms one by one, on the networks generated with the different community evolution events.

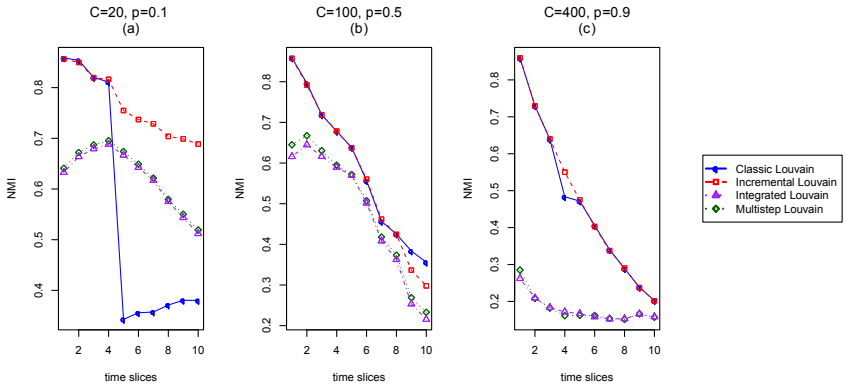

Fig. 5.4 NMI results on networks generated with evolution event type birth-death when $\mu = 0.3$. Evolution levels for plots (a), (b) and (c) are *medium*, *medium-high* and *high* respectively

Figure 5.4 shows the performances for *birth-death*. When the level of community evolution is *medium* (figure (a)), we see that the performance of Incremental Louvain decreases almost linearly while Classic Louvain exhibits a sudden drop at $t = 4$. The higher the number of changing communities, the faster the performance of two algorithms decreases. Although using the temporal information improves Incremental Louvain stability over time, it is still not enough to find a good community structure for all time slices. One reason for this performance decrease can be that the birth-death affects other properties of the network. A more detailed review based on the properties of the generated networks shows us that we have more isolated nodes not belonging to any community as time goes by. This is probably due to the death event. It might be a cause for the decreasing performance of these two algorithms. For the two other algorithms, Multistep and Integrated Louvain, the NMI performance values are not good for any case of birth-death events. As the number of evolving communities increases, the performance of these two algorithms becomes more stable, but the NMI values decrease. These two algorithms do not seem to be able of finding good community structure in birth-death networks.

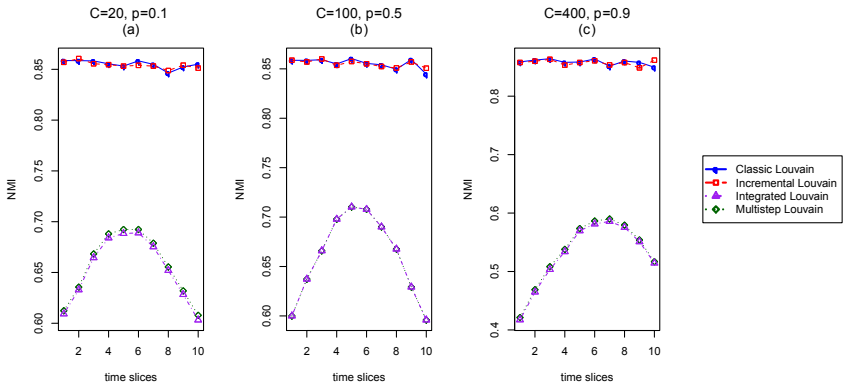

Fig. 5.5 NMI results on networks generated with evolution event type expansion-contraction when $\mu = 0.3$. Evolution levels for plots (a), (b) and (c) are *medium*, *medium-high* and *high* respectively

We can observe the performance of the four algorithms on networks generated with *expansion-contraction*, in Figure 5.5. Classic and Incremental Louvain seem to be affected neither by time nor

the number of nodes expanding or contracting. We can assume that when the communities already exist, it does not matter whether they grow up or shrink with time, for these two algorithms. Their performance is stable and good. For Integrated and Multistep Louvain, we see again a curved line, with clearly lower NMI values than for both other algorithms.

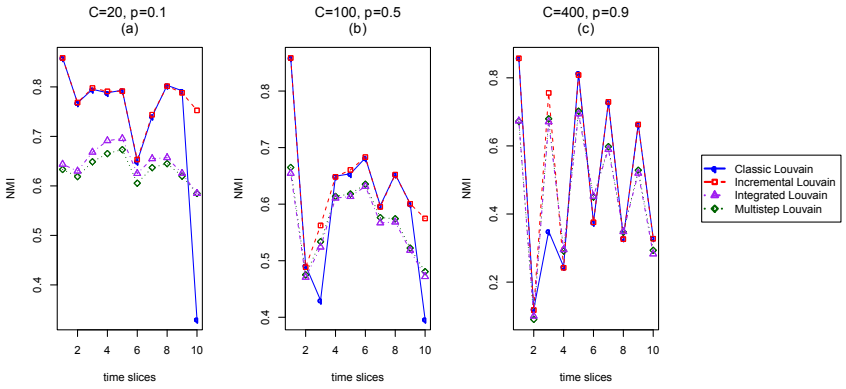

Fig. 5.6 NMI results on networks generated with evolution event type hide-appear when $\mu = 0.3$. Evolution levels for plots (a), (b) and (c) are *medium*, *medium-high* and *high* respectively

The performances obtained for *hide-appear* are represented in Figure 5.6. As it seems, none of the algorithms display a stable performance. This instability becomes more separated when the number of hiding communities increases. When $C = 20$, almost 4% of the communities *hide-appear* at randomly selected time slices. Incremental Louvain seems to get higher NMI and more stable performances in this case. But, it is still not completely stable, with a sudden drop at $t = 6$. The stability is the lowest with $C = 400$ for all algorithms. A more detailed study of the result sets shows us that at the drop points, in the reference structure, there are many communities of size 1. Thus, when a community hides, some of its nodes create their own community without any other member. However, their nodes are placed in some of the other existing communities by all four algorithms. It naturally causes a drop in NMI.

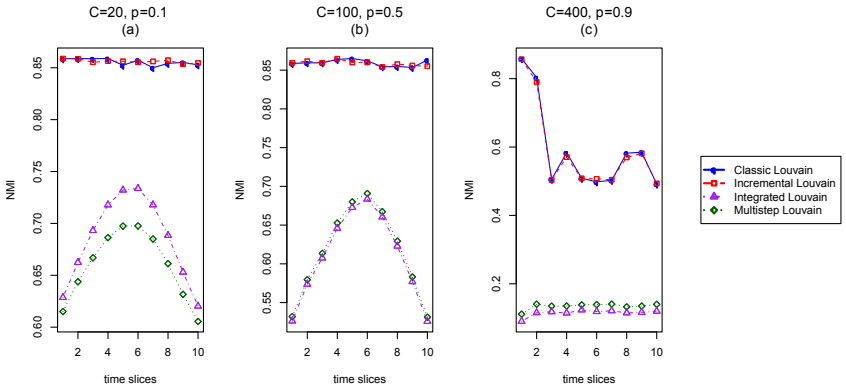

Fig. 5.7 NMI results on networks generated with evolution event type merge-split when $\mu = 0.3$. Evolution levels for plots (a), (b) and (c) are *medium*, *medium-high* and *high* respectively

We present the performances obtained with *merge-split* in Figure 5.7. All the algorithms exhibit a trend similar to that of the *expansion-contraction* networks, until the number of merging/splitting communities reaches 400. In this case, at each time slice, the majority of communities merge/split, and the community structure completely changes. A detailed review on the properties of communities and network at consecutive time slices shows us that both degree distribution and community size distribution

change dramatically. It can be the reason for the instability and drop in performance of Classic and Incremental Louvain.

Finally, for *switch*, we show the results in Figure 5.8. Incremental and Classic Louvain are not affected by the number of switching nodes, while the performance of Multistep and Integrated Louvain decreases. An overall observation of our results gives us a general idea of the performance of algorithms. The Multistep and Integrated Louvain are not producing a good community structure, for any event type. Moreover, they are very sensitive to large-scale events. Their performance is in general lower, and decreases when the number of evolving communities increases. Incremental and Classic Louvain obtain better performances than the other two algorithms. They seem to not be affected by small-scale events although they are affected by large-scale ones.

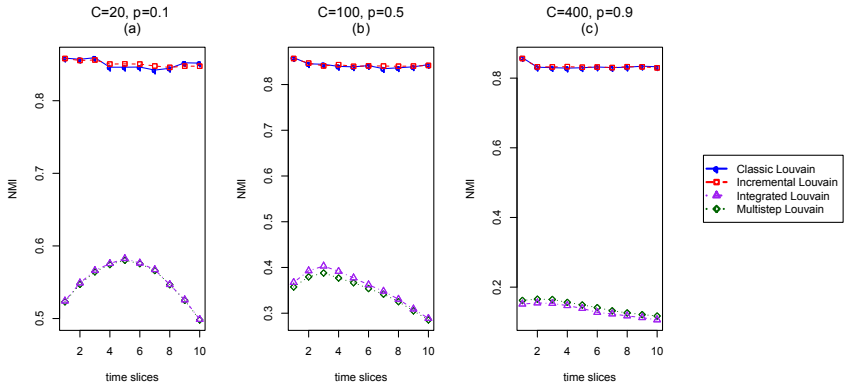

Fig. 5.8 NMI results on networks generated with evolution event type switch when $\mu = 0.3$. Evolution levels for plots (a), (b) and (c) are *medium*, *medium-high* and *high* respectively

5.2.2.3 Summary

Overall, our results show that one method (consensual vs. evolving community structures) does not completely dominate the other. In some cases, the algorithms finding evolving community structures (Classic and Incremental Louvain) obtain better results, whereas in some other cases those looking for consensual community structures (Multistep and Integrated Louvain) obtains better results. Among the former, Incremental Louvain is less sensitive to community events when the network is stable. The most difficult events for both these algorithms are *hide-appear*, *birth-death* and *merge-split*, in decreasing order of difficulty. For *switch* and *expansion-contraction*, both Incremental and Classic Louvain get good performances. Both other algorithms results are not good enough to conclude they are able to find a reliable community structure in general.

We observe a general decrease in performance when the level of community separation decreases. Algorithms finding evolving community structure seem to get good performances when the communities are well separated. However, algorithms finding consensual community structure seem less sensitive to the different levels of community separation. Among the algorithms detecting evolving community structures, here again, Incremental Louvain is more stable to the different evolution events. Its difference with Classic Louvain shows that taking advantage of the temporal information indeed improves the results. However, this information must be used in an appropriate way: the aggregation processes used by Integrated and MultiStep Louvain lead to poor performances. From our results, we can conclude that for the considered dynamic networks, communities should be detected though the use of local temporal information, which shows the relation between consecutive time slices.

Of course, these observations are not generalizable, because we considered only a reduced number of algorithms, and very basic generated data. But the point of this work was not to identify the best

community detection algorithm for dynamic networks, at task which would not even be theoretically sound. Instead, we wanted to evaluate the difference between a small numbers of selected algorithms, in order to pick the most appropriate one for testing our framework.

5.3 Finding Community Evolution Events

In subsection 4.4.1, we explained that our framework can also be used to follow major community evolution events. We want to show the relevance and the reliability of this approach here. For this reason, we at first generate networks with LFR-D, using the parameter values from table 5.2, as well as $C = 20$ and $p = 0.2$. Thus, we have a ground truth community structure for each time slice. Then, we create a database including only the community sequences. In this database, every node sequence has 10 itemsets, corresponding to 10 time slices. We apply CloSpan on this data set with minimum support 0.001. Here, we present only a few examples to show how to interpret this kind of results. However, it confirms that, by using closed pattern mining, we are able to catch all types of events for the communities representing a proportion of the network node greater or equal to the minimum support. Moreover, it is also possible to catch the important times t corresponding to remarkable evolution points. In the following subsection, we first present some of the interesting results for each type of considered evolution event, then we concentrate on how we can track the evolution of one community in particular.

5.3.1 Detecting Evolution Events

5.3.1.1 Birth-Death Evolution

After the application of CloSpan on the data set of birth-death event types networks generated with the parameter values indicated before, we have found 6649 patterns. Among them, we focused on patterns of size 10, which correspond to the considered time period, and are shown in Table 5.4. Pattern #1 in the table shows a group of nodes whose community ID changes at each time slice. All 793 of these nodes stay together, i.e. in the same community, for the whole time period of 10 time slices. By manually comparing the nodes of each one of the communities appearing at different time slices, with the nodes supporting this pattern, we see that these nodes correspond to the core part of an evolving community.

ID	CFS	Support	Size
1	<(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)(C182@6)(C175@7)(C168@8)(C163@9)(C155@10)>	793/10000	10
2	<(C435@1)(C420@2)(C404@3)(C391@4)(C0@5)(C0@6)(C0@7)(C0@8)(C0@9)(C0@10)>	598/10000	10
3	<(C121@1)(C116@2)(C0@3)(C0@4)(C0@5)(C0@6)(C0@7)(C0@8)(C0@9)(C0@10)>	189/10000	10

Table 5.4 Some Interesting Patterns Showing Birth-Death Events

For pattern #2, we see that, starting from $t = 5$, the community ID becomes 0. All 598 nodes sharing the same community until $t = 5$ become alone (i.e. each node constitutes its own community) afterwards, until the end. We understand that they were a part of a group of nodes all belonging to the same community from $t = 1$ till $t = 4$. Then, this group died at $t = 5$ (become community-less). Pattern #3 represents a group of 189 nodes staying together until $t = 3$. Then, they all get alone, too.

5.3.1.2 Expansion-Contraction Evolution

ID	CFS	Support	Size
1	<(C224@1)(C224@2)(C224@3)(C224@4)(C224@5)(C224@6)(C224@7)(C224@8)(C224@9)(C224@10)>	795/10000	10
2	<(C227@1)(C227@2)(C227@3)(C227@4)(C227@5)(C227@6)(C227@7)(C227@8)(C227@9)(C227@10)>	156/10000	10

Table 5.5 Some Interesting Patterns Showing Expansion-Contaction Events

For the networks generated using the expansion-contraction event types, we have 6814 patterns. For this specific type of events, the communities generated by LFR-D keep their ID over the while time period, which ease our study. As mentioned before, we consider expansion-contraction as small-scale. They are not related to whole communities appearing or disappearing, but rather to increases or decreases in their sizes. Thus, to interpret these patterns, we at first find out the patterns showing the core of the communities. These patterns are the ones lasting over the whole time period, i.e. 10 time slices in our case. In table 5.5, we list two patterns of this type (constant community ID). Note that this manipulation is relevant only for the expansion-contraction type of evolution. It seems 795 nodes of C224 and 156 nodes of C227 stay together during all time slices. To find out if any of those communities are expanded or contracted, we track their evolution in detail by studying these community IDs at each time slice alone. More clearly, we check if there is any closed pattern of size 1 containing these community IDs.

Community ID	CFS	Support	Size	Event
C224	<(C224@1)>	814/10000	1	stable
C224	<(C224@2)>	813/10000	1	stable
C224	<(C224@3)>	888/10000	1	expand
C224	<(C224@4)>	888/10000	1	stable
C227	<(C227@2)>	182/10000	1	stable
C227	<(C227@3)>	183/10000	1	stable
C227	<(C227@4)>	183/10000	1	stable
C227	<(C227@4)(C227@5)>	178/10000	2	contract
C227	<(C227@6)>	177/10000	1	stable
C227	<(C227@8)>	169/10000	1	contract

Table 5.6 Track Results of Community ID 224 and 227

We list the result of this tracking in Table 5.6. We see that there are 814 nodes in C224 at $t = 1$. At $t = 2$, one node leaves this community. One node is not enough to conclude there was a contraction event (it could be a simple switch), so we consider the community stayed stable. However, there are 888 nodes in C224 at $t = 3$. This means at least 75 new nodes joined this community between $t = 2$ and $t = 3$. Thus, we can say that C224 undergoes an expansion event at $t = 3$. Note that these 75 joining nodes might belong to one community from previous time slices or different node groups from 75 nodes might belong to many different communities. Thus, this expansion can be a conclusion of a merge event as well. To understand such details about evolution, one needs to observe all related CFS and their supporting nodes. When we observe C227, we see that from $t = 1$ to $t = 4$, there is no remarkable event. In the closed patterns, we do not find any pattern containing C227 with at $t = 5$. There can be two possibilities

for this: first, the support of the pattern <(C227@5)> does not exceeds the minimum support; second, there is a super sequence of <(C227@5)> with the same (or higher) support. We see that the second possibility occurs in our dataset, because the pattern <(C227@4)(C227@5)> is supported by 178 nodes. According to these support values, we can conclude that from $t = 4$ to $t = 5$, at least 5 nodes left C227. Thus, this community underwent a contraction event at time slice 5.

5.3.1.3 Hide-Appear Evolution

ID	CFS	Support	Size
1	<(C224@1)(C180@2)(C165@3)(C128@4)(C105@5)(C0@6)(C425@7) (C349@8)(C287@9)(C240@10)>	782/10000	10
2	<(C435@1)(C346@2)(C302@3)(C245@4)(C209@5)(C0@6)(C433@7) (C356@8)(C293@9)(C0@10)>	736/10000	10
3	<(C222@1)(C178@2)(C163@3)(C126@4)(C103@5)(C0@6)(C0@7) (C457@8)(C381@9)(C320@10)>	444/10000	10
4	<(C121@1)(C0@2)(C442@3)(C0@4)(C439@5)(C360@6)(C308@7) (C258@8)(C211@9)(C178@10)>	272/10000	10
5	<(C542@1)(C433@2)(C371@3)(C306@4)(C260@5)(C220@6)(C0@7) (C393@8)(C326@9)(C274@10)>	224/10000	10

Table 5.7 Some Interesting Patterns Showing Hide-Appear Events

We mined 6079 patterns on the networks generated based on the hide-appear event type. We list some patterns showing the communities that hide-appear at certain times, in table 5.7. For instance, pattern #1 corresponds to a group of 782 nodes which all belong the same community for the whole time period. A manual comparison of these nodes and of the communities whose IDs are contained in the pattern showed these 782 nodes constitute the core of an evolving community. This core, and therefore the community itself, disappears at $t = 6$ (the ID becomes 0) at $t = 6$. The core (and the community) is back starting from $t = 7$ until the end. Similar hide events van be observed for the other communities listed in table 5.7.

5.3.1.4 Merge-Split Evolution

For the networks based on merge-split event types, we find 6993 patterns. We list some examples in Table 5.8: let us start with the first three ones. They all contain the same community IDs until $t = 4$, which means they form a single group. At this point, the first pattern shows a first separation into two distinct groups: one is contained in community C533, while the other (patterns #2 and #3) belongs to community C534. This group again splits at $t = 9$, as shown by the second and third patterns.When considering the last two rows of the same table, we can observe a merge event. The two groups supporting these patterns are in two distinct communities until $t = 4$. At this point, they enter the same community until the end of the time period.

5.3.2 Tracking a Specific Community

One may want to focus on a specific community of interest, in which case he would want to follow its life cycle. Here, we present how we can track one community evolution by studying its related patterns. The principle is simple: we first select the most frequent pattern of size 1 and whose item corresponds

ID	CFS	Support	Event
1	<(C222@1)(C397@2)(C367@3)(C533@4)(C497@5)(C461@6)(C425@7)(C391@8)(C364@9)(C340@10)>	215/10000	
2	<(C222@1)(C397@2)(C367@3)(C534@4)(C498@5)(C462@6)(C138@7)(C128@8)(C517@9)(C480@10)>	113/10000	**Split**
3	<(C222@1)(C397@2)(C367@3)(C534@4)(C498@5)(C462@6)(C138@7)(C128@8)(C518@9)(C481@10)>	103/10000	
4	<(C19@1)(C14@2)(C517@3)(C211@4)(C200@5)(C181@6)(C164@7)(C151@8)(C142@9)(C133@10)>	12/10000	**Merge**
5	<(C276@1)(C252@2)(C232@3)(C211@4)(C200@5)(C181@6)(C164@7)(C151@8)(C142@9)(C133@10)>	65/10000	

Table 5.8 Some Interesting Patterns Showing Merge-Split Events

to the community of interest. To identify the ID of the community in the second time slice, we just need to select the most frequent pattern of size 2 whose first item is the ID of the community of interest. The second item is then the ID of the same community at $t = 2$. By repeating the same process for the next time slices, we can efficiently track the community until the end of the time period.

ID	CFS	Support	Size
1	<(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)(C182@6)(C175@7)(C168@8)(C163@9)(C155@10)>	793/10000	10
2	<(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)(C182@6)(C175@7)(C168@8)(C163@9)>	795/10000	9
3	<(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)(C182@6)(C175@7)(C168@8)>	796/10000	8
4	<(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)(C182@6)(C175@7)>	801/10000	7
5	<(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)>	802/10000	5
6	<(C224@1)(C216@2)(C208@3)(C201@4)>	804/10000	4
7	<(C224@1)(C216@2)(C208@3)>	810/10000	3
8	<(C224@1)(C216@2)>	811/10000	2
9	<(C224@1)>	814/10000	1

Table 5.9 Full Tracking of a Community with Related Patterns

In table 5.9, we present some patterns showing the life cycle of a community from a network generated using the birth-death evolution type. In the first row, we see the longest pattern has a size of 10 (full time period). On the last row, we see that there are 814 nodes in the same community at $t = 1$. Then, the number of supporting nodes, i.e. the number of nodes staying together, decreases from 811 to 793 between $t = 2$ and $t = 10$. Note that 810 of the initial nodes stay together until $t = 4$. But from $t = 4$ to $t = 5$ (rows #6 and 5 in the table), 6 nodes leave the community. Similarly, from $t = 7$ to $t = 8$ (rows #4 and 3 in the table), 5 nodes leave the same community. Although this network was generated with a birth-death type of evolution, we still observe a contraction for this community (probably due to switches). Another remark we want to underline is that from $t = 6$ to $t = 7$, this community does not change because there is no closed pattern like <(C224@1)(C216@2)(C208@3)(C201@4)(C190@5)(C182@6)> in the results. This means that the support of this pattern is the same than for pattern #4. Because pattern #4 is one of

its super-sequence, the pattern is not detected as a closed one.

Briefly, we can infer that this community contracts as time goes by. Here, we present only one example, however, it is possible to track any communities with this interpretation method. Although this way seems exhaustive, one can always increase or decrease the rate of detail to be observed. Finding Closed Patterns on Community IDs by using time slice numbers shows us all existing community events. We can adjust the range of these events in terms of node number, thanks to the minimum support parameter used when mining patterns.

5.4 Descriptor-Related Effects on Community Interpretation

In this section, we focus on the effect of descriptors on our method, especially in terms of scalability. Without loss of generality, we focus only on the attributes, but our conclusions are still valid for the topological measures. We perform three different experiments to analyze the behavior of Steps 3 and 4 of our framework, on networks including nodal attributes. We first generate networks using the LFR-DA extension, with the parameters values given in Table 5.10. During the attribute generation process, in first time slice, we respect the same distribution type for all attributes. Thus, it is not possible to generate one attribute in binomial distribution and another one in power law distribution. We made this choice for simplicity matters.

The values of the attribute-related parameters are described in table 5.11. For all three experiments, we used the same range for the attributes, i.e. integers between 0 and 9.

Parameter Symbol	**Value**
n	5000
$\langle k \rangle$	10
k_{max}	500
γ	3
$[min_c, max_c]$	$[3, 500]$
β	2
μ	0.3
s	10
e	{*switch, birth-death, merge-split, expansion-contraction, hide-appear*}
C	20
p	0.2
r	0.1

Table 5.10 LFR-DA Generation Parameter values

In the *first* experiment, we study the effect of the number of attributes. The attribute values are generated so that they are completely homogeneous inside of each community (*degenerate* distribution). Moreover, these values are not affected by time. In the *second* experiment, our aim is to see how our framework is affected by changes in the evolution percentage of attributes. For this reason, we fix the attribute number at three, again with a *degenerate* distribution, and modify only the percentage of nodes whose attributes evolve (q). Finally, in the *third* experiment, we change the distribution type of the attribute values (h), whereas the number of attributes and the attribute evolution are fixed at 3 and 5, respectively.

For each experiment, we create the node sequences, including the generated attribute values and ground truth community IDs, at each time of the considered period. We then apply CloSpan (Step 3 of

	Experiment 1	Experiment 2	Experiment 3
$\|A\|$	$\{1,3,5\}$	$\{3\}$	$\{3\}$
$[min_a, max_a]$	$[0,9]$	$[0,9]$	$[0,9]$
h	$\{degenerate\}$	$\{degenerate\}$	$\{degenerate,binomial,uniform,power\ law\}$
q	$\{0\}$	$\{5,20,50,100\}$	$\{5\}$

Table 5.11 Attribute Generation Parameter Values for Different Experiments

our framework) with $min_{sup} = 0.002$, which means the identified patterns should be supported by at least 10 nodes. Then, we process the growth rate by applying Step 4.B of our framework. In the following part, we show and discuss the results and performance of our tool.

5.4.1 Effect of the Number of Attributes

We represent the results of Experiment #1 with four plots in Figure 5.9.Plot (a) shows the execution times of Step 3 (application of CloSpan). The execution times of CloSpan on these networks increase dramatically when $|A| = 10$: it does not produce any result after having run for 3 days. For this reason, we did not use more than 5 attributes. When we increase the number of attributes from 1 to 5, the execution time of CloSpan also increases, for all types of events. In Chapter 4, we indicate that the complexity of the second step of CloSpan depends on the size of the dataset expressed in nodes. In practice, here, we see that overall complexity is also affected by the number of descriptors that we treat. The highest the number of descriptors, the longer the CloSpan process. The number of descriptor obviously affects the number of candidate sequences, thus, it naturally impacts the execution time. If we order the execution times in function of the type of evolution used for network generation, we get (in descending order): birth-death, hide-appear, merge-split, expansion-contraction and switch. So, when the community evolution undergoes small-scale events (i.e. communities do not appear or disappear, but they see their sizes change, or nodes switch between communities), mining the patterns requires less computation.

On the same figure, the plots (b) and (c) display the number of patterns found by CloSpan and the proportion of community-related patterns among them, respectively. Let us discuss them together. For experiment #1, the number of patterns increases with the number of attributes, which could be expected. Indeed, the number of possible non-closed sequences increases exponentially with the number of descriptors. We remark that most of those patterns (at least 93 %) are community-related. This could also be expected, because in this experiment, all the nodes of the same community have the same attribute values (degenerate distribution). So, a community-independent sequence is rarely closed, because there often exists a community-related super-sequence with the same numbers of nodes. The percentage of community-related patterns decreases when the number of attributes increases, though. This actually means that the number of community-related patterns increases slower than that of community-independent patterns. Indeed, when multiplying the number of attributes, the chance to see frequent patterns appearing in the whole network (without regards for community) is higher than that of having frequent patterns in specific communities. If we order the numbers of patterns identified, in function of the type of event used to generate the networks, we get (in ascending order): hide-appear, birth-death, merge-split, switch and expansion-contraction. Here, the most interesting point is about hide-appear events, for which the number of community-related patterns is clearly higher than for other events. When a network hides a community, all its nodes become their own community. For all of these communities of size is 1, we use the common ID 0, instead of giving them distinct community IDs. After hide events, this causes many nodes to have the same ID, and constitute a very large pseudo-community. This in turns

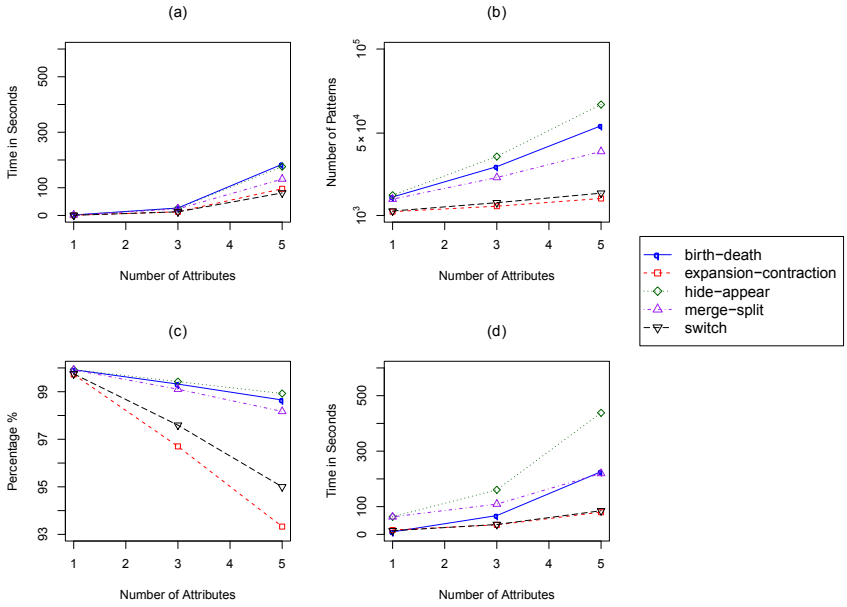

Fig. 5.9 Results of Experiment 1. Each plot shows the results for a different number of attributes. (a),(b),(c) and (d) show the execution time of CloSpan (Step 3), the total number of patterns found by CloSpan, the percentage of community-related patterns among all detected patterns and the execution time for post-processing (Step 4.B) respectively

causes CloSpan to extract numerous additional community-related patterns. This could be avoided by using distinct community IDs , but it would cause other practical problems.

The plot (d) shows the execution times of Step 4.B, i.e. the post-processing and the calculation of the growth rates. We can say that it clearly increases with the number of attributes, like for CloSpan, and even faster. This confirms our analytical estimation of the algorithmic complexity of this Step 4.B, which depends on the number of nodes, the number of time slices and, most of all, the total number of patterns fetched by CloSpan (c.f Chapter 4.). The longest computation times are obtained for the hide-appear event types. When we also observe the percentage of community-related patterns and the total number of patterns, we see that the values are higher for hide-appear event type, especially for large numbers of attributes. In the post-processing, we check each pattern. If they are community-related, we treat them. Checking a pattern does not consume too much time comparing to its treatment. For hide-appear events, there are more community-related patterns. Thus, this is naturally more costly in terms of post-processing. As a result, its post-processing is also higher than the other types of events. Note that none of those post-processing takes more than 400 seconds. Nevertheless, as mentioned before, we were not able to produce results for 10 attributes.

5.4.2 Effect of the Attribute Stability

The results of Experiment #2 are shown in Figure 5.10, with a plot placement similar to that of figure 5.9. We see that the fastest CloSpan processing is obtained for $q = 5$, i.e. when the attribute values of 5% of the nodes change at each time slice: it takes less than 20 seconds to identify the CFS. The computation time increases very irregularly for higher values of q. When $q = 20$, the execution time is the highest for large-scale events, whereas it is $q = 100$ for small-scale events. Note that changing q not only causes a change in the attribute evolution, but as a side effect, it also affects the distribution of attribute values inside the communities. The higher q, the earlier the communities become heterogeneous, in terms of attribute distribution. When $q \geq 50$, more than half the nodes see their attributes randomly

changed at each time slice. The attribute distribution therefore becomes more and more uniform (hence heterogeneous). A higher heterogeneity in the distribution of the attribute values should affect CloSpan negatively in terms of computational time. Indeed, if all the nodes have different values for one attribute, there is naturally more candidate patterns to generate. However, in our experiments, it is not clearly the case: the computational time even decreases, at some point: CloSpan works faster for $q = 50$ than for $q = 20$. This comes from our generative model. We have a limit on the number of values an attribute can take: the domains contain 10 different values. This has two consequences. First, in small communities (less than 10 nodes), it is impossible that all the nodes take a different value for a given attribute. Second, in large communities (more than 100 nodes), smaller homogeneous groups appear inside the communities, so we cannot say the community becomes more heterogeneous after a certain limit. In the latter case, the number of candidate patterns stays relatively stable, which explains how the performance of CloSpan evolves. Then, when $q = 100$, the computation time increases again compared to $q = 50$. This is due to an increase in the number of community-independent candidate patterns over the whole network, as shown by the plots of total number of patterns and percentage of community-related patterns. When the evolution percentage q increases, the number of patterns increases fast (seemingly exponentially), especially for $q \geq 50$. However, the percentage of community-related patterns decreases clearly. When $q = 100$, only 20% of the final patterns are community-related.

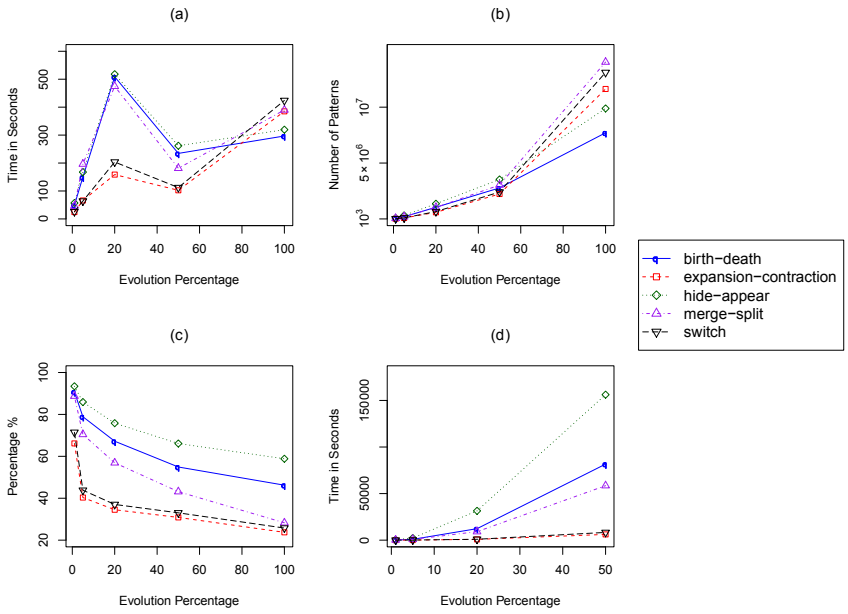

Fig. 5.10 Results of Experiment 2. Each plot shows the results for a different number of attributes. (a),(b),(c) and (d) show the execution time of CloSpan (Step 3), the total number of patterns found by CloSpan, the percentage of community-related patterns among all detected patterns and the execution time for post-processing (Step 4.B) respectively

When comparing the different types of evolution, we see that networks based on birth-death, hide-appear and merge-split event types are more difficult to process for CloSpan when $q \leq 50$, compared to switch and expansion-contraction. However, for $q > 50$, switch, expansion-contraction and merge-split event types are harder to process than birth-death and hide-appear. We can explain this with our previous assumption: when q increases, the communities get more heterogeneous in terms of attributes, but after a limit, smaller groups of homogeneous nodes appear. This causes fewer candidate patterns for switch and expansion-contraction event types. However, after the heterogeneity exceeds a limit, there are more candidate patterns (probably showing network trends and not community trends) for switch, expansion-

contraction and merge-split. Indeed, we see clearly the total number of patterns for merge-split is higher than for all the other event types when $q > 50$, followed by switch and expansion-contraction. At the same time, the percentage of community-related patterns of those three event types for $q > 50$ is lower than for other events.

When considering the execution time of our post-processing step, we see that the most difficult event type is hide-appear, which is consistent with the fact it also leads to more community-related patterns. It is followed by the birth-death, merge-split, expansion-contraction and switch. The time necessary to handle the community-related patterns for expansion-contraction and switch networks are very low when compared to the three other event types. It seems also stable, thus not affected by the increase in the percentage of modified nodes. Of course, regarding the percentage plot, we can say that there are fewer community-related patterns for these two events. Thus, the post-processing of small-scale events is not affected by the increase of the proportion of modified nodes. However, it increases for large-scale events.

5.4.3 Effect of the Attribute Distribution

Figure 5.11 presents our results in function of the different types of attribute distribution used during the network generation. The plots are organized similarly to the previous figures. CloSpan is the fastest when the attributes are completely homogeneous (i.e. degenerate distribution). For the most heterogeneous distribution (i.e. uniform distribution), it is also fast. This situation is similar to that exposed in the previous subsection: because of the finite size of the attribute domains, a uniform distribution leads to communities containing groups of nodes of approximately the same size and similar attributes. If the community is large enough, the trends of these node groups will be represented as different patterns of same community. However, if the community is small, because it is more risky that the size of these node groups cannot reach to the min_{sup} limit, their trend will not be identified as community-related patterns. However, because we use the same attribute values for all communities, the total size of these small node groups over the whole network is large enough to pass the min_{sup} threshold. Thus, their trend is represented as community-independent patterns. However, when the attributes are relatively homogeneous (binomial distribution), the execution time is at its highest. In this case, there are many nodes having the same value for a given attribute, and a few numbers of nodes with different values. It seems these few numbers of nodes with different values are causing the identification of many extra candidate patterns (cf. plot (b) of Figure 5.11). Here again, the reason can be explained by the usage of limited numbers of values for an attribute. For big communities, a large numbers of its nodes will have the same value for a given attribute. In average, the attribute values of %5 of those nodes will change. However, the sequences representing the trend of the remaining nodes (the ones which do not change) will be seen as closed patterns. For the remaining nodes, there are 9 more possible values to take. If the community is large enough, it is then possible for some of these remaining nodes to take the same values for an attribute. It naturally causes to have an enough support value for their trend to be a community-related pattern. Even if the community is not large enough, remaining nodes of different communities which have the same attribute value cause to the generation of community-independent patterns. It might be the reason why extra candidates and result patterns are identified. Our min_{sup} limit ensures to extract the closed patterns whose supporting node numbers are more than 10.

We see a distinction between the performance on small-scale and large-scale community events for binomial distributed attributes. Small-scale events are easier than large-scale ones. This distinction becomes clearer, especially for different events, on large-scale ones, when the attributes are power law distributed. In general, the execution time is less than binomial distribution for power law distribution. It seems the hide-appear type of evolution is the most difficult one for mining the patterns. The second and third most difficult evolution types are merge-split and birth-death respectively. The easiest events

are small-scale events.

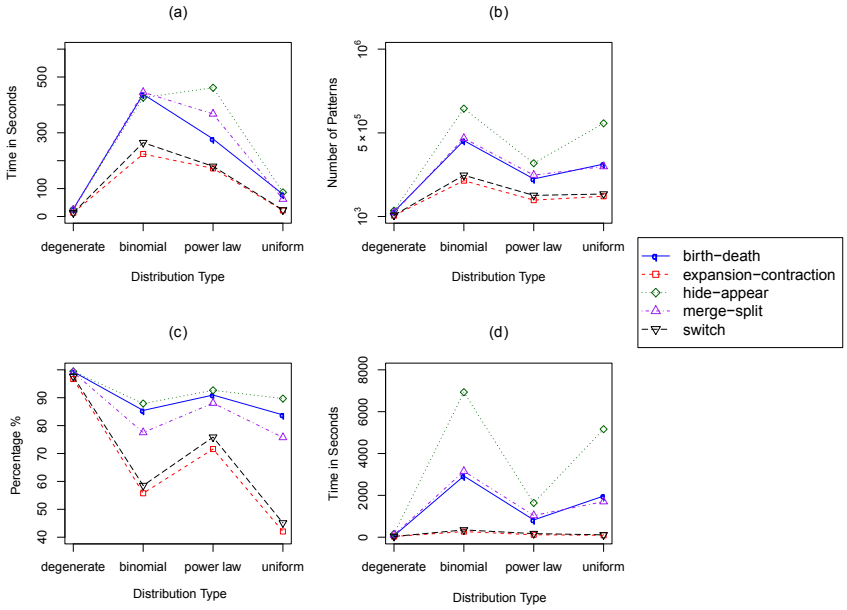

Fig. 5.11 Results of Experiment 3. Each plot shows the results for a different number of attributes. (a),(b),(c) and (d) show the execution time of CloSpan (Step 3), the total number of patterns found by CloSpan, the percentage of community-related patterns among all detected patterns and the execution time for post-processing (Step 4.B) respectively

The number of patterns obtained for Experiment #3 show that when the attributes are completely homogeneous, there are fewer patterns and a very high percentage of them ($\sim 90\%$) are community-related. For the other three distribution type, the descending order of pattern numbers is binomial, uniform and power law distribution. Regarding the percentage of community-related patterns, we see remarkable differences for the different event types. Hide-appear and birth-death events seem less affected by the distribution type. More than 85% of the patterns are community-related for every distribution type. Merge-split results show that around 75% of the patterns are community-related for binomial and uniform distributions. It is around 90% for power law distribution. For the other two types of events, we see the lowest percentage ($\sim 40\%$) at the uniform distribution. For power law distribution, there are $\sim 70\%$ of the patterns are community-related. Indeed we observe the effect of these percentages on the execution time of post-processing. For all events, when the communities are completely homogeneous, execution time of post-processing is very few. For small-scale events, execution time is fast and not affected by distribution type. However, for large-scale events, it seems affected by distribution type. Especially for hide-appear event, when the attribute values are in binomial distribution, the execution time clearly higher than all the other cases. For both birth-death and merge-split events, binomially distributed attribute values take more time for post-processing. When the attribute distribution follows a power law, it also seems difficult for post-processing on these events. Indeed, the execution time of post-processing directly related to community-related pattern number.

5.4.4 Summary

Briefly, we can conclude that increasing the number of attributes makes the processing harder for our framework. The higher this number, the higher the number of candidate patterns, and therefore the longer the execution time for both CloSpan and our post-processing of Step 4-B. Although the algorithmic time

complexity of CloSpan in not known exactly, we showed that, in practice, the number of descriptors affects the performance. When there are more descriptors, there are more possible subsequences of nodes sequences to treat, and more patterns are eventually treated. The number of patterns outputted by CloSpan directly affects the computation time of our post-processing step.

The distribution of the attribute values is also an important factor. The easiest case corresponds to a completely homogeneous distribution of values in each community. The opposite extreme case, where attribute values are heterogeneously distributed, using a uniform distribution, leads to slightly longer computation times, but clearly not as much as we expected. The worst case occurs for intermediary levels of homogeneity, i.e. binomial and power-law distributions. This is confirmed when considering the computation time in function of the proportion of nodes whose attribute values change over time. Indeed, due to the nature of this evolution, the distribution of attribute values tend to become more and more heterogeneous, until it reaches a limit caused by our use of finite attribute domains. For an initially homogeneous distribution, the computation times are the lowest, then they increase when the distribution becomes more heterogeneous, and they decrease once the limit is crossed, which corresponds to reaching a uniform distribution.

An overall observation shows that we have the longest execution times when the attribute values evolve often. The shortest execution time is obtained when changing the number of attributes. Thus, we can conclude that if the network is very dynamic, in the sense it changes a lot between two time slice, our framework works slower. It is more difficult to find the CFS and calculate their growth-rates. Our post-processing step is directly affected by the number of patterns found by CloSpan, as well as the proportion of community-related patterns they include. Another observation is that, among the different types of evolution considered in our evaluation, the large-scale ones (i.e. modification of whole communities) lead to the longest execution times, for both steps of the processing.

Finally, note that the model we used here is a very raw approximation of what a real-world system could be. It generates attributes according to some very simple rules. In a real-world system, we can suppose the value of the nodal attributes, as well as their evolution, are very domain-dependent. It is difficult, maybe impossible, to model them in a generic way. This is why our model should not be seen as a tentative to realistically mimic such a system, but rather as a plain evaluation tool. Our goal here was to have a general idea about the behavior of our framework when considering different types of data. In Chapter 6, we apply our framework to actual real-world data, taking the form of two dynamic attributed networks.

Chapter 6

Application on Real-World Data

We apply the framework we described in Chapter 4 to two dynamic attributed networks modeling real-world systems. The first one is based on data coming from the website DBLP, we treat it in section 6.2. The experiments related to DBLP by using *consensual community structure* are published in [100] and submitted in [99]. The second one is based on novel data we extracted from the website LastFM, we describe our analysis in Section 6.3. For both networks, we apply the same treatments to create the sequences databases. We detect the communities of both networks by using Incremental Louvain [5]. The modularity of the community structure of each time slice for both networks are higher than 0.6, which is a sign of well-separated communities. Incremental Louvain produces an *evolving* community structure, so we worked with an enlarged database. For the descriptors of the *enlarged* database, we use either attributes related to network of interest or topological measures. The treatments about attributes are explained in the sections related to each network when we explain. However, for the topological measures, we apply the same treatment for both networks. We explain these treatments in section 6.1.

We interpret the significant community sequences rather than the communities themselves, using the method described in section 4.4.2. Note that we interpret for both network, all the result patterns of our framework. Nevertheless, we choose some representative ones to present here. The communities or community sequences we characterized here present an example of how our framework can be used and how to comment the result patterns. Our aim is not making an exhaustive interpretation but presenting some examples which reveal the usage of the framework. In section 6.4, our conclusion & discussion is valid not only for the results of this section but also for all the results outputted of our framework.

6.1 Topological Situations and Their Meanings

We cluster the topological measures by using k-means. There are three types of topological situations for a node: *general topological situation*, *centrality-based topological situation* and *situation in the community*. The cluster numbers of different types of situation for different networks are different. Each cluster corresponds to an item of *enlarged* database. In order to be represented in M_{enl} and used by CloSpan, we encoded all the resulting items as integers. The item list and meaning is indicated in Appendix, Chapter 8 separately for DBLP and LastFM. Note that while creating items, we did not consider the isolated nodes, since they do not have any link to the other nodes, and are therefore of little interest to us.

Each cluster corresponds to the combination of different valued different topological measures. We represent in the patterns their item codes. But, we explain them in the sections with their meanings reflecting different values of measures. When we explain them, we use the words like *average*, *high*, *non-* or *low* to describe the value interval of the topological descriptor. These words represent the relative value of each descriptor. For example, *average betweenness* means a topological situation of the nodes

whose betweenness value is at the mean among the betweennes values for all the nodes of the network. When we say *non-between*, it means, the betwenness value is very low for the node compared to the rest of the network.

6.2 DBLP Network

6.2.1 Presentation of the Data

We selected the dynamic co-authorship network from [35], which was extracted from the DBLP database. It is a subset of the DBLP dataset [81]. In this network, each one of the 2145 nodes represents an author. Two nodes are connected if the corresponding authors published an article together at the considered time. Only authors who had at least 10 publications (in a selected set of 43 conferences/journals) from 1990 to 2012 are considered. Each time slice corresponds to a period of five years. There are totally 10 time slices ranging from 1990 to 2012. The consecutive periods have a three year overlap for the sake of stability. For each author, at each time slice, the database provides the number of publications in 43 conferences and journals. The conference/journals we use are related to the subjects of database, data mining, knowledge discovery, information retrieval or artificial intelligence. We use the number of publications to define 43 corresponding node attributes values, and we add two more: the total number of conference and journal publications, respectively. Finally we have a total of 45 integer attributes.

To lighten the computational load, we decided to bin these attribute values. For a journal/conference publication, we determined 5 categories, corresponding to the number of publications 1,2,3,4 and greater or equal to 5. For both attributes representing total conference and journal publications, we defined 5 categories as well, but they correspond to different ranges: $[1;5],]5;10],]10;20],]20;50]$ and $]50;\infty[$. These thresholds were determined according to our knowledge of the domain. In our analysis on DBLP, we want to learn the answer of some questions. *Is there a clear separation between the communities in terms of their journal/conference publications? Are the communities homogeneous about their journal/conference publications or they are homogeneous about their research theme? What are the anomaly types?*

6.2.2 Node Group Evolution

At first, we focus on community sequences, with a minimal support of $min_{sup} = 0.0045$, in order to study the evolution of node groups. As a reminder, what we call a *node group* here corresponds to a set of nodes jointly belonging to all the communities appearing in a given community sequence. The low min_{sup} we use allows us to catch patterns supported by ten nodes or more. We present visually evolving community structure by *alluvial diagram* at first. It is a type of flow diagram to represent the changes on community structure for dynamic networks. Let us at first have a look at the alluvial diagram given in Figure 6.1. It was generated by Edler & Rosvall's MapEquation tool [39]. It shows the major structural changes the community structure undergoes through time. Each point located on a time slice corresponds to a community, while the lines connecting them correspond to node groups evolving. The thickness of these lines is proportional to the size of the corresponding node group. To better understand these changes, one need to match the communities through time, i.e. to determine, for a given community ID at time t, what are the corresponding community IDs at times $t-1$ and $t+1$. However, as explained in chapter 4, we can use our framework for the same purpose, by simply focusing on the mining of community sequences, which allows to identify the most significant events around some communities of interest. Thus, here, we use the alluvial diagram not for tracking, but only for visualizing the general evolution of the community structure. There are a lot of nodes switching between the communities from one time slice to another. One reason of these switches is that we do not apply matching, and this prevents MapEquation to efficiently place the communities on the diagram. Another reason is that communities

are mixing. We also observe both the merge and split of some communities at almost all time slices. At $t = 3$, we see one dying community, while at $t = 4$, one of them is born. After $t = 6$, communities are getting larger than before. This diagram allows us to intuitively form an opinion about the major changes undergone by the community structure. However, this must be completed by a more detailed and objective analysis. For this matter, we study the identified community sequences.

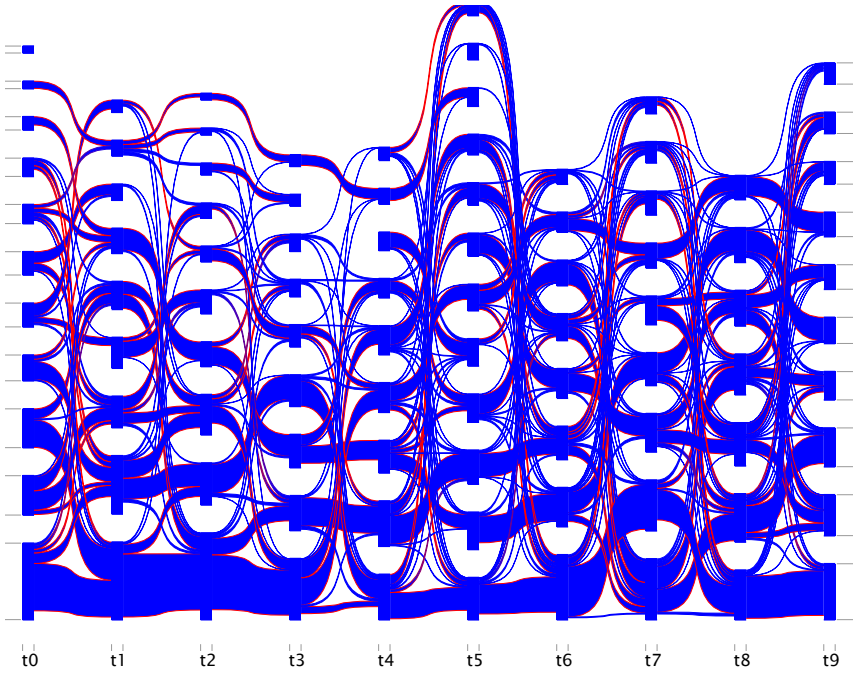

Fig. 6.1 Alluvial Diagram of DBLP evolving Community Structure. Diagram is generated by Edler and Rosvall's application [39]

It turns out the size of all community sequences supported by at least 100 nodes is 1. In other words, such patterns contain only one community ID, i.e. they span one single time slice. This means groups of more than 100 nodes are not lasting more than 1 time slice in this network. Therefore, in this specific case, we can say the node groups supporting these patterns correspond to communities (and not only parts of communities, as it the case for longer community sequences). Moreover, we notice that these larger communities mostly appear after $t = 7$. This confirms our observation made on the alluvial diagram, relatively to communities getting larger. We present the 5 most supported patterns in table 6.1.

s	$\lvert S(s)\rvert$
<(C161@9)>	183
<(C154@8)>	182
<(C389@7)>	148
<(C169@9)>	141
<(C201@9)>	140

Table 6.1 Community Sequences with Highest Supports for DBLP

We studied most of the communities and their related community sequences, which reveal the behavior of the different node groups composing the communities. Here, we present only a few of them just

to describe how our interpretation is done. When we evaluate one community, as explained in Section 4.4.2, we organize all related patterns in temporal order. We present the previous and next positions of the nodes belonging to the community of interest at the time of interest. If there is any related community sequence representing further past or further future situations for these same node groups, we also mention them.

6.2.2.1 Events Related to a Community of Interest

We first focus on community C161@9, which contains 183 nodes as indicated in table 6.1. We list the most frequent community sequences including C161@9 in table 6.2. We present the pattern, its supporting node size and the time order it presents (future, past,further future, further past or past-future together) in the table. Patterns #1, 2, 3 and 4 show that 43, 44, 17 and 16 nodes from C161@9 go to 4 different communities at the next time slice: C367@10, C189@10, C108@10 and C197@10, respectively. Some nodes of C161@9 are not supporting any of these patterns #1, 2, 3 and 4, and do not appear in other frequent patterns. This means they do not constitute node groups. We can suppose they are distributed over many different communities, or they stay alone at $t = 10$. In a sense, we can say that community C161@9 splits at $t = 10$, because none of the 4 considered node groups largely dominates the others in terms of size. However, note this split is not exactly similar to the events described for the LFR-D model in chapter 5. Indeed, in the model, the results of a community split are new smaller communities. Here, on the contrary, it is possible that the node groups resulting from the split joins existing communities at $t = 10$. Here, our community of interest is C161@9, which is why we limit our interpretation to the events directly related to it, and do not describe in detail the situation of each of the communities from $t = 10$. Maybe the node groups coming from C161@9 create the majority of these communities of $t = 10$ or maybe, they present only a small part of each of one these communities.

We now turn to patterns showing the past of C161@9, i.e. patterns #5, 6 and 7. We understand that three different node groups which belong to different communities at $t = 8$ get together at $t = 9$, joining C161@9. They only constitute a part of the community though (118 of its 183 nodes). Patterns #8 and 9 show the further past of two of these node groups. It seems a node group from C198@7 splits in two smaller node groups at $t = 9$ (belonging to C346@8 and C374@8) before reuniting at $t = 9$, in C161@9.

When we look in detail at the other related patterns showing both future and past, we see that a part of our community of interest is coming from C508@7. There are 11 nodes which continue to be in the same community from $t = 7$ to $t = 10$ (c.f. Pattern #10). Pattern #11 indicates the existence of a group of 24 nodes which are together in C374@8, then in C161@9 and in C189@10. The last two patterns show a node group existing at $t = 8$ and $t = 9$ split at $t = 10$: a part joined C197@10 and another joined C367@10.

6.2.2.2 Other Important Events

We also want to mention about some other community sequences although they do not have high supports. They show interesting trends about the evolution events. For example, the sequence $< (C471@1)(C3@2)(C1308@3) >$ is supported by 10 nodes. We cannot find any related patterns including any of those community IDs showing their situation after $t = 3$. It seems there is not any large node group belonging to these communities and staying together after this point. They might split in many small groups and join different other communities, or maybe become isolated.

Another sequence of interest is $< (C1165@4) >$, which is supported by 15 nodes. The interesting thing about this sequence is that there is no other sequence related to this community. Thus, there is no description of the past or future of this community. It appears at $t = 4$, and then disappears right away. In our pattern results, we do not have many sequences like this. Usually, community IDS appear in many patterns, which allows to study their past and future.

ID	s	$\|S(s)\|$	Time Order
1	<(**C161@9**)(C367@10)>	43	Future
2	<(**C161@9**)(C189@10)>	44	
3	<(**C161@9**)(C108@10)>	17	
4	<(**C161@9**)(C197@10)>	16	
5	<(C346@8)(**C161@9**)>	53	Past
6	<(C60@8)(**C161@9**)>	19	
7	<(C374@8)(**C161@9**)>	46	
8	<(C198@7)(C346@8)(**C161@9**)>	22	Further Past
9	<(C198@7)(C374@8)(**C161@9**)>	12	
10	<(C508@7)(C374@8)(**C161@9**)(C189@10)>	11	Past-Future
11	<(C374@8)(**C161@9**)(C189@10)>	24	Past-Future
12	<(C346@8)(**C161@9**)(C197@10)>	13	Past-Future
13	<(C346@8)(**C161@9**)(C367@10)>	18	Past-Future

Table 6.2 Some major events related to C161@9

The sequence $< (C561@4) >$ is also interesting. It is supported by 39 nodes. It means community 561 appears at $t = 4$, because there is no past history about this community. Although it might be constituted of smaller node groups existing before $t = 4$, they are not large enough to appear in the pattern list. When we track the future of this community, we see that one third of its nodes join C483@5. However, there is no other pattern representing the rest of its nodes, which means they disperse.

6.2.2.3 General Remarks

An overall observation to the community sequences of the DBLP network confirms us that the communities of this network are very dynamic and change very much. In the beginning of the considered time period, the communities are in general smaller. As time goes by, there is a trend of merging small communities to create larger ones. Moreover, we very often see that node groups coming from several communities of a given time slice get together and join other communities at the next time slice. We observe the apparition of large communities around $t = 7-9$, which tend to contract at $t = 10$.

No node group spans all 10 time slices constituting the considered time period: the longest life time is 5 time slices. The supports of these longest sequences show us that they do not represent the majority of the communities they successively belong to. Large communities of the last time slices generally contain a node group coming from $t = 5$ or $t = 6$. However, although it is not common, there are some relatively stable communities containing node group which were already formed at $t = 1$ or $t = 2$. These communities usually undergo small-scale events (expansion or contraction).

We see some small communities appearing suddenly at a given time slice, and getting later assimilated by other communities. More generally, we observe the disappearance of small communities especially during the first time slices. On the contrary, this is not the case for large communities. In the next section, we examine in detail the patterns of some large communities appearing at different time slices, in order to better understand their characteristic.

6.2.3 Community Interpretation

We create the enlarged sequence database and apply our framework on it. For CloSpan, we used $min_{sup} = 0.01$. Given the size of the network, this means patterns must be supported by at least 21 nodes to be

considered frequent. We determine $min_{gr} = 1.00$ for the limit of emergence. For the interpretation, we present the characteristic patterns found by our method in the tables, using numeric codes to display the items. These items and their meanings, as well as the attribute codes, are given in Appendix, Chapter 8. We at first focus on the most supported patterns at subsection 6.2.3.1. Then we have a detailed look at the most emerging patterns for some communities and community sequences at section 6.2.3.2 and section 6.2.3.3 respectively.

6.2.3.1 Most Supported Patterns

We show some communities from different time slices in table 6.3. Their ID appear in our result CFS set. We choose the largest ones to present here. We list the most supported patterns of each community, the support of these patterns and the size of these communities in the same table. We remark that the supports of all of these patterns are larger than 0.90 inside of the communities of interest. It shows that, for each of these communities, there is a consensus among their nodes, about the descriptor appearing in these patterns. Most of these patterns concern topological situations (topological descriptor), whereas few of them are related to the total number of conference publications (attribute descriptor).

The most supported pattern for C992@2 is related to the position of nodes in their community. Most of the nodes of this community have, at some point, many connections to other communities. Among them, there are mostly non-hub community nodes. When we looked at the future/past events of this community, we saw there is no node group lasting for a long time. The most supported pattern indicates that the majority of nodes have external connections, this can be the reason why the authors from the C992@2 do not stay together, since they already have connections and other interests related to other communities.

C	s	$Sup(s,C)$	$\|C\|$
C992@2	<(32)>	0.92	57
C1153@3	<(23)>	0.93	90
C1057@4	<(23)>	0.95	84
C262@6	<(23)>	0.94	102
C311@6	<(23)>	0.97	120
C389@7	<(31)>	0.91	148
C1@8	<(31)>	0.93	114
C8@8	<(31)>	0.95	113
C154@8	<(481)>	0.93	182
C161@9	<(31)>	0.95	183
C201@9	<(481)>	0.97	140
C383@9	<(482)>	0.90	105
C281@10	<(482)>	0.92	126

Table 6.3 Communities from different time slices (C), the most supported patterns of these communities (s), the support of these patterns ($Sup(s,C)$) and the size of these communities ($|C|$)

For C1153@3, C1057@4, C262@6 and C311@6, the most supported pattern is related to the centrality of their nodes. Most of the nodes in these communities have an average betweenness and eccentricity. It means the betweenness and eccentricity values are neither high nor low. The reason why a significant part of these communities has this situation is not only because of the position of their nodes in the community, but mostly because of the position of these communities in the network.

For C389@7, C1@8, C4@8 and C161@9, the most supported pattern signifies that the more than 90% of the nodes of these communities have a high embeddedness at some point of time period. Thus, they mostly have internal connections. It is an important sign, because it shows that the members of these communities tend to stay together, or even if they leave their community, they might do it together (at least subset of the community). Indeed, when we study the community sequences related to these communities, we see that a large node group from C389@7 (78 nodes) is staying together at C1@8.

For C154@8 and C201@9, the most supported pattern reflects that, for most nodes, the total number of conference publications, at some time slice, is between 1 and 5. For C383@9 and C281@10, the most supported pattern means the total number of conference publications number is between 5 and 10.

6.2.3.2 Trend of Communities Appearing in One Time Slice

In table 6.4, we present the trends appearing for the community sequences at size 1. Thus, these emerging patterns represent the trend of one community appearing at one time slice. In the table, each row presents the community, its characteristic pattern, growth rate and support of this pattern. The communities are indicated with the community sequences of size 1 in the first column of the table. For C311@6, we find 5 characteristic patterns. Three of them are shown as pattern #1, 2 and 3 in table 6.4. The most emerging one is pattern #1. It represents the existence of a node group in this community, which mostly belongs to their community and have no or very few external connections. Afterwards, they publish in the SDM once. Then, they are characterized by their belonging to tightly connected groups of nodes, having a high betweenness and having external connections. This topological situation continues in the future once more as well. Pattern #2 is an interesting supplementary pattern one this community. It describes nodes having average betweenness and closeness, then belonging mostly to their community, with very few external connections, while publishing in ICDM once. They then have a total number of journal publications between 1 and 5. Another interesting pattern of the same community shown as pattern #3 is referring to the existence of a node group having average betweenness, closeness and belonging mostly to its community with very few external connections. It is continued by having one publication in TKDE and a total number of conference publications between 1 and 5. It is continued by being mostly embedded in its community and still having a total number of conference publications between 1 and 5. The sequence follows with the same number of conference publications for two more time slices in the future. It seems the members of this community are interested in data mining mostly. They publish once in data mining events (SDM, ICDM and TKDE). In general, the community is not having too much external connection. However, the part which publishes in SDM starts to have external connections. We do not have an interesting pattern about the past or future of this community.

We have 4 characteristic patterns for the C8@8. We present here the most emerging one, which is also the most interesting one, and appears as pattern #4 in table 6.4. It has a growth rate of 59.08, which is very high. Thus, we can assume that it is very characteristic of this community. It refers to a node group having two publications in ILP, then having a high betwenness, low closeness and none or very few external connections. This community therefore includes a node group which has active internal connections but is only weakly connected to the rest of the network. There are two anomalies in this community: Jesse Davis and Deepak P. We notice that these authors do not have any publication in ILP in our database. We manually searched for more information regarding these people, and noticed Jesse Davis has published in ILP in 2012 and 2013; our database contains records only from 1990 to 2012. So, the anomalies outputted from our framework might have the possibility of being similar to their communities in the future.

Regarding C154@8, the most emerging pattern is presented as pattern #5 in table 6.4. It corresponds to a trend consisting in publishing in EDBT, then having high local transitivity while having a high betwenness, low closeness and some external connections. Nodes following this trend also have a total

	s_{wise}	$\lvert S(s_{wise})\rvert$	s_{less}	Gr	sup
1	<(C311@6)>	120	<(23 31)(261)(11 22 32)(11 22 32)>	6.86	0.2
2	<(C311@6)>	120	<(23)(31 61)(481)>	6.00	0.2
3	<(C311@6)>	120	<(23 31)(81 471)(31 471)(471)(471)>	4.00	0.2
4	<(C8@8)>	113	<(462)(22 31)>	59.08	0.20
5	<(C154@8)>	182	<(221)(11 22 32 472)>	9.28	0.17
6	<(C161@9)>	183	<(241)(203)>	11.07	0.17
7	<(C201@9)>	140	<(21 471)(471)(11 22 32 471) (11 22 32)(11 22 32)>	15.03	0.17
8	<(C201@9)>	140	<(11 22 32 81)(11 22 32 221)>	6.53	0.15
9	<(C201@9)>	140	<(222 482)>	3.06	0.15

Table 6.4 List of community sequences and their most emerging patterns related to C31@6, C8@8, C154@8, C161@9 and C201@9. From left to right, the columns correspond to a community sequence s_{wise}, the number of nodes $\lvert S(s_{wise})\rvert$ supporting it, a characteristic pattern of this sequences s_{less}, the growth rate $Gr(s_{less}, S(s_{wise}))$ and support $sup(s_{less}, S(s_{wise}))$ of this pattern relatively to the supporting node set of the community sequence.

number of conference publications between 5 and 10. The anomaly from this community is Muhammad Aamir Cheema. This author does not have any EDBT publication according to our database. We manually searched for additional information regarding this author as well. We noticed that he started publishing in EDBT after 2013. So, this anomaly also confirms our previous remark. Some of the anomalies might be similar with their communities in the future. Community C161@9 has 6 characteristic patterns. The most emerging and interesting one is shown as pattern #6 in table 6.4. It refers to a node group who publishes at first in ICML once, and then have three KDD publications. This community is rather large. It is difficult to cover it with only one pattern. However, it seems there is not a significant trend among the other nodes of this community except the ones who supports this pattern. The anomaly of this community is Yang Song. He has no publication in ICML or in KDD.

We find 7 characteristic patterns for C201@9. The most emerging and two interesting supplementary patterns are shown as pattern #7, 8 and 9 in table 6.4. Pattern #7 mostly indicates the topological situation of the nodes. A significant group in this community has a high eigenvector centrality, and they are close to the other nodes. Meanwhile, they have between 1 and 5 journal publications. Later, they have between 1 and 5 journal publications, again. Then, this group is connected to other communities, by having high betwenness and high local transitivity and low closeness. This topological situation continues two more time slices. One of the supplementary patterns, shown as pattern #8 in the table 6.4, tells us there is a node group which has external connections while having high betwenness and high local transitivity and low closeness. They meanwhile publish once in TKDE. Afterwards, the same topological situation continues and they publish once in EDBT. The last interesting supplementary pattern indicates the existence of a node group which publishes two times in EDBT and their total conference publication number is between 5 and 10. We do not find any anomaly in this community.

6.2.3.3 Trends of C992@2 and Related Node Groups

Let us have a look at C992@2. Because C1153@3 also appears in the same community sequence than C992@2, we comment them together. In table 6.5, we list the patterns related to these communities. We see that the most emerging one for C992@2 has a very high growth rate of 586.10. Thus, it is highly representative of this community. It represents a node group which have high eigenvector centrality and

who are close to other nodes. Meanwhile, they have 1 publication in VLDBJ and their total number of journal publication is between 1 and 5. Afterwards, these nodes have high local transitivity, more than average degree and they have high betweenness. Moreover, they have external connections. Then, these nodes have one publication in PODS. We have found 5 patterns to cover this community. One of the interesting supplementary patterns is shown as pattern #2 in Table 6.5. This pattern tells us that there is another node group having 1 publication in Commun ACM and a total number of journal publications between 1 and 5. They then have a total number of journal publications between 1 and 5 again in a subsequent time slice.

When we focus on pattern #3 in table 6.5, we have an idea about the node group of C992@2 which keep being together at $t = 3$. The trend they follow is quite similar to the characteristic patterns of C992@2, with some additional items. It seems, the nodes which keep being together have high eigenvector and close to other nodes and have more than 4 publications in VLDB conference and their total journal publication number is between 1 and 5. Afterwards, still their total journal publication number is between 1 and 5. It is followed by having high transitivity and more than average degree and having external connections. Then, again their total journal publication number is between 1 and 5.

	s_{wise}	$\lvert S(s_{wise})\rvert$	s_{less}	Gr	sup
1	<(**C992@2**)>	57	<(21 301 471)(11 32) (291)>	586.10	0.28
2	<(**C992@2**)>	57	<(71 471)(471) (471)>	5.00	0.28
3	<(**C992@2**)(C1153@3)>	40	<(21 215 471)(471)(11 32)(471) >	64.76	0.40
4	<(C1153@3)>	90	<(21 301 471)(21 301 471)(32)>	77.06	0.30
5	<(C1153@3)>	90	<(21)(215 471)(471)>	10.74	0.17

Table 6.5 List of community sequences and their most emerging patterns related to C992@2 and C1153@3. From left to right, the columns correspond to a community sequence s_{wise}, the number of nodes $\lvert S(s_{wise})\rvert$ supporting it, a characteristic pattern of this sequences s_{less}, the growth rate $Gr(s_{less}, S(s_{wise}))$ and support $sup(s_{less}, S(s_{wise}))$ of this pattern relatively to the supporting node set of the community sequence.

We see a similar trend for C1153@3, as illustrated by pattern #4 in table 6.5. We found 5 characteristic patterns for C1153@3, but we discuss only the most emerging and interesting ones. A node group of this community have a high eigenvector centrality and they are close to other nodes. Meanwhile, they have one publication in VLDBJ and a total number of publications between 1 and 5. The same situation is reproduced at a later time slice. Later, these nodes are connected with other communities. The pattern #5 in table 6.5 represents a trend which is similar enough. It points out the nodes which have high eigenvector centrality, then have more than 4 publications in VLDB and a total number of journal publications between 1 and 5. Then, their total number of journal publications is again between 1 and 5. To sum up, we remark that the core part of C992@2 publishes in VLDB and VLDBJ. These nodes stay together at $t = 3$. Although there are new comers (the size of C1153@3 is 90 while the size of C992@2 is 57), it seems they follow a similar trend, in the sense they also publish in VLDB or VLDBJ. Afterwards, this core part starts to have external connections. Although there is another part of C992@2 who publishes in CommunACM, it seems they disappear, or the nodes of this part do not continue being together.

6.2.4 Conclusion

In general, DBLP communities are getting crowded as time goes by. In the beginning we have asked some questions about DBLP communities. Our framework helped us to reply these questions. *Is there a clear separation between the communities in terms of their journal/conference publications?* We have

seen that the communities of DBLP do not constituted around one interest but they have different node groups having common interests in terms of their attributes. These node groups tend to keep being together as time goes by. *Are the communities homogeneous about their journal/conference publications or they are homogeneous about their research theme?* For this question, we see that in general communities are not completely homogeneous. However, although there might be different node groups publishing at different conferences/journals, the theme of these activities can be same for some communities. The other question was *What are the anomaly types?* We can group the anomalies into three groups. The first ones are authors not publishing in the same conferences than the other authors from the same community. Our manual verifications showed that some of these authors have started to publish in the same conferences recently that we do not have these records in our database. The second anomalies are nodes which are topologically different from their communities, although they publish in the same conferences. The third anomalies are nodes which are much more active or passive relatively to the rest of their community.

6.3 LastFM Network

6.3.1 Presentation of the Data

LastFM is a music website that allows its members to register and listen to music online [88]. It is also a social network platform, because the members can declare friendship relationships. In LastFM, members can join a predefined groups related to their music tastes, and participate in music-related events such as concerts. Using the LastFM API, One can retrieve the information of the artist and track a user has listened to, with the exact timestamp. Moreover, it is also possible to get some information regarding the music-related events the users joined, including the exact timestamps.

We extracted a network by focusing on the members of the group *Jazz*, which is supposed to include users appreciating this type of music. We took advantage of the LastFM API to retrieve the members of this group and the existing friendship connection between them. In the end, our network contains 3478 nodes representing the Jazz users. The friendship relationships between them is static, though, in the sense the LastFM API does not give access to any temporal information regarding their beginning or end. So, we decided to take advantage of some additional information to get a dynamic structure. We put a link between two nodes if two conditions were simultaneously true: 1) both considered users listened to at least one common artist for a specific period of time, and 2) they are friends on the LastFM platform. For the mentioned period of time, we decided to use 1 month, after having analyzed the dynamics of the platform. In other words, we extracted a dynamic network in which each time slice represents one month of LastFM usage for our 3478 users of interest. We define 37 nodal attributes based on the users' music tastes: 21 of them represent the most popular artists in the Jazz group, and the 16 others correspond to the types of music-related events the members can joined. The popularity of the artists is determined depending on their numbers of listeners in the Jazz group. For each time slice, the attribute values correspond to the total numbers of times a given user listed to a given artist, and the number of times he joined an event of a given type, respectively. The complete list of these attributes is given in Appendix, Chapter 8. For this work, we focused on the data describing only the year 2013, and separated it into 12 time slices of one month each.

It is interesting noticing that, although all our nodes are from the Jazz group, the selection of most popular artists include acts from music types very different from Jazz, such as the Beatles or Pink Floyd. We nevertheless kept these artists, because their playing rate is very high in the Jazz group. But most of the popular artists we evaluate play Jazz-related music. Event types also span a wide range of music types, from jazz to folk. Our aim while analyzing this network is to get an idea about the trends among the jazz listeners. Some questions we want to answer are: *Is there a clear separation among the listeners*

of different jazz types? Do jazz listeners follow jazz events as well? What are the communities of jazz listeners? Do jazz listeners from same communities have similar tastes of music?

Once the network was created, we processed the required topological measures and clustered them like we did for the DBLP network in the previous section. The attribute values were binned. For artist attributes, we determined 5 intervals to group the numbers of listenings: $[1;5],]5;30],]30;90],]90;200]$ and $]200;\infty[$. Regarding the event types, we defined 4 intervals: $[1;4],]4;7],]7;11]$ and $]11;\infty[$. The intervals and their meanings are listed in Appendix, Chapter 8.

6.3.2 Node Group Evolution

As we did for the DBLP network, we first apply CloSpan on a sequence database including only community sequences, in order to see how the communities evolve. We used $min_{sup} = 0.001$, which allows us to obtain community sequences supported by at least 4 nodes. Here, we follow for LastFM the same interpretation method than the one used for DBLP. Like in the DBLP network, the most supported community sequences have of size 1. They represent one community appearing at one time slice. We present the alluvial diagram of the evolving community structure of LastFM in Figure 6.2. Differently from DBLP, there are not death or birth events in LastFM. Moreover, we notice that the sizes of the communities of different time slices are almost similar. In DBLP, the communities were getting larger with time. However, in LastFM, there is not a significant difference between the first and last time slices.

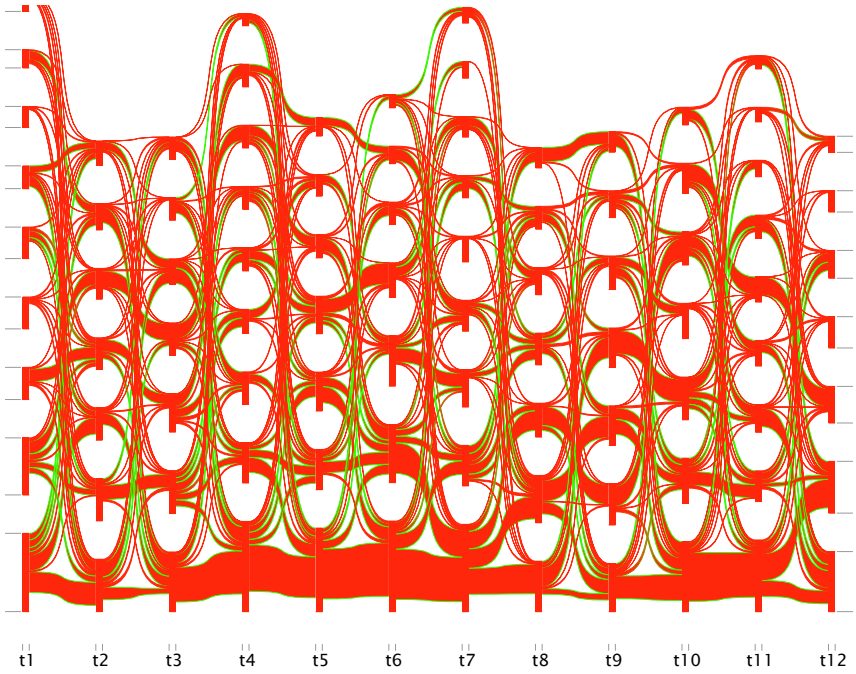

Fig. 6.2 Alluvial Diagram of LastFM evolving Community Structure. Diagram is generated by Edler and Rosvall's application [39]

Let us now have a look in detail at the evolution of the main communities. The community sequences whose sizes are larger than 100 are all of size 1. We list the first five community sequences with the highest support in table 6.8. Some of these sequences are the continuation of each other. More clearly, we obtain community sequences including some of the listed communities together. Here, we present an evaluation around one community to show a sample of community evolution events of LastFM. In the subsection which concludes this section, we try to express our general observations of community evolution in LastFM.

s	$\lvert S(s)\rvert$
<(C593@5)>	161
<(C4@4)>	159
<(C668@7)>	151
<(C11@6)>	150
<(C19@7)>	140

Table 6.6 Community Sequences with highest supports for LastFM

6.3.2.1 Events Related to a Community of Interest

We show some important community sequences related to C593@5 in table 6.7. We see that three node groups leave C593@5 and join three communities at the next time slice (C11@6, C12@6 and C22@6). Two of these new node groups are smaller (patterns #4 and 5: 13 nodes each) while one of them is clearly larger (pattern #3: 62 nodes). Moreover, at a longer term, we see that some parts of this larger node group stay together at $t = 7$ (c.f. pattern #2) and at $t = 8$ (c.f. pattern #1). Regarding the past, we observe that three node groups from different communities of $t = 4$ get together in C593@5. Among them, C4@4 have the highest intersection with C593@5 (c.f. pattern #8, 48 supporting node). When we look at the further past, we see that there is a node group coming from C68@1 which end up in C593@5 (c.f. pattern #9 with 41 nodes), even if these nodes do not all stay together in the meantime. Some of them do, though, in C4@4, as shown in pattern #11 (28 nodes). Similarly, when we regard patterns #10 and 12, we understand that there is a node group of C16@2 which are also together in C593@5. Some of those nodes were also together in C4@4.

ID	s	$\lvert S(s)\rvert$	**Time Order**
1	<(**C593@5**)(C11@6)(C13@8)>	20	Further Future
2	<(**C593@5**))(C11@6)(C668@7)>	20	
3	<(**C593@5**))(C11@6)>	62	Future
4	<(**C593@5**))(C22@6)>	13	
5	<(**C593@5**))(C12@6)>	13	
6	<(C586@4)(**C593@5**))>	19	Past
7	<(C547@4)(**C593@5**))>	12	
8	<(C4@4)(**C593@5**))>	48	
9	<(C68@1)(**C593@5**))>	41	Further Past
10	<(C16@2)(**C593@5**))>	37	Further Past
11	<(C68@1)(C4@4)(**C593@5**))>	28	Further Past
12	<(C16@2)(C4@4)(**C593@5**))>	20	
13	<(C237@3)(C4@4)(**C593@5**))>	19	

Table 6.7 Some major events related to C593@5

6.3.2.2 General Remarks

Considering all community evolution events occurring in LastFM, we can say that although the evolution is very dynamic, it is not as eventful as for DBLP. We do not observe major large-scale events like birth or death. However, we observe frequent separations or joining of different node groups. In general, two

communities of a previous time slices do not merge completely to create a larger one in the time slice of interest: the general trend is more for some parts of communities to get together with some parts of other communities. They then create a part of another community at the following time slice. It is the same for the groups leaving existing communities as well. Differently from DBLP, communities are not getting larger with time. We even observe dense communities at the 1st and 2nd time slices. Moreover, community C68@1 affects many major communities of the next time slices. Different node groups of this community later appear in many other communities from $t = 4$ or $t = 5$, which keep surviving until $t = 9$.

Regarding the size of the community sequences, we notice that we have very long sequences (representing an evolution during 10 to 12 time slices) supported by up to 6 nodes. This means some groups of 6 nodes stay together in the same community during the whole considered period of time. For the sequences supported by more than 10 nodes, there are many shorter sequences representing an evolution included between $t = 1$ and $t = 12$ (but not spanning the whole period of time). Although the size of these sequences is not longer than 6 (which means they do not include each time slice), they still represent a continuation for a long time. In general, we can say that the evolution of the LastFM communities is smoother than that of the DBLP communities, mainly because of the absence of large-scale events, meaning the communities evolve most of all in terms of increasing or decreasing size. We want to remind the reader that this LastFM network describes an evolution over one year, while the DBLP network represents a 18 years evolution. So not only the systems, but also the temporal scales are different.

6.3.3 Community Interpretation

We create the enlarged sequence database of LastFM and apply our framework on it. For CloSpan, we used $min_{sup} = 0.005$. Thus, we get patterns supported by at least 20 nodes. As we did for DBLP, we at first present the most supported patterns at section 6.3.3.1 and then, we also explains the interesting emerging patterns for some communities in section 6.3.3.2. We present the interesting emerging patterns related to some community sequences at sections 6.3.3.3 and 6.3.3.4.

6.3.3.1 Most Supported Patterns

We show a selection of most interesting communities from different time slices in table 6.8. Each row displays a community that we choose according to its size (larger than 80 nodes), together with its most supported pattern, the support of this pattern and the community size. We want to underline here that supports are not as high as with the DBLP network. For most of the communities, the highest support is between 0.60 and 0.70, which is still high. This shows that these communities are not as homogeneous as the DBLP ones, in terms of similarity of their descriptor values.

The most part of the most supported patterns expresses a topological situation. We see that for communities C68@1, C602@1, C301@1,C593@5, C11@6, C668@7, C551@8 and C12@9, the most supported pattern represents node groups which do not have a central position especially about betweenness and eigenvector centralities. They can be close to other nodes, but they do not have a high eigenvector centrality. They do not have a high degree or transitivity. They mostly are embedded in their own communities. Even if they have some external connections, these are very few. Moreover, they do not hold a community hub role. This does not appear in this specific table, but among these communities, C68@1 is the root of many other communities of the next time slices: C593@5, C11@6, C668@7 and C12@9.It means, we obtain many community sequences starting with C68@1 and including also C593@5, C11@6, C668@7 and C12@9. The trend of belonging mostly to its community for the nodes of C68@1 can be a sign of that this node group keep being together in following time slices C593@5, C11@6, C668@7 and C12@9. For being sure, we need to check the characteristic patterns related to

community sequences including both C68@1 and other community ID's. We examine in detail the trends related to C68@1 in section 6.3.3.3.

C	s	$Sup(s,C)$	$\|C\|$
C68@1	<(11 24 32)>	0.68	124
C602@1	<(11 24 32)>	0.87	81
C301@1	<(11 24 32)>	0.77	105
C16@2	<(131)>	0.62	116
C237@3	<(131)>	0.66	98
C4@4	<(12)>	0.63	159
C593@5	<(11 24 32)>	0.68	161
C11@6	<(11 24 32)>	0.69	150
C668@7	<(11 24 32)>	0.64	151
C551@8	<(11 24 32)>	0.75	111
C12@9	<(11 24 32)>	0.71	91
C123@10	<(12)>	0.69	119
C57@11	<(12)>	0.65	129
C1256@12	<(12)>	0.65	107
C38@12	<(131)>	0.68	85

Table 6.8 Communities from different time slices (C), the most supported patterns of these communities (s), the support of these patterns ($Sup(s,C)$) and the size of these communities ($|C|$)

For communities C4@4, C123@10, C57@11 and C1256@12, the most supported pattern shows that the majority of the nodes of these communities have at least once a central position. At least at one time slice, they might be have a high degree and a high local transitivity. They might have a high eigenvector centrality. For communities C16@2, C237@3 and C38@12, we see that the majority of the users listen to John Coltrane between one to five times, at least at one time slice. It is indeed very interesting that there is a consensus among the nodes of these communities about listening to the same artist. We discuss in detail the emerging trends of some of these communities representing the common evolution of their nodes.

6.3.3.2 Trend of Communities Appearing in One Time Slice

We start with the emerging patterns containing a single community ID. These communities and their characteristic patterns are shown in table 6.9 and 6.10. We find 8 characteristic patterns mentioning C301@1. The most emerging one is shown as pattern #1 in table 6.9 . It represents a node group which listens to Miles Davis between one and five times in a time slice. Then, they take a central situation and high local transitivity. Meanwhile they have some external connections. However, these external connections are not distributed over many different communities, but rather concentrated in a few ones. In the future, these nodes keep this topological position, while also having a low betweenness and listening to Miles Davis between one and five times. We want to indicate that there are other communities characterized also by listening Miles Davis. For C301@1, having external connection but especially to few number of other communities can be a sign that Miles Davis listeners of different communities are highly connected with each other. Thus, there might be overlapping between different communities. Two other interesting patterns among the supplementary ones are shown as pattern #2 and 3. Pattern #2 refers to a node group which have a central position with average degree, while listening to Ella Fitzgerald

and Chet Baker between one and five times. It is continued by keeping the same topological position. Here listening Ella Fitzgerald and Chet Baker at same time might be explained by the fact that these two artists are vocal. Chet Baker is famous with his virtuosity on Trumpet but he also sings. Indeed, when we search in detail the data of LastFM, we see that there are users listening several times *My Funny Valentine* from Chet Baker and also from Elle Fitzgerald. Pattern #3 describes a node group listening to Miles Davis between one and five times, followed by having some external connections and having a low betweenness. Then, these nodes keep their topological position and also listen to John Coltrane between once and five times. It is followed by having the same topological position. For C301@1, we see a common topological situation referring to the nodes which have some external connections. Their external connections are usually to one community. They have a non-hub role in their community. It seems there are Miles Davis lovers in this community. The two different node group of C301@1 represented in patterns #1 and #3 refers them. However these two groups differs in terms of their topological situations. There are also Ella Fitzgerald, Chet Baker and John Coltrane lovers in this community. The interesting thing is some of these artists represents very different Jazz types. Elle Fitzgerald is a singer of Classic Jazz while John Coltrane or Miles Davis are the most important representatives of modal jazz. Thus, there is a consensus in this community neither about the artist nor the jazz type they listen. Nevertheless, we cannot say that the nodes of this community is very dispersed in terms of their interest. There is two major group : Miles Davis-John Coltrane fans and Ella Fitzgerald-Chet Baker fans.

	s_{wise}	$\lvert S(s_{wise})\rvert$	s_{less}	Gr	sup
1	<(C301@1)>	105	<(41)(12 34)(12 22 34 41)>	20.36	0.18
2	<(C301@1)>	105	<(12 81 101)(12)>	8.84	0.18
3	<(C301@1)>	105	<(41)(12 22 34)(12 131)(12)>	16.49	0.18
4	<(C593@5)>	161	<(22)(22 42)(22 172)(22) >	37.34	0.18
5	<(C593@5)>	161	<(61)>	2.06	0.13
6	<(C11@6)>	150	<(12 22)(12 22 34)(12 34)(22)(22 42)(22)>	35.03	0.20
7	<(C11@6)>	150	<(151)(151)>	6.93	0.20

Table 6.9 List of community sequences and their most emerging patterns related to C301@1, C593@5 and C11@6. From left to right, the columns correspond to a community sequence s_{wise}, the number of nodes $\lvert S(s_{wise})\rvert$ supporting it, a characteristic pattern of this sequences s_{less}, the growth rate $Gr(s_{less}, S(s_{wise}))$ and support $sup(s_{less}, S(s_{wise}))$ of this pattern relatively to the supporting node set of the community sequence.

Regarding C593@5, we find 5 characteristic patterns. The most emerging one is shown in pattern #4 in Table 6.9. This pattern refers to nodes in this community which are non-between and have average eigenvector centrality. They keep their topological position in the future, while also listening to Mile Davis 5-10 times. The same topological situation continues in the future and they listen to Charles Mingus 5-10 times. They then keep the same topological position. The supplementary pattern is shown as pattern #5. It refers to a node group listening to Keith Jarrett at least at one time slice from once to five times. We have 11 anomaly nodes in this community, which do not follow any of the detected patterns. We check in detail these nodes. They do neither have a listening activity nor participate to any event at any time slice. Moreover, we see that these nodes are isolated (i.e. their degree is 0). So, we can conclude they were misplaced by the community detection algorithm. Indeed, they should not be placed in any community, because they are not connected to other nodes.

When we look at community C11@6, we find 5 characteristic patterns. The most emerging one is

shown as pattern #6 in table 6.9. It mostly refers to a topological position. The nodes supporting this pattern have a low betweenness and average closeness. They have high local transitivity. They keep this position while also having some external connections and they are non-hub in their communities. Indeed this topological position continues in the future. Then, they listen to Miles Davis 5-10 times. One interesting supplementary pattern of this community is shown as pattern #7. It refers to the nodes which listen to Thelonious Monk 1-5 times. Afterwards, they again listen to same artist 1-5 times. We detected four anomalies in this community, which are also nodes misplaced by the community detection algorithm, like for the previous community. These nodes do not have any activity at any time slice.

There are 6 characteristic patterns mentioning C668@7. The most emerging one is pattern #1 in Table 6.10. It is referring to a node group listening to Miles Davis 1-5 times. Meanwhile, these nodes are central, especially in terms of their closeness. They have high transitivity. They then have a low betweenness and are mostly connected to their community, while being community non-hubs. Two interesting supplementary patterns are shown as patterns #1 and 2. The former one corresponds to a node group having high local transitivity and which are central, especially in terms of closeness. These nodes meanwhile listen to Charlie Parker 1-5 times. Afterwards, they listen to Herbie Hancock 1-5 times. The latter pattern describes a node group listening 1-5 times to the Beatles and then to Pink Floyd. We again detect the misplaced nodes as anomalies for this community. It is very interesting that in C668@7, three patterns refer to three different node groups that seem having different music tastes. One of them listens to Miles Davis, the other listens to Charlie Parker and Herbie Hancock and the last one listens to the Beatles and Pink Floyd. Here, we notice a clear separation of non-jazz listeners. It can be expected that the Beatles and Pink Floyd fans stay apart from other listeners because of their taste of music.

	s_{wise}	$\lvert S(s_{wise})\rvert$	s_{less}	Gr	sup
1	<(C668@7)>	151	<(12 42)(22 32)>	26.82	0.18
2	<(C668@7)>	151	<(12 91)(51)>	8.22	0.18
3	<(C668@7)>	151	<(221)(241)>	2.44	0.18
4	<(C551@8)>	111	<(12 34)(12 34)(111)(171)>	13.67	0.20
5	<(C551@8)>	111	<(191)>	1.55	0.20
6	<(C1256@12)>	107	<(12)(12 42)(22)(41)(12 34)(12 34) >	34.82	0.19
7	<(C1256@12)>	107	<(281) >	1.46	0.21
8	<(C1256@12)>	107	<(251) >	2.15	0.20

Table 6.10 List of community sequences and their most emerging patterns related to C668@7, C551@8 and C1256@12. From left to right, the columns correspond to a community sequence s_{wise}, the number of nodes $|S(s_{wise})|$ supporting it, a characteristic pattern of this sequences s_{less}, the growth rate $Gr(s_{less}, S(s_{wise}))$ and support $sup(s_{less}, S(s_{wise}))$ of this pattern relatively to the supporting node set of the community sequence.

For C551@8, there are 6 characteristic patterns. The most emerging one is shown as pattern #4 in table 6.10. It refers to a node group which are community non-hubs, have a few external links, but are mainly connected internally, and are central especially in term of their closeness. This topological position continues once in the future again. Then, they listen to Nina Simone 1-5 times. Afterwards, they listen to Charles Mingus 1-5 times. One interesting supplementary pattern is given as pattern #5 in the same table. It refers to a node group once belonging to C551@8, which listens to Norah Jones 1-5 times, at least at one time slice. Here again we see a separation between the node groups which listens different artists. However, we can identify this community as soul, blues female vocal lovers. Norah Jones is vocal of 2000s. She is singing soul, blues etc. Nina Simone and Charles Mingus represent an

older age. Nevertheless, Nina Simone also sings blues, soul, etc.

For community C1256@12, we find 10 characteristic patterns. The most emerging one is given as pattern #6 in table 6.10. It mostly refers to the continuation of a topological situation. In the beginning, the concerned nodes have a high closeness centrality high local transitivity. This situation continues while also listening to Miles Davis 5-10 times. Afterwards, these nodes have a high eigenvector centrality and a have low betweenness. Then, they again listen to Miles Davis, but this time 1-5 times. It is followed by having high closeness centrality, high local transitivity and also having some external connections. Two supplementary patterns refer to the existence of two different node groups. The first and second groups participate to rock and jazz events, respectively, at least at one time slice, 1-4 times. The four anomalies of this community have the same property than the other anomalies we indicated before: they seem to be misplaced by the community detection algorithm, because they have no activity and they are isolated.

After an overall observation together these characteristic patterns related to one community appearing at one time slice, we can say that they usually refer to a continuation of a topological situation at least twice. This topological situation indicates the existence of nodes which have some external connections, generally focusing only on one or a few communities. Moreover, they do not have a high betweenness, but they seem have a high closeness and a high local transitivity. We find at least one attribute-related descriptor during the characterization. However, usually the most emerging and supplementary patterns refer to different attributes. Thus, we can assume that the communities are not completely homogeneous, but there are different homogeneous groups inside of the same community. We notice that many different communities are characterized by Miles Davis. He is a great artist who is the creator of different types of Jazz. Thus, it can be expected that we find his lovers in many different communities. For some communities, it is possible to distinguish them from the rest of the network by their interest (c.f. C551@8 - soul, blues female vocal lovers). For some other communities, there is not one interest but few of them (c.f. C668@7 - rock lovers and jazz lovers). Regarding the anomalies, we notice that all the anomalies we find for these communities are misplaced. So, our framework helps us to correct the defect of community detection algorithm. Now, let us look in detail to the patterns appearing in more than one community of different time slices. These patterns help us to understand the characteristic of switching nodes from one community to another at different time slices.

6.3.3.3 Trends of C68@1 and Related Node Groups

In the section 6.3.3.1, we underlined that there are different node groups of C68@1 appearing in different communities of the next time slices. That is why, we want to interpret this community. Indeed, we find many emerging patterns showing the trend of switching nodes from this community. In table 6.11, we present the most emerging and most interesting patterns related to C68@1. We find 8 characteristic patterns for C68@1.

Pattern #1 in table 6.11 is the most emerging one. It represents the nodes listening to Thelonious Monk 5-10 times. Then, they hold a topological situation of having high eigenvector and low betweenness. This topological situation continues for a long time in the future. The three interesting supplementary patterns are shown as patterns #2, 3 and 4. Pattern #2 refers to the same situation than before, but these nodes listen to Charles Mingus. Pattern #3 represents the nodes listening to Miles Davis 5-10 times and Nina Simone 1-5 times. Then, these nodes have a high closeness but low betweenness. They also have a high eigenvector centrality. We see the continuation of these nodes in patterns #5, 6, 9, 10, 14 and 15. It seems these nodes listening to Miles Davis migrate to C237@3, then C4@4 , C459@5 , C11@6, C12@9 and C38@12 respectively. We make out this result by comparing the supporting nodes sets of patterns #5, 6, 9, 10, 14 and 15. Their supporting nodes are almost the same, although there are some small differences for each community. An interesting remark for community sequence <(C68@1)(C4@4)(C593@5)(C38@12)> is that the support of the most emerging pattern is 1.00 (c.f. pattern #15 in table 6.11). Thus all the nodes which belong to C68@1, then 4@4, then C593@5 and

then 38@12 listens to Miles Davis in at least at one time slice. We obtain one anomaly for pattern #9. It is user *bentelhaj*, which listens very routinely to Miles Davis: 1-5 times at almost at all time slices. The general trend of this community sequence includes listening to Miles Davis 5-10 times.

Regarding pattern #4, we understand that there are nodes which have a central position in terms of their closeness and a high local transitivity. Afterwards, these nodes listen to John Coltrane 5-10 times, and in the future they again listen to the same artist 5-10 times. Focusing on the nodes belonging to C68@1 and then C4@4, we find three interesting trends. They are shown by patterns #6, 7 and 8. Pattern #7 represents a node group belonging to these two communities, respectively, and which at least once have a high closeness, a high local transitivity and listen to Charles Mingus 1-5 times. Pattern #8 represents a group among the nodes switching from C68@1 to C4@4, which have a low betweenness and have average eigenvector centrality. This topological situation continues once more, while also listening to John Coltrane. We find one anomaly for community sequence <(C68@1)(C4@4)>: it is user *JazzyFJuice*, which mostly listens to the Beatles, Radiohead and Django Reinhardt. He does not follow any of the patterns we detected for the community sequence he belongs to. For community sequence <(C68@1)(C38@12)>, the related characteristic patterns are shown as patterns #11 and 12. They represent two nodes groups of this community sequence. The first ones have some connection from outside of their community. These friends belong usually to one community. The nodes have a high closeness, then listen to John Coltrane, then to Miles Davis. The second one indicates a similar topological situation but the external connections of the nodes are distributed over many communities. Afterwards, these nodes listen to Charles Mingus.

	s_{wise}	$\lvert S(s_{wise})\rvert$	s_{less}	Gr	sup
1	<(**C68@1**)>	124	<(152)(22) (22) (22) (22) (22) (22)>	36.59	0.18
2	<(**C68@1**)>	124	<(22) (22 172)>	7.53	0.13
3	<(**C68@1**)>	124	<(42 111) (12 22)>	14.81	0.18
4	<(**C68@1**)>	124	<(12) (132) (132) (22) >	8.21	0.13
5	<(**C68@1**)(C237@3)>	30	<(42)(12)>	7.91	0.63
6	<(**C68@1**)(C4@4)>	59	<(12 22) (22 42)(22)(22)(12) >	22.43	0.40
7	<(**C68@1**)(C4@4)>	59	<(12 171)>	6.17	0.38
8	<(**C68@1**)(C4@4)>	59	<(22)(22 131)>	7.75	0.28
9	<(**C68@1**)(C593@5)>	41	<(42)(22)>	12.76	0.56
10	<(**C68@1**)(C11@6)>	46	< (22) (42)>	12.52	0.50
11	<(**C68@1**)(C38@12)>	43	< (12 34)(132)(42)(42)>	21.55	0.39
12	<(**C68@1**)(C38@12)>	43	< (12 33)(172) >	19.97	0.39
13	<(**C68@1**)(C4@4) (C38@12)>	31	<(12 22)(12)(131)>	13.79	0.54
14	<(**C68@1**)(C12@9) (C38@12)>	31	<(12 42)>	8.92	0.68
15	<(**C68@1**)(C4@4) (C593@5)(C38@12)>	17	<(42)>	5.91	1.00

Table 6.11 List of community sequences and their most emerging patterns related to C68@1, C237@3, C4@4, C11@6, C12@9 and C38@12. From left to right, the columns correspond to a community sequence s_{wise}, the number of nodes $|S(s_{wise})|$ supporting it, a characteristic pattern of this sequences s_{less}, the growth rate $Gr(s_{less}, S(s_{wise}))$ and support $sup(s_{less}, S(s_{wise}))$ of this pattern relatively to the supporting node set of the community sequence.

6.3.3.4 Trends of C602@1 and Related Node Groups

We obtain 5 characteristic patterns for community C602@1. The interesting ones, as well as the community sequences including them, are listed in table 6.12. The most emerging one is pattern #1. It shows a specific topological position of having no central situation, few connections, low centrality, being a community non-hub and being mostly connected to its own community. This situation continues in the future. Then, the users corresponding to these nodes listen to Pink Floyd 5-10 times. Afterwards, they keep their topological situation while also listening to the Beatles 5-10 times, and then they again hold the same topological situation. Although the artist they listen to changes, their structural features do not change at all.

One interesting supplementary pattern is shown as pattern #2. It refers to an emerging node group which listen at first to the Beatles 5-10 times, then to the Beatles again in the same amount, and then to Frank Sinatra 1-5 times too. A part of C602@1 seems to not be interested in Jazz at all. The emerging group of this community is mostly connected with the nodes of their community. In fact, intuitively, it makes sense that the people not interested in jazz too much but which are still a part of a jazz listeners network stay close to each other and do not interact much with other users. A very interesting fact concerning these users is that some of these groups keep surviving at $t = 10$. Pattern #4 shows that the nodes supporting the community sequence <(C602@1)(C246@10)> have the same topological situation and listen to Pink Floyd. It is very similar for pattern #3 as well. When we compare the supporting nodes of these two patterns (patterns #3 and 4), we notice they have a large intersection. Two anomalies of community sequence given #3 are users *Negozio Fama* and *Absurd_Maers*. We checked their node sequences and observed that these nodes are listening to Pink Floyd much. Although the common trend of this community sequences includes also listening to Pink Floyd, the difference of these anomalies is that they listen to this band more than 100 times per month.

	s_{wise}	$\lvert S(s_{wise})\rvert$	s_{less}	Gr	sup
1	<(**C602@1**)>	81	<(11 24 32)(11 24 32)(242)(11 24 32 222)(11 24 32)>	14.97	0.18
2	<(**C602@1**)>	81	<(222)(222)(121)>	7.23	0.18
3	<(**C602@1**)(C188@4)>	22	<(11 24 32)(242)>	6.21	0.68
4	<(**C602@1**)(C246@10)>	24	<(11 24 32)(242)(11 24 32)>	10.74	0.625

Table 6.12 List of community sequences and their most emerging patterns related to C602@1, C188@4 and C246@10. From left to right, the columns correspond to a community sequence s_{wise}, the number of nodes $\lvert S(s_{wise})\rvert$ supporting it, a characteristic pattern of this sequences s_{less}, the growth rate $Gr(s_{less}, S(s_{wise}))$ and support $sup(s_{less}, S(s_{wise}))$ of this pattern relatively to the supporting node set of the community sequence.

6.3.4 Conclusion

In section 6.3, we asked some questions related to the analysis of the LastFM network. Our framework helps us replying to these questions. We asked the question *Is there a clear separation among the listeners of different jazz types?* We see that there are well separated communities at each time slice. Some of those communities are associated to different jazz artists. However, each community does not correspond to a different artist, but it corresponds to the combination of artists. Moreover, we also notice by the help of emerging topological situations that one community including non-jazz listeners (fans of the Beatles or Pink Floyd) are less connected with the rest of the network but keep being well-connected

with each other as time goes by. Another question was *Do jazz listeners follow jazz events as well?* We did not see any trend showing a relation between the artists and events. In general, the most followed events are related to rock or metal. We also asked the question *if jazz listeners from same communities have similar taste of music*. In general, we can say that Miles Davis listeners appear in most of the communities. For some communities, we notice some common points about the different artists that different node groups listen. For those communities, we can say that their nodes have similar taste of music. But for some other communities, we notice very different trends inside of same community.

The anomalies we detected for LastFM can be grouped into three groups. The first ones are the anomalies appearing for one community. These are misplaced nodes which should be isolated but placed in one community via detection algorithm. The second and third types of anomalies are appearing at community sequences. The second type is the nodes listening to different artists than their communities. The third type is the nodes which listen same artist but much more times than the rest of their node groups following same community sequence.

6.4 Conclusion & Discussion

In this section, after having analyzed the communities in both networks, we want to summarize some common and different properties of DBLP and LastFM. Our results also show some facts about the evolution of communities in time. We also want to discuss them.

Both of the networks have a very dynamic community structure, changing very much at each time slice. We have seen that the communities of DBLP network are getting larger as time goes by, whereas it is not the case for LastFM. In LastFM, we can have large communities even at $t = 1$ or $t = 2$. This difference between the two networks might be due not only to differences in the modeled systems, but also in the considered durations of both the whole considered time period and the time slices. The DBLP network represents an evolution of 18 years, while the LastFM one corresponds to 1 single year. We also notice that characteristic topological situations of the communities of DBLP are changing with time. Thus, tracking these changes helps us to better understand common evolution in the communities. These evolutions of topological situations help us for reasoning on the changes of attributes as well. We benefit from both types of descriptors during the interpretation. For example, we see that node groups which are more embedded in their communities keep publishing in the same conferences. However, in LastFM, although we observe topological situations in a characteristic pattern more than once, they do not change very much.

For both DBLP and LastFM, we cannot claim that the communities are created around one interest. The characteristic patterns show us that there are some node groups having the same interest (publishing in same conference of listening same artist) inside of the communities, but they do not constitute entire communities. On the contrary, these groups leave and join different communities, or even disperse. Regarding the trends on community sequences at size larger than 1, we see that the node groups having same interest stay together. This means temporal communities are not completely homogenous. Their homogeneous parts have a trend of keep being together as time goes by. Nevertheless, we also want to remark that here, we use the term *homogeneous* just for the context of having same attribute values. We notice that some communities for both networks can be interpreted by a common theme although they include different interest. For example, we characterized a community of DBLP with the interest of Data Mining although it has more than one node group that each of them publishes at different conference or journal. Thus, here, our comments about homogeneity reflect only the similarity of attributes although there might be different homogeneity levels. We do not use attributes in our experiments representing more common features of the nodes like the subject of the conference/journal (data mining, data base, etc.) for DBLP or jazz-type of the artists (cool, modal, classic, fusion etc.), we might have different patterns showing more homogeneity for each community.

Regarding the homophily of the topological positions, we notice that the communities of DBLP can be characterized by topological situations (the most supported patterns are representing specific topological situations and the supports are all larger than 0.90). But LastFM communities do not have such a consensus about the topological situations because the most supported patterns' supports are mostly around 0.70.

The communities of one time slice include different homogeneous node groups. The characteristic patterns detected for the community sequences show that homogeneous node groups tend to stay together as time goes by. They either migrate to other communities together, or stay together in the same community. Especially, for LastFM, we had high supports for the most emerging patterns of community sequences. It shows that the nodes which are together at different time slices are highly similar. Here, if we recall the definition of community in social science (c.f Chapter 2), it expresses groups of persons sharing emotions, having the feeling of belonging together. Our experiments on these two networks reveal that not the structural temporal communities but the node groups having same interest inside of them tend to have the feeling of belonging together. Thus, maybe these node groups should be named as community.

Community evolution events are limited with merge, split, birth, death, hide, expand, contract and switch. We also had these events in our networks. The most common events we had are small-scale events. We want to underline that neither merge nor split occurs purely in our experiments. The general descriptions of these events are like two (or more) communities are getting together to create a new one or one community splits to create new ones. However, in our case, we see that not two (or more) communities completely merges but homogeneous node groups of different communities are getting together to be a part of another community in the next time slices. Thus, one community of a previous time slice can be the root of many other communities at the next time slices. It is same trend for split event. When communities are splitting, different homogeneous node groups are separating to different communities. We do not observe an expansion or contraction at the communities in terms of only increasing/decreasing their node number. However, in our experiments, it is mostly like that different parts of different communities are getting together in the next time slices and their size is thus increasing or decreasing. Briefly, in our experiments, we have seen that communities are including different homogeneous node groups such that some of them might move together in the next time slices. It would be interesting to analyze if there is a hierarchy between the different node groups of same community or if some trends coming from the homogeneity of one node group is absorbing some other trends. Empirically, we notice that these different homogeneous groups are linked to each other with bridges inside of the community. It would also be interesting to analyze deeper these bridge nodes.

Regarding the anomalies, we can say that our framework is capable of detecting the nodes which are different from than their communities. This difference can be both topologically or about attributes evolution. The anomalies can also be misplaced nodes by community detection algorithm. We have seen that some anomalies which are different in terms of their attributes may have same attribute in the future. Thus, our framework can be used for prediction.

Chapter 7

Conclusion

In this thesis, we worked on community structures in complex networks. A *community* is roughly defined as node group which has more links inside of the group relatively to the rest of the network. The presence of a community structure is a common property appearing in networks modeling many real-world complex systems of different domains. They constitute a rich part of complex network analysis. Naturally, there are many different studies performed in a wide range of areas ranging from biology to sociology. Most of these studies tackle the problem of community detection in different types of networks. There are right now hundreds of algorithms using different formal definitions of communities, developed for *plain*, *attributed* or *dynamic networks*. Some of the studies are oriented to the evaluation of the performance of community detection algorithms or generation realistic artificial network with predefined community structures. Although these studies are not directly related to the development of a community detection algorithm, they still serve the partitioning performance evaluation of these algorithms. We see that community detection is an important part of community structure studies. However, unless we do not interpret the communities, they are not more than a node group. Thus, they stay meaningless. For this reason, in this thesis, we specifically concentrate on *community interpretation*.

To the best of our knowledge, there is neither a formal definition nor a dedicated work for the community interpretation problem. The first and most prominent contribution of this thesis is proposing a formal definition of the problem of community interpretation. For us, it consists of two sub-problems: *finding an appropriate representation of the communities* and *choosing the most characteristic part of this representation*. Our second contribution is proposing a solution for these problems. We represent each community under the form of chronologically ordered sequences modeling the evolution of the topological measures and attribute values describing the nodes. Thus, we consider three important types of information presented in the network modeling: (1) the network topology reflecting node-to-node connections, (2) the attributes corresponding to an individual description of nodes and (3) the temporal information representing the network dynamics. For the characterization, we select the sequences fulfilling two criteria, *prevalence* and *distinctiveness*, and one condition, *being informative*. We propose a framework to tackle these problems. The different steps of our framework are: detecting the communities in the dynamic attributed network, creating a sequential database containing all the mentioned information, finding the frequent closed sequential patterns among this database thanks to the algorithm CloSpan [136], processing the growth rates of the sequential patterns, select the most characteristic patterns able to best cover the community.

Our third contribution is to perform an empirical analysis of the behavior of our framework. For this purpose, we used artificial network produced by two generative models. The first one is the LFR-D model by Greene et al. [52], which is based on the model developed by Lancichinetti et al. [76] to produce realistic static networks. LFR-D generates dynamic networks which have a predefined community structure evolving from one time slice to another. The evolution can be controlled through different

types of community-related events: birth-death, merge-split, expansion-contraction,hide-appear, switch. In the first part of our evaluation, we focused on the community detection step. We performed an empirical comparison of a selection of community detection algorithms based on the Louvain approach developped by Blondel et al. [15]. Some of these algorithms output what we call a *consensual community structure*, which is supposedly valid for the whole considered time period. Some others output what we call a sequence of community structures (one for each time slice), which we call *evolving community structure*. The algorithms we selected are called *Classic*, *Incremental*, *MultiStep* and *Integrated* Louvain. The MultiStep and Integrated methods produce consensual community structures, whereas the Classic and Incremental methods find evolving ones. We evaluated them by comparing the community structures they estimated, with the reference community ones, generated by the model. Our performance analysis reveals that when the reference communities are well separated, the algorithms producing evolving community structure estimate better communities than the others. Among them, Incremental Louvain gets more stable results. Another observation is that large-scale community events (merge-split, birth-death and hide-appear) introduce more difficulty in the community detection process than small-scale ones (expansion-contraction and switch).

In the second part of our evaluation, we focused on the pattern mining steps. We needed attributed dynamic networks, so we proposed LFR-DA, an extension of LFR-D able to add nodal attributes to an existing dynamic network. These attributes are controlled through three parameters: number, evolution percentage and distribution type. The first parameter determines the number of attributes defined for each node. The second controls the proportion of nodes whose attribute values change at each time slice. The third specifies how the values of each attribute are distributed over the nodes of a community. Our results show that the higher the number of attribute, the longer it takes for the framework to extract the patterns. Moreover, when the attributes values are completely homogeneous (community-wise), it is easier to extract the patterns. Nevertheless, when the attributes are heterogeneous, the execution times are not as long as expected. The longest execution times are obtained when the attribute values change very often. Thus, if a network is very dynamic, it is more difficult to extract its patterns. Regarding the different steps of our framework, mining the closed patterns is faster than calculating their growth rate procedure. The performance of the growth rate calculation directly depends on the number of patterns outputted by CloSpan.

Our last contribution is to have applied our framework to two real-worlds networks. The first is a DBLP network showing the evolution of co-authorship connections over 18 years . The second is a LastFM network, whose nodes represent users of this service, and links represent a combination of declared friendship and observed listening behavior, for the members of the Jazz user group. The total time span for the LastFM network is 1 year. Using our method, we interpret the communities of these two networks. For DBLP, we can say that the communities get larger with time. The most supported patterns involve mainly topological descriptors. Inside a given community, the majority of the nodes ($\sim 90\%$) have the same topological situation. Moreover, the topological situation of the nodes of a community are changing over time. Communities are characterized by more than one emerging pattern, each one involving attribute descriptor, and related to the publication of articles in specific conferences or journals. The community growth noticed for DBLP is not observed for LastFM. In this network, some of the most supported patterns represent topological roles, while some others reflect an interest towards a specific Jazz artist. The proportion of nodes following the most supported patterns are not as high as for DBLP ($\sim 70\%$), which means, the nodes of LastFM communities are less homogeneous about than the DBLP ones. The communities are covered by more than one emerging pattern. Thus as for DBLP, there are several node groups with different interests in a given community.

For both networks, the most interesting result is that the communities are not completely homogeneous. There are constituted of different node groups, which are homogeneous, but only in terms of certain descriptors. We notice that some of these node groups are staying together, in the sense they keep

on belonging to the same community over time, sometimes for a large number of time slices. Regarding the community evolution, we reveal that as time goes by, different homogeneous node groups of different communities of previous time slices are getting together in the next time slices. Additionally, our framework is able to detect the outlier nodes, i.e. those different from the rest of their community. We detect three types of outliers in DBLP. The first ones are not publishing at the same conferences/journals than their community-mate. The second outliers are those which are topologically different, although they publish at the same conferences/journals. The third outliers are those which are much more active or passive. The outliers detected for LastFM can be grouped into three groups, too. The first ones are misplaced nodes, in the sense the community detection algorithm wrongly assigned them to the considered community. These nodes are isolated, they should not belong to any community. The second outliers represent users listening to artists different from the community general habits. The third outliers are the nodes listening to the same artists than the rest of their community, but much more often. We also notice that some outliers outputted from our framework might have the possibility of being similar to their communities in the future. So, it can be used for prediction.

Our framework includes different steps. For each step, we choose some algorithms or methods. It is possible to replace these algorithms/methods with different ones. For the first step, we evaluate only the detection algorithms finding community structure under the form of partition. It would be very interesting to interpret the overlapping communities. In our study, we find out that homogeneous groups of different communities get together in next time slice. These node groups might be the overlapping parts of the communities. Making a comparative interpretation on both overlapping and non-overlapping communities can be very interesting to have a deepen knowledge about the community structures. Moreover, to have an idea about the sensitivity of our framework to the community detection methods, it can be complementary to interpret the result community structure of many different algorithms. In our second step, we use some specific attributes related to studied system. Finally, the characteristic patterns of communities are not around one attribute but some of them. For some communities, although there are different characteristic attributes, their theme are common. Thus, while choosing the attributes, one can determine different levels of attributes which reflect the features of the nodes at different level. For example, in DBLP application, it can very interesting not only using different conference/journal publication but also their theme. At the data preparation step (step 2), we try to determine different topological situations over the network by consider all the time slices together. However, for the networks like DBLP which shows a connection of more than 15 years, topological positions can vary too much. Thus, one can consider each time slice independently when preparing the data.

Our third and fourth step is related to pattern mining and determining emergence. From the applicative point of view, these steps consume the most time and resource. Depending on network and descriptor size, this time increases too much. In some experiments on Chapter 5, we cannot extract the results because of this drawback. The slowness of our framework makes its field of application limited. For instance, our framework can be used for offline analysis. It is not compatible for online systems. In our fifth step, we select the most representative characteristic patterns. Here, we define some criteria for the similarity of patterns. For different criteria from ours, the framework can output different results. We extract anomalies according to our selection criteria. For other criteria, these anomalies cannot be outputted either. Thus, it would be very interesting to extend this step in the way to order the nodes of communities according to the patterns they follow. More clearly, one can extend pattern selection with also a hierarchical order of the nodes. For example, the nodes which follow the most emerging/supported patterns can be at the higher level of the hierarchy. It could ensure more flexibility and robustness of anomaly extraction. Rather than outputting one or few nodes, the weakest level and its nodes can be outputted as anomaly.

Regarding the artificial network generator LFR-DA we propose, it is right now depending on three parameters we determined just to see the limits of our framework. This model can be extended in some

ways. Attributes are very system depended. It makes their simulation difficult. However, it would be very interesting to experimentally analyze different networks and determine some common points about structure+attribute relation. Then, these common points can be reflected to the generator model. Moreover, we take into account only node attributes. It can also be interesting to tackle with link attributes as well in the generator model.

Our experiments here are limited with the functioning of our framework. Nevertheless, it can be complementary to have a statistical analysis for different tasks. For example, when we follow the community evolution events, we notice that some node groups survive during long time while some others disappear. It might be because some powerful trends are absorbing the weak ones. So, a detailed statistical analysis on the different homogeneous node groups and their evolution can be useful to have an idea about why they survive or they disappear. Another perspective to our study is regarding the matches of the nodes of communities of different time slices and node groups which stay together at those communities over time. A detailed analysis on these matches can give us an idea about the proportions of the nodes of communities which join or leave from them. It is also necessary and could be very interesting to see the results of our framework on the networks from different domains. Here, the two networks we use have a size smaller than 5000. They are not very big. Both of them are social networks. Thus, we want to see the performance of our framework on larger networks having different types of attributes.

References

[1] Akoglu, L., Tong, H., Meeder, B., and Faloutsos, C. (2012). Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*, Series PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs, pages 439–450. SIAM / Omnipress.

[2] Asur, S., Parthasarathy, S., and Ucar, D. (2009). An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Trans. Knowl. Discov. Data*, 3(4):1–36.

[3] Aynaud, T., Fleury, E., Guillaume, J.-L., and Wang, Q. (2011). Communities in evolving networks: Definitions, detection, and analysis techniques. In *Dynamics On and Of Complex Networks, Volume 2*, Modeling and Simulation in Science, Engineering and Technology, pages 159–200. Springer New York.

[4] Aynaud, T. and Guillaume, J.-L. (2010a). Long range community detection. In *in Proceedings of Latin American Workshop on Dynamic Networks*, Series Long range community detection.

[5] Aynaud, T. and Guillaume, J.-L. (2010b). Static community detection algorithms for evolving networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 513–519.

[6] Aynaud, T. and Guillaume, J.-L. (2011). Multi-step community detection and hierarchical time segmentation in evolving networks. In *The 5th SNA-KDD Workshop 2011 (SNA-KDD 11)*, Series Multi-Step Community Detection and Hierarchical Time Segmentation in Evolving Networks.

[7] Bagrow, J. P. (2008). Evaluating local community methods in networks. *Journal of Statistical Mechanics: Theory and Experiment*, page P05001.

[8] Balasubramanyan, R. and Cohen, W. (2010). Block-lda: Jointly modeling entity-annotated text and entity-entity links. In *ICML 2010 Workshop on Topic Models: Structure, Applications, Evaluation, and Extensions*.

[9] Barabasi, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

[10] Barabasi, A. and Albert, R. (2002). Statistical mechanics of complex networks. *Reviews of Modern physics*, 74(1):47–96.

[11] Barabasi, A.-L. (2002). *Linked*. Perseus Publishing.

[12] Birkhoff, G. (1961). *Lattice Theory*. American Mathematical Society.

[13] Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022.

[14] Blondel, V. D. (2011). The louvain method for community detection in large networks. http://perso.uclouvain.be/vincent.blondel/research/louvain.html.

[15] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, page P10008.

[16] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D. (2006). Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308.

[17] Bonacich, P. (1987). Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182.

[18] Bonacich, P. and Lloyd, P. (2001). Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23(3):191–201.

[19] Borgnat, P., Fleury, E., Guillaume, J.-L., Magnien, C., Robardet, C., and Scherrer, A. (2008). Evolving networks. In *NATO ASI on Mining Massive Data Sets for Security*, NATO Science for Peace and Security Series D: Information and Communication Security, pages 198–204. IOS Press.

[20] Brandes, U. (2001). A faster algorithm for betweenness centrality. In *Journal of Mathematical Sociology*, volume 25, pages 163–177.

[21] Chakrabarti, D., Kumar, R., and Tomkins, A. (2006). Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Series Evolutionary clustering, Philadelphia, PA, USA. ACM.

[22] Chang, J. (2009). Relational topic models for document networks. In *In Proc. of Conf. on AI and Statistics (AISTATS)*.

[23] Clauset, A. (2005). Finding local community structure in networks. *Physical Review E*, 72(2):026132.

[24] Combe, D., Largeron, C., Egyed-Zsigmond, E., and Gery, M. (2012). Detection de communautes dans des reseaux scientifiques aă partir de donnees relationnelles et textuelles. In *3eme Conference sur les modeles et l'analyse de reseaux : approches mathematiques et informatiques*.

[25] Coscia, M., Giannotti, F., and Pedreschi, D. (2011). A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546.

[26] Costa, L. F., Oliveira Jr., O. N., Travieso, G., Rodrigues, F. A., Villas Boas, P. R., Antiqueira, L., Viana, M. P., and Rocha, L. E. C. (2011). Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412.

[27] Cruz, J. D., Bothorel, C., and Poulet, F. (2011). Entropy based community detection in augmented social networks. In *International Conference on Computational Aspects of Social Networks*, page 163 168.

[28] da Fontura Costa, L., Rodrigues, F. A., Travieso, G., and Villas Boas, P. R. (2007). Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242.

[29] Dang, T. A. and Viennet, E. (2012). Community detection based on structural and attribute similarities. In *6th International Conference on Digital Society*, Series Community Detection based on Structural and Attribute Similarities, pages 7–12.

[30] Danon, L., Diaz-Guilera, A., and Arenas, A. (2006). The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, page 11010.

[31] Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, page P09008.

[32] Danon, L., Duch, J., Arenas, A., and Diaz-Guilera, A. (2007). Community structure identification. In *Large Scale Structure and Dynamics of Complex Networks: From Information Technology to Finance and Natural Science*, pages 93–113. World Scientific.

[33] de Nooy, W., Batagelj, V., and Mrvar, A. (2005). *Exploratory Social Network Analysis with Pajek*. Structural Analysis in the Social. Cambridge University Press.

[34] Derenyi, I., Palla, G., and Vicsek, T. (2005). Clique percolation in random networks. *Physical Review Letters*, 94(16).

[35] Desmier, E., Plantevit, M., Robardet, C., and Boulicaut, J.-F. (2012). Cohesive co-evolution patterns in dynamic attributed graphs. In Ganascia, J.-G., Lenca, P., and Petit, J.-M., editors, *Discovery Science*, volume 7569 of *Lecture Notes in Computer Science*, pages 110–124. Springer Berlin Heidelberg.

[36] Donetti, L. and Munoz, M. A. (2004). Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, (10):P10012.

[37] Dugué, N., Labatut, V., and Perez, A. (2014). Identification de rôles communautaires dans des réseaux orientés appliquée à twitter. In *14eme conference Extraction et Gestion des Connaissances*.

[38] Duncan, J. W. and Steven, H. S. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

[39] Edler, D. and Rosvall, M. (2013). Mapequation.

[40] Erdos, P. and Renyi, A. (1959). On random graphs. *Publicationes Mathematicae*, 6:290–297.

[41] Falkowski, T., Barth, A., and Spiliopoulou, M. (2007). Dengraph: A density-based community detection algorithm. In *IEEE/WIC/ACM International Conference on Web Intelligence*, Series DENGRAPH: A Density-based Community Detection Algorithm, pages 112–115.

[42] Fleury, E., Guillaume, J. L., Robardet, C., and Scherrer, A. (2007). Analysis of dynamic sensor networks: Power law then what? In *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*, pages 1–13.

[43] Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75–174.

[44] Fortunato, S. and Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Science of the USA*, 104(1):36–41.

[45] Fortunato, S. and Lancichinetti, A. (2009). Community detection algorithms: A comparative analysis: Invited presentation, extended abstract. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 27:1–27:2, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[46] Fournier-Viger, P. (2010). Spmf: A sequential mining framework.

[47] Fred, A. L. N. and Jain, A. K. (2003). Robust data clustering. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *Series Robust Data Clustering*, pages 128–136. IEEE Computer Society.

[48] Freeman, L. (1979). Centrality in social networks i: Conceptual clarification. *Social Networks*, 1(3):215–239.

[49] Gan, G., Ma, C., and Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability. Society for Industrial and Applied Mathematics, Philadelphia, US-PA.

[50] Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826.

[51] Good, B., Montjoye, Y. D., and Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106.

[52] Greene, D., Doyle, D., and Cunningham, P. (2010). Tracking the evolution of communities in dynamic social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 176–183.

[53] Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018.

[54] Görke, R., Maillard, P., Schumm, A., Staudt, C., and Wagner, D. (2013). Dynamic graph clustering combining modularity and smoothness. *ACM Journal on Experimental Algorithmics*, 18:1.5:1.1–1.5:1.29.

[55] Guillaume, J.-L., Blondel, V. D., and Lambiotte, R. (2008). Louvain method: Finding communities in large networks.

[56] Guillaume, J.-L. and Latapy, M. (2006). Bipartite graphs as models of complex networks. *Physica A: Statistical Mechanics and its Applications*, 371(2):795–813.

[57] Guimera, R. and Nunes Amaral, L. (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900.

[58] Guimera, R., Sales-Pardo, M., and Amaral, L. A. N. (2004). Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101.

[59] Guimera, R.ă, R. and AmaGuimerai, N. (2005). Cartography of complex networks: modules and universal roles. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(02):P02001.

[60] Gusfield, J. R. (1975). *Community: a critical response*. Harper & Row, 1978.

[61] Harary, F. (1969). *Graph Theory*. Addison-Wesley Publishing Company.

[62] Hartigan, J. A. and Wong, M. A. (1979). A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108.

[63] Hoff, P., Raftery, A., and Handcock, M. (2002). Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098.

[64] Hopcroft, J., Khan, O., Kulis, B., and Selman, B. (2004). Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5249–5253.

[65] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

[66] igraph Project, T. (2005). The igraph library.

[67] Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

[68] Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43.

[69] Kim, M.-S. and Han, J. (2009). A particle-and-density based evolutionary clustering method for dynamic networks. *Proc. VLDB Endow.*, 2(1):622–633.

[70] Kleinberg, J. M. (1999). Hubs, authorities, and communities. *ACM Comput. Surv.*, 31(4es):5.

[71] Kunegis, J. (2014a). The koblenz network collection: Clustering coefficient distribution.

[72] Kunegis, J. (2014b). The koblenz network collection: Wikinews, english.

[73] Labatut, V. and Balasque, J.-M. (2012). Detection and interpretation of communities in complex networks: Practical methods and application. In Abraham, A. and Hassanien, A.-E., editors, *Computational Social Networks*, pages 81–113. Springer London.

[74] Labatut, V. and Balasque, J.-M. (2013). Informative value of individual and relational data compared through business-oriented community detection. In *The Influence of Technology on Social Network Analysis and Mining*, volume 6 of *Lecture Notes in Social Networks*, pages 303–330. Springer Vienna.

[75] Lamberson, P. (1 March 2013). Social dynamics.

[76] Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4 Pt 2):046110.

[77] Lancichinetti, A., Kivela, M., Saramaki, J., and Fortunato, S. (2010). Characterizing the community structure of complex networks. *PLoS ONE*, 5(8):e11976.

[78] Lancichinetti, A., Radicchi, F., Ramasco, J., and Fortunato, S. (2011). Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961.

[79] Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, Series Graphs over time: densification laws, shrinking diameters and possible explanations, Chicago, Illinois, USA. ACM.

[80] Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. (2008). Statistical properties of community structure in large social and information networks. In *17th International Conference on World Wide Web*, Series Statistical Properties of Community Structure in Large Social and Information Networks, pages 695–704.

[81] Ley, M. and Dagstuhl, S. (1993). The dblp computer science bibliography.

[82] Li, Z., Lu, S., Myagmar, S., and Zhou, Y. (2006). Cp-miner: Finding copy-paste and related bugs in large-scale software code. *IEEE Transactions on Software Engineering*, 32(3):176–192.

[83] Liu, Y., Mizil, A., and Gryc, W. (2009). Topic-link lda: joint models of topic and author community. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 665–672, Montreal, Quebec, Canada. ACM.

[84] Liu, Y., Wang, Q., Wang, Q., Yao, Q., and Liu, Y. (2007). Email community detection using artificial ant colony clustering. In *Advances in Web and Network Technologies, and Information Management*, Lecture Notes in Computer Science, pages 287–298. Springer, Berlin / Heidelberg.

[85] Mabroukeh, N. R. and Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Comput. Surv.*, 43(1):1–41.

[86] McMillan, D. W. and Chavis, D. M. (January 1986). Sense of community: A definition and theory. *Journal of Community Psychology*, 14.

[87] McPherson, M., Lovin, L., and Cook, J. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444.

[88] Miller, F., Stiksel, M., Breidenbruecker, M., and Willomitzer, T. (2002). Lastfm.

[89] Mitchell, M. (2006). Field review: Complex systems: Network thinking. *Artif. Intell.*, 170(18):1194–1212.

[90] Molloy, M. and Reed, B. (1995). A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6(2/3):161–179.

[91] Moody, J. (2001). Race, school integration, and friendship segregation in america. *American Journal of Sociology*, 107(3):679–716.

[92] Moser, F., Ge, R., and Ester, M. (2007). Joint cluster analysis of attribute and relationship data withouta-priori specification of the number of clusters. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 510–519, San Jose, California, USA. ACM.

[93] Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Science of the USA*, 103(23):8577–8582.

[94] Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45:167–256.

[95] Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133.

[96] Newman, M. E. J. (April 19, 2013). Network data.

[97] Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113.

[98] Orman, G. and Labatut, V. (2009). A comparison of community detection algorithms on artificial networks. In Gama, J., Costa, V., Jorge, A., and Brazdil, P., editors, *Discovery Science*, volume 5808 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg.

[99] Orman, G., Labatut, V., and Plantevit, M.and Boulicaut, J.-F. (2014a). A method for characterizing communities in dynamic attributed complex networks. In *ASONAM*.

[100] Orman, G., Labatut, V., and Plantevit, M.and Boulicaut, J.-F. (2014b). Une méthode pour caractériser les communautés des réseaux dynamiques à attributs. In *EGC 2014*, volume RNTI-E-26, pages 101–112. 14ème Conférence Extraction et Gestion des Connaissances.

[101] Orman, G. K. and Labatut, V. (2010). The effect of network realism on community detection algorithms. In *ASONAM*, Series The Effect of Network Realism on Community Detection Algorithms, pages 301–305, Odense, DK.

[102] Orman, G. K., Labatut, V., and Cherifi, H. (2011). Qualitative comparison of community detection algorithms. In *Digital Information and Communication Technology and Its Applications*, volume 167 of *Communications in Computer and Information Science*, pages 265–279. Springer Berlin Heidelberg.

[103] Orman, G. K., Labatut, V., and Cherifi, H. (2012). Comparative evaluation of community detection algorithms: a topological approach. *Journal of Statistical Mechanics: Theory and Experiment*, 2012(08):P08001.

[104] Palla, G., Barabasi, A., and Vicsek, T. (2007). Quantifying social group evolution. *Nature*, 446(7136):664–667.

[105] Palla, G., Derenyi, I., Farkas, I., and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818.

[106] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, Washington, DC, USA. IEEE Computer Society.

[107] Plantevit, M. and Cremilleux, B. (2009). Condensed representation of sequential patterns according to frequency-based measures. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, Series Condensed Representation of Sequential Patterns According to Frequency-Based Measures, pages 155–166, Lyon, France. Springer-Verlag.

[108] Poncela, J., Gomez-Gardeles, J., Floria, L. M., Sanchez, A., and Moreno, Y. (2008). Complex cooperative networks from evolutionary preferential attachment. *PLoS ONE*, 3(6):e2449.

[109] Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks. In Yolum, p., Güngör, T., Gürgen, F., and Özturan, C., editors, *Computer and Information Sciences - ISCIS 2005*, volume 3733 of *Lecture Notes in Computer Science*, pages 284–293. Springer Berlin Heidelberg.

[110] Porter, M. A., Onnela, J.-P., and Mucha, P. J. (2009). Communities in networks. *Notices of the American Mathematical Society*, 56(9):1082–1166.

[111] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Science of the USA*, 101(9):2658–2663.

[112] Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106.

[113] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

[114] Reichardt, J. and Bornholdt, S. (2004). Detecting fuzzy community structures in complex networks with a potts model. *Physical Review Letters*, 93(21):218701.

[115] Rosvall, M. and Bergstrom, C. T. (2007). An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104(18):7327–7331.

[116] Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Science of the USA*, 105(4):1118.

[117] Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65.

[118] Ruan, Y., Fuhry, D., and Parthasarathy, S. (2013). Efficient community detection in large networks using content and links. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1089–1098, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[119] Sabidussi, G. (1966). The centrality index of a graph. *Psychometrika*, 31(4):581–603.

[120] Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1):27–64.

[121] Schank, T. and Wagner, D. (2005). Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9.

[122] Sheng, P., Changjia, C., and Ting, W. (2009). A realtime community detection algorithm: incremental label propagation. In *Future Information Networks, 2009. ICFIN 2009. First International Conference on*, pages 313–317.

[123] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905.

[124] Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., and Schult, R. (2006). Monic: Modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 706–711, New York, NY, USA. ACM.

[125] Stattner, E. and Collard, M. (2012). Social-based conceptual links: Conceptual analysis applied to social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 25–29.

[126] Stattner, E. and Collard, M. (2013). Frequent conceptual links and link-based clustering: a comparative analysis of two clustering techniques. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Series Frequent conceptual links and link-based clustering: a comparative analysis of two clustering techniques, pages 134–141, Niagara, Ontario, Canada. ACM.

[127] Steinhaeuser, K. and Chawla, N. (2008). Community detection in a large real-world social network. In *Social Computing, Behavioral Modeling, and Prediction*, pages 168–175.

[128] Strogatz, S. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

[129] Takes, F. W. and Kosters, W. A. (2013). Computing the eccentricity distribution of large graphs. *Algorithms*, 6(1):100–118.

[130] Tanja, H., Andrea, K., and Dorothea, W. (2014). Clustering evolving networks. *arXiv:1401.3516v1*.

[131] Tasgin, M. and Bingol, H. (2006). Community detection in complex networks using genetic algorithm.

[132] Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, 32(4):425–443.

[133] Tumminello, M., Miccichè, S., Lillo, F., Varho, J., Piilo, J., and Mantegna, R. N. (2011). Community characterization of heterogeneous complex systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(01):P01019.

[134] van Dongen, S. (2008). Graph clustering via a discrete uncoupling process. *SIAM J Matrix Anal Appl*, 30(1):121–141.

[135] Yan, X. (2003). Software - clospan: Closed sequential pattern mining package.

[136] Yan, X., Han, J., and Afshar, R. (2003). Clospan: Mining closed sequential patterns in large datasets. In *Proc. 2003 Int. SIAM Conf. on Data Mining (SDM '03)*, pages 166–177.

[137] Yang, J., McAuley, J., and Leskovec, J. (2013). Community detection in networks with node attributes. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1151–1156.

[138] Yang, T., Chi, Y., Zhu, S., Gong, Y., and Jin, R. (2011). Detecting communities and their evolutions in dynamic social networksâĂŤaÂăbayesian approach. *Machine Learning*, 82(2):157–189.

[139] Zaki, M. J. and Hsiao, C.-J. (2002). Charm: An efficient algorithm for closed itemset mining. In Grossman, R. L., Han, J., Kumar, V., Mannila, H., and Motwani, R., editors, *SDM*. SIAM.

[140] Zhou, Y., Cheng, H., and Yu, J. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2:718–729.

[141] Zurich, E. (2013). A network perspective on software modularity. http://www.sg.ethz.ch/research/topics/social-se/modular-coherence/.

Chapter 8

Appendix1

When we prepare the enlarged database, we use 9 topological measures explained in Chapter 2. As explained in Chapter 4, we clustered the values of these measures to determine final items of database. We detect 3 topological field. Each node takes a value for all of those fields. We present those fields and the role inside of the topological fields by integer codes because CloSpan algorithm accepts integer items. Coding is very simple. It is two digit integer numbers. The first digit represents the order of topological field and the second digit represents the role in this field. Here, we present these codes for not only topological situation but also attribute values. In section 8.1, we present the codes of DBLP network and in section 8.2,we present the ones for LastFM network.

8.1 DBLP Network

All related item codes to topological situations are given in Table 8.1. We not only use topological measures but also node attributes. These attributes are the number of conference/journal publications. We determine 5 ranges: $1,2,3,4$ and $[5;\infty[$ for each conference/journal publications and 5 ranges : $[1;5],]5;10],]10;20],]20;50]$ and $]50;\infty[$ for total journal/conference publication. Each of them is coded by $1,2,3,4$ and 5 respectively. We list the attributes names and the codes we give for them in Table 8.2. The items related to attributes are created by the same coding method we explained for topological roles. We use two digit integers. The first one is attribute field code and second one is the code representing its range. For example, code 72 means the number of publication in CommunACM is 2.

8.2 LastFM Network

All related item codes to topological situations are given in Table 8.3. We use two types of attributes of nodes. First one is the artist they listen and second one is event types that they attend. We determine 5 categories that the intervals of the numbers of artist listening are $[1;5],]5;30],]30;90],]90;200]$ and $]200;\infty[$. Each of them is coded with $1,2,3,4$ and 5 respectively. For the number of the events types, we have 4 categories with the intervals $[1;4],]4;7],]7;11]$ and $]11;\infty[$. They are coded with $1,2,3$ and 4 respectively. The names of the all attributes are listed in Table 8.4. We apply same coding method with DBLP. So, for example item code 224 means that the number of listening The Beatles is between 90 and 200. Or, 401 means the number of attending a different event not listed in the Table 8.4 (other events) is between 1 and 4.

	Item code	Meaning
General Topological Role	11	Having especially high betweenness, high local transitivity, average to high degree
	12	Having average local transitivity, average degree, high betweenness
	13	Having no or very few degree, no centrality, highly belong to its community, no connections from other communities
Centrality Oriented Role	21	having high eigenvector and closeness centrality
	22	Having high betweenness, low closeness and high eccentricity
	23	Having average betweenness and average closeness
	24	Not central at all
Community Oriented Role	31	Belonging completely to its community with no or few external connections, mostly non-hub but still there are some hubs
	32	Connections from other communities, connections distributed among other communities, mostly non-hub but still there are some hubs

Table 8.1 Items Related to Topological Roles and Their Meanings for DBLP

Field Code	Field Name
4	DMKD
5	SAC
6	ICDM
7	CommunACM
8	IEEE Transaction Knowledge and Data Engineering
9	IEEE Intelligent Systems
10	SIGKDD Explorations
11	DASFAA
12	IJCAI
13	IDA
14	ICDE
15	PAKDD
16	AAAI
17	PVLDB
18	CIKM
19	ECML_PKDD
20	KDD
21	VLDB
22	EDBT
23	Intelligent Data Analysis
24	ICML
25	ICDT
26	SDM
27	Knowledge Information Systems
28	TKDD
29	PODS
30	VLDBJ
31	Statistical Analysis and Data Mining
32	SIGMOD
33	Machine Learning
34	ACM Transactions on Database Systems
35	JMLR
36	ACM Transactions on Information Systems
37	Journal of Intelligent Information Systems
38	Information Systems
39	Data Knowledge Engineering
40	Pattern Recognition
41	BioInfo
42	BMC Biology
43	ECAI
44	WWW
45	DEXA
46	ILP
47	TOTAL JOURNAL
48	TOTAL CONFERENCE

Table 8.2 Attribute Field Codes and Their Names for DBLP

	Item code	Meaning
General Topological Role	11	Having no central position, average degree, high within module degree (hub), low participation coefficient
	12	Having central position, high degree, high local transitivity
Centrality Oriented Role	21	Having high eigenvector, betweenness and closeness centrality
	22	Having average eigenvector and average closeness centrality, non-between
	23	Not central at all
	24	Having high closeness but other centralities are not very low
Community Oriented Role	31	Community Hub with some external connections, its connections are distributed uniformly
	32	Community Non-hub with very few external connections, mostly belong to its community
	33	Community Non-hub with some external connections, its connections are distributed uniformly
	34	Community Non-hub with some external connections, but its connections to other communities are mostly to one community

Table 8.3 Items Related to Topological Roles and Their Meanings for LastFM

Field Code	Field Name	Field Type
4	Miles Davis	Artist
5	Herbie Hancock	Artist
6	Keith Jarrett	Artist
7	Louis Armstrong	Artist
8	Ella Fitzgerald	Artist
9	Charlie Parker	Artist
10	Chet Baker	Artist
11	Nina Simone	Artist
12	Frank Sinatra	Artist
13	John Coltrane	Artist
14	Django Reinhardt	Artist
15	Thelonious Monk	Artist
16	Count Basie	Artist
17	Charles Mingus	Artist
18	Amy Winehouse	Artist
19	Norah Jones	Artist
20	The Rolling Stones	Artist
21	Daft Punk	Artist
22	The Beatles	Artist
23	Radiohead	Artist
24	Pink Floyd	Artist
25	jazz	Event
26	bebop	Event
27	swing	Event
28	rock	Event
29	punk	Event
30	classic	Event
31	pop	Event
32	fusion	Event
33	hip-hop	Event
34	funk	Event
35	blues	Event
36	metal	Event
37	folk	Event
38	R&B	Event
39	soul	Event
40	other	Event

Table 8.4 Attribute Field Codes and Their Names for LastFM