Detection and dynamic of local communities in large social networks

Christel Blaise Ngonmang Kaledje

UNIVERSITÉ PARIS 13

THÈSE

Présentée pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE PARIS 13, SORBONNE PARIS CITE
Spécialité
INFORMATIQUE

Detection and Dynamic of Local Communities in Large Social Networks

---

Christel Blaise NGONMANG KALEDJE

Jury

Rapporteurs:	
Jean-Loup GUILLAUME	Professeur, Université de La Rochelle
Sihem AMER-YAHIA	Directeur de Recherche CNRS, Université de Grenoble
Examinateurs:	
Younes BENNANI	Professeur, Université Paris 13
Francoise SOULIE-FOGELMAN	Conseiller Scientifique, Institut Mines Telecom
Maurice TCHUENTE	Professeur, Université de Yaoundé I
Directeur:	
Emmanuel VIENNET	Professeur, Université Paris 13

Soutenue publiquement le: 27 Novembre 2014

"La connaissance s'acquiert par l'expérience, tout le reste n'est que de l'information."

- Albert Einstein

A Charlotte, Tiphanie et Emmany.

REMERCIEMENTS

Je ne saurais présenter cette thèse sans exprimer ma gratitude tout d'abord au seigneur mon Dieu qui n'a jamais cessé de me combler de ses grâces. Ma gratitude va ensuite au professeur Emmanuel Viennet qui m'encadre depuis mon stage de master et qui n'a pas cessé de m'apporter tout le support dont j'avais besoin durant ces années. Merci pour tout. Ensuite je souhaite exprimer toute ma gratitude au professeur Maurice Tchuenté qui m'a apporté son support depuis le début de mon master et ne cesse d'être disponible jusqu'aujourd'hui et sans qui je n'aurais sans doute jamais rencontré le professeur Viennet. Un livre entier ne saurait suffire pour dire tout ce que vous avez fait pour moi. Merci "grand prof.".

Je remercie tous les membres du jury qui ont pris, malgré des agendas toujours très chargés, le temps d'évaluer cette thèse. Merci pour vos remarques et suggestions qui ne peuvent qu'améliorer la qualité de ce travail.

Ma gratitude va ensuite à toute ma (grande) famille qui a toujours cru en moi et m'a soutenu. Je remercie chacun de vous. Je remercie chacune des personnes que j'ai rencontré dans ma vie et qui a d'une manière ou d'une autre contribué à ce que je suis aujourd'hui. Je ne saurais terminer ces remerciements sans citer les personnes exceptionnelles que j'ai eu à rencontrer durant ces années de thèse et avec qui j'ai pu partager un bureau, travailler sur des projets, ou tout simplement discuter.

Thanks to all of you.

RÉSUMÉ

Les réseaux sont présents dans plusieurs contextes et applications: biologie, transports, réseaux sociaux en ligne, etc. De nombreuses applications récentes traitent d'immenses volumes de données personnelles. Les liens entre les personnes dans ces données peuvent traduire des liens d'amitiés, des échanges de messages, ou des intérêts communs. Les entités impliquées dans les réseaux, et spécialement les personnes, ont tendance à former des communautés. Dans ce contexte, une communauté peut être définie comme un ensemble d'entités qui interagissent beaucoup plus entre elles qu'avec le reste du réseau.

La détection de communautés dans les grands réseaux a largement été étudiée pendant ces dernières années, suite aux travaux précurseurs de Newman qui a introduit le critère de modularité. Toutefois, la majorité des algorithmes de détection de communautés supposent que le réseau est complètement connu et qu'il n'évolue pas avec le temps.

Dans cette thèse, nous commençons par proposer de nouvelles méthodes pour la détection de communautés locales (en considérant uniquement le voisinage d'un nœud donné et sans accéder à la totalité du réseau). Nos algorithmes sont plus efficaces que ceux de l'état de l'art. Nous montrons ensuite comment utiliser les communautés détectées pour améliorer la prévision de comportements utilisateurs. Dans un deuxième temps, nous proposons des approches pour prévoir l'évolution des communautés détectées. Ces méthodes sont basées sur des techniques d'apprentissage automatique. Enfin, nous proposons un framework général pour stocker et analyser les réseaux distribués dans un environnement "Big Data".

Les méthodes proposées sont validées en utilisant (entre autre) des données réelles issues d'un partenaire industriel fournissant un des réseaux en ligne les plus utilisés en France (40 millions d'utilisateurs).

Mots-clés: Réseaux sociaux dynamiques, apprentissage automatique, communautés locales, Big Data.

ABSTRACT

Complex networks arises in many contexts and applications: biology, transports, online social networks (OSN). Many recent applications deal with large amount of personal data. The links between peoples may reflect friendship, messaging, or some common interests. Entities in complex network, and especially persons, tend to form communities. Here, a community can be defined as a set of entities interacting more between each other than with the rest of the network.

The topic of community detection in large networks as been extensively studied during the last decades, following the seminal work by Newman, who popularized the modularity criteria. However, most community detection algorithms assume that the network is entirely known and that it does not evolve with time. This is usually not true in real world applications.

In this thesis, we start by proposing novel methods for local community identification (considering only the vicinity of a given node, without accessing the whole graph). Our algorithms experimentally outperform the state-of-art methods. We show how to use the local communities to enhance the prediction of a user's behaviour. Secondly, we propose some approaches to predict the evolution of the detected communities based on machine learning methods. Finally we propose a framework for storing and processing distributed social networks in a Big Data environment.

The proposed methods are validated using (among others) real world data, provided by an industrial partner operating a major social network platform in France (40 millions of users).

Keywords: social network analysis, machine learning, local community detection, Big Data.

CONTENTS

1 Introduction **1**
- 1.1 Context . . . 2
- 1.2 Contribution . . . 3
- 1.3 Organisation . . . 5

2 Basic definitions and notations **7**
- 2.1 Basic notions on social networks . . . 7
- 2.2 Basic notions on supervised learning for classification . . . 11
  - 2.2.1 Problem definition . . . 11
  - 2.2.2 Model evaluation . . . 11
  - 2.2.3 Introduction to support vector machines . . . 12
- 2.3 Conclusion . . . 14

3 Global community detection in static networks **15**
- 3.1 Introduction . . . 15
- 3.2 Different definitions of communities . . . 17
- 3.3 Quality functions . . . 17
  - 3.3.1 Conductance . . . 17
  - 3.3.2 Performance . . . 18
  - 3.3.3 Modularity . . . 19
- 3.4 Some problems similar to community detection . . . 19
  - 3.4.1 Graph Partitioning . . . 20
  - 3.4.2 Data clustering . . . 20
- 3.5 Global community detection . . . 21
  - 3.5.1 Divisive methods . . . 21
  - 3.5.2 Agglomerative methods . . . 23
  - 3.5.3 Spectral methods . . . 25
  - 3.5.4 Random walks . . . 26

3.5.5 Information diffusion methods . . . 27
3.5.6 Ensemble methods . . . 27
3.5.7 Community detection in attributed graphs . . . 28
3.5.8 Methods to detect overlapping communities . . . 28
3.5.9 Summary on global community detection . . . 31
3.6 Evaluation of community detection methods . . . 32
3.6.1 Evaluation with ground truth . . . 32
3.6.2 Evaluation without ground truth . . . 34
3.6.3 Presence of a community structure in a network . . . 34
3.7 Conclusion . . . 34

4 Local community identification in social networks 35
4.1 Introduction . . . 35
4.2 Previous methods . . . 36
4.2.1 General greedy scheme for community detection . . . 37
4.2.2 Quality functions for local community identification . . . 39
4.3 Improvements for local community identification in networks . . . 42
4.4 Identification of overlapping local communities . . . 45
4.5 Performance evaluation . . . 46
4.5.1 Tests on toy examples . . . 46
4.5.2 Results on NCAA 2000 Football League . . . 46
4.5.3 Comparison with the LFR benchmark . . . 48
4.5.4 Results on Netscience . . . 50
4.5.5 Results on Amazon 2006 . . . 52
4.5.6 Results on Skyrock friendship network . . . 54
4.5.7 Results on Skyrock words network . . . 55
4.6 Related work . . . 55
4.6.1 Egomunities of the first level . . . 59
4.6.2 Multi-ego-communities . . . 60
4.7 Conclusion . . . 61

5 Local community identification applied to the prediction of user behaviours 63
5.1 Introduction . . . 63
5.2 Churn prediction in social networks . . . 65
5.2.1 Proposed methodology . . . 66
5.2.2 Dataset description and attributes extraction . . . 67
5.2.3 Experimentations and results . . . 70
5.3 Recommendation . . . 75
5.3.1 Introduction to recommender systems . . . 75
5.3.2 Proposed Social recommendation model . . . 76

5.3.3 Evaluation 77
5.4 Conclusion and discussions 79

6 Communities in dynamic networks 81
6.1 Introduction 81
6.2 Dynamic communities 83
6.2.1 Community tracking 83
6.2.2 Community updating 84
6.2.3 Long term-communities detection 85
6.3 The link prediction problem 86
6.3.1 Probabilistic methods 87
6.3.2 Transitivity-based methods 88
6.3.3 Attributes-based methods 90
6.4 A supervised method for local Community prediction 91
6.5 Community prediction through interactions prediction 92
6.5.1 Interaction prediction in complex networks 92
6.6 Evaluation and discussion 96
6.6.1 Dataset Description 96
6.6.2 Evaluation of interaction prediction 98
6.6.3 Local community prediction evaluation 100
6.7 Conclusions and perspectives 102

7 Towards a Framework for storing and analysing distributed networks 105
7.1 Introduction 105
7.2 Tools for analysis and storage of large social networks 106
7.2.1 Tools for social network analysis 107
7.2.2 Hadoop Map Reduce 108
7.2.3 Pregel and Giraph 109
7.2.4 Pegasus 109
7.2.5 Graph databases and NoSQL 110
7.2.6 Summary of network analysis tools 111
7.3 Framework description 111
7.3.1 Distributed Database Layer (DDL) 112
7.3.2 Data Analysis Layer (DAL) 113
7.3.3 Data Visualisation Layer (DVL) 113
7.4 An implementation of the framework 113
7.4.1 Distributed Database Layer 114
7.4.2 Data Analysis Layer 116
7.4.3 Data Visualisation Layer 117
7.5 Implementation of our methods 118

7.5.1 Experimental Setup 118
7.5.2 Local community detection 118
7.5.3 Local community prediction 118
7.6 Conclusion 121

8 Conclusions and Future Directions **123**
8.1 Summary of the contributions 123
8.2 Future directions 125

LIST OF FIGURES

2.1 Some graph examples: an undirected and unweighed graph (a), an undirected and unweighed graph (b), a directed and unweighed graph (c) and a directed and unweighed graph (d). 8
2.2 Example of community structure. 10
2.3 ROC curves examples. 13
2.4 SVM's Margin illustration. 14

3.1 Example of community structure for the computation of the conductance. 18
3.2 A dendogram structure. 22
3.3 Hierarchic Optimisation of modularity with the Louvain's method [18]. 24
3.4 A *4-cliques.* percolation example. 29
3.5 Link partitioning. 30

4.1 Subsets used in greedy algorithms. 37
4.2 A node highly connected with nodes outside its community. . 39
4.3 Pathological cases for quality functions R and M. 40
4.4 Node 7 is on the border of the local community on the right. . 42
4.5 Example of community structure. 43
4.6 The two local communities identified by IOLoCo for the central node. 46
4.7 No community is found by IOLoCo for the central node. . . . 47
4.8 Four communities identified by IOLoCo for node 0. 47
4.9 Performance (acc_m) of the 5 algorithms on six datasets (lower values of ν mean more separated communities). 49
4.10 Performance (NMI) of the 4 algorithms on six datasets (lower values of ν mean more separated communities). 50
4.11 Size distributions of the local communities found for Netscience 51

4.12 Netscience - Analysis of the nodes for which Chen et al.'s method returns no community. 51
4.13 Netscience - sub-communities of nodes for which Chen et al.'s method fails. 52
4.14 Skyrock - a community of firemen. 54
4.15 Skyrock - a community of *carp* fishers. 55
4.16 Skyrock - a community of futur mothers. 56
4.17 Skyrock - local community of the word *animaux* 57
4.18 Skyrock - local community of the word *aimer* 58
4.19 Skyrock - local community of the word *automobile* 58

5.1 Experimental setup used for churn prediction. 67
5.2 The main page of the Skyrock's social site. 67
5.3 Global Skyrock friendship degree distribution 68
5.4 A central node and its first neighbourhood (inner circle), its second neighbourhood (outer circle) and its local community (nodes in bold). 70
5.5 Variable contributions to the K2C/K2R model 71

6.1 Interaction prediction problem definition. 93
6.2 Interaction prediction problem definition. 94
6.3 A simple example showing a node with its neighbors (first circle), second circle and local community (nodes and links filled in red). 94
6.4 Some Statistics on DBLP dataset. 97
6.5 Some statistics on Facebook Walls dataset. 98
6.6 Evaluation of local community prediction for the DBLP dataset 101
6.7 Evaluation of local community prediction for the Facebook wall dataset 102

7.1 Hadoop Map Reduce execution model. 108
7.2 Layered model for social network analysis. 112
7.3 HBase architecture. 114
7.4 A toy network (a) and the representation of node 5(b). 115
7.5 Main classes of the Data Access Layer. 116
7.6 API component of the DAL. 117
7.7 Speedup results: the computation time of local communities identification, for all the nodes of the network, linearly decreases with the number of processors. 120
7.8 Common Neigbours computation using a map/reduce approach. 120
7.9 Common neighbours speedup on DBLP. 121

LIST OF TABLES

2.1 Binary classifier confusion matrix. 12

3.1 Summary of some community detection algorithms. 32

4.1 Quality evaluation 48
4.2 community of the book "Merlin Trilogy". 53
4.3 Local community of the book "Specter of the Past (Star Wars)". 53

5.1 Some attributes for churn prediction 69
5.2 Evaluation with Support Vector Classifiers 71
5.3 Ranking of attributes (only the top ten) using the absolute value of predicted coefficients of the linear regression. 74
5.4 Ranking of attributes (only the top ten) using the p-value of the Wald Chi-Square Test. 75
5.5 Evaluation of social recommendation 79

6.1 Evaluation (AUC) of interaction prediction models 100

7.1 Comparison of graph analysis tools 111

CHAPTER 1

Introduction

"You can do anything, but not everything."

- David Allen

Back to the early 2000*s*, Barabasi in his book *Linked: The New Science of Networks* [12] has changed the way of thinking about real-world networks and has also shown that many real complex systems can be modelled as networks. Since then, interest in social networks analysis and mining has considerably grown. Moreover, the popularization of online social networks like Facebook, Twitter and LinkedIn has reinforced the importance accorded to social networks by the scientific community.

A network is a set of entities, like people or organisation, connected by links representing relations or interactions (friendship, messages,...). Example of such networks are online social networks (Facebook, Twitter, Skyrock, etc.), Internet (with routers as nodes and their physical connections as links) and the Web (with pages as nodes and hyper-links as links). An introduction to the field of complex networks analysis can be found in [109].

One of the hottest topics in the social network analysis literature is the community detection problem. Following the seminal work by Newman, introducing the modularity function [106], hundred of papers have been published on that topic. A good recent review of these methods has been proposed by Fortunato [47]. The community detection problem consists in the automatic detection of groups of related nodes, according to a defined quality function.

Many methods for community detection exist, however they usually suppose that the network is entirely available in one computer at the analysis time. This assumption is usually false for very large, dynamic or inherently distributed networks. Moreover, for some applications, one can be only interested in the communities to which a particular node belongs to. Despite

the fact that it is possible to first detect all the communities and then select the ones containing the target node, the time complexity of these global community detection methods makes them infeasible for real time applications on very large networks.

In this thesis we are interested on the communities a given node belongs to: their detection, their usage and their evolution assuming that the network is large, dynamic and distributed.

Other possible analysis tasks on social networks that we will explore in this thesis (either as application, tools or for comparison purposes) are:

- node categorization: given a social network, a node can have many different states (active or not active; infected, not infected or recovered). The node categorization problem consists in predicting the state of an unlabelled node;
- link prediction: the problem here is to predict whether or not a link that does not exist at time t will be created at time $t' > t$;
- information diffusion: given a piece of information held by some users, the problem is to study how it will spread across the network;

The remainder of this chapter is organised as follows: section 1 presents the context of this thesis, section 2 presents its contributions and finally section 3 draws the organisation of the rest of this dissertation.

Contents

1.1 Context **2**
1.2 Contribution **3**
1.3 Organisation **5**

1.1 Context

Algorithms for community detection usually make the four following assumptions:

1. the network is entirely known: the community detection algorithm has a global knowledge of the data composing the network;
2. each node has (at least) one community: each node will be assigned to at least one community even if the node does not really belong to any one;

3. the network does not evolve with time: the dynamics of the network is simply ignored;

4. all the data of the network are available at a unique location: it is not possible to deal with distributed data.

These assumptions are not always true in real social networks. Moreover, for such networks, these algorithms can produce communities with thousands or millions of nodes and this may not be useful in practice. Our aim is to propose solutions which do not require that the above mentioned assumptions hold.

1.2 Contribution

The objective of this thesis is to present local community detection methods as an alternative to global community detection which usually require the entire knowledge of the network. In practice, having this global knowledge of the network is not always possible. The detected communities are then used in predicting some users' behaviours in a real online social network. Because real networks are dynamics, we also investigate the dynamic of these detected communities. Finally, because the data of very large social networks are usually distributed, we propose a framework that enables the computation of local communities in particular and more generally social network analysis in the context of distributed data. The contributions of this thesis are summarized as follow:

- we propose new algorithms for local community detection which can be used even if the entire structure of the network is not known. These algorithms can also detect that a node does not belong to any community;

- we propose a procedure to detect all the overlapping communities a given node belongs to, using the above mentioned new local community detection methods;

- to show that the detected local communities are meaningful, we apply them in the prediction of users' behaviours using machine learning;

- we propose some methods, using machine learning, to predict the evolution of local communities;

- we finally propose a model to store and process social networks with distributed data.

This work has been valued by the following publications:

- Peer-reviewed international journals
  - Blaise Ngonmang, Emmanuel Viennet, and Maurice Tchuente. Predicting users behaviours in distributed social networks using community analysis. Lect. Notes Social Networks. Springer, 2014.
  - Blaise Ngonmang, Maurice Tchuente, and Emmanuel Viennet. Local communities identification in social networks. Parallel Processing Letters, 22(1), March 2012.
- Book Chapters
  - Blaise Ngonmang, Vanessa Kamga, Emmanuel Viennet and Maurice Tchuente. Community Analysis and Link Prediction in Dynamic Social networks. Computing in Research and Development in Africa - Benefits, Trends, Challenges. Springer, 2014.
- Peer-reviewed international conferences
  - Blaise Ngonmang, Emmanuel Viennet, S. Sean, Françoise Fogelman-Soulié, and Rémi Kirche. Monetization and services on a real online social network using social network analysis. In ICDM workshop on Data Mining Case Studies and Success Stories, December 2013.
  - Blaise Ngonmang, Emmanuel Viennet, and Maurice Tchuente. Churn prediction in a real online social network using local community analysis. In IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM'12), pages 282-290, August 2012.
  - Blaise Ngonmang and Emmanuel Viennet. Toward community dynamic through interactions prediction in complex networks. In IEEE SITIS 2013, Complex Networks and their Applications, Kyoto, Japan, December 2013.
- Peer-reviewed national conferences
  - Blaise Ngonmang and Emmanuel Viennet. Dynamique des communautés par prévision d'interactions dans les réseaux sociaux. In Atelier Fouille de Données Complexes de la 14e Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC'14), Rennes, January 2014.

- Blaise Ngonmang and Emmanuel Viennet. Dynamique des communautés par prédiction d'interactions dans les réseaux sociaux (poster). In 14e Conférence Internationale Francophone sur l' et la Gestion des Connaissances (EGC'14), Rennes, January 2014.
- Blaise Ngonmang and Emmanuel Viennet. Framework d'analyse de grands réseaux sociaux distribués. In CRI 2013, Yaoundé, Cameroun, December 2013.

1.3 Organisation

The content of this thesis is organized as follow:

- Chapter 2- "Basic definitions and notations": introduces helpful notions to easily follow the remaining chapters.
- Chapter 3- "Global community detection in static network": introduces the problem of detecting the community structure of static networks.
- Chapter 4- "Local community identification in social networks": presents local methods for the identification of the community a particular node belongs to.
- Chapter 5- "Local community applied to the prediction of users' behaviours": presents the application of local community identification to the prediction of a users' behaviours.
- Chapter 6- "Communities in dynamic networks": presents an overview of methods for detecting community in dynamics networks and our contribution to predict local communities.
- Chapter 7: "Towards a Framework for storing and analysing distributed networks": introduces a general Framework to analyse social networks with distributed data.
- Chapter 8: "Conclusions and perspectives": draws some conclusions and perspectives.

CHAPTER 2

Basic definitions and notations

"The Web is more a social creation than a technical one."

- Tim Berners-Lee

Throughout this thesis, some social networks and machine learning concepts will be used. This chapter recalls the main concepts that will help to better understand the rest of this thesis.

Contents

2.1 Basic notions on social networks 7

2.2 Basic notions on supervised learning for classification 11

2.2.1 Problem definition 11

2.2.2 Model evaluation 11

2.2.3 Introduction to support vector machines 12

2.3 Conclusion 14

2.1 Basic notions on social networks

A social network can be represented by a graph $G = (V, E)$ [109], where V is the set of vertices (or nodes), and E is the set of edges (or links), formed by pairs of vertices. The two nodes u and v are the end vertices of the edge $e = (u, v)$. If the order of end vertices matters in an edge, then the graph is said to be directed; otherwise, it is undirected. Figure 2.1 shows examples of undirected (a and b) and directed (c and d) graphs.

The neighbourhood $\Gamma(u)$ of a node u is the set of nodes v such that $(u, v) \in E$. In figure 2.1 (a), the neighbour of the node 0 is $\{1, 2\}$. The

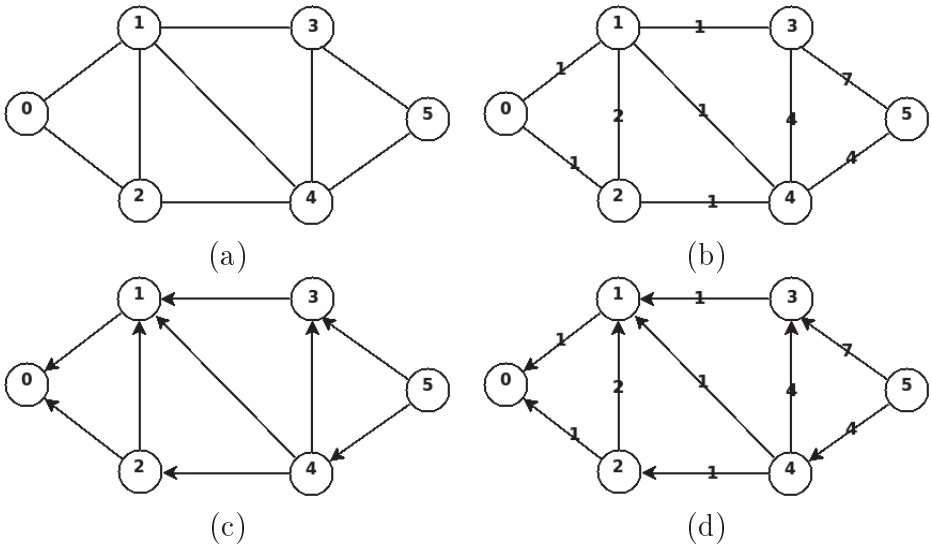


Figure 2.1: Some graph examples: an undirected and unweighed graph (a), an undirected and unweighed graph (b), a directed and unweighed graph (c) and a directed and unweighed graph (d).

degree of a node u is the number of its neighbours or the cardinality of $\Gamma(u)$, i.e. $degree(u) = |\Gamma(u)|$. The degree of node u will also be denoted by d_u. In figure 2.1 (a), $d_0 = 2$. Given this model, all graph theoretic tools can be reused in network analysis. We recall in the rest of this section, the main concepts of graph theory that will be used in this thesis.

Hereafter, the number of nodes of the network will be denoted by n and the number of edges will be denoted by m. The adjacency matrix of G is an $n \times n$ boolean matrix A defined by $a_{ij} = 1$ if there is a link from i to j, and $a_{ij} = 0$ otherwise. In some applications, it is useful to model the strength of the link between i and a neighbour j. In this case a_{ij} is a real number. Such graphs are said to be weighted. Example of weighted graphs are presented in figure 2.1 (b and d).

A graph algorithm is global if its time complexity scales at least with the number of node ($O(n)$). A global graph algorithm thus requires access to a vast portion (usually all the nodes) of the graph. Conversely, a local graph algorithm only needs to access a little sub-set of the graph.

The spectrum of a graph G is the set of eigenvalues of its adjacency matrix A. The Laplacian matrix L of a graph G is defined by $L = D - A$, where D is the diagonal matrix of order n defined by $d_{ii} = degree(i)$.

A finite path of length k in G, is a sequence of edges $\langle e_1 = (u_1, u_2), e_2 = (u_2, u_3), \ldots, e_k = (u_k, u_{k+1})\rangle$, such that two consecutive edges $e_i = (u_i, u_{i+1})$

and $e_{i+1} = (u_{i+1}, u_{i+2})$ share a common end vertex u_{i+1}. Such a path connects u_1 to u_{k+1}. A path of length k is said to be closed if $u_1 = u_{k+1}$. In figure 2.1(a) an example of path is given by the sequence $\langle(0,1),(1,3),(3,5)\rangle$ and a closed path is given by $\langle(1,2),(2,4),(4,3),(3,1)\rangle$.

A connected component of an undirected graph G is a maximal sub-graph in which any two vertices are connected to each other by paths. Maximal means that such a component is not connected to any additional vertex in G. A clique is a set of nodes that forms a complete graph i.e. with all possible links. The term $k-clique$ is used to denote a clique of k nodes. A triangle is a $3-clique$ or closed path of length 3.

The clustering coefficient of a network is related to the number of triangles (close path of length 2). It corresponds to the probability that two nodes u and v, connected to a common neighbour w, are also connected. There are two different ways for computing the clustering coefficient of a network. The global clustering coefficient is the number of triangle divided by the number of paths of length 2. The clustering coefficient of a node is the ration between the number of pairs of its neighbors that are connected and the total number of pairs of its neighbors. The average local clustering coefficient is an average of the clustering coefficients for all nodes of the network. In figure 2.1(a) the global clustering coefficient is 0.66 and the average local clustering coefficient is 0.72.

A link $e = (i, j)$ is internal to a sub-graph $G' = (V', E')$ if i and j are in V'. A link $e = (i, j)$ is external to a sub-graph G' if either i or j is in V', but not both. The density δ of a graph corresponds to the proportion of its existing links compared to the total possible links. The internal density δ_{in} corresponds to the proportion of internal links of a sub-graph compared to the possible internal links. Similarly, the external density δ_{out} corresponds to the proportion of external links of a sub-graph compared to the possible external links.

The observation of a dynamic network during T time steps is modelled by $G = (G_1, G_2, ..., G_T)$, where $G_t = (V_t, E_t)$ is the network observed at time t.

It has been observed that many real world complex networks share some characteristics [109, 13]:

- **scale-free property:** the degree distribution follows a power law i.e. the probability that a node has degree k is given by [12, 109]:

$$P(k) = k^{-\gamma} \tag{2.1}$$

  for a given constant γ usually between 2 and 3. This power low has been justified by the preferential attachment [14] which states that new

links tend to connect to high degree nodes.

This property is relevant for the analysis in this thesis because it leads to very sparse networks with only some very high degree nodes. It is then safe to consider that the average degree is small compared to the total number of nodes (with removing very high degree nodes if needed) in complexity estimation.

- **Small world property:** the shortest path between any given pair of nodes is usually small. This property is relevant for us because we can consider that "close friends" in a social networks must be connected with short paths.

- **High clustering coefficient:** this implies that the probability to have a link between two nodes having the same third node as neigbour is high, compared to a random network. This property will help us to predict missing interactions in social networks.

- **Presence of a community structure**. A community is a set of nodes having a high internal density and a low external density. Fig. 2.2 presents an example of community structure in a network. We however notice that community structure is not always present or easy to detect. This topic is the subject of active research (see for instance [73, 21]).

  Detecting communities from a local point of view and predicting their dynamic will be at the core of our contributions.

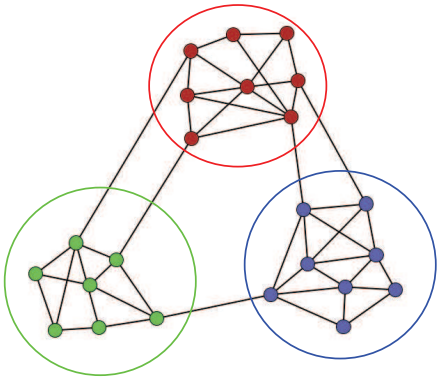


Figure 2.2: Example of community structure.

2.2 Basic notions on supervised learning for classification

In this thesis some supervised learning models are used for prediction. This section presents the supervised learning problem, some evaluation techniques and Support Vector Machines(SVM), the tool used in our experiments.

2.2.1 Problem definition

Supervised learning is the machine learning task of inferring a function from labelled training data [97]. The training data consists of a set of training examples. Each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

The classification task is an example of supervised learning where the target value to predict is a categorical finite variable. One particular case is a binary target variable, for example whether or not a user will leave a social network.

More formally, a classification task can be defined as follow: given M training examples of the form $\{(x_1, y_1), ...(x_M, y_M)\}$, where x_i is the feature vector (set of variables describing x) of example i and y_i is the label (also called class) of example i. The objective of the classification task is to infer a function h, usually called the hypothesis or classifier, that will be able for a previously unseen feature vector x_i to compute the probability $P_c(x_i)$ that it belongs to the class c.

2.2.2 Model evaluation

To measure the generalisation power of the classifier on previously unseen examples, the available data must be divided into two disjoint sets: a *training set* used to train the classifier and a *test set* used to evaluate it. For each example of the test set, the trained model is used to predict its class. A comparison is then done between the predicted and the real value of the example. Different measures can then be used to give an evaluation score. In this thesis we will used Precision/Recall/F-score and the Area Under the Curve(AUC).

True label->	True	False
True	a	b
False	c	d

Table 2.1: Binary classifier confusion matrix.

Precision/Recall/F-Score

After using a classifier on a test set for a binary classification problem, one gets a confusion matrix as presented in table 2.1. Based on that confusion matrix, the *precision* is defined by:

$$Precision = \frac{a}{a+b} \tag{2.2}$$

and can be interpreted as the proportion of returned positives that are really positives. The *recall* is defined by:

$$Recall = \frac{a}{a+c} \tag{2.3}$$

and can be interpreted as the proportion of returned positives compared to the total number of positives examples. Finally the *F-Score* that combines the two is defined by:

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

Area under the curve

A receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system while its discrimination threshold is varying. It is created by plotting the fraction of true positives out of the total actual positives (TPR = true positive rate) vs. the fraction of false positives out of the total actual negatives (FPR = false positive rate), at various threshold settings. TPR is also known as sensitivity or recall in machine learning. Examples of ROC curve are presented in figure 2.3

The area under the ROC curve (AUC) gives a good measure to compare classifiers: the higher the value, the better the classifier is.

2.2.3 Introduction to support vector machines

Support vector machines (SVM) are one of the most powerful tools for classification. The idea behind SVM is to find the linear separator that gives the

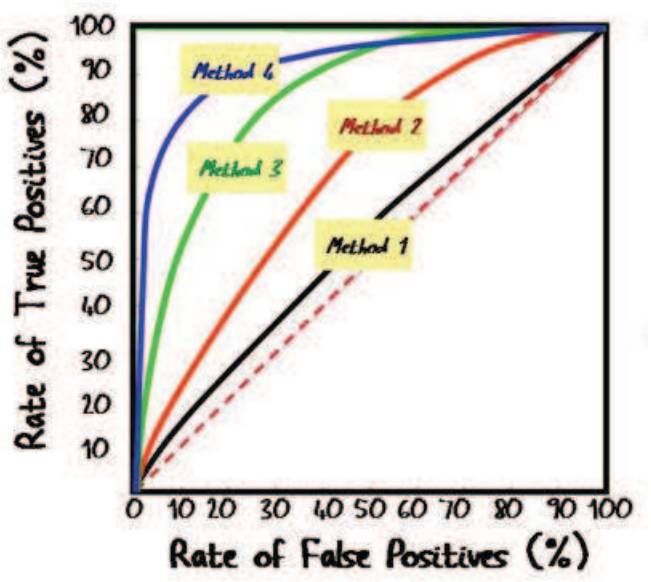


Figure 2.3: ROC curves examples.

maximal margin between the two classes in a binary classification problem. This *large margin* intuition is illustrated in figure 2.4. In that figure, W_{opt} is the optimal weight vector. The SVM learning problem can be formally defined as the following optimisation problem:

$$min\frac{1}{2}||W||^2 + C\sum_{j=1}^{M}\xi_j \tag{2.5}$$

$\xi_j > 0$ correspond to the eventual violation of the example j to the constraint of being behind the margin. $C > 0$ is a user defined parameter that controls the trade-off between large margin and constraint violation.

When the problem is not linearly separable, a kernel trick can be used to project the data into a higher dimensional space where the data can be linearly separated. The most used kernels are:

- the polynomial kernel: $K(x_i, x_j) = (x_i^T x_j + 1)^d$. d is the polynomial degree and leads to one more user defined parameter.
- the gaussian kernel: $K(x_i, x_j) = exp(-\frac{||x_i - y_j||^2}{2\sigma^2})$. σ is the standard deviation and must also be provided by the user.

All the parameters can be automatically set using a validation set or *k-fold* cross validation and an heuristic search technique like grid-search[67].

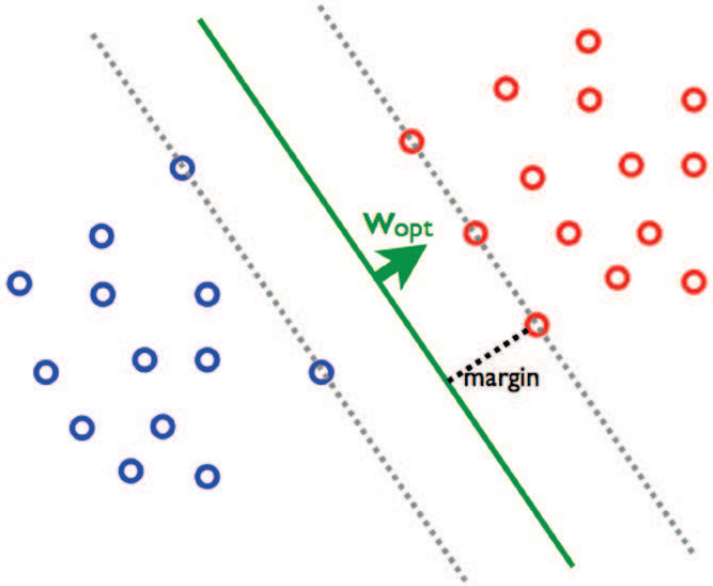


Figure 2.4: SVM's Margin illustration.

2.3 Conclusion

With these basic concepts in mind, it will be more easy to follow the remainder of this thesis. The next chapter will start addressing community detection in static networks.

CHAPTER 3

Global community detection in static networks

"In theory, there is no difference between theory and practice. But in practice, there is."

- Yogi Berra.

3.1 Introduction

A graph is a set of items called vertices (or nodes), with connections between them, called edges (or links) [106]. Many complex systems can be represented using graphs. Examples are online social networks (Facebook, Skyrock, Twitter, etc.), Internet and web pages. An interesting problem studied in these networks is the detection of communities. A community is a set of nodes highly connected together than with the rest of the network. Community detection is an important task in networks because, in the one hand, it can help better understand the structure and the functions of the network [109] and can be used as component for more complex tasks like social recommendation [34] or network visualisation [83] in the other hand.

In this thesis, we are interested in detecting the communities a particular node belongs to. If one can efficiently detect all the communities of the network, it is then possible to select the ones where we have the node of interest. In this chapter, we present an overview of these global community detection methods. This presentation will help us better understand the limitations that make them unsuitable to detect the communities of a given node in very large, dynamic and distributed networks.

An important notion in community detection is the quality function. The role of quality functions is to evaluate how well a particular community de-

tection performs on a given network. Moreover, some community detection algorithms are based on the optimisation of a particular quality function.

This chapter is organised as follows: section 2 presents the different meaning of the word community in the literature, section 3 presents the most popular quality functions, section 4 presents some problems similar to community detection, section 5 presents an overview of some community detection methods, section 6 presents how to evaluate community detection methods, section 6 shows how to check whether or not a community structure is present in a network and finally, section 7 concludes this chapter.

Contents

3.1 Introduction 15
3.2 Different definitions of communities 17
3.3 Quality functions 17
3.3.1 Conductance 17
3.3.2 Performance 18
3.3.3 Modularity 19
3.4 Some problems similar to community detection . 19
3.4.1 Graph Partitioning 20
3.4.2 Data clustering 20
3.5 Global community detection 21
3.5.1 Divisive methods 21
3.5.2 Agglomerative methods 23
3.5.3 Spectral methods 25
3.5.4 Random walks 26
3.5.5 Information diffusion methods 27
3.5.6 Ensemble methods 27
3.5.7 Community detection in attributed graphs 28
3.5.8 Methods to detect overlapping communities 28
3.5.9 Summary on global community detection 31
3.6 Evaluation of community detection methods . . . 32
3.6.1 Evaluation with ground truth 32
3.6.2 Evaluation without ground truth 34
3.6.3 Presence of a community structure in a network . . 34
3.7 Conclusion 34

3.2 Different definitions of communities

The term community has different meanings in the scientific literature. It may refer to interest community, similarity community or connectivity community. A community of interest [45] is a community of people who share a common interest or passion. In the context of online social networks (OSN), these people exchange ideas and thoughts about the given passion, but may know (or care) little about each other outside of this area. An example of such a community is the set of fans of "The Beattles".

A similarity community is in fact a cluster: a set of similar objects [153]. The social links are ignored and only the node attributes are considered. Here also, members of these communities may know or care little about others. An example of such community is the community that contains mainly PhD students, located in France, working on data mining and having started their thesis in 2011.

A connectivity community is a set of vertices having a high number of links between them and few with the rest of the network [106]. In the rest of this thesis, if not stated explicitly, this is the definition of community that is assumed. Examples of such communities will be presented in the following chapter.

3.3 Quality functions

Most of the community detection methods use a function to estimate the quality of the detected community structure. This function is called the *quality function*. The estimation of the value for this quality function is either done directly during the optimisation, because it is used as the objective function to optimise, or indirectly because it is used a posteriori to evaluate the mined community structure. Quality functions are very useful when one wants to compare many algorithms that do not optimise the same objective function. Among the most commonly used quality functions, there are: the conductance, the performance and the modularity.

3.3.1 Conductance

The conductance is one of the simplest definitions for the quality of a community. It is defined as the ratio between the number of links having one end in the community and the minimum between the number of links inside the community and the number of links outside it. Given a community S, its conductance is computed using the following expression:

$$\Phi(S) = \frac{|(u,v) : u \in S, v \in \overline{S}|}{min(\sum_{u \in S} d_u, \sum_{v \in \overline{S}} d_v)} \tag{3.1}$$

The conductance can be explained as the probability that a random walk starting inside the community will leave it. As a probability, its values range from 0 to 1. A value close to 0 corresponds to a good partition. Figure 3.1 presents an example of community structure. In that figure, $\Phi(A) = \frac{2}{14} > \Phi(B) = \frac{1}{11}$. A is a better community than B as far as the conductance is concerns. Using the conductance, a community structure is as good as all its communities have a good conductance (close to 0).

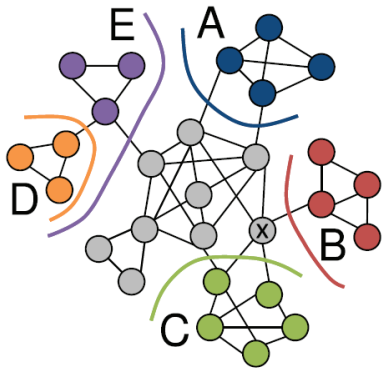


Figure 3.1: Example of community structure for the computation of the conductance.

3.3.2 Performance

The performance [47] gives the proportion of pairs of nodes correctly interpreted by the algorithm i.e. nodes belonging to the same community and connected with links or nodes belonging to different communities and not connected. Given a partitioning p, its performance score is given by:

$$P(p) = \frac{|(i,j) \in E, C_i = C_j| + |(i,j) \notin E, C_i \neq C_j|}{n(n-1)/2} \tag{3.2}$$

where C_i is the community of node i.

By definition, $0 \leq P(p) \leq 1$. A value of performance close to 1 means a good partitioning.

3.3.3 Modularity

The modularity defined by Girvan and Newman [110] is the most used quality function. The intuition behind this quality function is that a random network is not supposed to have a community structure. For each community, the density of its links is compared to the expected density in a random network with the same number of nodes and the same degree distribution but without community structure. The value of the modularity is given using the following formula:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2m}) \delta(C_i, C_j) \tag{3.3}$$

where m is the number of links of the network, A the adjacency matrix, d_i and C_i the degree and the community of the node i respectively. The function δ is equal to 1 if and only if $C_i = C_j$ and 0 otherwise. The value of Q range from -1 to 1 and it is generally admitted that a good community structure has a modularity score greater than 0.4 [108].

The modularity optimisation gives a good way to detect community in very large networks [107, 108, 18]. However, this quality function has two most important drawbacks:

1. the *resolution limit*: it has been shown in [48] that the size of each community depends on the number of nodes in the network. It is then difficult to detect small communities, even well separated.

2. the *instability*: the assumption behind the modularity is that a random network is not supposed to have community structure. The actual community structure is then compared with a *null model* expressed in term of expectation as presented with the term $\frac{k_i k_j}{2m}$ in equation 3.3. This leads to many possible realisations of the null model and it is possible to have many different community structures with a high modularity score, even in random networks [74].

3.4 Some problems similar to community detection

In this section, we present some problems which consist in grouping related objects of a given set into classes. We have chosen to present these problems because tools used to solve them can be adapted for community detection.

3.4.1 Graph Partitioning

The objective of graph partitioning is to group the nodes of a graph in a predetermined number of partitions, with predetermined sizes. This is usually done by minimizing the number of links falling between the partitions. The most successful historical methods are presented bellow:

- Spectral bisection [124]: this method consists in computing the eigenvector corresponding to the smallest eigenvalue of the Laplacian matrix of the graph. The graph is then divided into two parts corresponding to the sign of the projection of each node according to this eigenvector. The time complexity of this method is $O(n^3)$ and the results are good when the graph really contains two partitions of similar sizes. That is only a particular case in the context of community detection.
- Minimal cut [10]: this algorithm tries to minimize the cut of the graph by minimizing the number of links between two groups. This method requires the sizes of the partitions as a parameter and starts with a random initial partition. The optimisation is performed using a greedy scheme. At each step, the quality of the bisection is improved by switching nodes between partitions. The switch which gives the best improvement is performed, with the condition that a node changes its partition at most once. There is then exactly n^2 computations and the best bisection is returned. The worst case complexity is $O(n^3)$.

These methods can be used for community detection if the number of communities and their sizes are known a priori. However, this is not usually the case and others techniques must be developed.

3.4.2 Data clustering

This problem has been introduced for data analysis. It consists in partitioning data objects based on a similarity measure (a pseudo distance, because the triangular inequality is not required) between data points in some defined space. Similar objects are grouped together and dissimilar one are separated in different groups.

Lets A = (a_1, a_2, ..., a_k) and B = (b_1, b_2, ..., b_k) be two data points in a k dimensional space. The distance functions mostly used as similarity measures are:

- the euclidean distance (or L_2 norm) : $d_{AB} = \sum_{i=1}^{n} \sqrt{(a_i - b_i)^2}$
- the Manhattan distance (or L_1 norm) : $d_{AB} = \sum_{i=1}^{n} |(a_i - b_i)|$

- the cosine similarity : $d_{AB} = \frac{A.B}{||A||.||B||}$

One of the mostly used method for data clustering is the $k - means$ algorithm [153]. It works as follows:

1. for each of the k classes, one point is chosen at random as the initial mean;
2. for each data point, its distance to each mean is computed and it is moved to the cluster of the closest mean (breaking ties randomly);
3. for each of the k updated clusters, the new mean is computed;
4. the process restart from step 2 until the clusters do not change or a maximum (predefined) number of iterations is reached.

As for the graph partitioning, one needs to know the number of clusters. Choosing the number of cluster for $k - means$ is widely studied and some heuristics such as the *elbows' methods* [61, 132] exist to solve it.

Some others methods that do not require the number of clusters as parameters also exist. They usually perform an hierarchical clustering by either starting with all the data points in one cluster and separate them at each step (top down approaches) or starting with each point in its own cluster and grouping clusters at each step (bottom up approaches).

The data clustering methods can be used in community detection by choosing an appropriate distance function between the nodes of the network. An example of such an application will be presented when presenting spectral methods (see sub-section 3.5.3).

3.5 Global community detection

Given a function measuring the quality of a community structure, optimizing it to get the best partitioning is an *NP-hard* problem [150, 20, 47]. Some heuristics are then used to approximate the best partitioning. The most important classes of methods are presented below with some examples of methods ranging from each class.

3.5.1 Divisive methods

These methods proceed by starting with all the nodes of the network in one community and at each step, they use a criterion to divide the existing communities. This optimisation procedure gives a dendrogram as the one shown

in figure 3.2. Examples of methods ranging in this category are the *link betweenness centrality-based* and the *link clustering coefficient-based* methods.

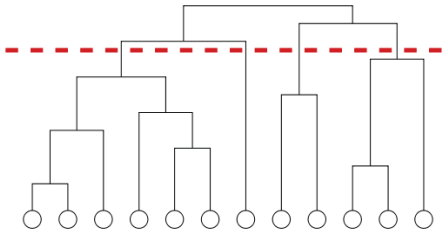


Figure 3.2: A dendogram structure.

Link betweenness centrality based method

This method has been proposed by Girvan and Newman in 2002 [110]. The betweenness centrality of a link is the number of shortest paths passing through this link. The idea is that a link with a high betweenness is more likely to be an inter-community link. At each step, the algorithm removes the link with the highest betweenness and the process is repeated until there is no more increase for the modularity score (used as quality function in this algorithm).

The time complexity of this method is $O(m^2n)$. In fact, the computation of all the betweenness centralities is done in $O(mn)$ and one can need m steps in the worst case. The detected communities using this method are quite good (with a high modularity score) but, due to the time complexity it can not be used on large networks.

Link Clustering coefficient based method

This method was proposed by Radicchi et al. in 2004 [129]. The k-order clustering coefficient of a link is the ratio between the number of cycles of length k passing through this link and the total number of cycles of length k in the graph. The idea is that links with high clustering coefficients are more likely to be inter-community links. The optimisation procedure is then the same as the one using the link betweenness.

When removing a link, one just need to update the clustering coefficient values for the links adjacent to it. That leads to a more efficient algorithm, than the one using the betweenness centrality, with a time complexity of $O(m^2)$. However, it is still not suitable to deal with very large graph (millions

of nodes). We will then move to agglomerative methods where we have one of the most efficient methods for global community detection in networks.

3.5.2 Agglomerative methods

The main idea behind this class of methods is that, at the start, each node is alone in is community and, at each step the two more similar neighbour communities (having adjacent edges) are merged to form one new community. Like divisive methods, this optimisation procedure also leads to a dendrogram. Examples of methods from this class are *modularity optimisation* and *the Louvain's* methods. We will detail them in the following sub-sections.

Modularity optimisation

This method has been proposed by Newman in 2004 [107]. It consists in a greedy optimisation of the modularity: starting with each vertex being the sole member of a community of one, at each step, the two communities whose amalgamation produces the largest increase in modularity are joined. The number of partitioning found by this method is $n-1$, each partitioning having a different number of communities between n and 1. The partitioning with the highest value of modularity is retained as the approximation of the best partitioning.

The time complexity of this algorithm is in $O(n^2)$ in the case of sparse networks (as it is usually the case for social networks). Indeed, the evaluation of the change in modularity when merging two communities is done in constant time. This evaluation for all pairs of adjacent communities is then done in $O(m)$. Afterwards, at each step, one needs to to update the matrix containing the inter-community links, needed to compute the modularity. This update is done in $O(n)$ in the worst case. Finally, the algorithm has $n-1$ steps. The complexity is then in time $O((m+n)n)$ which is approximated by $O(n^2)$ for sparse network. As reported by the author in [107], this method is able to analyse networks up to 10^5 nodes.

The time complexity of this method has been considerably improved and reduced to $O(n \log^2 n)$ by Clauset et al. [30]. In fact, these authors have noticed that, because the adjacency matrix of the network is sparse, one can use adapted data structures for this type of matrices. They then have proposed an implementation using a heap. This new implementation enables to analyse network up to 10^6 nodes. This implementation is freely available here: `http://cs.unm.edu/~aaron/research/fastmodularity.htm`.

With this method, it is then possible to analyse quite large networks but still not very large networks with millions of nodes. The next presented

method enables us to take that step.

Louvain's method

The Louvain's method [18] is one of the fastest methods for community detection in complex networks. It can be used in the general case of weighted networks. As an agglomerative algorithm, it starts with each node in a separated community. The algorithm has two main steps: in the first step, all the nodes of the network are evaluated one by one and for each of them, the weighted modularity gain is computed if the current node i is added to the community of its neighbour j. The node i is added to the community which produces the maximal positive gain. These sweeping are repeated until there is no more positive gain. At the end of this step, one has a partitioning of the network.

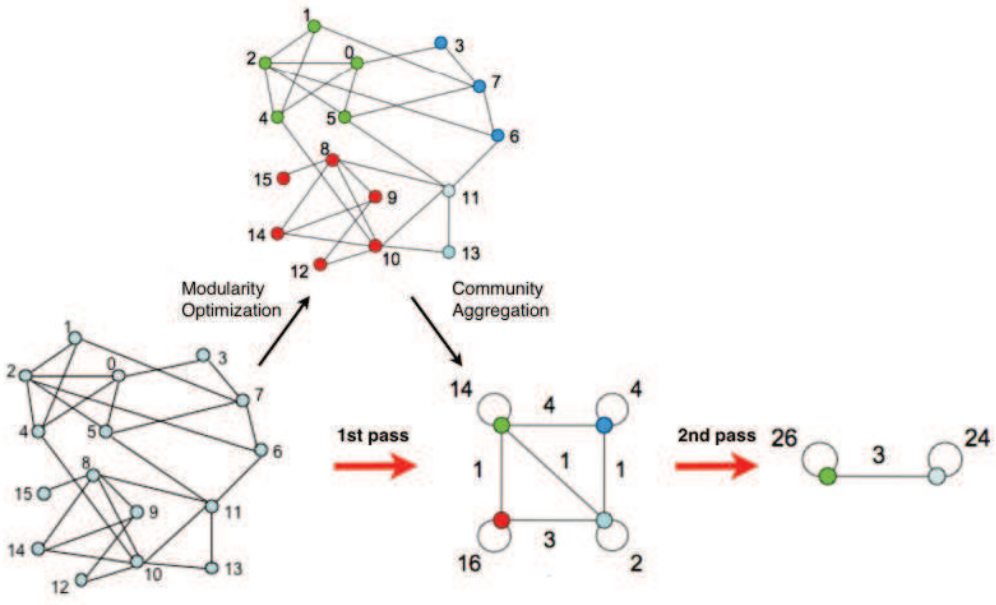


Figure 3.3: Hierarchic Optimisation of modularity with the Louvain's method [18].

In the second step, each of the previous discovered communities becomes a *super-node*. Two super nodes are linked if and only if there is at least one link between two members belonging to each community. The link between two communities is weighted according to the sum of all the weights of their inter-communities links.

These two steps are repeated until no more gain in modularity is observed. It is worth noting that the gains on modularity are always computed from the starting network, in order to be able to compare the partitioning during each pass of the algorithm. The figure 3.3 presents an execution of this method on a toy network. On this figure, one can see two passes starting from the network at the left.

The speed of this method comes from the observation that the modularity can be optimised locally: only the neighbours are considered during the evaluation, that enables to update the modularity gain in linear time ($O(m)$). This method is actually only limited by the storage (main memory) capacity and allows to analyse networks of millions of nodes in some minutes.

However, the result of this algorithm extremely depends on the order in which the node are processed. In fact, two different orders can lead to quite different partitions. This is because many partitioning can lead to a high and close value for the modularity [21, 141]. The authors have proposed to run the algorithm many time and get the partitioning that gives the highest value [18].

3.5.3 Spectral methods

Spectral methods have been initially used for the particular case of graph partitioning. Many methods have been afterwards proposed in the more general case of community detection. The main idea of this class of methods is to segment the network using a projection of the nodes in a space constructed with the eigenvectors of one of its characteristic matrices. The mostly used matrices are:

- the adjacency matrix A;
- the Laplacian matrix L;
- the normalized Laplacian matrix.

The nodes of the graph are represented in the space of the k firsts eigenvectors of the chosen matrix. A similarity function is then defined in this space and one can applied a classical clustering algorithm.

The Laplacian matrix is the mostly used because it has the following properties:

- it is positive definite: all its eigenvalues are positives;
- the sum of each of its lines or columns gives 0.

These properties ensure that it always has an eigenvalue equals to 0 which can be associated to the eigenvector with all elements equals to 1.

The intuition behind spectral methods is the fact that when a graph has k connected components, it has exactly k eigenvalues equal to 0 and, the projection of the nodes of the graph in the space associated with the eigenvectors of these k eigenvalues gives the same coordinates to all points belonging to the same connected component [47, 91]. For a graph with a community structure, the points corresponding to nodes in the same community are then expected to be close in that space. Examples of spectral methods can be found in [154, 108, 105].

Spectral algorithms for community detection differ mostly on the choice of the topological matrix and the clustering algorithm used. The time complexity mostly depends on the computation of the eigenvalues. The exact computation is done in time $O(n^3)$. Because one only needs the k first eigenvalues, some quick approximations, such as the Lanczos method for example [80], can be used.

3.5.4 Random walks

A random walker on a graph $G = (V, E)$ follows a stochastic process that starts at a node $i \in V$ and, at each step, selects with probability P_j among its neighbours the next node j to visit at time $t+1$ [87]. Usually, this selection is done randomly and uniformly i.e. $P_j = \frac{a_{ij}}{d_i}$. The length of a random walk is its number of steps. The transition matrix of the random walk is $P = AD^{-1}$, and the probability of going from a vertex i to a vertex j in t steps is $(P^t)_{ij}$.

The idea behind random walk methods is that a random walker on G tends to get trapped into communities. As a consequence, two vertices i and j of the same community tend to see all the other vertices in the same way, i.e. if the length t of the random walk is long enough, the i^{th} row $(P^t)_{i.}$ and the j^{th} row $(P^t)_{j.}$ will be similar. This leads to a definition of distance between nodes and community detection becomes a clustering problem that can be solved using for example hierarchical methods. One algorithm of this class is *WALKTRAP* [122].

Random walks on graphs are strongly related to methods that study community structure using spectral properties of graphs. For instance, in [122] it is shown that the distance r_{ij} defined between two nodes i and j, using random walks, is related to the spectral property of the matrix P by the formula:

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} (v_\alpha(i) - v_\alpha(j))^2 \tag{3.4}$$

where $(\lambda_\alpha)_{1\leq\alpha\leq n}$ and $(v_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and eigenvectors.

3.5.5 Information diffusion methods

The idea behind this class of methods is that if one member of a community has a piece of information, this information can reach easily other members of the same community and this information will reach more difficultly non-members of the community. One of the most famous algorithms ranging from this category is the label propagation method [130] which can be described as follows:

1. at the start of the algorithm, each node has a unique label;
2. at each iteration, each node receives the label of its neighbours and adopts the must frequent one breaking ties randomly;
3. the process stops when there is no more label change or after a maximum number of iterations (used as parameter);
4. each set of nodes with the same label are assigned to the same community.

Notice that the step number 2 can either be synchronous or asynchronous. The asynchronous update cause the node labels to be updated as each node is considered. Conversely, with the synchronous updates, the node labels are fixed until all the nodes are considered, and then all of them are updated accordingly. Experimentations have shown that this can cause more oscillatory actions than the Asynchronous option (see for example this online simulation: `http://www.opcoast.com/demos/label_propagation/index.html`).

3.5.6 Ensemble methods

Statistical estimates can often be improved by fusion of data from different sources. It is the idea behind the so-called ensemble methods which have been successfully applied in some machine learning problems like classification or clustering [40]. The same idea has been applied for community detection in networks. The general framework for ensemble methods for community detection can be stated as follow:

1. compute k community structures using either a stochastic algorithm or k different algorithms;

2. build a $n \times n$ consensus matrix F where F_{ij} is the frequency of instances where node i and node j are in the same community;

3. construct a new graph with the frequency matrix F as adjacency matrix;

4. perform a partitioning of the new constructed graph.

Methods of this class mainly differ by the algorithms used for community detection in step 1 and the partitioning method used in step 4. Examples of ensemble methods for community detection are presented in [32, 141].

3.5.7 Community detection in attributed graphs

All the above mentioned methods do not take into account the attributes of the nodes in the network. Some works have been proposed to deal with attributed graphs. In [33], the authors have proposed a new quality function that combines the structure and the attributes of the graph. This new quality function is defined by:

$$Q_{hybrid} = \sum_{C} \sum_{i,j \in C} (\alpha \cdot S(i,j) + (1-\alpha) \cdot simA(i,j)) \tag{3.5}$$

where the link strength $S(i,j)$ between two nodes i and j is measured by comparing the true network interaction G_{ij} with the expected number of connections $(d_i \cdot d_j)/2m$. $simA(i,j)$ is a user defined attribute similarity and α is the weighting factor, $0 \leq \alpha \leq 1$ and enables to give more importance to the structural or attribute similarity.

3.5.8 Methods to detect overlapping communities

All the algorithms presented until now always assume that each node belongs to only one community. This assumption is rarely true in real social networks. In fact, in his/her Facebook contacts, a user can have friends that was classmates, some that are his/her colleagues and some that are members of his/her family, leading to three different communities he/she belongs to, with eventually some overlaps between them (a past classmate can now be a colleague). The last part of this section presents some methods to detect overlapping communities.

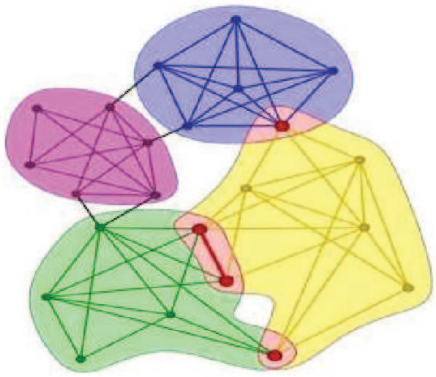


Figure 3.4: A *4-cliques.* percolation example.

Cliques Percolation Method

One of the most popular method for overlapping community detection in network is the Clique Percolation Method (CPM) proposed by Palla et al. in 2005 [119]. This technique is based on the fact that internal links in a community generally form cliques because of their high density. similarly, the inter-community links are not expected to form cliques due to their low density.

The authors of this method used the notion of *k-cliques* to define a full connected sub-graph of k nodes. Two *k-cliques* are adjacent if they share $k-1$ nodes. The union of many adjacent *k-cliques* forms a *k-cliques chain.* Finally, a *k-cliques* community is the biggest sub-graph composed of a *k-clique* and all the *k-cliques* that are adjacent to it. By construction, the k-cliques community can share nodes, it is why they can overlap.

To detect *k-cliques communities*, one firstly find all the maximal cliques of the network (all the complete sub-graphs). Afterwards, an overlapping matrix O of order n_c, the number of maximal cliques, is built. The element O_{ij} gives the number of common nodes between the cliques i and j. Then, a matrix O' is built by setting O'_{ij} to one if and only if O_{ij} is greater than or equal to $k-1$ and setting O'_{ij} to zero otherwise. The *k-cliques communities* are finally obtained by finding the connected components of the graph induced by the matrix O'.

The time complexity of this method depends on many factors, including the number of maximal cliques, and do not have been given in closest form. Experimentations on real networks shows that it is only applicable on small or very sparse graphs. The implementation done by the authors is freely available here: `www.cfinder.org`.

Link partitioning

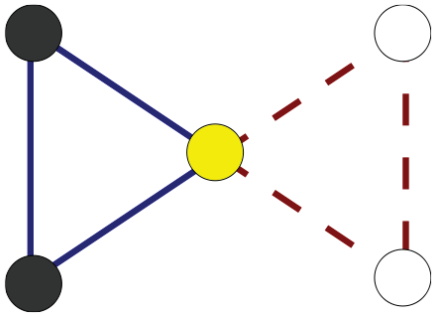


Figure 3.5: Link partitioning.

It is sometimes more easy to identify the intra and inter-communities links than to detect overlapping nodes. For example, in figure 3.5, it is easier to identify the links of each community than to detect the communities themselves. The idea of link partitioning is then to define communities in terms of group of links rather than groups of nodes.

Evans and Lambiotte [44] have, for example, proposed to used the *line graph* constructed using the starting graph as follows: the nodes are the links of the starting graph and a link exists between two nodes if the corresponding links are adjacent in the starting graph. One can then use a usual community detection method to partition the *line graph*. In the resulting graph, inter-communities links corresponds to overlapping nodes in the initial graph.

The idea of grouping the links is quite interesting, however it is not a priori better to group links rather than nodes. In fact, the two situations are equivalent: when partitioning links, a link crossing two communities will be assigned to only one of them.

The implementation provided by the authors is freely available here: `http://sites.google.com/site/linegraphs/`.

Graph Expansion Method

If one is able to detect the overlapping nodes, it is possible to create multiple copies of that nodes, one for each community they belong to, in order to remove the overlapping and then be able to use an existing non-overlapping method. That is the idea behind graph expansion proposed by Gregory [57]. This method, called COPRA(Community Overlap PRopagation Algorithm), has three steps:

1. the starting graph is expanded to remove the overlaps;
2. a non overlapping community detection method is applied to the expanded graph;
3. the nodes of each partition are converted to their equivalent in the non-expanded graph.

The transformation step is done by identifying the nodes with the highest betweenness centrality. These nodes are divided in many copies connected by links. This step is applied until the maximum value of the node betweenness is greater than a threshold s, used as the parameter of the method. Using this procedure, most of the nodes of the network remain the same, only nodes belonging to many communities are transformed into many copies. The overlaps in the result are obtained by looking nodes having copies belonging to different communities.

The time complexity of the method mostly depends on the transformation phase which exact computation require time in $O(n^2)$. However the author has proposed an approximation which only considers the shortest paths of length k. This optimisation leads to an algorithm in time $O(n \log n)$ for the expansion phase.

3.5.9 Summary on global community detection

Tab. 3.1 summarizes the complexity of several community detection algorithms (adapted from [47]) The complexity (Compl.) is measured under the assumption that the graph is sparse. The scale of graph in which the method is appropriate is also provided : S for small scale (up to 10^4 nodes), M for medium (up to 10^6 nodes) and L for large scale (up to 10^9 nodes). Whether or not the method is overlapping (Overlap.) or (Hierarchical) is also indicated.

Table 3.1: Summary of some community detection algorithms.

Method	Compl.	Scale	Overlap.?	Hier.?
CPM ([119])	$O(exp(n))$	M	Y	N
WALKTRAP ([122])	$O(n^2 \log n)$	S	N	Y
Fast Greedy ([107])	$O(n \log^2 n)$	M	N	Y
Spectral optimization ([108])	$O(n^2 \log n)$	M	N	N
Louvain ([18])	$O(n)$	L	N	N
Girvan-Newman ([110])	$O(n^3)$	S	N	Y
Link community ([44])	$O(n^2)$	M	Y	Y
Label propagation ([130])	$O(n)$	L	N	N
COPRA ([57])	$O(n \log n)$	L	Y	N

3.6 Evaluation of community detection methods

Given a community detection method, how can we be sure that it performs well? And how can we compare two different methods that eventually optimise two different objectives functions? These are the principal questions addressed by evaluation techniques for community detection. There are two principal configurations, depending on whether the ground truth is known or not.

3.6.1 Evaluation with ground truth

When the ground truth is known, the evaluation consists in comparing how well the algorithm recovers the known communities. For that purpose, among others, the following partitions comparison indexes can be used: the Rand Index and its variations, the Mutual Information and its variations.

Given a set of elements $S = \{e_1, e_2, ..., e_n\}$ and two partitions P and Q of S to compare, lets define:

- a, the number of pairs of elements in S that are in the same set in P and in the same set in Q;
- b, the number of pairs of elements in S that are in different sets in P and in different sets in Q;
- c, the number of pairs of elements in that are in the same set in and in different sets in

- d, the number of pairs of elements in S that are in different sets in P and in the same set in Q.

The rand index [137] is :

$$RI = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{3.6}$$

RI ranges from 0 to 1, 1 meaning a perfect matching. There are some known problems with RI such as the fact that the expected value of the RI of two random partitions does not take a constant value (say zero) or that the Rand statistic approaches its upper limit of unity as the number of clusters increases. With the intention to overcome these limitations the Adjusted for chance Rand Index (ARI) was introduced. The ARI is computed by:

$$ARI = \frac{\binom{n}{2}(a+b) - [(a+b)(a+c)(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)(c+d)(b+d)]} \tag{3.7}$$

with expected value zero and maximum value one.

The Mutual Information (MI) allows to compare two partitions by quantifying the information they share. The mutual information of two partitions X and Y is given by:

$$MI(X,Y) = \sum_{i=1}^{r}\sum_{j=1}^{s} P(i,j) log \frac{P(i,j)}{P(i)P(j)} \tag{3.8}$$

With $P(i) = \frac{|X_i|}{N}$ the probability that a node in the community X_i in the first partition, $P(j) = \frac{|Y_j|}{N}$ is the probability that a node in the community Y_j in the second partition and $P(i,j) = \frac{|X_i \cap Y_j|}{N}$ the joint probability. MI actually measures the difference between the entropy of X and the conditional entropy of X knowing Y.

A normalized version of mutual information that takes values ranging between 0 (different partitioning) and 1 (identical partitioning) is given by:

$$NMI(X,Y) = \frac{2\,MI(X,Y)}{H(X)+H(Y)} \tag{3.9}$$

with $H(X)$ being the entropy of X.

Usually, the ground truth is not available (if we already know the community, why do we need to compute them?), for that reason, some graph generator with known community structure have been introduced. The mostly used of these generators is the LFR benchmark [78].

3.6.2 Evaluation without ground truth

When the ground truth is unknown, the only way to evaluate a community detection method or to compare two methods is by defining a quality measure. The mostly used measure in this case is the modularity defined in section 3.3. For each algorithm, the value of the quality function is computed and one can compare the algorithms based on this value.

3.6.3 Presence of a community structure in a network

All the community detection methods presented in this chapter suppose that a community structure is present in the network. However, a community structure is not always present or easy to detect. This topic is the subject of active research. The works in [73] and [21] use the notion of *consensus* to decide whether or not a network has a community structure. More precisely, different partitions are found in the network by using many executions of a non deterministic algorithm and the nodes frequently in the same communities form the consensus (also called community cores). It has been shown in [21] that in a network without community structure, community cores are trivial, either containing all the nodes of the graph or one node each.

3.7 Conclusion

In this chapter, we have presented the problem of global community detection and show how it has been addressed in the literature. Due to the lack of formal definition of community structure, hundreds of methods have appeared these past years. To get the communities a given node belong to, one can mine all the communities of the network, using one of the above described methods and then select the communities of this target node. However, the methods described in this chapter usually assume that the entire network is available and can be analysed as a whole. They also assume that the network do not evolve with time. Because these assumptions are not always true, and also because it is sometime usefull to get in realtime the community a given node belongs to, in a network that change quickly, we will present in the next chapter methods that can be used in these contexts.

CHAPTER 4

Local community identification in social networks

"Beware of bugs in the above code; I have only proved it correct, not tried it "

- Donald E. Knuth

4.1 Introduction

In the previous chapter, an overview of the global community detection problem has been presented. Algorithms like the ones published in [111, 18] have been described. These algorithms generally assume that the complete structure of the network is known i.e., it can fit into the memory of a single machine. This assumption is not realistic for very large networks like the Web. Moreover, for such networks, these algorithms produce communities with thousands or millions of nodes and this may not be useful in practice. Alternatively, we can focus only on the communities a given node belongs to. This is called the local community identification problem. It is solved by exploring a portion of the graph starting from the given node.

To overcome the complexity of graph exploration problems, most of the proposed algorithms for local community identification are based on greedy heuristics. They proceed in an iterative manner: starting from a node, they add at each step one of the external nodes that maximize a quality function. Such a method is described in [28]. However, these algorithms have difficulty in identifying a local community when starting from a node which is at its boundary. Moreover when a node belongs to several local communities, these algorithms fail to produce the overlapping sub-communities. In this chapter, we propose to improve the existing methods for local community

identification and we propose a method to identify the local overlapping communities starting from a specific node.

This chapter is organized as follows: section 4.2 presents previous methods for the identification of local communities; section 3 presents our improvement for local community identification, section 4.4 presents our proposed method for the identification of local overlapping communities and finally section 4.5 presents the performance of our algorithms on synthetic and real datasets, and compares them to some state-of-the-art algorithms.

Contents

4.1 Introduction 35
4.2 Previous methods 36
4.2.1 General greedy scheme for community detection . . 37
4.2.2 Quality functions for local community identification 39
4.3 Improvements for local community identification in networks 42
4.4 Identification of overlapping local communities . . 45
4.5 Performance evaluation 46
4.5.1 Tests on toy examples 46
4.5.2 Results on NCAA 2000 Football League 46
4.5.3 Comparison with the LFR benchmark 48
4.5.4 Results on Netscience 50
4.5.5 Results on Amazon 2006 52
4.5.6 Results on Skyrock friendship network 54
4.5.7 Results on Skyrock words network 55
4.6 Related work 55
4.6.1 Egomunities of the first level 59
4.6.2 Multi-ego-communities 60
4.7 Conclusion 61

4.2 Previous methods

Hereafter, a network is represented by an undirected and unweighed graph $G =< V, E >$, where V is the set of nodes and E is the set of edges. A neighbour of node u is a vertex v such that $(u, v) \in E$. $\Gamma(u)$ denotes the set

of neighbours of u. Let D be a subset of V. An edge $e = (u, v)$ is internal to D if both ends u and v are in D. An outgoing link is an edge $e = (u, v)$ that has only one end in D.

The density δ of links present in a graph is a generic notion that refers to the number of links divided by the number of nodes to which they are incident. The internal density δ_{in} corresponds to the number of internal links of a sub-graph divided by the number of vertices. Similarly, the external density δ_{out} corresponds to the number of outgoing links divided by the number of nodes.

A community of a network G is a subset of nodes D such that δ_{in} is high, and δ_{out} is low. In the case where the network is not fully known, the community produced by exploring G starting from a node n_0 is called the local community of n_0.

4.2.1 General greedy scheme for community detection

When we have started our work on local community identification in social network, there where some existing methods starting with the one by Clauset [29] in 2005 and ending with the one of Chen and colleagues [28] in 2009. Most of this existing algorithms for local community identification are based on a greedy and iterative scheme in which, at each step we have a local view of the graph G as follows:

- the set D of nodes already identified as members of the local community. This set D can be divided into two subsets:
  - the subset C of nodes u such that $\Gamma(u) \subseteq D$, called the core of D;

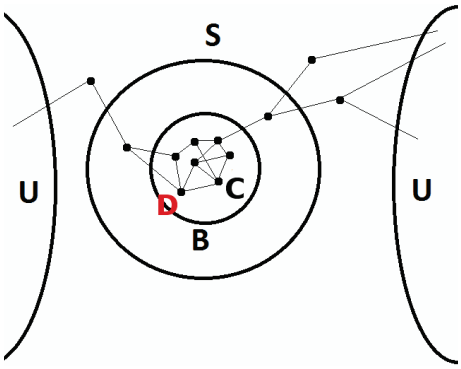


Figure 4.1: Subsets used in greedy algorithms.

  - the subset B of nodes u that have at least one neighbour outside D, called the border of D;

- the set S of nodes that do not belong to D and are adjacent to at least one node of D;

- and the set U of nodes with no neighbor in D.

Fig. 4.1 illustrates the subsets D, C, B and S. It is further assumed that the only way to get more information about G is to visit the nodes of S one at a time.

The basic greedy scheme for the identification of local communities using the maximisation of a quality function is given in Algorithm 1.

Algorithm 1 Identification of local communities based on greedy maximization of a quality criterion Q.

Algorithm: *Local community identification*
Input: a graph G and a starting node n_0.
Output: a subset D : the local community of n_0.
Initialize D with n_0
Initialize B with n_0
Initialize C with the empty set
Initialize S with the neighbors of n_0
Q = 0
Repeat
  For each $s_i \in S$ **do**
    Compute the quality criterion obtained if s_i is added to D
  End for each
  Select the node s^* that produces the maximal quality Q^*, breaking ties randomly.
  If $Q^* > Q$ **then**
    Add s^* to D and remove it from S.
    Update B, S, C.
  End if
Until $(Q^* \leq Q)$
Return D

Initially, the local community D of n_0 is reduced to n_0, $B = D$, C is the empty set, $S = \Gamma(n_0)$ and the quality of this initial community is 0. At each step, the node $s^* \in S$ that maximizes the quality function Q used by the algorithm is considered. If its inclusion into D increases the quality criterion Q, then it is removed from S and added to D, and the subsets D, C, B and S are updated. This procedure is repeated until there is no vertex $s \in S$

whose inclusion into D increases the quality Q. At the end of the algorithm, D contains the local community of n_0.

4.2.2 Quality functions for local community identification

The idea introduced by Clauset [29] is that nodes on the border of a community (nodes of B) must have more links with D (the nodes of the community) than with S (the nodes outside the community with some neighbors in D). The local modularity of D is then defined by formula (4.1):

$$R = \frac{B_{in}}{B_{in} + B_{out}} \tag{4.1}$$

where $B_{in} = \sum_{u \in B} |\Gamma(u) \cap D|$ is the number of links between B and D, and $B_{out} = \sum_{u \in B} |\Gamma(u) \cap S|$ is the number of links between B and S.

Note that $R = 1/(1 + \frac{1}{R'}) = f(R')$ where $R' = B_{in}/B_{out}$. Since f is an increasing function of R', it follows that the quality functions R and R' are equivalent.

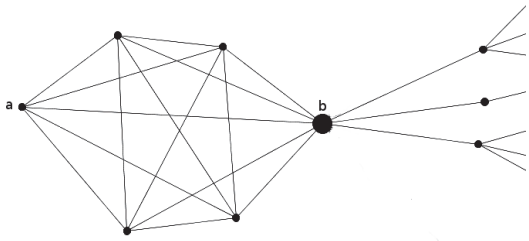


Figure 4.2: A node highly connected with nodes outside its community.

This quality function does not treat correctly nodes that are highly connected to neighbors outside their communities. For instance, the application of this quality function to the network of Fig. 4.2 will give a solution where the local community of node a does not contain node b. Indeed, before the insertion of b, $B_{in} = 10$ and $B_{out} = 5$, hence $R' = 2$. The inclusion of b leads to $B_{in} = 5$, $B_{out} = 3$ hence the new value of R' is 1.67. This is because the inclusion of node b decreases drastically B_{in} which in this case is good for the community, but is wrongly taken into account by the quality function R' that decreases.

Luo [90] has proposed another quality function that takes into account all the internal links rather than just those edges that link B to D. The new

quality function is defined by :

$$M = \frac{D_{in}}{D_{out}} \tag{4.2}$$

where $D_{in} = \sum_{u \in D} |\Gamma(u) \cap D|$, and $D_{out} = B_{out}$. When applied to the network of Fig. 4.2, this quality function produces for node a a local community that contains node b. The limitation of this quality function, when using the greedy approach, is that it can introduce unwanted nodes in the community. Indeed, let us consider the example of Fig. 4.3 where, starting from n_1, nodes n_2, n_3 and n_4 have already been inserted into D. Clearly, nodes $O_1, ..., O_5$ of this diagram don't belong to the local community of n_0. On the other hand, it is easily checked that using the quality function M, these nodes will be inserted successively into D. Note that the same observation holds for the quality function R.

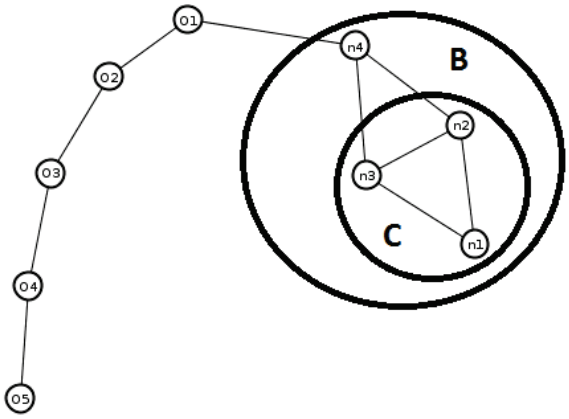


Figure 4.3: Pathological cases for quality functions R and M.

To solve this problem, Chen et al. [28] have proposed a new quality function L and a new method. This method introduced a first innovation: it considers the densities of intra-community edges and outer edges and not their numbers. More precisely the density of intra-community links L_{in} is defined according to the expression:

$$L_{in} = \frac{\sum_{i \in D} |\Gamma(i) \cap D|}{|D|} \tag{4.3}$$

Similarly, the density of external links is defined by:

$$L_{ex} = \frac{\sum_{i \in B} |\Gamma(i) \cap S|}{|B|} \tag{4.4}$$

Chen et al. then use the ratio of these two densities to define a quality function:

$$L = \frac{L_{in}}{L_{ex}} \tag{4.5}$$

To avoid the improper inclusion of vertices into the local community as illustrated above for the example of Fig. 4.3, an increase of the quality function L does not induce the automatic inclusion of a new node into D. More precisely, let L'_{in} and L'_{ex} denote the new densities if n_i is added to D. We can have $L' > L$ and one of the two following cases:

1. $L'_{in} > L_{in}$: the addition of n_i increases the density of internal links, hence n_i, with respect to the internal density, deserves to be integrated into the community. However, we must distinguish two sub-cases:
   - $L'_{ex} \leq L_{ex}$: this means that the addition of n_i does not increase the density of external links; clearly n_i must be added to D.
   - $L'_{ex} > L_{ex}$: the addition of n_i increases the density of external links; in this sub-case, there is a doubt because n_i may belong to a nearby community and a final check will be done at the end of the algorithm.
2. $L'_{in} < L_{in}$ and $L'_{ex} < L_{ex}$: this means that the density of internal links decreases, reducing the cohesion of the local community D; at the same time, L_{ex} decreases even more, leading to $L' > L$; we are therefore in the presence of a node n_i that is an outlier for the local community D.

At the end of the main loop, i.e. when there is no extra outside node that produces a positive gain when added to D, the algorithm reviews the nodes of D. Each node of D is removed and a test to add it following the above procedure is carried out. During this confirmation stage, a node is maintained in D only if it falls into the first case.

This method performs correctly when applied to the network of Fig. 4.3.

However Chen's algorithm suffers from a serious shortcoming because it can fail to identify the local community of a node n_0 which is located on the border of another local community D'. Indeed, when looking for the local community D of such a node, the search can be extended to the community D' which is close to D after adding members of D', one after the other. In this case, node n_0 will be removed from its own community in the confirmation phase of the algorithm, which is not correct. For instance, when applied to node 7 in Fig. 4a, phase one of this algorithm may produce the subgraph on the right, generated by nodes i, for $i \geq 7$, and after the confirmation phase,

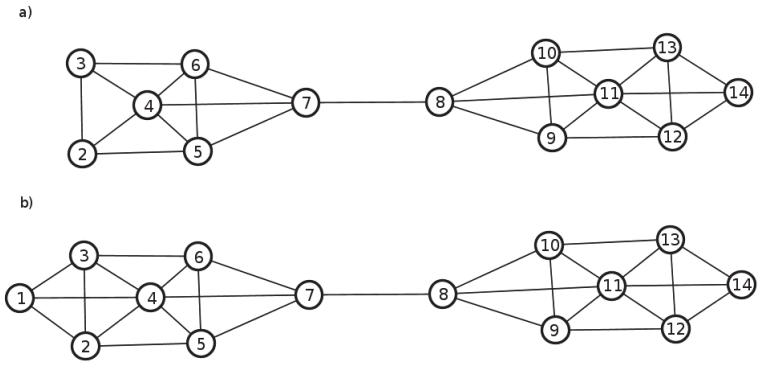


Figure 4.4: Node 7 is on the border of the local community on the right.

node 7 will not belong to D. This happens because at the first step, the neighbors of 7 that maximise the quality function are $5, 6, 8$, and node 8 may be chosen.

4.3 Improvements for local community identification in networks

To address the problem mentioned above, we proposed [112] to improve the algorithm of Chen et al. by adding at each step, and at the same time, all the nodes that maximize the quality function, instead of selecting one at random.

This algorithm, hereafter named BME0a according to the first letters of the first names of each author, runs correctly for the example of Fig. 4.4a when applied to node 7. Indeed it produces the subgraph on the left in Fig. 4.4a.

However, when applied to the network of Fig. 4.4b, it produces for node 7 a local community that does not contain node 1. Indeed, if node 8 is chosen at the first step, and node 5 or 6 is chosen at the second step, then after 6 steps, we obtain the situation where $D = \{2, 3, 4, 5, 6, 7, 8\}$ with $L = 2.29$. The inclusion of node 1 leads to $L' = 1.25 < L$, hence this node will not be included in the local community of 7, which is not correct.

This mistake comes from the fact that, since $|B|$ is at the denominator of L_{ex}, there can be a situation where the inclusion of a good node with many neighbors in C and no neighbor in S decreases drastically $|B|$, induces a drastic increase of L_{ex} and leads to $L' < L$. This is why we propose another improvement of the algorithm proposed by Chen et al. More precisely, we

modify the quality function proposed in [28] by using $|D|$ instead of $|B|$ in the denominator of L_{ex}. The new formula for the quality function is:

$$\widetilde{L} = \frac{\widetilde{L}_{in}}{\widetilde{L}_{ex}} \tag{4.6}$$

where $\widetilde{L}_{in} = L_{in}$ and $\widetilde{L}_{ex} = \frac{\sum_{i \in B} |\Gamma(i) \cap S|}{|D|}$. With this modification, and the new algorithm hereafter named BME0b, the example of Fig. 4.4b is treated correctly i.e., the local community of node 7 is the subgraph on the left.

However BME0b fails to identify the local community of node 5 in Fig. 4.5. In this method, the node with the smallest degree n^* is systematically

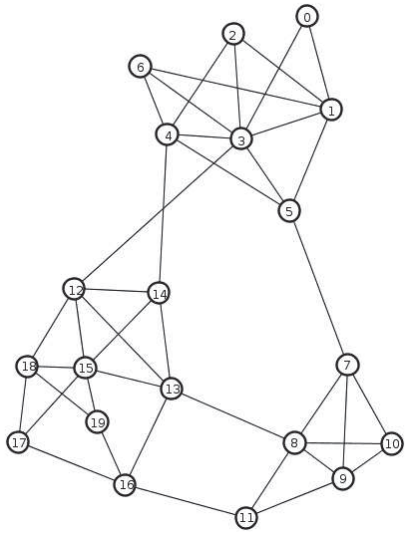


Figure 4.5: Example of community structure.

inserted into D

6 D

$7, 8, 9, 10, 11$ 5

T

$$T_{in} = \frac{\sum_{i \in D} \frac{|\Gamma(i) \cap D|}{(1+d_i)}}{D}$$

Table 1. Summary of methods.

Method	Fig. 2 (a)	Fig. 3 (n_1)	Fig. 4a (7)	Fig. 4b (7)	Fig. 5 (5)
Clauset et al.	No	No	No	No	No
Luo et al.	Yes	No	No	No	No
Chen et al.	Yes	Yes	No	No	No
BME0a	Yes	Yes	Yes	No	No
BME0b	Yes	Yes	Yes	Yes	No
BME1	Yes	Yes	Yes	Yes	Yes

where d_i is the length of the path from the starting node n_0 to node i in the tree generated by the algorithm. In this expression, internal links that are close to the starting node are favored by the multiplicative factor $1+d_i$. For external links, the contribution is defined by:

$$T_{ex} = \frac{\sum_{i \in D} |\Gamma(i) \cap S|(1+d_i)}{D} \tag{4.8}$$

In this expression, external links that are far away from the starting node are penalized by the multiplicative factor $1+d_i$. This leads to the quality function:

$$T = \frac{T_{in}}{T_{ex}} \tag{4.9}$$

BME1 computes correctly the local communities for the network in Fig. 4.5. Table 1 summarizes the performance of the presented algorithms when applied to networks of Figs. 3 to 6 and the starting node in parenthesis.

A final improvement of all the methods presented so far is to modify the condition of the confirmation phase. In this new method, another condition for maintaining a node n_i is $|\Gamma(n_i) \cap D| \geq D_{in}/|D|$. This improvement works for many cases presented previously and where some nodes were wrongly excluded during the confirmation phase.

The time complexity of the algorithms derived from Chen et al.'s method, depends only on the average degree of the network, the size of the local community found and the number of neighboring nodes of that community. It is in the worst case $O(|D|d|S|)$, were $|D|$ and $|S|$ are usually very small compared to the size of the whole network.

4.4 Identification of overlapping local communities

A natural idea for identifying overlapping communities is to take an algorithm A for local community identification and apply the scheme of Algorithm 2. In this scheme, A corresponds to one pass through the loop, V is the set of nodes confirmed in the second phase of the algorithm A and LocalCom is the table of the local overlapping communities found, indexed by idcom. After each execution of A, the links that are internal to $V \setminus \{n_0\}$ are deleted.

Algorithm 2 Identification of Overlapping Local Communities

Algorithm: IOLoCo
Input: a graph G and a starting node n_0.
Output: a table $LocalCom$[..] of local overlapping communities of n_0
idcom = 0
Initialize $LocalCom$ with the empty table
Repeat
V = the local community of n_0 produced by algorithm A
if $n_0 \in V$ **or** $|\Gamma(n_0) \cap V| >= \sum_{i \in V} \frac{|\Gamma(i) \cap V|}{|V|}$ **then**
$LocalCom$[idCom] = V
idcom = idcom + 1
end if
Mark all the internal links of $V \setminus \{n_0\}$ as "deleted"
Until ($\Gamma(n_0) = \emptyset$)
Return $LocalCom$

IOLoCo executes the loop for local community detection k times, where k is the number of sub-communities explored; k is small for all datasets tested. As a consequence, IOLoCo runs efficiently on huge graphs such as those produced by social web applications. As mentioned before, we can use any non overlapping local community detection method inside IOLoCo. For the evaluation presented in the following section, we have used $BME0b$ with which we have obtained the better results.

The execution of IOLoCo can produce communities that are highly overlapped. In other to merge communities with high overlap, we can follow the procedure proposed in [50] which is based on the computation of the $overlap(C_1, C_2) = |C_1 \cap C_2|/min(|C_1|, |C_2|)$ between two communities C_1 and C_2. The authors then propose to build a graph where each node is a community and there is a an edge between two nodes C_1 and C_2 if $overlap(C_1, C_2)$ is greater than a threshold O_{min}. Finally, communities belonging to the same connected component are merged.

4.5 Performance evaluation

To validate our algorithms, we present results on some simple datasets with known communities. Then we make a comparison with existing algorithms on benchmark graphs. After that, we present some results on the collaboration network between scientists working in the field of network analysis. We also present the results on the social network of co-purchases on the Amazon web site. Finally, results on a real online social network are presented. Besides the datasets provided by Skyrock, all the others were choosen because they are publicly available and they represent many different type of real networks. The dataset provided by Skyrock are not publicly available but have the advantage that the result can be verified by the provider of the data.

4.5.1 Tests on toy examples

In Figs. 4.6, 4.7 and 4.8 we present examples of results obtained with IOLoCo. On all these figures, the starting node is the central one and each community is represented by an ellipse. In Fig. 4.8 for instance, node 1 belongs to the two local communities $\{0, 1, 2, 3, 4\}$ and $\{0, 1, 5, 6, 7\}$ of node 0.

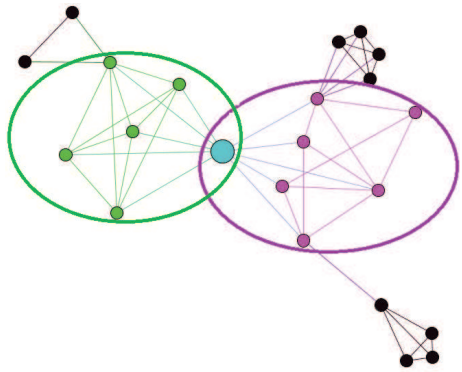


Figure 4.6: The two local communities identified by IOLoCo for the central node.

4.5.2 Results on NCAA 2000 Football League

The NCAA 2000 Football league [106] is a real dataset which represents football matches between 110 teams group in 11 conferences. The underlying network is constructed as follow: the nodes are the team and a links is present

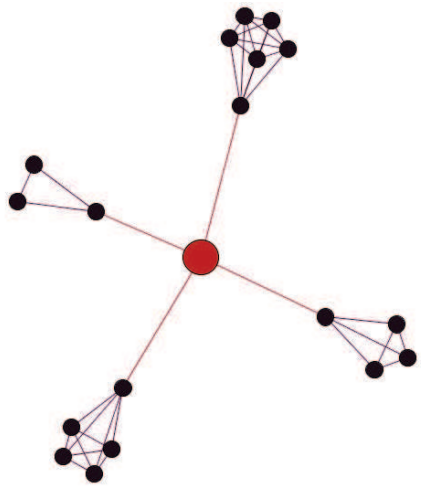


Figure 4.7: No community is found by IOLoCo for the central node.

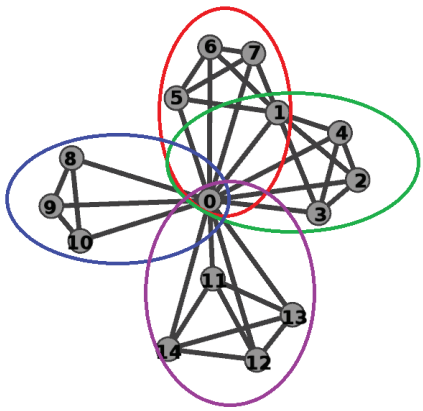


Figure 4.8: Four communities identified by IOLoCo for node 0.

Table 4.1: Quality evaluation

2000 NCAA League		Results							
		Chen et al.'s Algorithm				BME0b			
Conference	Size	Null.	Préc.	Rap.	F.	Null.	Préc.	Rap.	F.
Atlantic coast	9	2	1	1	1	0	1	1	1
Big East	8	7	1	0.50	0.667	0	0.95	1	0.972
Big 10	11	6	1	1	1	0	1	1	1
Big 12	12	2	1	1	1	0	0.99	1	0.997
Conference USA	10	8	0.93	0.50	0.639	0	0.91	0.83	0.878
Mid American	13	0	1	1	1	0	0.98	0.84	0.893
Mountain West	8	5	1	0.5	0.667	0	0.96	1	0.975
Pac. 10	10	1	1	1	1	0	1	1	1
SEC	12	1	0.99	1	0.996	0	1	1	1
Sunbelt	7	7	-	-	-	0	0.64	0.51	0.566
Western Atletic	10	2	0.84	0.74	0.758	0	0.75	0.66	0.700

between two node if and only if the two team have played a match together. The teams of the same conference usually play together and the matches between conferences are uncommon. That leads to a natural division into communities: the conferences.

We compare our non overlapping method, BMEb, with the method by Chen et al. on this dataset. For each node, it local community is computed using each algorithm, supposing that it must return all is conference as result. We can then compute the precision, the recall and the F-score and average for each conference. The table 4.1 gives the result. In this table *Null* gives the number of nodes that each algorithm says they do not have a community.

The results in table 4.1 show that our algorithm is able to find the communities for each team, and that these communities are of good quality.

4.5.3 Comparison with the LFR benchmark

We compare BME0b, BME1, and IOLoCo to the algorithms presented in [28] and [26] on networks generated by the benchmark proposed by Lancichinetti et al.[78], using among others, a parameter ν that is the mean value of $D_{out}/|D|$, where D denotes a community. A network of size 1000 is generated for each value $\nu \in \{0.1, 0.2, ..., 0.6\}$. For each given value of ν, a network $G =< V, E >$ is generated. Then, for each algorithm, we proceed as follows: for each node n of G, we compare the partitions $D \cup (V \setminus D)$ and $L \cup (V \setminus$

L) where D is the community of n in G and L is the local community of n generated by the algorithm. This is done using the normalised mutual information as performance index.

acc_m introduced by Papadakis et al. [121] for comparing estimated communities S'_i, $i \in \{1, ..., k\}$ and actual ones S_i, $i \in \{1, ..., k\}$, and is defined as:

$$acc_m = \frac{acc - \frac{1}{k}}{1 - \frac{1}{k}} \tag{4.10}$$

where

$$acc = \frac{1}{k} \sum_{i=1}^{k} \frac{|S_i \cap S'_i|}{|S_i \cup S'_i|} \tag{4.11}$$

Note that in our context $k = 2$. This index is interesting because $acc_m = 0$ for random partitions and $acc_m = 1$ when the two partitions are equal.

Given an algorithm and a network with parameter ν, Fig. 4.10 gives the average value of acc_m when all nodes are considered. On this benchmark graphs, the three algorithms introduced here perform better than the algorithms presented in [28, 26], and IOLoCo is the best.

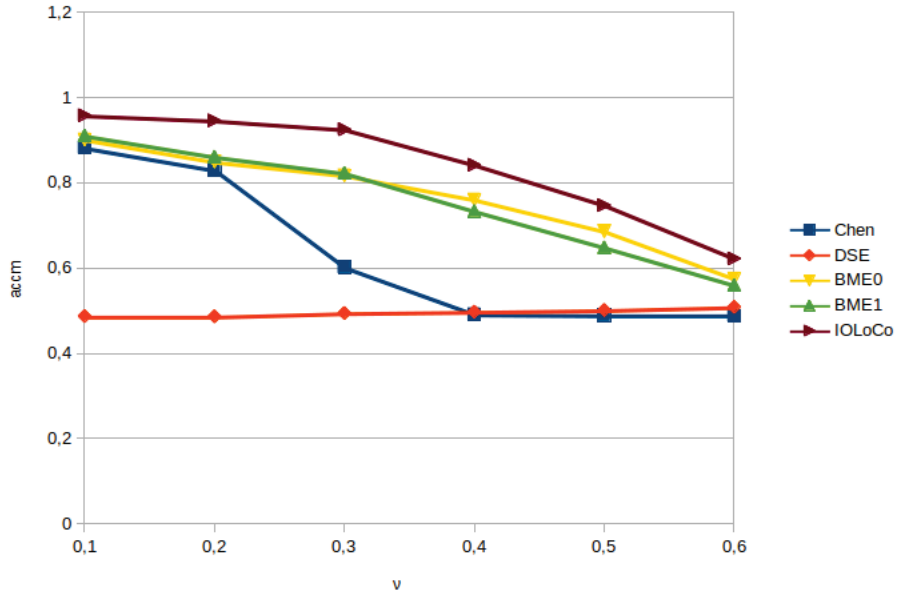


Figure 4.9: Performance (acc_m) of the 5 algorithms on six datasets (lower values of ν mean more separated communities).

For comparison purposes, we also reports result using normalised mutual information (NMI) in figure 4.10. This results are consistent with the previous one.

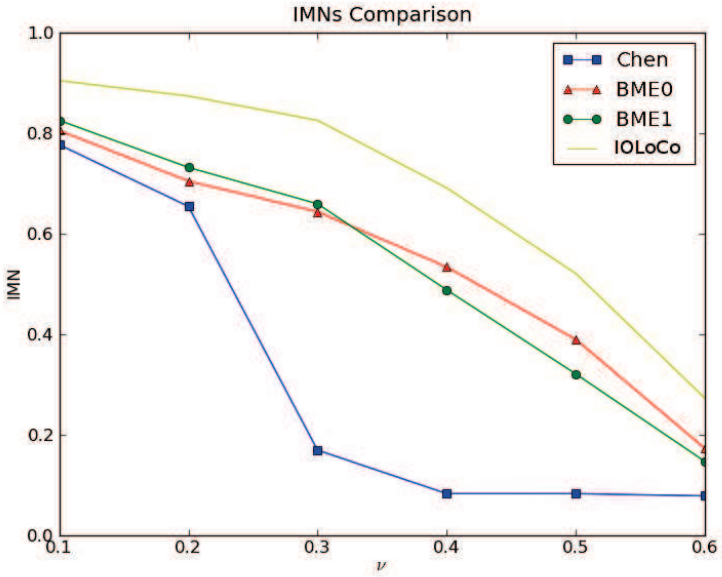


Figure 4.10: Performance (NMI) of the 4 algorithms on six datasets (lower values of ν mean more separated communities).

4.5.4 Results on Netscience

The Netscience dataset [108] represents the collaboration network between scientists working in the field of network analysis. Two authors are linked if they have co-published at least one paper. This network has 1589 nodes and 2742 links. It has several connected components with the largest one having 379 nodes. It also has several connected components of size two and three, leading to trivial communities. Fig. 4.11 gives the distribution of the sizes of the local communities found by Chen et al., BME1 and IOLoCo. The quality of the local communities cannot be easily compared on this dataset, because no ground truth is available (we ignore what are the "true" communities). However, we observe although for each size of community all the algorithms gives similar results, Chen et al.'s algorithm fails to find a community in 20% of the cases (more than 300 nodes with a local community of size zero). This is probably a consequence of the fact that these nodes belong to several

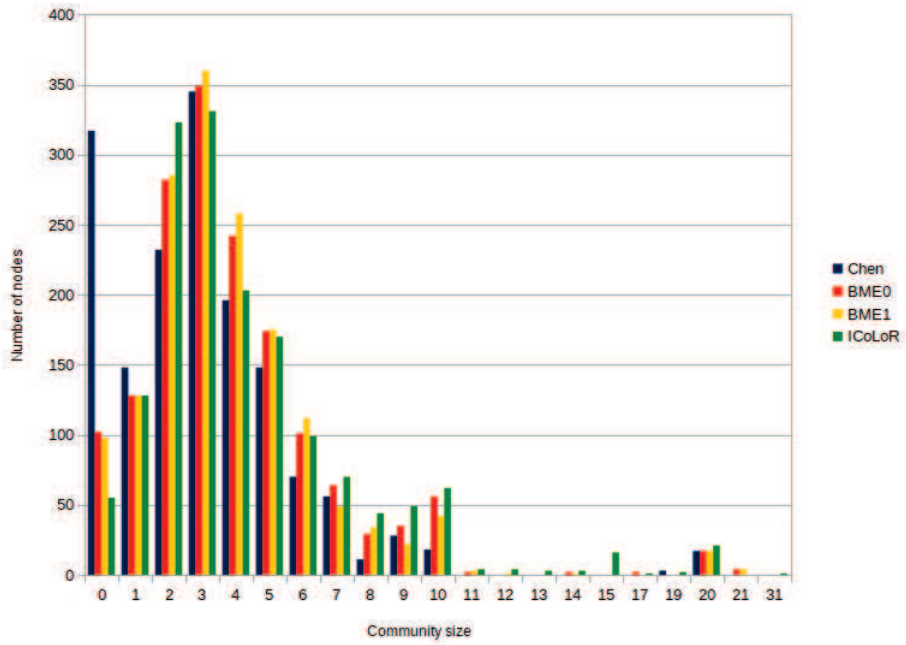


Figure 4.11: Size distributions of the local communities found for Netscience

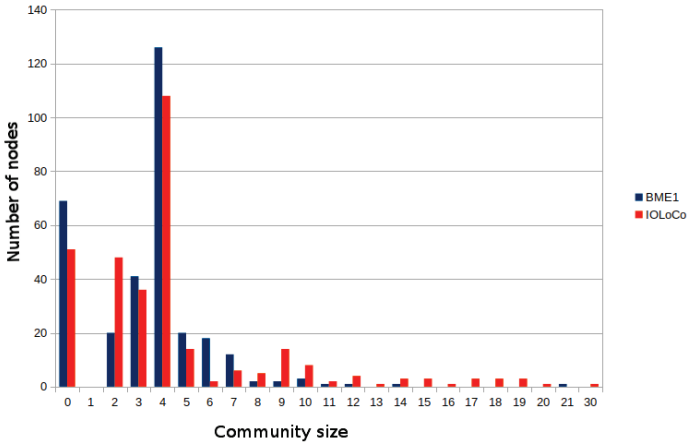


Figure 4.12: Netscience - Analysis of the nodes for which Chen et al.'s method returns no community.

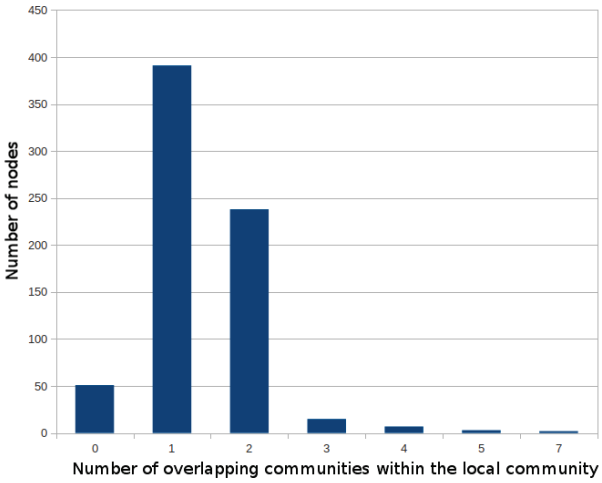


Figure 4.13: Netscience - sub-communities of nodes for which Chen et al.'s method fails.

communities. To check that, we have plotted the distribution of community sizes found by the two other algorithms for the 317 nodes which do not have local communities according to Chen et al.'s method. Fig. 4.12 presents this distribution; 218 (around 69%) of these nodes have a local community of more than 3 nodes. Fig. 4.13 shows that some communities found by IOLoCo for the nodes without local communities according to Chen et al.'s method are divided into several sub-communities. These observations confirm the fact that the algorithm proposed by Chen et al. sometime fails to identify the local community of a node because it does not consider the possibility of overlapping.

4.5.5 Results on Amazon 2006

The Amazon 2006 dataset [82] represents the network of co-purchases on the Amazon website collected during the Summer of 2006. The dataset contains product meta-data and review information about 548,552 different products (Books, music CDs, DVDs and VHS video tapes). For each product the following information is available: title, sales rank, list of similar products (that get co-purchased with the current product), Detailed product categorization, time, customer, rating, number of votes, number of people that found the review helpful. A network is constructed using this data as follows: each node is a product and there is a link between two nodes if the corresponding products are frequently purchased together. This network has $548,552$ nodes

Table 4.2: community of the book "Merlin Trilogy".

Num.	Members	description
1	The Wicked Day (Arthurian Saga, Book 4) The Last Enchantment (Arthurian Saga, Bk. 3.) The Crystal Cave (The Arthurian Saga, Book 1)	Arthurian Saga
2	The Sword in the Stone The Book of Merlyn The Once and Future King	Merlin and Arthur

Table 4.3: Local community of the book "Specter of the Past (Star Wars)".

Star Wars
The Last Command (Star Wars)
Children of the Jedi
Heir to the Empire (Star Wars)
The Last Command (Star Wars)
Dark Force Rising (Star Wars Vol. 2)
Champions of the Force (Star Wars)
Vision of the Future (Star Wars)
Star Wars Darksaber
Dark Apprentice (Star Wars)
Planet of Twilight (Star Wars (Random House Paperback))
Jedi Search (Star Wars)
The Last Command (Star Wars)
"I, Jedi"

and $1,788,725$ edges.

The overlapping local communities found by IOLoCo for the book *"Merlin Trilogy"* (see table 4.2) of *Mary Stewart* are $L_1 = \{$The Wicked Day, The Last Enchantment, The Crystal Cave$\}$ and $L_2 = \{$The Sword in the Stone, The Once and Future King,The Book of Merlyn$\}$. The books of L_1 belong to the series Arthurian Saga and the books of L_2 have Merlin and Arthur as main actors.

Another example is given in table 4.3 where the community of the book "Specter of the Past (Star Wars)" is presented. In these simple examples, the local communities found by IOLoCo are meaningful: they correspond to the same series or to the same theme. These communities can clearly be exploited by a recommendation system.

4.5.6 Results on Skyrock friendship network

Figures 4.14, 4.15 and 4.16 show some examples of local communities in the Skyrock Friends network. The first community (Fig. 4.14) was obtained starting from node pompier-france37 (pompier means fireman). It has 59 members and is divided into three sub-communities. The user's pseudos in this community are in majority related to fireman. Not surprisingly, one sub-community is related to police forces. The second local community presented in Fig. 4.15 was detected starting from node carpistedu62 ("carpiste" is French for "carp's fisher"). This local community has 19 members. The majority of the pseudos contain carp and 62. 62 is the number of a French department called Pas-de-Calais. This is thus a local community composed of carp's fishers from this region.

Finally, the last figure show the local community of a *future mother*. This community has three sub-communities with all the pseudo related to *new mother*.

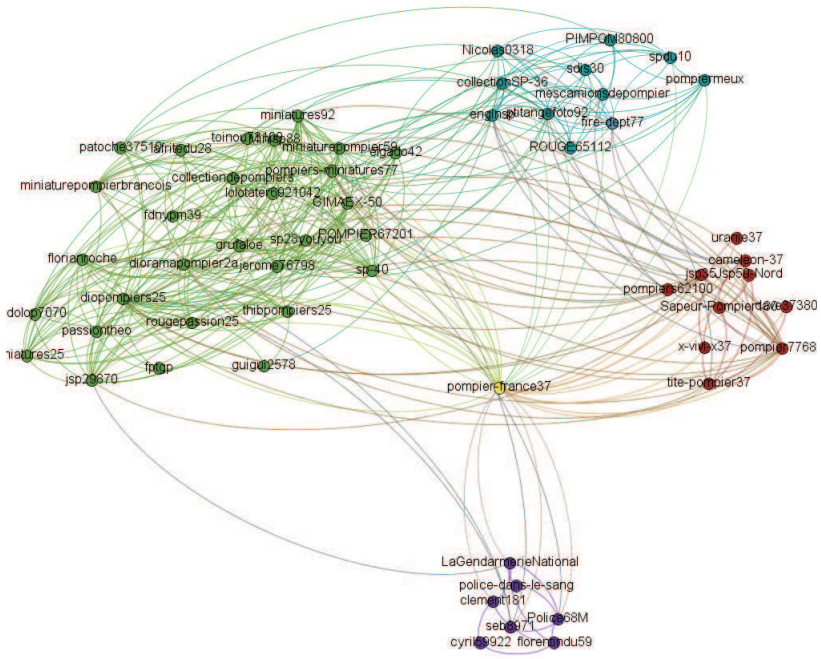


Figure 4.14: Skyrock - a community of firemen.

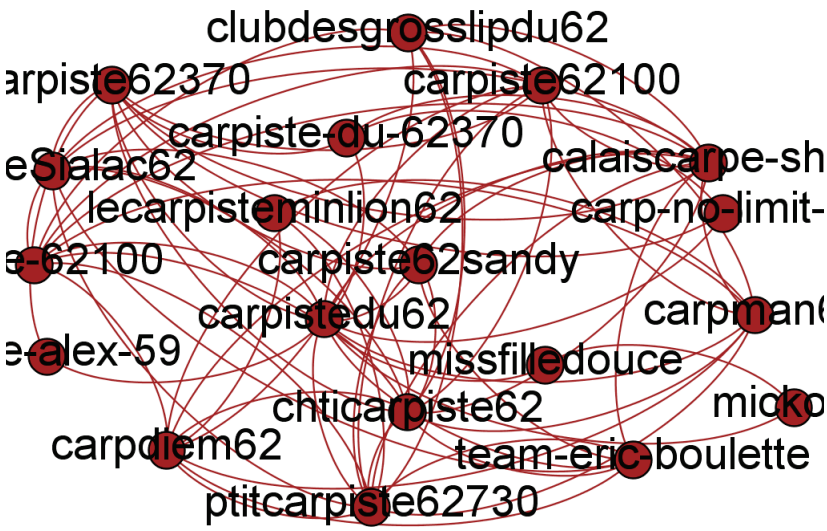


Figure 4.15: Skyrock - a community of *carp* fishers.

4.5.7 Results on Skyrock words network

In Skyrock, user can also used *tags* to describe their profiles. With this *tags*, a network can be built as follow: each tag is a node and a link is created between two nodes if at least k user have this tag.

We have applied our local community detection to this network and bellow are some detected communities. Fig. 4.17 presents the local community of the tag *animaux* which has two sub-community. One seem to be related to names (a) and the other to abuse.

Fig. 4.18 presents the local community of the tag *aimer* with word related to sentiments. And Fig. 4.19 presents the community of the tag *audi* with many word related to racing games.

4.6 Related work

In this section we recall some works quite similar to local community identification. The first one named *egomunities* [50] consists in detecting the overlapping communities on the neighbours of a given node (discarding the rest of the network). The second one called *multi-ego-communities* consider multiple starting nodes and detect the communities that contain them [36].

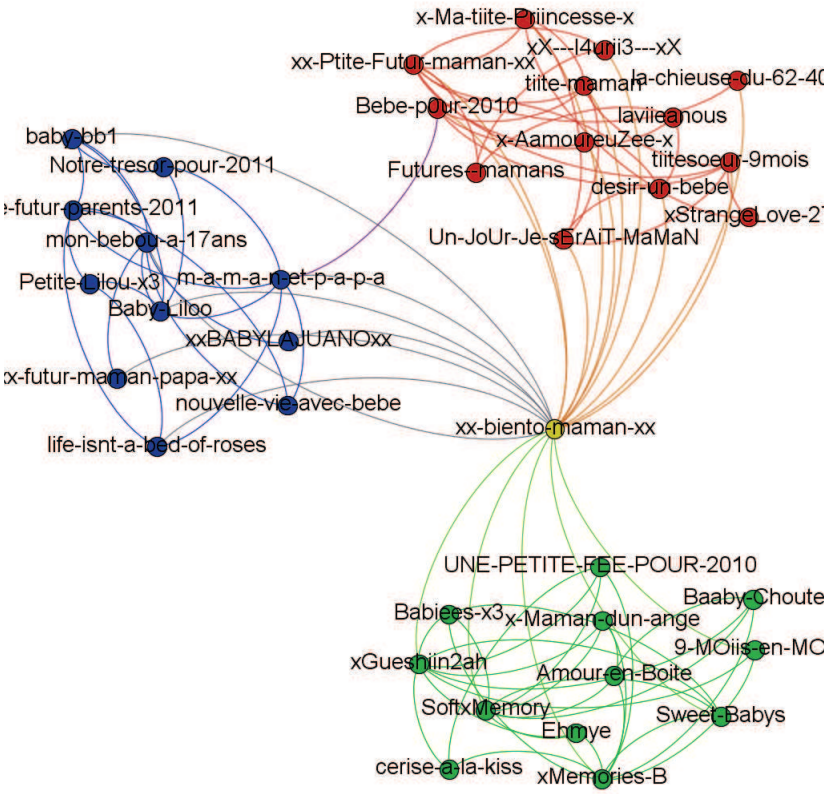


Figure 4.16: Skyrock - a community of futur mothers.

(a)

(b)

Figure 4.17: Skyrock - local community of the word *animaux*

Figure 4.18: Skyrock - local community of the word *aimer*

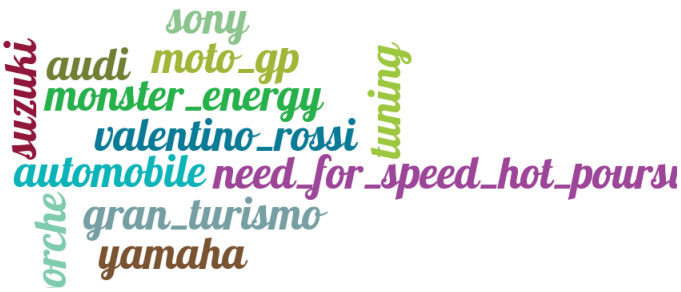


Figure 4.19: Skyrock - local community of the word *automobile*

4.6.1 Egomunities of the first level

To detect all the overlapping communities on the neighbourhood of a particular node, the authors in [50] have proposed to used the cohesion of a set of node defined as:

$$C(D) = \frac{\Delta_{in}(D)}{\binom{|D|}{3}} \frac{\Delta_{in}(D)}{\Delta_{in}(D) + \Delta_{out}(D)} \tag{4.12}$$

where D is a community, $\Delta_{in}(D)$ and $\Delta_{out}(D)$ are respectively the number of internal triangles of the set D (with all three nodes in D) and the number of triangles pointing out of the set D (with exactly one of the nodes out of D).

Let E be set of nodes. The cohesion is used to detect the set of overlapping communities D on the neighbourhood of given node u in a graph G as follows:

1. from the graph G extract the graph G' containing only the set of nodes $u \cup \Gamma(u)$ and all their links;

2. from G' remove all the nodes with degree one as they can not form triangles;

3. select the node v with the highest degree from $\Gamma(u)$ and add u and v to E;

4. while there is a node $w : w \in \Gamma(u)$ and $w \notin E$ which produces a maximum positive cohesion gain, add w to E;

5. add E to D;

6. repeat this process from step 2 until there is no more discovered community

7. return D.

We do not have reported any comparison with this method because it do not solves the same problem: it finds the all the community in the subgraph composed of a node and its neighbours. From our point of view, due to the size of the networks analyzed using this method, other overlapping community detection techniques presented in chapter 3 can be suitable to solve this problem and it lacks such kind of comparison.

4.6.2 Multi-ego-communities

Multi-ego-communities are defined in [36] as a local community centred around k ($k \geq 1$) nodes. To compute the multi-ego-communities, the authors of [36] have first proposed a new node similarity measure based on information diffusion on networks. To compute the similarity between a node u and all the other nodes of the network, the following process is proposed:

1. initialise the similarity score of u with one and the score of all the other nodes to zero;

2. at each iteration, the similarity of each node is set to the average similarity of its neighbours;

3. rescale all the similarities so that the lowest similarity is set to 0;

4. reset the similarity of node u to 1;

5. repeat steps $2-4$ until convergence.

This procedure produces the values for the similarity call *carryover opinion metric* by the authors. It gives a way to have a similarity between a given node and all the others. The similarity of node u is always reset to one as it must always be perfectly similar to itself. The ego-community of a node is then the set of nodes more similar to it.

Based on this similarity measure, and a set of k nodes $U = \{u_1, u_2, ..., u_k\}$ the multi-ego-communities are detected as follows:

1. for all nodes of the network, evaluate the similarities to each node of the set U;

2. the similarity of a node v to the set U is given by the minimum or the geometric mean of all its similarities to nodes of U;

The multi-ego-community of a set of nodes is then the set of nodes more similar to this set.

We do not have reported any comparison with this approach because it was posterior to our contributions.

4.7 Conclusion

We have presented new algorithms for local community identification in social networks that perform better than all the compared methods in term of accuracy. IOLoCo is able to identify the overlapping local communities of a given node, a situation which arises frequently in real social networks and is not handled correctly by the algorithms with which the comparisons were performed.

The performance of our approaches were assessed on synthetic and real world data. The application to the dataset *Amazon 2006* shows that the communities produced are meaningful and may be used to enhance recommendation systems.

The detection of local communities is just a tool for more complex network analysis tasks. In real world social applications, the networks are usually very dynamic (nodes and links are created and deleted). The following chapter will present an application of local community identification to enhance the prediction of some users' behaviours in social networks.

CHAPTER 5

Local community identification applied to the prediction of user behaviours

"I believe that if you show people the problems and you show them the solutions they will be moved to act."

- Bill Gates

5.1 Introduction

The behaviours of users are often social or viral in the sense that one user will influence or be influenced by his or her friends. It thus makes sense to use the social connections of a user in order to predict his/her behaviours more accurately. The main intuition behind using social ties is that *close friends* influence more a user in the adoption of a behaviour [96]. The challenge is then to identify the set of *close friends* of each user. In this chapter we propose to model this notion of *close friends* using local communities.

In this chapter, local community analysis is applied to two different dynamic users' behaviours. The first phenomenon that will be studied is the churn. The term churn is derived from *change* and *turn*. It means the discontinuation of a contract. Churn prediction is a well studied data mining task in the context of telecommunication networks [155]. It consists of predicting whether, in a near future, a user will leave his present operator for another one. With the multiplication of social network platforms that generally provide similar functionalities, a subscriber of a particular platform can decide to churn and switch to another network. This phenomenon was observed on several platforms when Facebook arrived on the online social network market.

The contribution of *local community-based* attributes for churn prediction will be illustrated here on a dataset provided by Skyrock (`http://www.skyrock.com`). Skyrock is a large online blog network where users can establish friendship relations. It is then possible to compute the communities in this friendship network. The aim of this work is to show that the local communities of users in real friendship social networks such as Skyrock, produce relevant attributes for the construction of a supervised learning model for churn prediction.

The second application of local community analysis to predict users' behaviours is social recommendation. Given the incredible amount of items available in web stores like Amazon [85], it is very difficult for a user to find the right item he/she may want to get or purchase. Recommender Systems assist users in this task. In a well studied class of recommendation systems called user-based collaborative filtering [76, 3], a user gets, as recommendations, items already purchased[1] by users similar to him/her. This similarity is generally based on common purchase habits. However, when a user do not already have purchased many items, he/she cannot get recommendations (or just a few) because he/she is more similar to users also without many items. To overcome this problem (referred as cold start [43]) recommendations can rely on the "real friends" of the user based on his/her explicit social network. These methods are called social recommendation. Although when a user is new to a platform, he/she does not have friends, it is usually easiest to let him/her connect to existing friends (by importing his emails' contacts, for example, and finding already registered addresses) than make him buy many products. In this chapter, we propose a way to use local communities for social recommendation. The contribution of local communities to social recommendation will be illustrated using two well-known real datasets: *Flixter* and *LastFM*.

The rest of this chapter is organized as follows: in section 5.2, we present the churn prediction problem in the context of social networks; section 5.3 presents the application of local community analysis to social recommendation; and finally section 5.4 presents some conclusions and perspectives.

Contents

5.1 Introduction 63
5.2 Churn prediction in social networks 65
5.2.1 Proposed methodology 66
5.2.2 Dataset description and attributes extraction . . . 67

[1] We will used the term purchase in this manuscript but it can be replaced by clicked, view, rated, etc. depending on the context and the particular application.

5.2.3 Experimentations and results 70
5.3 Recommendation 75
5.3.1 Introduction to recommender systems 75
5.3.2 Proposed Social recommendation model 76
5.3.3 Evaluation 77
5.4 Conclusion and discussions 79

5.2 Churn prediction in social networks

In business applications, the churn corresponds to customer loss. In telecommunications, a subscriber is said to have churned when he leaves one carrier to move to another [68], and this phenomenon has been extensively studied [101, 37]. Indeed, it is generally admitted that retaining an existing customer is less expensive than winning a new one [59]. As a consequence, telecommunication companies tend to move important marketing efforts from customers acquisition to customers retention. Churn prediction is therefore an important task for Customer Relationship Management (CRM).

The objective of churn prediction in the context of social networks is to estimate the likelihood that a given user will stop using a social network platform in the near future. A churner can thus be defined as a user who has become inactive for a certain period of time. This knowledge can be exploited by the platform operator to take preventive actions: if the user is likely to stop using the platform, it could be interesting to send him some incentives (personalized recommendations, free applications, ...).

Most of the methods for churn prediction belong to three main categories: feature-based methods[68], network-based methods[37, 155] and hybrid methods. In feature based methods, to predict churn, dozen to hundred of attributes are generally derived from the customer's profiles (age, location, gender,...) and service usages (last usage date, frequency of usage, amount spent, ...). These features are then used to build a statistical model for churn prediction based on supervised learning [68]. This pure *feature-based* churn prediction has the limitation that it does not take into account the social relations between subscribers.

Network-based methods use the social links to detect the churners. The methods of this category usually model the churn prediction as a diffusion or contagion process [37]: starting with the known churners as seeds, each seed tries to activate its neighbours at each iteration. This processes is repeated until convergence or up to a maximum number of iterations. The analysis

carried out in the work proposed by Dasgusta et al. [37] explores the propensity of a subscriber of a mobile operator to churn, depending on the number of friends that have already churned.

Other studies have also been conducted in P2P networks. For example, authors of [16] have studied the bias induced by the length of the observation period while predicting the churn in P2P networks.

The limitation of network-based approaches is that they usually do not consider other available information (socio-demographic information, age, gender, address, profession,...) when they are available.

Finally hybrid methods combine the two previous ones. The proposed methodology of this chapter falls in this category and will be explained in the next section.

5.2.1 Proposed methodology

This section presents a new methodology for churn prediction in social networks. Our methodology is hybrid and combines the node attributes and the network information. In social networks, although node attributes are usually important, social ties or links are also relevant for churn prediction [37] because people form communities and are more active with members within their local communities than with members outside their local communities. It follows that if many members of a users' community stop using a service, the number of people with whom this user can interact through that service decreases, and the probability that he also churns gets higher. The local community of a user can therefore be mined to provide *community-based* attributes for churn explanation.

Our methodology follows a supervised learning approach and can be summarised as follows:

- at the start of the learning period, the network of active user is extracted;
- at the end of the learning period, node attributes and network-derived attributes are extracted and a supervised learning model using these attributes is constructed;
- the constructed model is used to detect the churners for the prediction period.

This methodology is general and is illustrated in figure 5.1. In the next section a concrete example will be presented on a real dataset.

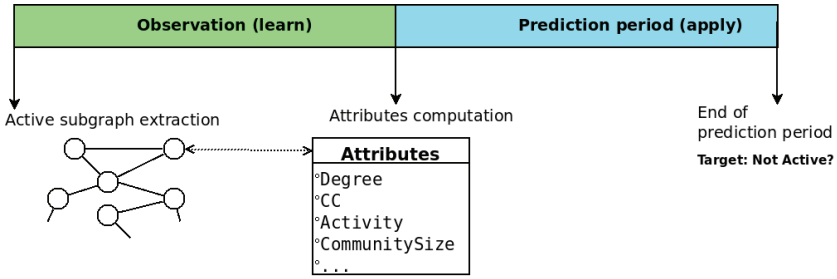


Figure 5.1: Experimental setup used for churn prediction.

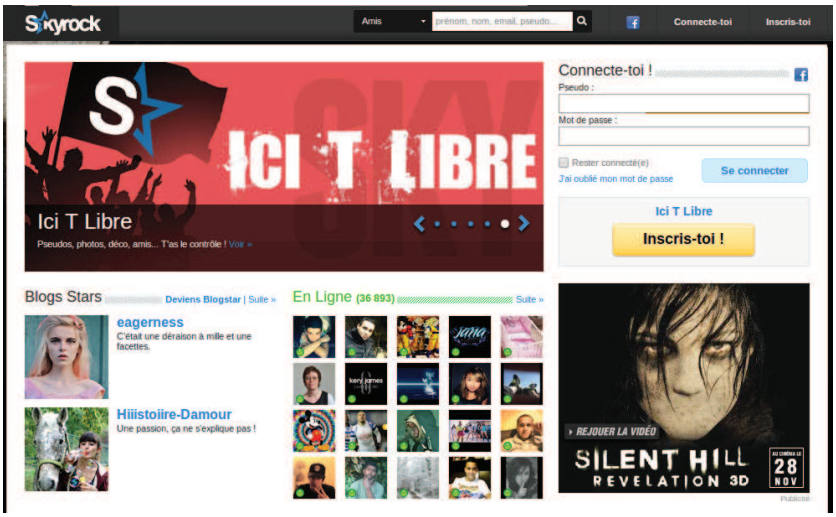


Figure 5.2: The main page of the Skyrock's social site.

5.2.2 Dataset description and attributes extraction

Skyrock (`http://www.skyrock.fr`) is a social platform where users (called "skynautes") can (among other actions) create blogs, add comments, create tags and define explicitly friendship relations. The global dataset of Skyrock has 31.3×10^6 nodes (the users' profiles) and 1.17×10^9 links (the friendship relations). Fig. 10 shows a screenshot of the Skyrock Social Network.

The dataset used for the experimentations is constructed as follows (Fig. 9): from the global network, the sub-graph of active users in March 2011 is extracted, because churn prediction is only relevant for active users that become inactive. A user is active if he has made at least one connection to the platform during the considered period of time. In the following, only this graph formed by active users is taken into account.

After that, all the nodes that have more than $5,000$ (this threshold was provided by the Skyrock) friends are removed because they generally repre-

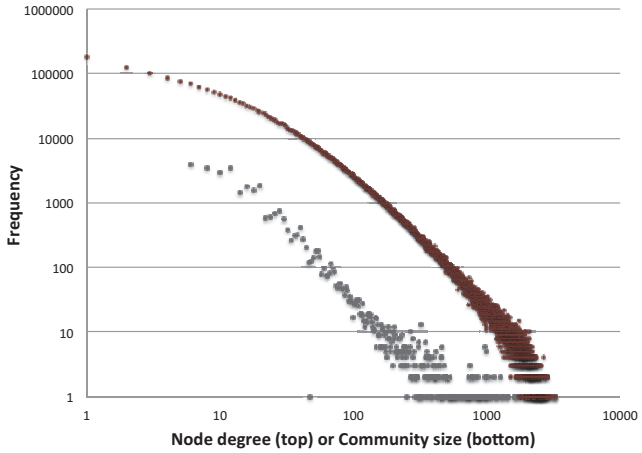


Figure 5.3: Global Skyrock friendship degree distribution

sent celebrities ("mega-hubs" in the network) with abnormal behaviour. This new sub-graph has 2.24×10^6 nodes and 127.428×10^6 links compared to the 31.3×10^6 nodes and 1.17×10^9 friendship links of the whole network. Fig. 5.3 presents the degree distribution of the nodes in this network. Not surprisingly, the degrees seem to follow a power law.

Local communities are then computed using the algorithm IOLoCo presented in chapter 4. The distribution of sizes of these local communities is shown in Fig. 5.3. For the sake of comparison, the global communities detected by the Louvain algorithm [18] are also computed.

Then, some simple attributes are defined to characterize a node and its vicinity. We consider only very simple attributes (degree, community size, proportion of active nodes, etc.) which can be computed quickly and have straightforward interpretation. Some of these attributes are related to the starting node and some of its social circles such as its local community, the first neighborhood , the second neighborhood (see Fig. 6.3) and the Louvain's community (the community structure with the highest modularity after ten executions). All these attributes are presented in table 5.1.

Then $50,000$ users are selected at random to create the learning dataset. The test set contains all the users of the period following the training period that where known in the training period. A supervised learning algorithm is trained and estimated on this dataset.

The results of attribute extraction using the methods introduced above, can now be presented. It's worth looking at the sizes of the sets of nodes used to compute the attributes: the *local community* size (attribute *Com-Size*) ranges from 1 to 547 with a mean of 21. The first neighborhood size

Table 5.1: Some attributes for churn prediction

	Attribute name	Description
1	Degree	The degree of the node
2	CC	The local clustering coefficient of the node
3	Activity	The number of time the node has made a connexion during the learning month
4	DaysAfterLastCon	The number of days since the last connexion of the node during the learning month
5	LocalComSize	The size of the local community i.e. the number of nodes of the local community
6	LocalInProp	The internal proportion i.e. the proportion of local community's node directly connected to the starting node
7	LocalAvgDegree	The average degree of the nodes inside the local community
8	LocalPropInact	The proportion of nodes inside the local community that are already inactive
9	LocalAvgAct	The average activity for the nodes of the local community
10	NeigSize	The size of the first neighborhood
11	NeigAvgDegree	The average degree of the first neighborhood
12	NeigPropInact.	The proportion of nodes inside the first neighborhood that are already inactive
13	NeigAvgAct	The average activity for the nodes of the first neighborhood
14	Neig2Size	The size of the second neighborhood
15	Neig2AvgDegree	The average degree of the second Neighborhood
16	Neig2PropInact	The proportion of nodes inside the first Neighborhood that are already inactive
17	Neig2AvgAct	The average activity for the second Neighborhood
18	LouvainSize	The size of the Louvain's global community the node belongs to
19	LouvainAvgDegree	The average degree of the Louvain's global community the node belongs to
20	LouvainPropInact.	The proportion of nodes inside the Louvain's global community the node belongs to that are already inactive
21	LouvainAvgAct	The average activity for the Louvain's global community the node belongs to
22	Not active?	The target attribute we want to predict

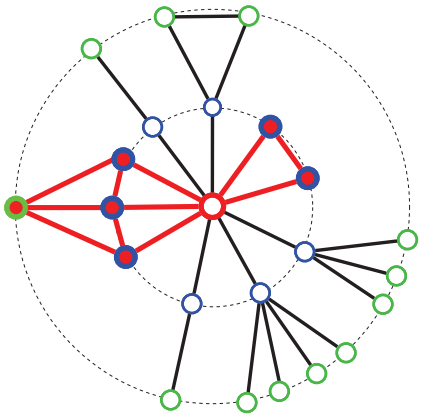


Figure 5.4: A central node and its first neighbourhood (inner circle), its second neighbourhood (outer circle) and its local community (nodes in bold).

(*NeigSize*) is large, ranging from 1 to 4,644 with a mean of 669. The second neighbourhood size (*Neig2Size*) is even larger (4 to 423,089 with a mean of 79,702). Finally, the sizes of global communities (*LouvainSize*) are as usual very heterogeneous, ranging from 2 to 511,457 with a mean of 359,625. Thus, the local communities are by far the smallest sets of nodes that we will consider. This leads to a faster algorithm and allows us to focus on the most important neighbours.

In this application, the churn problem consists, starting from a set of active users by the end of March, to observe their activity during the month of April in order to predict which users will churn in May. The objective is to show the contribution of the (local) community of a node in churn prediction. This is presented in the next section.

5.2.3 Experimentations and results

Various models, based on different subsets of attributes, were built. *Infiniteinsight*, a tool provided by KXEN (`www.kxen.com`) (that is known to be very efficient [25, 46]) is used to identify the most relevant attributes, amongst the attributes enumerated above. This is done using their contributions to the underlying model. In this context, the output of Infiniteinsight is a ranking of the attributes considered as explanatory variables. Some classical ranking tools for logistic regression are also used to confirm this ranking and for reproducibility purposes.

Table 5.2: Evaluation with Support Vector Classifiers

Attributes sets	Avg #nodes used	AUC
All	431,978	0.855
All without Louvain's global community	72,353	0.854
Node & local community	21	0.832
SPA method	-	0.829
Node & second Neighborhood	71,734	0.826
Node & first Neighborhood	598	0.824
Node & Louvain's global community	359,625	0.823
Node only	1	0.815

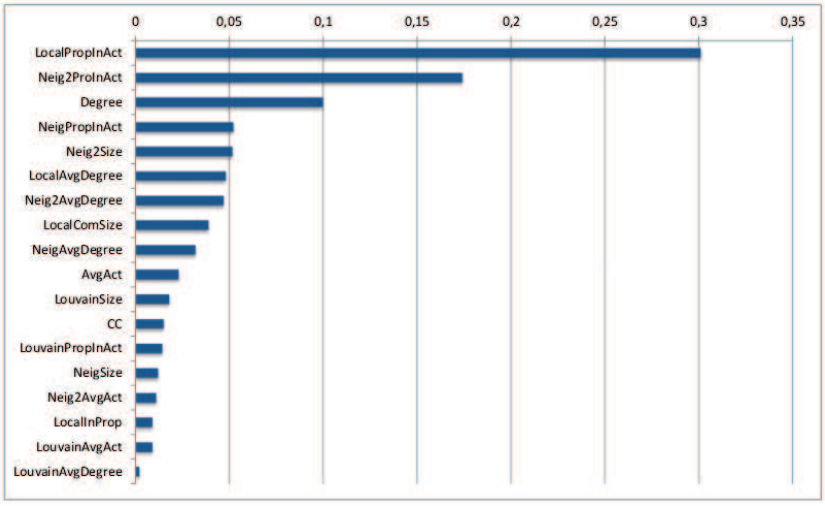


Figure 5.5: Variable contributions to the K2C/K2R model

Experiments with various attributes sets

All attributes are numerical and have been normalized between -1 and 1. The experiments are based on standard Support Vector Machines - a well established statistical classifier. The ***LIBSVM*** software [23] is used.

The Radius Basis Function (RBF) kernel has been chosen. This kernel non-linearly maps samples into a higher dimensional space and allows to build non-linear decision frontiers. The linear kernel is a special case of RBF [75]. The parameters C and γ of the model have been selected using grid search optimisation technique.

To train the different models, the following sets of attributes were used:

1. all the attributes;

2. the node attributes;
3. the node attributes and the first neighborhood attributes;
4. the node attributes and the second neighborhood attributes;
5. the node attributes and the local community attributes;
6. the node attributes and the Louvain community attributes;
7. all the attributes except Louvain community attributes;
8. the first neighborhood attributes;
9. the node second neighborhood attributes;
10. the local community attributes;
11. the Louvain community attributes.

The models were trained and then applied to the test set in order to compute the Area Under ROC Curve (AUC) was computed. For sake of comparison, the AUC obtained with the method by Dasgusta et al. (SPA) [37] is also displayed.

The Receiver Operating Characteristic (ROC) curve is a plot of the *true positive rate* against the *false positive rate* for the different possible *decision thresholds*. The area under this curve represents how well the model can separate the two classes.

Table 5.2 presents the results of the experiments using the above performance indicators. The results are ranked by the value of AUC which is the most relevant performance indicator because the classes are unbalanced.

One can observe from table 5.2 that the best model is obtained by using all the attributes. This illustrates the well known robustness of SVM models, which are able to cope with a lot of variables. The second model is the one with all variables except those extracted from global communities. This is not a surprise, because global communities are too large for this application and thus are not able to capture the local dynamics of the network. The local circle that gives the best prediction is the local community computed with IOLoCo.

The second most efficient model (bolded in table 5.2) is based only on the attributes computed from the user and its local community. On average, this model considers the state of 21 nodes, while the average degree of the nodes is 669: the local community allows to focus on the most important neighbours.

In an application where local communities are incrementally computed and maintained, this leads to very efficient estimators.

The relevance of the local community is confirmed by the observation that when the nodes attributes are not considered, the best performance is achieved using the local community only. This shows clearly that the most important contribution to churn prediction is provided by the local community.

One can observe in table 5.2 that the method by Dasgupta et al. (SPA) [37] has a performance that is close to what is obtained with the local community. However, contrary to SPA, the method proposed here can easily take into account non graph-based attributes such as age, gender, profession and salary. in the statistical model. Moreover, the approach proposed in this chapter is more flexible and modular with a clear separation between the computation of the local community and the estimation of the outcome: both the community detection algorithm and the statistical model could be replaced by other versions adapted to specific applications.

Finally, the second neighbourhood leads to a better AUC than the first one. However, it still leads to an algorithm that is less accurate and much slower than the one based on the local community.

Ranking of attributes

The second experiment uses InfiniteInsight, a tool developed by KXEN (a data mining company). The K2C/K2R (Kxen Consistent Coder and Robust Regression) modules are able to select the most relevant attributes and compute their contribution to the model. With these modules, one can built a model using all the attributes except those related to the activity of the node itself. The aim of this test is to identify the topological attributes that have the most important contribution. The results are shown in figure 5.5. It can be seen from figure 5.5 that the most relevant topological attribute for churn prediction is *PropInact*: the proportion of inactive members of the local community. This result reinforces the intuition that nodes are highly influenced by the behaviours of local community members: the most important explicative factor of churn is the number of friends in the local community that churned during the period of observation. This generalizes the result of [37] where it was shown that the number of friends (neighbours with strong links) that have churned, has a great impact on churn prediction.

Figure 5.5 also shows that the second most relevant topological attribute is the proportion of nodes that are inactive in the second neighbourhood (*Neig2InactProp*). This is consistent with the previous test : after the local community, the second neighbourhood produces the most relevant attributes.

Attribute	Coefficient	Rank
LocalPropInact	**7.96**	**1**
Activity	-7.31	2
Neig2PropInact.	4.34	3
LocalAvgAct.	-4.05	4
DaysAfterLast	3.15	5
Neig2AvgDegree	1.92	6
Neig2size	-1.82	7
NeigAvgAct.	1.58	8
LouvainPropInact	-1.45	9
LouvainAvgAct.	-0.86	10

Table 5.3: Ranking of attributes (only the top ten) using the absolute value of predicted coefficients of the linear regression.

Because InfiniteInsight is a commercial tool, some more classical ranking of variable are presented. For that purpose, a linear regression model is built using all the attributes (after normalization). The AUC of this model is 0.84, which very close to the result of the SVM model. The first ranking method is simply by using the absolute value of the estimated coefficients of the logistic regression. The coefficient of the logistic regression are given in table 5.3. Although this ranking is not the same as the one produced with InfiniteInsight, it is coherent in the fact that the attribute *PropInact* is still ranked first. The attribute Neig2PropInact. also has a good ranking with the third position.

The second considered approach is to rank predictors by the probability of the Wald chi-square test [63], $H_0 : \beta_i = 0$; the null hypothesis is that there is no association between the predictor i and the outcome after taking into account the other predictors in the model. Small p-value indicates that the null hypothesis should be rejected, meaning that there is evidence of a non-zero association. This metric only indicates the strength of evidence that there is some association, not the magnitude of the association. Thus the ranking should be interpreted as a ranking in terms of strength of evidence of non-zero association. This ranking is presented in figure 5.4. Here also the attribute *LocalPropInact.* rank first and the attribute *Neig2PropInact.* rank third.

Using a consensus from this two rankings, the two most relevant attributes are *LocalPropInact.* and *Neig2PropInact.* as suggested by the ranking performed by InfiniteInsight.

Attribute	p-value	Rank
LocalPropInact	**< 2e-16**	**1**
DaysAfterLast	**< 2e-16**	**1**
Neig2PropInact.	2.98e-16	3
Neig2size	1.17e-13	4
Neig2AvgDegree	8.63e-13	5
LocalInProp.	3.21e-06	6
Degree	4.36e-05	7
NeigAvgAct.	0.00100	8
NeigPropInact.	0.00392	9
LouvainPropInact	0.01183	10

Table 5.4: Ranking of attributes (only the top ten) using the p-value of the Wald Chi-Square Test.

5.3 Recommendation

The second application of local community analysis discussed here is recommendation. Recommender Systems (RSs) help users deal with information overload by proposing to them items suited to their interests. The aim of this section is to show that the local communities can provide a good model for the notion of *close friends* in the context of recommendation.

5.3.1 Introduction to recommender systems

The history of RSs started in the late 1990s with work by the GroupLens team at University of Minnesota [43] to recommend news and by MovieLens in 1996 to recommend movies, which demonstrated that automated recommendations were very well received by users. Then Amazon, which had been incorporated in 1994, published its patent in 2001 and has been serving recommendations ever since, acting as a de facto reference showcase for the efficiency of RSs [85]. The first papers on collaborative filtering showed how to use the opinions of similar users to recommend items to the active user [3]. Since then, research in RSs has become very active (see for example a recent RS survey including more than 250 references [43]) and RSs have been successfully used in many industry sectors to recommend items: movies (Netflix [17], MovieLens[98]), products (Amazon.com [85] , La Boîte à Outils [125]), songs [4], jobs to Facebook users (Work4 Labs.com [39]), friends, banners or content on a social site (Skyrock.com [114]) etc.

Recommender systems can be classified into three main classes: content

based, collaborative filtering, and hybrid systems [118] which combine the two previous ones. In content based recommender systems [86] , each user and each product is described with a set of attributes. A similarity measure is then used to compare each user to each product and each user gets the N items that are most similar to is profile as recommendations. The main problem of this approach is to describe all the items and the user. Because this description is very difficult to collect, it is usually incomplete and error prone [39].

In collaborative filtering recommendation systems [139] a user gets recommendations based on what other users have already chosen. Two main classes of collaborative filtering systems exist: user-based and item-based. In user-based collaborative filtering, a given user u get recommendation from the users that are most similar to him/her. In item-based collaborative filtering, a user u get recommendations from items that are more similar than the items he/her has already purchased. In the particular case of user based collaborative filtering, an important question is how to define the similarity between users in order to perform the recommendations. Given a rating matrix R with $R(u,p) = r$ if user u has given the rating r to the product p the following similarity functions can be defined between users:

$$Cos(u,v) = \frac{R(u).R(v)}{||R(u)||||R(v)||} \tag{5.1}$$

$$Pearson(u,v) = \frac{Cov(R(u),R(v))}{\sigma_{R(u)}\sigma_{R(v)}} \tag{5.2}$$

One problem of user-based collaborative filtering using these functions, usually referred as *cold start*[140] is that new users cannot get recommendations because they are most similar with other people without any products. If the content is available, a content-based or an hybrid method could help solve this problem. Otherwise, one can solve this problem using social recommendation as discussed in the following sub-section.

5.3.2 Proposed Social recommendation model

Social recommendations are recommendations that rely on one's social connections in order to make personalized recommendations of ads, content, products, and people [92]. The main question here is how to choose the social connection to rely on for recommendation. As for the churn problem described above, many social circles can be considered.

Users belonging to the local community of a particular user u are expected to provide goods recommendations to him. Moreover, if a member v of the

local community of u if connected to u with a shorter path than another member w, then v is expected to be more reliable for u than w. These two hypothesis form the basis of our proposed local community-based social recommendation score. For user u and an item i, this score produces a value by computing a weighted average of all the rating, for item i, for all users belonging to the local community of u. The aim of this weighting average is to make more close friends in the community participate more. More formally, the score for a user u on an item i is then given by:

$$LComScore(u,i) = \frac{\sum_{v \in LC(u)} R(v,i) \times f(d(u,v))}{\sum_{v \in LC(u)} f(d(u,v))} \tag{5.3}$$

$LC(u)$ is the local community of node u, $(d(u,v))$ is the distance between nodes u and v, in terms of shortest paths and f is a user defined function to weight the distance. Two particular choices for f are : the constant function $f_1(x) = 1$ and the function f_2 such that $f_2(x) = 1$ if $d(u,v) = 1$ and 0 otherwise.

5.3.3 Evaluation

Datasets

To demonstrate the usefulness of our method, we present some results on various datasets traditionally used in the literature. These datasets are characterized by the number of users, items, preferences (implicit or explicit) and existing explicit social relationships:

- Lastfm: this dataset contains 92 834 listening information of 17,632 artists by 1,892 users of lastfm.com. There is an explicit friends' network with 25,434 links.
- Flixster: this dataset contains 8,2 M ratings of about 49,000 movies by about 1 M users. There is an explicit friends' network with 26.7 M links.

Evaluation metrics

To evaluate whether recommended items were adequate for the user; for example, recommended items were later consumed. We thus have a target set for each user which represents the set of items he consumed after being recommended. This can be implemented by splitting the available dataset into Training / Testing subsets (taking into account time stamps if available). In this case, metrics are those classically used in information retrieval:

- *Recall@k* and *Precision@k* are defined as:

$$Recall@k = \frac{1}{L} \sum_{a} \frac{|R_a \cap T_a|}{|T_a|} \tag{5.4}$$

$$Precision@k = \frac{1}{L} \sum_{a} \frac{|R_a \cap T_a|}{k} \tag{5.5}$$

  where $R_a = \{i_1^a, i_2^a, ..., i_k^a\}$ is the set of k items recommended to a, and T_a is the target set for a.

- $F\beta - measure$: $F\beta$ is designed to take into account both recall and precision; $F1$ is the most commonly used. $F\beta$ is defined as:

$$F\beta = \frac{1 + \beta^2 \times Precision@k \times Recall@k}{\beta^2 \times Precision@k + Recall@k} \tag{5.6}$$

- *MAP@k* (Mean Average Precision) was used, for example, in the Million Song Dataset challenge(`http://kaggle.com/c/msdchallenge`) ; it is defined as:

$$MAP@k = \frac{1}{L} \sum_{a=1}^{L} \frac{1}{k} \sum_{i=1}^{k} \frac{P_{ai}}{i} \times 1_{ai} \tag{5.7}$$

  where P_{ai} is the number of correct recommendations to user a in the first i recommendations (*precision@i* for user a) and $1_{ai} = 1$ if item at rank i is correct (for user a), 0 otherwise.

Results and discussion

The evaluation of the proposed local community based model for social recommendation is presented in table 5.5. In that table, *LCR* stands for local community recommender and *WLCR* stands for weighted local community recommender. In that weighted version we have used the f_2 weighting function defines above. All results are the mean for all the users. As one can see in table 5.5, *WLCR* gives the best results with the smallest set of users. Although it is possible to use other weighting functions, our objective was to show that local communities are adapted to model *close friends*.

@10	Flixster				lastfm			
Meth./Ind.	Size	MAP	Prec	Recall	Size	MAP	Prec	Recall
1st circle	32.93	0.025	0.025	0.025	26.88	0.015	0.008	0.0212
LCR	**6.32**	0.023	0.024	0.024	**5.92**	0.013	0.007	0.0186
WLCR	**6.32**	**0.030**	**0.029**	**0.032**	**5.92**	**0.016**	**0.011**	**0.0256**

Table 5.5: Evaluation of social recommendation

5.4 Conclusion and discussions

In this chapter, we have studied some users' behaviours prediction problems in social networks. Local communities, quickly and accurately computable, can be used to extract attributes that are relevant for those problems.

In the next chapter some methods to predict the dynamic of local communities will be presented. Indeed, the analysis of the dynamics of local communities can clearly help to better understand the users' interactions and future behaviours.

CHAPTER 6

Communities in dynamic networks

"The dynamic of a relationship changes when one person gets sober."

- Trent Reznor

6.1 Introduction

Social Networks are dynamics by nature: new nodes arrive, existing nodes leave. This is also true for links. Community detection methods in complex networks have for long only considered the static aspect of networks: a snapshot of the graph is taken at a particular time and the communities are computed. Recently, many works on community detection in dynamic networks have started. Some authors try to follow the evolution of communities at different time-steps [120, 146, 7], other dynamically update the existing communities with the new events (creation or deletion of nodes/links) [117]. Finally, the last class of methods try to discover communities which are consistent in many time-steps [9].

One problem actually unexplored is the prediction of community dynamics: knowing the evolution of the network until the time-step t, can we predict the communities at time-step $t+1$?

In this chapter, we propose two general approaches for communities prediction. The first approach consists in directly predicting whether or not a particular node will be member of a community. This is done by computing some local attributes and then constructing a machine learning model for prediction.

The second method is based on interaction prediction. In this approach, given the evolution of the network until time-step t, the interactions are predicted for time-step $t+1$ and the communities are computed in the predicted network. The assumption behind this approach is the following: if one is able

to predict the structure of the network with a high accuracy, he just needs to compute the communities on that predicted network to have the prediction for the communities.

Although many methods for links prediction exist [88], they generally consider that the network is growing: new links are created but existing ones remain forever. However, in many real social networks, new links are created and existing links are removed. For example, in a call graph, the contacts a particular person calls evolve with time: he stops calling some of them (e.g. ex girlfriend) and start calling others (e.g. new course mate). The interaction prediction problem considered in our second approach is then more general than link prediction.

This chapter is organized a follow: Section 1 presents different approaches for community detection in dynamic networks. Section 2 presents our attempt to directly predict the members of a community . Section 3 presents the proposed models for interactions prediction in social networks and how to apply them to communities prediction. The performances are evaluated and discussed in section 4. Finally, section 5 draws some conclusions and perspectives.

Contents

6.1 Introduction 81
6.2 Dynamic communities 83
6.2.1 Community tracking 83
6.2.2 Community updating 84
6.2.3 Long term-communities detection 85
6.3 The link prediction problem 86
6.3.1 Probabilistic methods 87
6.3.2 Transitivity-based methods 88
6.3.3 Attributes-based methods 90
6.4 A supervised method for local Community prediction 91
6.5 Community prediction through interactions prediction 92
6.5.1 Interaction prediction in complex networks 92
6.6 Evaluation and discussion 96
6.6.1 Dataset Description 96
6.6.2 Evaluation of interaction prediction 98

6.6.3 Local community prediction evaluation 100
6.7 Conclusions and perspectives 102

6.2 Dynamic communities

There are two different dynamics that can be considered in networks: interaction networks or evolving networks. In interaction networks, only the events that have taken place during a time-step form the links of this period and only the corresponding nodes are considered. For example, in a scientific collaboration network, if the time-step is one month, only collaboration during each month are considered. Conversely, in evolving networks, new events are just appended to the existing network.

When talking about community detection in dynamic networks, there is actually no consensus and formal definition of what is a community. The proposed methods also take into account one or both of the above presented dynamic networks.

We present here some methods for communities tracking, communities updating, long-term communities detection.

6.2.1 Community tracking

The general idea behind this class of methods is that one can compute the communities in each time-step independently and match them for each pair of consecutive time-steps. Between two consecutive time-steps, for each community, the following events are possible: continuation, fusion, division, birth, death. The methods falling in this category differ on the one hand by the choice of the algorithm used to detect the communities in each time-step, and by the matching method used between the time-steps on the other hand.

Palla et al. [120], for example, have proposed to use the clique percolation method [119] to detect the communities in each time-step. For the matching between two consecutive time-steps of an interaction network, these authors first compute the communities in the *union graph*. Due to the properties of the clique percolation method, each community present in at least one of the two consecutive time-steps is contained in exactly one community of the *union graph* (see [120] for justifications). The matching is finally done as following:

- if a community of the *union graph* contains only one community of the time-step t and one community of the time-step $t+1$ then it is a *continuation* event.

- if a community of the *union graph* contains more than one community for one of the two consecutive time-steps then the communities are matched with the number of common nodes descending.

The principal limitation of this method is that it is based on the particular properties of the clique percolation method and can not be directly applied to other community detection methods.

Greene et al. [56] have proposed a more general method which can be used with any static community detection algorithm. These authors proposed to first use a static community detection method at each time-step and then to match them using the Jaccard similarity [133]. However, this method is not suitable for non deterministic community detection methods because the result depends on the previous ones. This non determinism can be reduced using ensemble-based community detection methods as described in [141]. Although this method can be applied to both types of dynamic networks, experimentation by the authors have been performed in evolving networks.

Tantipathananandh et al. [146] have defined the tracking problem as a graph colouring problem. Despite the nice formulation of the problem, the solution proposed is computationally expensive.

6.2.2 Community updating

The basic idea of this class of methods can be described as follows: detect the communities at a reference time-step t_0. For each following step, update the community structure obtained at the precedent step with the elementary events that are produced during the two consecutive time-steps. The elementary events that can be considered are:

- a new link is added;
- an existing link is deleted;
- a new node with k links are added;
- an existing node with k links are deleted.

Methods from this category consider evolving networks only.

In their works, Nguyen et al. [117] have proposed to manage the elementary events as follows:

- a new link is added: if the link is internal to a community, it reinforces the cohesion and the community structure remain the same. Else, one need to check if the new edge can initiate the fusion of the two communities it crosses.

- an existing link is deleted: if the link crosses two communities, it reinforces the separation between them and the community structure remain the same. Else, one need to check if the new edge can initiate the split of the community it belongs to.

- a new node u with k links are added: if the node has no link then a new community is created containing only the node u. Else the node u comes with one or more links connecting one or more communities. In that case, for each community c the new node is connected to, a local computation is performed to obtain the gain in the quality function. u is then moved to the community that produces the maximal gain.

- an existing node with k links are deleted: when a node u is removed from a community C it can divide C into many communities. The community structure is adapted as follows: a *3-clique* (triangle) is chosen in the neighborhood of u. Starting with this triangle, a *3-clique community* is built using the clique percolation method [119]. All the nodes in this *3-clique community* are kept in C . The other nodes (members of C not in the *3-clique community*) are examined one by one to choose the community they must join.

In another work, Cazabet et al. [22] have proposed to start with an empty network and, at each step, the new created edges (ordered by their creation time, breaking ties randomly) are added to the network one by one (the connecting nodes are created if they do not already exist) and the communities are updated.

6.2.3 Long term-communities detection

This last class of methods try to detect communities which are consistent in many time-steps i.e. that are present in several of these time-steps. These methods are applicable to both type of dynamic networks.

Aynaud et al. [9] have proposed two methods based on the *modularity* [111]. The first method consists in building a *sum network* and then apply a static community detection method on that network. Alghtout any static-community detection can be used, for their experimentations, these authors have used the Louvain method for community detection [18]. The second method consists in defining a *mean value for the modularity* on all the time-steps. This mean value is then optimized with a procedure similar to the Louvains' method[18].

Mitra et al. have proposed a method designed for *citation networks*. This method consists in first building a *summary network* as follows:

- a node A_i is created if author A has published a paper at time i.

- a link is created between the nodes A_i and B_j if and only if the paper published by author A at time i cites the paper published by author B at time j.

A static community detection method can then be used to mine the community structure in this constructed network.

The principal drawback of these methods is the lack of evaluation methods for the results.

6.3 The link prediction problem

Given a snapshot of a social network at time t, the link prediction problem is to accurately predict the edges that will be added to the network from time t to a given future time t'. The link prediction problem therefore tackles the following question: to what extent can the evolution of a social network be modelled using features intrinsic to the network itself [84]? Formally, consider a network $G = (V, E)$ where V is the set of vertices and E is the set of links. The set of edges $(u, v) \subseteq V$ with $u \neq v$, that are absent in E is denoted $\overline{E}$. In a practical application, $\overline{E}$ can be divided into two parts: the set E' of links that will appear in the future, also called missing links, and the set E'' of edges that will never appear. Clearly, $E' \cup E'' = \overline{E}$ and $E' \cap E'' = \emptyset$. The challenge of link prediction is to produce quickly, accurate approximations for E', even for huge social networks.

Link prediction is a very active research area because of its wide range of applications. For instance, if G is a social network representing recorded interactions between terrorists, the link prediction can be used to detect underground relationships between them. On the other hand, a link prediction algorithm can be applied to a clients/products network produced by an e-commerce platform, to suggest products that a client is likely to purchase in the near future. Other algorithms and applications related to link prediction in complex networks can be found in [89].

We now present some basic link prediction algorithms according to the following nomenclature: probabilistic methods, transitivity-based methods and attributes-based methods. After that we introduce an extension of the link prediction problem to dynamic networks.

6.3.1 Probabilistic methods

The most naive probabilistic model of link prediction is the Random predictor which randomly chooses a subset of links that are not present in the network, and predicts them. Since the subset selection is done randomly, the accuracy of the algorithm is based on luck. The probability of failure of the Random predictor is $\frac{1}{2}$ for a given link. This method can't be taken seriously when dealing with an application. It only serves as reference point: any serious algorithm must have a better accuracy.

The probabilistic approaches can nevertheless be useful when there is a prior knowledge on the problem. For instance, many complex natural and social systems assemble and evolve through the addition and removal of nodes and links. This dynamics often appears to be a self organizing mechanism governed by evolutionary laws that lead to some common topological features. One of such features is the power-law degree distribution, i.e. the probability that a node has degree k is $P(k) = k^{-\gamma}$, usually with $2 < \gamma \leq 3$. Such networks are said to be scale-free. For such networks, the "preferential attachment principle" states as follows: when a new node is added to the network with m edges that link this new node to m nodes already present, the probability that this new node will connect to a node i with degree d_i is proportional to d_i i.e $\pi(d_i) = \frac{d_i}{(\sum_i d_i)}$. It can be shown that a network evolving according to this principle, tends to a scale-invariant state with $\gamma = 3$ [109]. Clearly, such a model of network growth constitutes an a priori information that can help to design efficient link prediction algorithms. The preferential attachment principle is also known in economy as cumulative advantage: the rich gets richer [14, 142].

The preferential attachment is a good illustration of Zhu and Kinzel's observation. Indeed, it gives the worst performance when applied to physical internet networks where high degree nodes are routers that have a very low probability of being connected by new physical lines.

Recently, Freno et al. [49] have proposed a new approach that is not based on parametric assumptions concerning the modelled distributions. More precisely, they have introduced the Fielder random field model, called Fielder delta statistic that, for each binary edge variable $X_{u,v}$, defines a potential that encapsulates the measure of its role in determining the connectivity of its neighbourhood. The trick is that these potentials can be estimated from data by minimizing a suitable objective function. Experiments on some real-world networks have resulted in link prediction algorithms that outperform the solutions proposed by Watts-Strogatz [151] and Barabasi-Albert [14]. Other probabilistic methods for link prediction are reported in [89].

6.3.2 Transitivity-based methods

In mathematics, a binary relation $\Re$ defined on a domain D is said to be transitive if whenever u is in relation with $v (u \Re v)$ and v is in relation with $w (v \Re w)$, then u is in relation with $w (u \Re w)$.

In topological transitivity applied to a complex network $G = (V, E)$, the domain D consists of the set V of nodes of the network, and the relation $\Re$ is represented by the set E of edges. The application of topological transitivity to link prediction is based on the assumption that, as a complex network evolves, it tends to become transitive i.e: if at time t, u is related to v and v is related to w, then there is a high probability that at a future time t', u will be related to w. This assumption follows from a common observation made for instance on friendship networks: a friend of your friend is likely to be or become your friend. This corresponds to triangles in G, i.e triples of edges (u, v), (v, w) and (u, w). In graph-theoretic terms, the degree of transitivity of a network G can be measured by the so-called clustering coefficient [151]:

$$C = \frac{\sum_{u \in V} C_u}{|V|} \tag{6.1}$$

where

$$C_u = \frac{number\ of\ triangles\ connected\ to\ vertex\ u}{number\ of\ triples\ centred\ on\ vertex\ u} \tag{6.2}$$

As reported in [104], this coefficient has remarkable values for many current networks: greater than 0.75 for film actors and power grids; between 0.6 and 0.74 for biology co-authorship, train routes and metabolic networks; between 0.30 and 0.59 for math co-authorship, Internet and word co-occurrences in web pages, and less than 0.20 for email messages and fresh water food web.

The basic link prediction methods based on topological transitivity, use some local or global properties of the network G, to assign a connection weight $Score(u, v)$, to pairs of nodes (u, v) of V. All non-observed links are then ranked in decreasing order of $Score(u, v)$. In this approach, links with the highest scores are supposed to be of higher existence likelihoods and are produced by the algorithm. Such a measure must reflect the proximity or similarity between nodes u and v. The problem is to design good similarity measures.

Let us denote $\Gamma(u)$ the set of neighbours of node u, and let $|A|$ be the cardinality of a set A. $CN(u, v) = |\Gamma(u) \cap \Gamma(v)|$ corresponds to the number of common neighbours of u and v[103]. The idea is that if u and v have many neighbours in common, then there is a high probability that they will become neighbours in the future. The efficiency of this measure has been experienced with collaborative networks [104]. However, this measure suffers from serious

drawbacks. For instance, in a friendship network, the fact that two nodes u and w have a common very popular neighbour v, i.e. with a very high degree d_v, does not necessarily mean that u and w will become friends in the future. They may even be from different continents and never meet. In the same way, in an allocation network, if node u sends a unit of resource to a very popular neighbour v that serves as intermediary, and if node v subdivides the resource and sends equal parts to his neighbours, then the portion received by any neighbour $w \in \Gamma(v)$ will be $\frac{1}{d_v}$. This means that the contribution of an intermediate node v for the "future connection" between u and w is divided by the degree of v. This has motived some authors to introduce $RA(u,v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{d_w}$ [156] and the log form $AA(u,v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log(d_w)}$ [2]. Many other variants of CN have been proposed, but extensive experiments on real world networks have shown that RA is the best whereas CN is the second best.

A link prediction method based on topological transitivity has been introduced by Latapy et al.[5]. Consider a bipartite clients/products network $G = (\perp, \top, E)$ where $\perp$ is the set of clients, $\top$ the set of products and E the set of purchases. The $\perp$-projection of G is the graph $G_\perp = (\perp, E_\perp)$ in which $(u,v) \in E_\perp$ if u and v have at least s neighbours in common in G, where s is a given threshold, i.e. $|\Gamma(u) \cap \Gamma(v)| \geq s$. The underlying intuition of the internal link prediction method is that, in a clients/products network, if two clients have bought in the past many common products, then they will probably acquire new common products in the future. This method falls within the transitivity framework as follows: if client A is related to client B in $G_\perp$ and if client B is related to product p in G, then there is high probability for A to be related to p in the future.

Another topological transitivity measure for link prediction is based on random walks already introduced in section 3.2.1 for community detection. In the simplest version of this method, it is assumed that, when a random walker is at node u, it can go in one step to any node $v \in \Gamma(u)$ with probability $\frac{1}{d_u}$. Let $m(u,v)$ denote the average number of steps necessary for a random walker to go from u to v. The commute time is the symmetrical measure $CT(u,v) = m(u,v) + m(v,u)$. This transitivity measure is then used to predict missing links: the smaller $CT(u,v)$ is, the greater is the probability for u and v to establish a connection in the future.

Association rules originally defined for large databases of sales transactions can be adapted for link prediction on a network $G = (V,E)$. Consider $D = \{\Gamma(u) | u \in V\}$. Define frequent groups of nodes as subsets that are included in at least s elements of D, where s is a given threshold. An association rule is an implication of the form $A \rightarrow B$, where $A \subseteq V$, $B \subseteq V$, $A \cap B = \emptyset$, and $A \cup B$ is frequent. A rule $A \rightarrow B$ holds with confidence c if c

percent of neighbourhoods in D that contain A also contain B. Hereafter, we denote $A \to B : c$. The transitivity principle states as follows: if $A \to B : c$ and $B \to C : c'$ then, $A \to C : c \times c'$. In the context of an application to co-authorship[70]: A, B and C are sets of co-authors. $A \to B : c$ means that c percent of articles co-authored by A are also co-authored by B, and $B \to C : c'$ means that c' percent of articles co-authored by B are also co-authored by C. As a consequence, if $A \to B : c$ and $B \to C : c'$ are observed, then $A \to C$ is predicted with probability $c \times c'$(i.e a new article with $A \cup C$ as co-authors is predicted)[70].

6.3.3 Attributes-based methods

The great specificity for graphs that model social networks is that nodes and links usually have attributes. Consider a phone network in which a node represents a person and each link represents a call. Phone numbers can be used as node attributes and the average number of calls between nodes can be used as link attributes.

The link prediction problem can be expressed as a classification problem for pairs (u, v). The following attributes may be considered when dealing with co-authorship networks [64]: the number of common neighbours $(\Gamma(u) \cap \Gamma(v))$, the number of common key words $(Kw(u) \cap Kw(v))$ or the total number of articles published by u and v. The class attribute is a binary variable with value 1 if the link will appear and 0 otherwise. All attributes values are normalised to have zero mean and one standard deviation. A classification model such as Decisions Tree(DT), Support Vector Machine(SVM) or Artificial Neural Network(ANN), can then be used. Hasan et al. [64] have shown on two networks(DBPL and BIOBASE) that SVM beats all the most used classification methods.

The similarity between two nodes can use attributes of nodes and links. This is the case for *Abstract* proposed in [70], which takes into account summaries of articles in the bipartite graph Authors/Articles. The idea is that articles already published contain information on topics that interest the co-authors. It is then natural to suppose that authors working in the same domain are more likely to collaborate and co-publish an article in the future. The attributes-based similarity between two u and v authors is then defined as:

$$score(u, v) = cos(V(u), V(v)) \tag{6.3}$$

where $V(u)$ is a descriptor that encapsulates the attributes for vertex u. It has been shown in [70] that this approach produce very good predictions for some well known co-authorship networks.

Link prediction is not sufficient to analyze the dynamics of communities in complex networks. Indeed, it supposes that all the created links will last forever. However, in a real interaction complex networks, the links between nodes come and leave. For example, in a collaboration network, a publication between two scientists in a particular year do not guarantee that they will still work together in the future. To really predict the dynamics of communities one needs a more general model for interaction prediction which determine whether or not a particular link will exist no matter if it has already appeared or not.

6.4 A supervised method for local Community prediction

In this section, a general method is described to predict whether or not a node belongs to a local community. The method first extracts some attributes from existing communities and then, a learning model is used for the prediction. More precisely, to predict the local community of a node u we proceed as follow: For each past time-step, the local community of u is computed and, for each node belonging to the local community or to its neigborhood (not in the community but connected to some member of the community), the following attributes are extracted :

1. the community size
2. the internal degree of the node
3. the external degree of the node
4. the position of the node in the community.
5. the distance to the starting node

The position can take three values: 1 if the node is in the community without any connection with the outside of the community, 2 if it is in the community but as some connections with the outside, and 3 if it is outside but with some connections with the community.

The set of candidates to be members of the local community of u is then taken from the set of nodes having already been in the community or in the neigborhood of the community in the past.

Given the candidate sets for each local community, a dataset is constructed and a machine learning model is learned given the true membership

for the training period. The experiments are based on Support Vector Machines and will be presented in the evaluation section.

6.5 Community prediction through interactions prediction

In this section we present the general framework for predicting the community using the prediction of interactions. This process is presented in figure 6.2 and the steps are presented in the rest of this section.

6.5.1 Interaction prediction in complex networks

In this subsection, the interaction prediction in complex network is first formalized and then the proposed similarity-based and supervised models are presented.

Problem definition

The interaction problem in complex networks can be stated as follows: given a dynamic network $G = (G_1, ..., G_n)$ what is the structure of the following snapshot (G_{n+1})? This problem is a generalization of the link prediction problem: here, not only the non-previously seen links are predicted but also the existent one to check whether or not they will still exist in the following snapshot.

The problem formalization is presented in figure 6.2.

As a generalization of the link prediction problem, the same class of solution can also be used to solve it. In the following, we present similarity-based and supervised solutions. On all these models, *time* plays an important role. Indeed, the history of interactions must be taken into account to accurately predict the interactions.

Similarity model

We start with a very simple model based on a similarity measure between two nodes, which will be used a baseline model for experimental comparisons. The similarity takes into account *time*, the *existing links* and the neighborhood of the pair of nodes into consideration. These equation gives a measure of the similarity between two nodes. The general form of this equation is:

$$Sim(i,j) = \sum_{t \in T} f(t) \times (\alpha A[i,j] + \beta g(neigh(i), neigh(j))) \qquad (6.4)$$

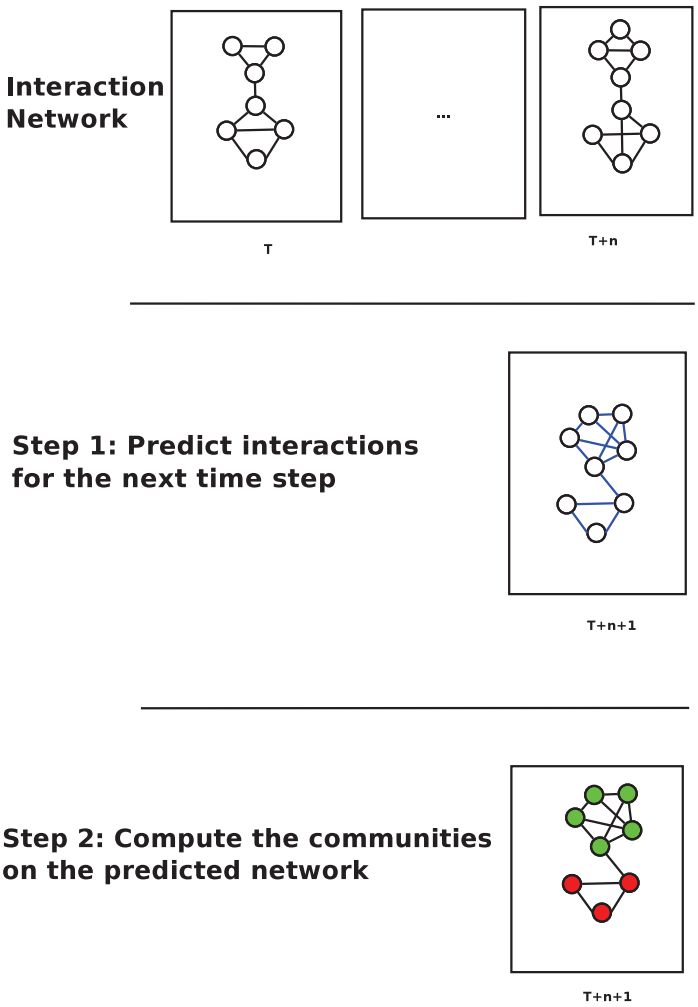


Figure 6.1: Interaction prediction problem definition.

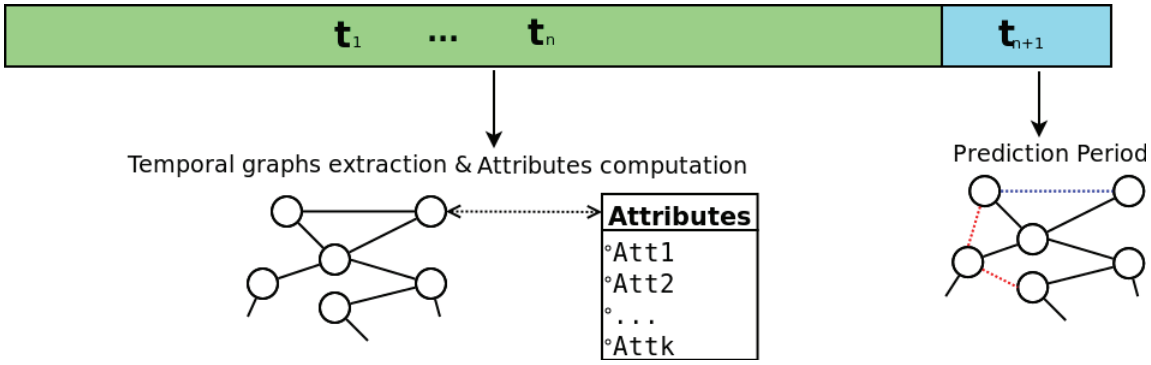


Figure 6.2: Interaction prediction problem definition.

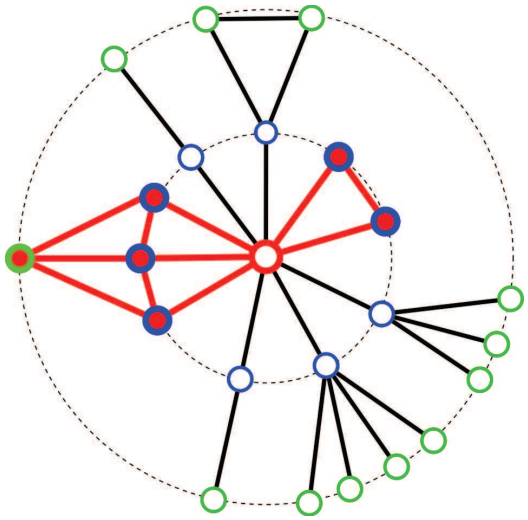


Figure 6.3: A simple example showing a node with its neighbors (first circle), second circle and local community (nodes and links filled in red).

In eq. 6.4, functions f and g and parameters α and β have to be defined by the user. The function f is a time function which enables to take into account the age of the relationship (give more importance for recent interactions for example) and g the topological similarity function which measures the proximity in the graph. A is the adjacency matrix, and $neigh(i)$ is a general neighborhood function. Some examples of neighborhood that can be considered are the first neighborhood (friends only), the second neighborhood (friends and their friends) and the local community. See Fig. 6.3 for an illustration.

The parameters of this model can easily be optimized using a random

restart Hill Climbing [135] method which can be described as follow:

1. Start at an arbitrary point
2. Calculate values for neighboring points
3. Move to the point with increased value
4. Terminate if no higher value could be found, otherwise continue at 2
5. Restart at 1 if the number of iterations is not reached.

The restart help prevent to stay at a local maximum during the optimisation. The parameters used in the evaluation section are obtained using this procedure.

As for the link prediction similarity-based models, this model is used as follows: the value of the similarity is computed for each pair of nodes and a threshold is chosen to decide whether or not the interaction should be created.

This model is quite intuitive. However, because the real relationship between inputs attributes and the target variable is not known, we propose in the following subsection a more general approach based on machine learning.

Supervised model

For the supervised approach we propose the following procedure: for each snapshot t of the training period, the following features are computed for each couple of nodes:

- the number of common neighbors
- the number of common community members
- a boolean indicating whether an interaction is present or not between the two nodes
- the attribute similarity between the two nodes (if available)

The real classes are obtained on the test period. It is worth nothing that to reduce the complexity (the number of possible interactions is in the order of $O(n^2)$), we only consider interactions that are likely to appear based on the computed similarity scores (topological or attribute based). Here also, a support vector machine is used to build the model.

This approach is more general because it does not suppose the form of the similarity function, and is more flexible in case we have other information attached to the nodes.

Community prediction

Given the predicted network using the methods proposed in the previous section, one has only to compute the communities on that network to get the prediction.

6.6 Evaluation and discussion

In this section, the datasets used are first described. After that, the evaluation of the interaction prediction model proposed is presented. Finally, the evaluation of the application to the communities prediction is presented.

6.6.1 Dataset Description

The datasets used to evaluate the proposed methods are *DBLP* and *Facebook Wall*.

DBLP dataset

DBLP is a collaboration network between authors indexed on http://dblp.uni-trier.de/. The time-steps are the years. The nodes are the authors and, for each year, a link exists between two nodes if the corresponding authors have at least one common publication for that year. The links are weighted by the number of common publications corresponding to the period of time (one year).

Figure 6.4 presents some statistics on the dynamics of the DBLP network. One can remark that the number of active authors (having at least one publication) follows a quasi-linear law (a). The same apply for the number of publications (b). One can also note that the number of new authors each year is quite high (c) and many authors do not publish in two consecutive years (d). Finally the number of authors that (re-)publish after a year off (e) and the average degree (f) are also presented. All these observations confirm that it is very difficult to take only one snapshot in order to predict the following one.

Facebook Wall dataset

Facebook Wall [1] is a network built with a small subset of posts to other users' wall on Facebook. The nodes of the network are from the New Orleans' Region. For each year, there is a link between two nodes if there is a wall publication between them. The links are weighted by the number of wall

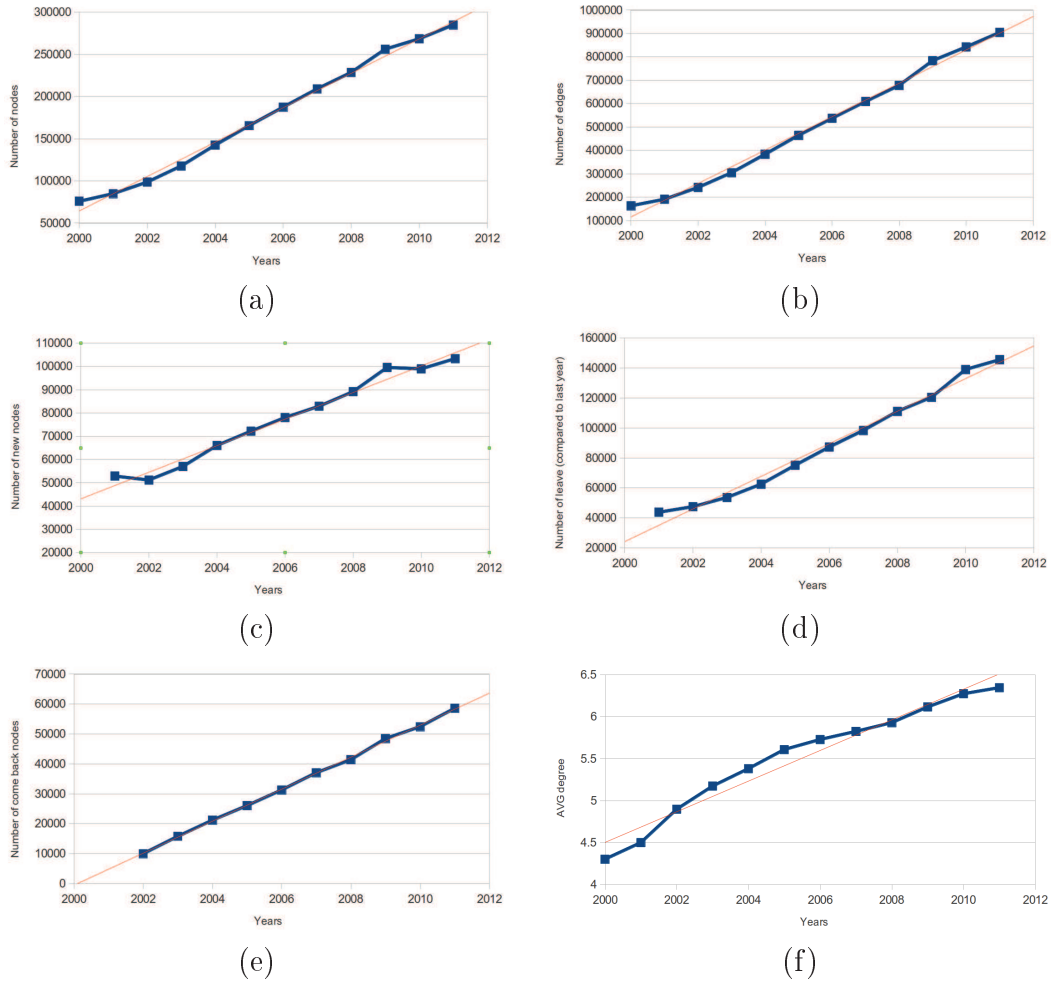


Figure 6.4: Some Statistics on DBLP dataset.

messages between them for the corresponding year. The direction of the link is not taken into account in our experiments.

As for the DBLP dataset, some statistics are presented in figure 6.5. Figure 6.5 (a) presents the evolution of the number of nodes, figure 6.5 (b) shows the evolution of the total number of interactions, figure 6.5 (c) let us see the dynamics of the number of new nodes, figure 6.5 (d) deals with the number of new interactions, figure 6.5 (e) presents the evolution of the average degree and finally figure 6.5 (f) illustrates the evolution of the number of new interactions between existing nodes.

One can see from these statistics that this network is very dynamic and that the number of nodes and edges grows very fast with time.

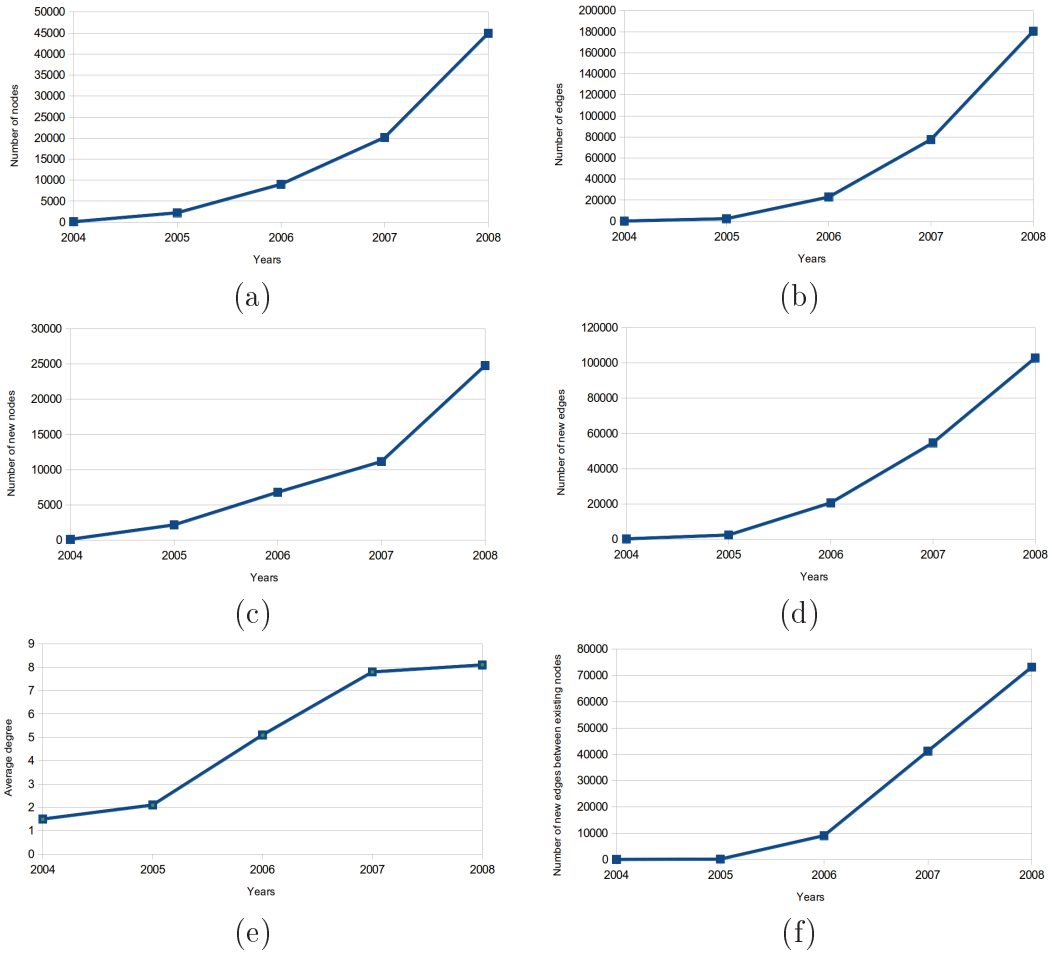


Figure 6.5: Some statistics on Facebook Walls dataset.

6.6.2 Evaluation of interaction prediction

true positive rate
decision thresholds

false positive rate

DBLP Dataset

The similarity-based model presented in section 6.5.1 is general. For each particular dataset, one has to find the most suitable parameters. We have used the random-restart hill climbing optimisation method[135] to set the parameters α and β. For the DBLP dataset, the instantiation of the model we have used is as follows:

$$Sim_{DBLP}(i,j) = \sum_{t \in T} (1/(n-t+1)) \times (0.67 \times ||A[i,j]|| + (0.33 \times J(C(i), C(j)))) \tag{6.5}$$

with n the number of time-steps, $||x||$ the normalization of the variable x between 0 and 1 and $J(C(i), C(j))$ the Jaccard similarity between the local community of i and the local community of j. The local communities are computed with the algorithm described in [115]. As shown in [116] local communities are the best compromise between the first and the second neighborhood.

This model produces an AUC of 0.69 which is already better than a random predictor which has an AUC of 0.5. Because the parameters are set manually, we are not sure that it is the best result we can achieve with this model. It is for that reason that the supervised model is used.

The supervised model for this dataset is built as described above with the same attributes involved in the similarity-based model. With this model, we get an AUC of 0.87. This model is much better than the previously constructed similarity-based one. We the used this last model for community prediction.

Facebook walls Dataset

As for the DBLP, we have used a particular instantiation of the model similarity-based model using the optimisation technique described above to set the parameters. It is stated as follows:

$$Sim_{Fcb}(i,j) = \sum_{t \in T} (1/(n-t+1)) \times (0.74 \times ||A[i,j]|| + (0.24 \times J(C(i), C(j)))) \tag{6.6}$$

This model produces an AUC of 0.84 which is also better than a random predictor. As stated above, this result is probably sub-optimal.

	DBLP dataset	Facebook wall dataset
Random predictor	0.50	0.50
similarity-based model	0.69	0.84
Supervised model	**0.87**	**0.92**

Table 6.1: Evaluation (AUC) of interaction prediction models

With the supervised model, we get an AUC of 0.92. This model is better than the previously constructed similarity-based one. It is then this last one that will be used for community prediction.

Table 6.1 summarizes the performances of our models for interaction prediction.

6.6.3 Local community prediction evaluation

Having a model which predicts quite well the future interactions, one can now apply it to local community prediction. for sake of comparison, the evaluation of directly predicting the membership of the nodes, with the supervised method presented in section 3 is also shown.

The performance index that we used to compare algorithms is the Normalized Mutual Information as presented in chapter 2.

The local community method used for our evaluations is our method, *IOLOCO*, presented in chapter 4. This algorithm locally optimize a defined quality function and is able to uncover the overlapping community structure a particular starting node belongs to and also detect when a node does not belong to a community. It is worth noting that the same can be done with any other community detection method.

For the evaluation, the local communities are computed on the real network and on the predicted one, based on the supervised model described above. The predicted network is constructed by keeping only the predicted interactions. The two results are then compared based on the NMI. This evaluation only take into account the local communities of nodes that appear in the predicted and the real network. The results are presented below for each dataset.

DBLP dataset

The supervised method to directly predict whether or not a node will belong to a community leads to an AUC of 0.87 and an accuracy of 86.10%. That leads to an average value for the NMI index of 0.32. This score is low because

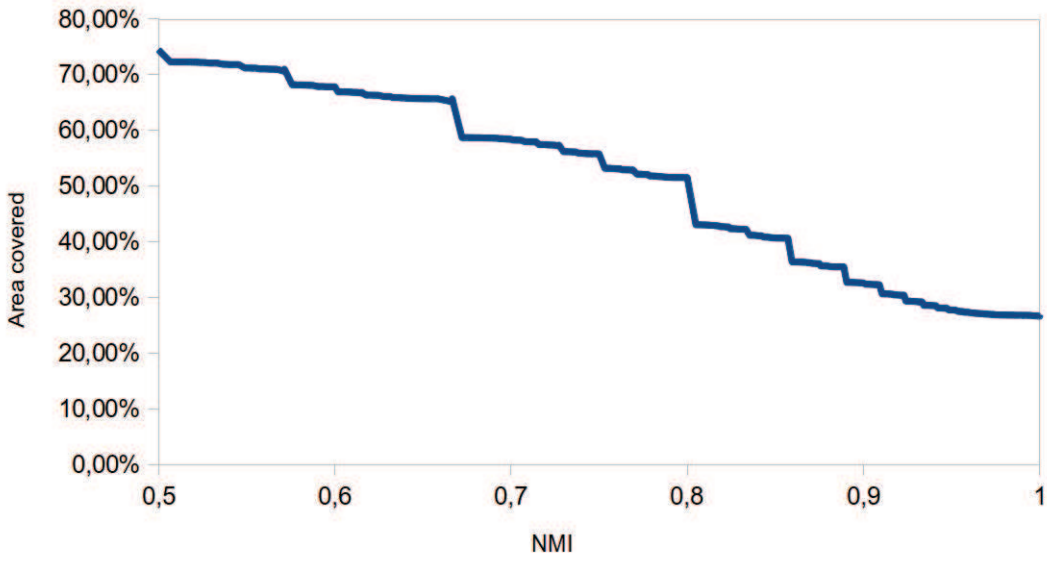


Figure 6.6: Evaluation of local community prediction for the DBLP dataset

many nodes that exist in the past do not exist in the future (if the author have not published again).

The results of the evaluation of the prediction of communities through the interactions prediction for this dataset are presented in Fig 6.6. In this figure, the area covered represents the percentage of nodes having a value of NMI equal or greater than the corresponding value of NMI on the x-axis. One can see that for more than 30% of the nodes, the prediction is perfect with a Jaccard index value of 1. More than 50% of nodes produce a score greater than or equal to 0.67 and more than 60% have their scores greater or equal to 0.5. The average value of NMI score is 0.65.

It worth noting that in this evaluation the nodes that only appear in the target period are not considered as starting nodes for evaluation because they cannot be predicted. However they are considered when computing the NMI of a particular local community.

Facebook walls dataset

The supervised method to directly predict whether or not a node will belong to a community leads to an AUC of 0.78 and an accuracy of 89.84. That leads to an average value for the NMI index of 0.45. This method is more suitable for this dataset because all the nodes existing in the past are present in the future.

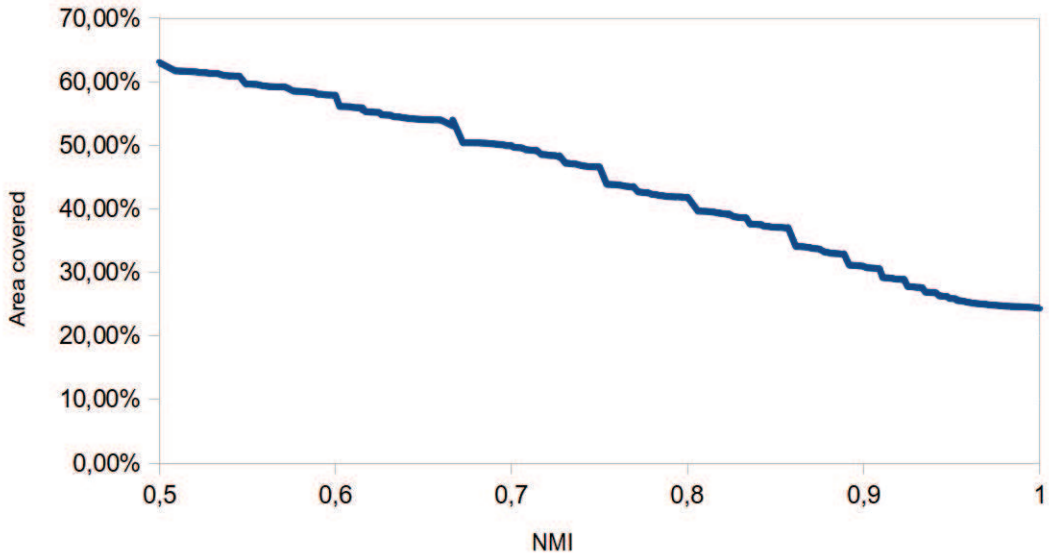


Figure 6.7: Evaluation of local community prediction for the Facebook wall dataset

As for the previous one, the results of the evaluation for the prediction of communities through the interactions prediction for this dataset are presented in fig 6.7. One can see that for approximately 25% of the nodes, the prediction is perfect with a Jaccard index value of 1. More than 50% of nodes produce a score greater than 0.5 and the average value of the index is 0.56.

6.7 Conclusions and perspectives

Recently, many works have started to deal with community detection in dynamic complex networks. To the best of our knowledge, community prediction in complex interaction networks is not actually studied in the complex network literature.

In this chapter, we have stated the community prediction problem and proposed two general approaches to solve it. The first approach tries to predict the membership of a node in a particular community based on a supervised learning model. The second approach first predicts the interactions and then computes the communities in the predicted network.

Directly predicting the memberships of nodes is more suitable for growing interaction networks where existing nodes do not leave the network. Predicting the interactions is more complex (and computationally expensive) than

directly predicting the memberships, but it gives better results because it does not require exact predictions: if a missing internal interaction in a community is predicted, it reinforces the cohesion. Likewise, if an inter community links is not predicted, it better separates the communities.

Tests on real datasets publicly available show the feasibility of the proposed approaches.

One direct perspective of this work is to improve this approach by adding a model for nodes evolution which will enable to predict the new nodes in the network. We will also test the approach on other datasets.

CHAPTER 7

Towards a Framework for storing and analysing distributed networks

"simplicity is the ultimate sophistication"

- Leonard de Vinci

7.1 Introduction

In the previous chapters we have proposed some methods to detect local communities and analyse their dynamics in large scale social networks. Although these methods can be implemented using existing social networks analysis tools, many of these tools usually consider that the data of these networks can be loaded into the main memory of one computer to process them. That is not always true in practice. Indeed, Online Social networks (OSN) like Facebook, Twitter or Skyrock usually generate huge amount of data. For that reason their data are generally distributed across many physical servers and eventually many datacenters around the world. Social network analysis (SNA) algorithms must take into account this distribution of data.

Besides, many distributed computing models for big data processing exist and can be used for SNA. However, they are (sometime) not straightforward in their usage and, as models, they are not always suitable for every real world needs. More importantly, they are not always mature and the environment is rapidly changing.

The aim of this chapter is to propose a general framework to analyse social networks with distributed data that can take advantage of the state-of-the-art tools in the big data era. This framework should be suitable for most of the possible usages. We have tested a first implementation of this framework using *Hadoop Map Reduce*, *HBase*, and *Apache Giraph*.

This chapter is organised as follows: section 7.2 presents existing tools for analysis and storage of graphs and some big data processing tools. Section 7.3 presents our model for social network analysis with distributed data. Section 7.5 shows how to implement our methods for detecting and predicting local communities and discusses the results.

Contents

7.1 Introduction 105
7.2 Tools for analysis and storage of large social networks 106
7.2.1 Tools for social network analysis 107
7.2.2 Hadoop Map Reduce 108
7.2.3 Pregel and Giraph 109
7.2.4 Pegasus 109
7.2.5 Graph databases and NoSQL 110
7.2.6 Summary of network analysis tools 111
7.3 Framework description 111
7.3.1 Distributed Database Layer (DDL) 112
7.3.2 Data Analysis Layer (DAL) 113
7.3.3 Data Visualisation Layer (DVL) 113
7.4 An implementation of the framework 113
7.4.1 Distributed Database Layer 114
7.4.2 Data Analysis Layer 116
7.4.3 Data Visualisation Layer 117
7.5 Implementation of our methods 118
7.5.1 Experimental Setup 118
7.5.2 Local community detection 118
7.5.3 Local community prediction 118
7.6 Conclusion 121

7.2 Tools for analysis and storage of large social networks

Large social networks analysis with distributed data requires adapted tools for processing and storage. The aim of this section is to give an overview of

existing solutions.

7.2.1 Tools for social network analysis

Gephi

Gephi [15] is an open-source tool, written in Java, to analyse and visualise social networks. Many classical algorithms are implemented: connected components, page rank, betweenness centrality, community detection (Louvain's method), etc.

The most important limitation of this tool is the size of the graphs it can handle. Indeed, because the network visualisation is always presented to the user, it is only possible to display relatively small graphs.

NetworkX

NetworkX [60] is a library for the Python programming language. It is quite complete and up to date and allows to analyse complex networks with attributes for nodes and links.

Its limitations are that it is only available for python and that it can only analyse networks with hundreds of thousands of nodes.

Igraph

Igraph [51] is a library in *C* language which allows to analyse complex networks like NetworkX. Its advantages compared to NetworkX is that it is available for several high level programming languages: *C/C++*, python and *R*.

Its limitations are that it do not take advantage of multi cores/processors and it require all the network to be available in the main memory.

GraphCT/STINGER

The Spatio-Temporal Interaction Networks and Graphs (STING) Extensible Representation [42] is a parallel framework to analyse very large networks (with billions of nodes). As Igraph, several algorithms are already implemented, it allows to define new algorithms using the proposed data structure and it is available for many programming languages: *C/C++*, Java and Python.

Although it enables efficient analysis of very large networks, it requires all the data to be available in the main memory of a single computer.

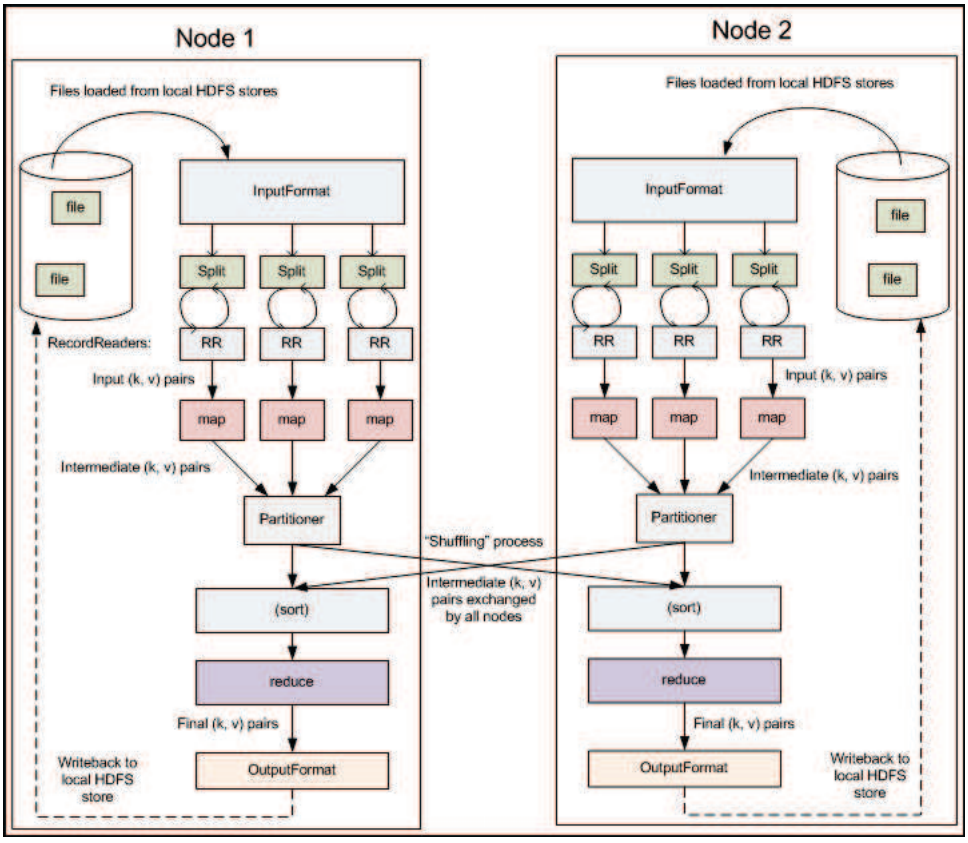


Figure 7.1: Hadoop Map Reduce execution model.

7.2.2 Hadoop Map Reduce

All the solutions presented above are all mono-site. To deal with distributed data, one can use the *Hadoop Map Reduce* framework[152]. In this framework, data are modelled using *key-value* pairs. The computations in this framework are divided into one or several iterations. Each iteration consists in two principal operations: a *Map* and a *Reduce*. During the Map step, an operation is applied to each data row and a result containing a (eventually new) key and the value of the operation is emitted. All the results having the same key are transmitted to the same reducer. The Reduce operation consists in aggregating all the values with the same key. Each iteration is usually performed simultaneously on many physical servers containing the distributed data. See fig. 7.1 for an illustration of this process between two computing nodes.

The main limitation of this framework is that it considers that all the

analysis tasks can be represented as map/reduce tasks. Due to this limitation, many algorithms for networks analysis are difficult if not impossible to implement using this framework (for example, algorithms that require global sharing states). Moreover, for each Map task, one need to process the entire network, which is not always suitable.

7.2.3 Pregel and Giraph

Google has recently proposed Pregel [94], a framework to processes large graphs. In this framework, each node of the graph is a logical computing unit which has a state and is able to receive and send messages from and to its direct neighbours. A processing task is done using one or several iterations during which each node compute in parallel. At each iteration, each node:

- receives the messages sent to him during the previous iteration;
- executes the function defined by the user;
- changes eventually it value and send a message to its neighbours;
- changes if necessary the topology of the network
- votes to halt if it has no more work to do.

Classical graphs algorithms like connected components, pageRank, can be easily implemented in this framework.

Pregel is the private property of Google. Giraph is an open source implementation of this computing framework. Pregel and Giraph are more computing models than graph social network analysis framework. As Hadoop MapReduce some processing tasks are difficult or even impossible to implement using it. Also, they do not provide ways to query particular parts of the network.

7.2.4 Pegasus

Pegasus [71] is a graph processing framework built on top of Hadoop MapReduce [152]. It allows to analyse distributed large graphs using the Hadoop distributed file system (HDFS). This allows to process very large distributed networks.

Its limitations are that it does not take into account attributes of nodes and edges and does not allows to query parts of the network.

7.2.5 Graph databases and NoSQL

Many graph databases and general purposes distributed databases have been developed during the recent years, mainly to overcome the limitations of traditional relational databases. Here are some of these solutions selected for their relevancy for our context.

AllegroGraph

Allegrograph is a Resource Description Framework (RDF) which allows to implement graph databases. Its query language is an implementation of the *SPARQL* language. Its limitation is that it is commercial and mono-site only.

Neo4J

Neo4J is a graph oriented database. It defines graph traversal operations to visit the nodes of the graph following a user defined pattern. Its main query language is called ***Cyphers***. Its is available both in commercial and open source licences (for non commercial use only).

Although some effort are actually done to enable it dealing with distributed data, the main supported version is actually only usable on one single computer.

FlockDB

FlockDB is a distributed graph database for storing adjancency lists, with goals of supporting:

- a high rate of add/update/remove operations
- potentially complex set arithmetic queries
- paging through query result sets containing millions of entries
- ability to "archive" and later restore archived edges
- horizontal scaling including replication
- online data migration

Non-goals include: multi-hop queries (or graph-walking queries) and automatic shard migrations.

FlockDB is much simpler than other graph databases such as Neo4J because it tries to solve fewer problems. It scales horizontally and is designed for on-line, low-latency, high throughput environments such as web-sites.

Tools	Storage	Cons.	Vis?	DM	RT query?	HS?	Flex.?
Gephi	M	NA	yes	PG	yes	no	F
Networkx	M	NA	yes	PG	yes	no	F
Igraph	M	NA	yes	PG	yes	no	F
STINGER	M	NA	no	EAL	yes	No	P
Allegrograph	M	FL	no	RDF	yes	no	P
FlockDB	DD	FL	no	AL	yes	yes	N
Neo4J	LD	ACID	yes	PG	yes	no	F
Giraph	M	NA	no	AL	no	no	P
Pegasus	DD	FL	no	KV	no	yes	N

Table 7.1: Comparison of graph analysis tools

Twitter uses FlockDB to store social graphs (who follows whom, who blocks whom) and secondary indices. As of April 2010, the Twitter FlockDB cluster stores 13+ billion edges and sustains peak traffic of $20k$ writes/second and $100k$ reads/second.

7.2.6 Summary of network analysis tools

From the overview presented above, one can note that many solutions for network processing exist. However, they are either storages or processing frameworks. In the rest of this chapter, an integrated framework that can take advantage of the state-of-the-art of processing and storage tools is presented. Table 7.1 presents a summary of the presented tools. In that table, "M" means "in memory", "DD" is Distributed Database, "LD" is local database "NA" is "not applicable", "RT" is "real time", "HS" is for "horizontal scalability", "Vis" is a short-cut for visualisation and "Flex" for flexibility(adding new attributes, node and/or edge types). For flexibility, "F" means fully flexible, "P", partially and "N" not flexible at all(with not attribute support). "DM" means data model and "PG" means property graph, "AL" is for Adjacency list, "EAL" for extended adjacency list and "KV" for key value.

7.3 Framework description

In this section, a general framework for the analysis and storage of social networks with distributed data is presented. The idea of this framework is to take advantage of the most up-to-date tools and to interchange them transparently as needed. For that reason, the framework is defined as a layered

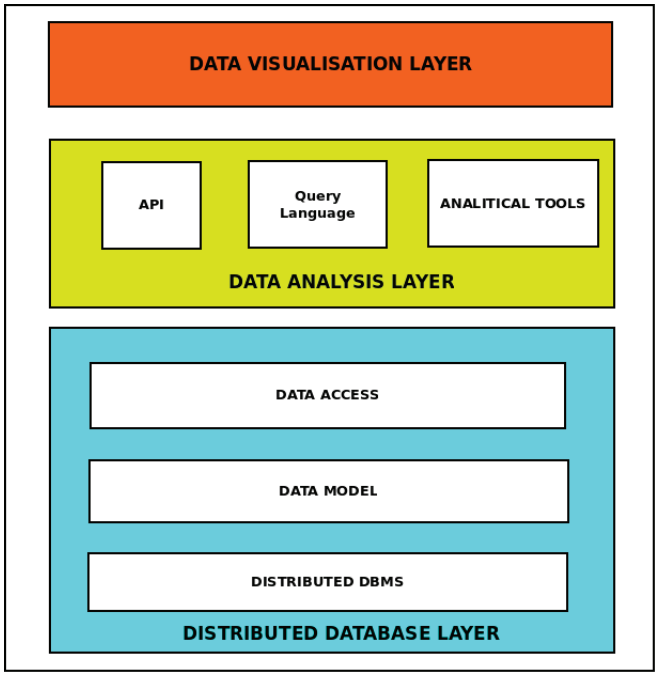


Figure 7.2: Layered model for social network analysis.

model like the Open System Interconnection (OSI) in physical interconnection networks. The proposed model has the following layers: the Distributed Database Layer (DDL), the Data Analysis Layer and the Data Visualisation Layer. Fig. 7.2 present these layers. The following sub-sections describe these layer in depth.

7.3.1 Distributed Database Layer (DDL)

The first part of the framework is the Distributed Database Layer. It handles the storage of and access to the distributed network's data. It has three main components: the Distributed Database Management System, the Data Model and the Data Access.

Distributed Database Management System

The Distributed Database Management System handles the persistence of the distributed network. As a database management system (DBMS), It is in charge of the management of the consistency and availability of the data. Possible choices for the implementation of this layer are HBase, Apache Cassandra, couchDB and MongoDB.

Data Model

The Data Model describes how the network is represented on the database. The chosen representation must facilitate the queries and the processing of the network. It must also be tightly coupled to a particular architecture i.e., a data model is supposed to be used with different DBMS.

Data Access Layer (DAL)

The role of the Data Access Layer is to define how the network data is accessed and updated. This is done by defining some procedures to manipulate the network. These procedures take into account the defined data model.

7.3.2 Data Analysis Layer (DAL)

Given the description of the DDL, the Data Analysis Layer is able to query and interact with the data and process them. This layer consists in three components:

- API (Application Programming Interface): enables to access to the framework using programming languages. This API must allow to perform efficiently operations like the neighbordhood exploration, the computation of local clustering coefficients, computing the triangle closures, and more generally, graph traversal operations.
- CLI (Command Line Interface): allows to analyze the network interactively. It must provide to the user all the classical analysis tasks.
- Analytic tools: provides batch processing tools which require more execution time and are not suitable in an interactive mode.

7.3.3 Data Visualisation Layer (DVL)

The Visualisation Layer consists in visualisation algorithms that can display sub-graphs queried using the DAL. Examples of such algorithm are layout algorithms to spatially arrange node and edges and nodes/edges selection.

7.4 An implementation of the framework

In this section, the details of one implementation of this framework we have performed is presented.

7.4.1 Distributed Database Layer

This sub-section shows how we have managed the distribution and the access to the data. It presents all the components of this layer: the distributed DBMS, the Data model and the Data Access.

Distributed DBMS

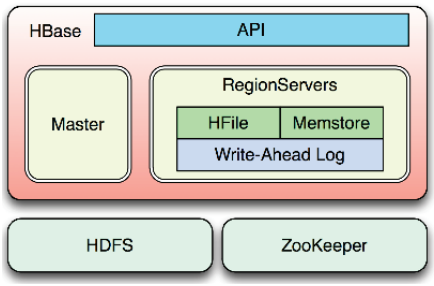
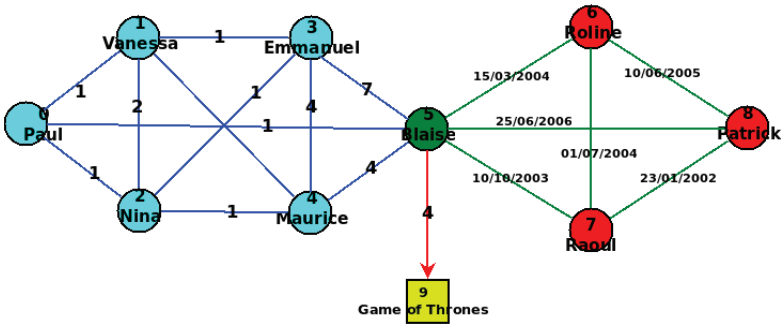


Figure 7.3: HBase architecture.

In the implementation we have done, the Distributed Database Layer is managed by HBase [52]. HBase is an open-source distributed database. Its is column oriented, versioned (multiple versions of the same row). Fig. 7.3 presents an overview of the HBase architecture:

- data are stored using the Hadoop Distributed File System (HDFS) [152];
- the Master is responsible to assign regions to regions servers (RegionServers) and it uses Apache ZooKeeper, a reliable, persistent and highly available distributed coordination service;
- the RegionServers manage the regions and the data storage on HDFS;
- the API is the component which allows the access to HBase using programming languages.

More information on HBase are available on [52] or by browsing on the official web site: `http://hbase.apache.org/`.



(a)

Row ID	Column Family	Column Qualifier	Value
5	Attribute	Type	People
5	Attribute	Name	Blaise
5	Collaboration	0	1
5	Collaboration	3	7
5	Collaboration	4	4
5	Friendship	6	15/03/04
5	Friendship	7	10/10/03
5	Friendship	8	25/06/06
5	Rating	9	4

(b)

Figure 7.4: A toy network (a) and the representation of node 5(b).

Data Model

Given the distributed database management system, a data model must be defined. The implemented Data Model is a table in which each line corresponds to an attribute of a node or a link:

- a link (u,v) with the label l is represented by (u, Neigh, v, l), where "Neigh" represents the type of the link;
- an attribute of the node u of type T and value V is represented by (u, Attribute, T, V).

Figure 7.4 presents an example of network (a) and the rows representing the information of node 5 (b). In that figure, examples of types of links are "Collaboration" and "Friendship" and examples of nodes' attributes are "Type" and "Name". One can see for example that the node *Blaise* with id 5 has (among others) a friendship relation, started the *10/10/2003* with the node *Raoul*. *Blaise* has also rated the movie *Game of thrones* with 4 stars.

This model is quite general and can model a large range of applications in social network analysis.

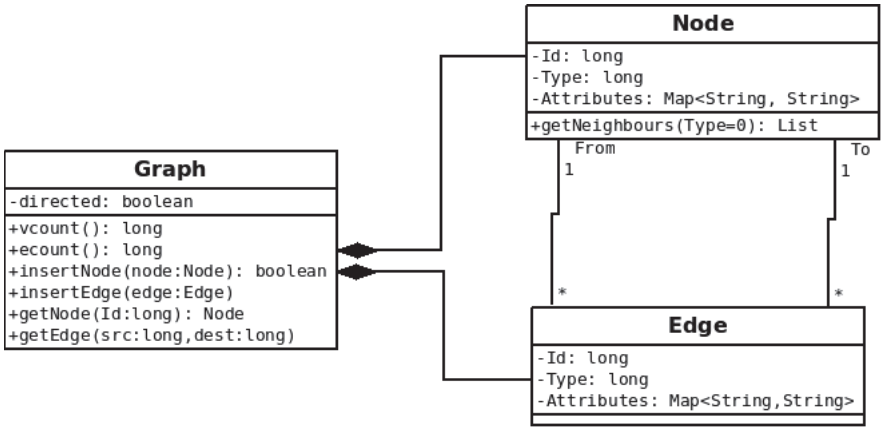


Figure 7.5: Main classes of the Data Access Layer.

Data Access

Our implementation of this component contains the main following classes: Node, Edge and Graph. Fig. 7.5 presents this classes and their relations using a *UML diagram*.

Given this description of the Data Access Layer, the upper layers can implements social network algorithm as needed.

7.4.2 Data Analysis Layer

Our DAL has been implemented using the Java language. Among the three components of this layer, we have implemented the API and started to provide some analytic tools. The main methods provided by the API are:

- insertNode(Node node): insert a node to the database
- Node getNode(): get a node from the database
- insertEdge(Edge edge): insert an edge to the database
- Edge getEdge(): get an edge from the database
- long vcount(): return the number of nodes
- long ecount(): return the number of edges
- List<Node> getAllNode(long type=0): return all nodes of a particular type (0 means all types)

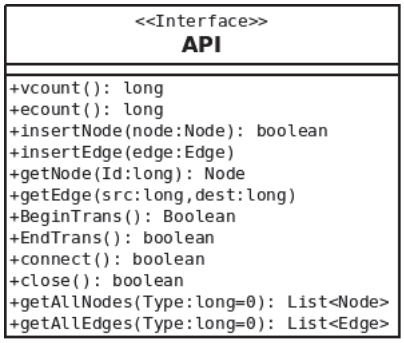


Figure 7.6: API component of the DAL.

- List<Edge> getAllEdge(long type=0): return all edges of a particular type (0 means all types)
- List<Node> getNeighbours(long id, long type=0): return all neighbours of the node *id* having a particular type (0 means all types)
- boolean beginTrans(): begin a transaction mode
- boolean endTrans(): end a transaction mode
- boolean connect(): open the connection to the database
- boolean close(): close the connection to the database

These operations are provided through the interface presented in figure 7.6.

The Analytic Tools component is implemented using Hadoop Map Reduce and Apache Giraph described in the section 7.2. The analytic algorithms already implemented are:

- common neighours (MapReduce)
- single source shortest path (Giraph)
- Connected components(Giraph)
- Degree Distribution(Map educe)
- PageRank (Giraph)

7.4.3 Data Visualisation Layer

In our implementation of the framework, we have used Gephi[15] to visualize the sub-graphs generated by our algorithms.

7.5 Implementation of our methods

In this section, we present how we have used this framework to implement some of the methods proposed in this thesis. The selected methods are local community identification and local community prediction. Before presenting these implementation, we first give an overview of the experimental setup.

7.5.1 Experimental Setup

Hardware configuration

For our preliminary experimentations, we have used an Hadoop cluster of *11* commodity desktop computers. The replication factor, which is the number of copies of a block of data, is set to 3 and the block size is set to $64MB$.

Data

The dataset used is DBLP as described in chapter 6.

7.5.2 Local community detection

The distributed algorithms are the same as in the case where the data is located on the computation node. We only need to replace each sequential instruction that gives access to a node or to an edge, by the corresponding one that is distributed. The basic greedy scheme for the identification of local communities using the framework is given in Algorithm 3. In this algorithm, the change are in red italic and boldface.

Since the computation of each social circle is independent of the others, the total computation time of the distributed program decreases linearly with the number of processing nodes. We have conducted experiments with one to ten processors and the results are reported in Fig. 7.7.

7.5.3 Local community prediction

In this subsection, the preliminary results obtained with this framework for community prediction are presented. Recall that one of the most computational part of the proposed method is the computation of the number of common neighbours between each pair of nodes. Here we present an algorithm with complexity in $O(n)$ for power-law degree distributed networks. This algorithm is summarised in fig. 7.8. This algorithm used a map/reduce approach and can be described as follows:

Algorithm 3 Identification of local communities using the framework

Algorithm: *Local community identification*

Input: an instance of the API G **and a starting node** n_0.

Output: a subset D : the local community of n_0.

node := G.getNode(n_0)

$D := \{node\}$

$B := \{node\}$

$C := \emptyset$

S := node.getNeighbours()

Q = 0

Repeat

For each $s_i \in S$ **do**

Compute the quality criterion obtained if s_i is added to D

End for each

Select the node s^* that produces the maximal quality Q^*, breaking ties randomly.

If $Q^* > Q$ **then**

$D.add(s^*)$

$S.remove(s^*)$

S.add(s*.getNeighbours())

S.add(s*.getNeighbours())

Update B, C.

End if

Until $(Q^* \leq Q)$

Return D

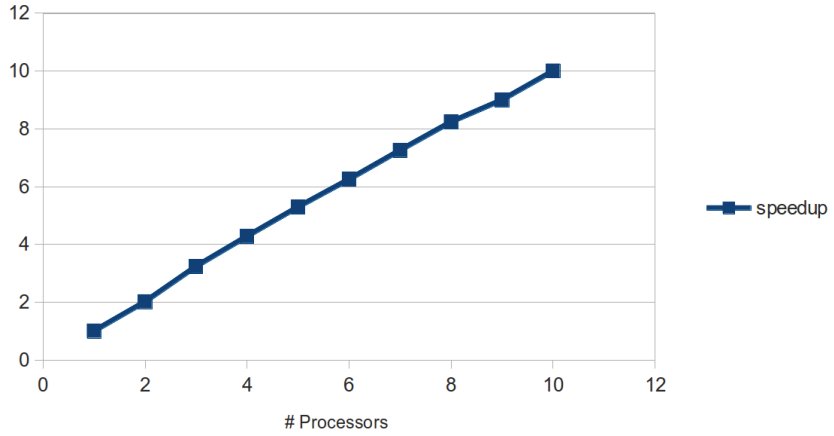


Figure 7.7: Speedup results: the computation time of local communities identification, for all the nodes of the network, linearly decreases with the number of processors.

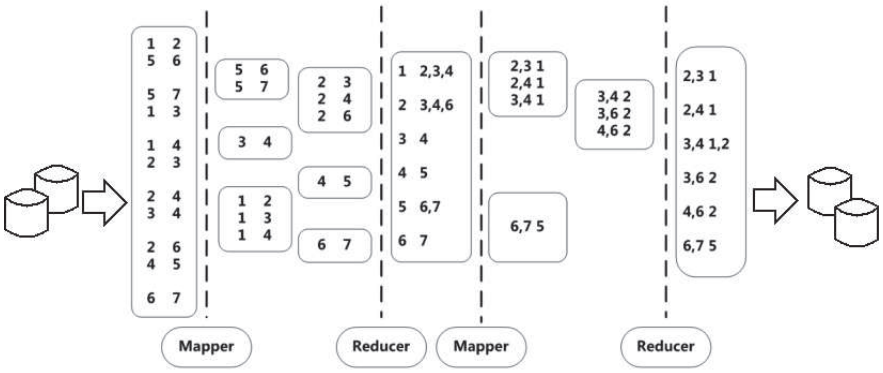


Figure 7.8: Common Neigbours computation using a map/reduce approach.

1. each split of the data with the source node as key and the destination node as value is provided to a first set of mappers;

2. these first mappers just emit the keys and the values to the first set of reducers ;

3. the first set of reducers produce the adjacency list for each node;

4. the adjacency lists are passed to a second set of mappers with the node as key and the list as value;

5. for each list passed to a mapper and each pair of node in the list, the mapper emits as key the pair of node and as value the id of the node

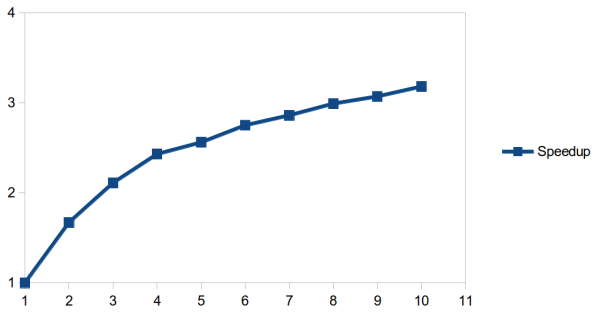


Figure 7.9: Common neighbours speedup on DBLP.

having this adjacency list;

6. the second set of reducers just add up the list of common neighbours.

All the steps except step 5 are computable (without taking into account communication) in time $O(n)/K$ with K the number of workers. The step 5 requires to compute all the non ordered couples of a list of size L that can be done in time $O((L^2) \times (n/K))$ also with K workers which reduce to $O(n)$ for power law degree distributed networks with a preprocessing consisting in removing very high degree nodes (according to the target application).

Fig. 7.9 shows the speedup obtained. The horizontal axis corresponds to the number of processors and the vertical axis correspond to the to the speed-up. One must notice that the speedup obtained is by comparison to the execution on one node.

We observe on Fig. 7.9 a gain growing with the number of processors with a speedup of more than 3 on ten computers. The experimentation was not perform on a real cluster and where we expect better performances than the presented results.

7.6 Conclusion

In this chapter, we have presented a general framework to deal with data distribution for social networks. We also have proposed an implementation of this framework using up to date open-source big data tools based on Hadoop technologies. A preliminary evaluation shows that this framework is promising.

We plan to implement other algorithms, test the framework on a real cluster and distribute the framework as an open-source tool.

CHAPTER 8

Conclusions and Future Directions

"You have to start with the truth. The truth is the only way that we can get anywhere. Because any decision-making that is based upon lies or ignorance can't lead to a good conclusion."

- Julian Assange

The observation that many real worlds systems can be modelled as networks and the democratisation of online social networks have provided many interesting new challenges to the scientific community. In this thesis we have contributed to the analysis of community in dynamic and distributed social networks from a node-centric point of view. Our motivation for this local analysis is that it is often difficult to perform a global analysis in very large and dynamic networks. Moreover, for some applications like churn prediction or social recommendation, a local view is sometimes more pertinent than a global one.

In this thesis we focused on four main objectives. The first objective was to develop a local community detection method. The second was to validate the detected local community on some real applications. The third was to predict the evolution of the detected local communities. The fourth was to propose a distributed model to store the network with efficient support of local community detection in distributed networks.

8.1 Summary of the contributions

We summarize our contributions by grouping them into major themes.

Literature review

- In chapter 3, we have presented a review of global communities detection in complex networks.

- In Chapter 4, we have presented a state-of-the-art of local community detection methods.

- Finally, in chapter 7, we have presented an overview of solution to store and process large social networks.

Design of algorithms

- In Chapter 4, we have presented two new local community detection methods that performs better than the state-of-the-art on real and synthetic datasets.

- In Chapter 4 we also have proposed a procedure to detect local overlapping communities. Results on real dataset have shown the quality of this approach.

Applications of network communities

- In chapter 5, we have considered using local communities in churn prediction. Local community based attributes perform better than all the others considered neighbourhood.

- In chapter 5, we have considered using local communities in social recommendation. The propose model using local community as nearest neighbours gives good results on the considered datasets.

Distributed network model

- In chapter 7, we have proposed a general framework to store and analyse large dataset in a distributed way. This model can handle complex networks with nodes/links attributes and temporal evolution.

- In chapter 7 we also have proposed an implementation of this framework using Hadoop technologies. Experimentations on the detection of local communities and prediction of their dynamic have shown that this framework is promising.

8.2 Future directions

We propose several ideas and perspectives built on the work presented in the thesis.

Network structure and communities

- Include user's attributes while keeping the computation local: how can we take advantage of the node attributes while having a local view of the network? Locality Sensitive Hashing (LSH) [6] may be a promising direction towards this objective.

- Detect the influential members: who are the most important community members and what are their properties? Taking into account the localization of each node (whether it is on the border or the center) and its connections can be a starting point.

- Automatic labelling of communities(interpretation of community structure): what is a community talking about? We can inferred topics based on the data exchanged by community member using text categorization techniques. This can help an Online Social Network Provider like Skyrock to detect all the communities around a given topic to target its advertising campaigns.

Local community in dynamic networks

Can we directly model the dynamic of local communities in evolving networks? The Link Flow model, recently proposed by Latapy et al. [81], which takes simultaneously into account the topological and temporal dimensions can be a good direction.

Framework for storing and processing Large Social Networks

Building a software framework helping to handle large and distributed social networks is an ambitious project. We proposed some directions during this thesis. This effort should be pursued and the result could be published as an open source library, for the benefit of the research community and the industrials.

Applications

So far we only used communities in churn prediction and social recommendation. There are other possible applications of network communities that we have yet to discover such as group profiling, trend monitoring, sentiment analysis, link prediction.

The long-term perspective of this thesis is to apply the developed and the studied techniques in some important domains for the society such as health

(propagation of diseases, collect of information), education (collaborative learning, resources sharing) and social development (collaboration between actors).

BIBLIOGRAPHY

[1] Facebook wall posts network dataset - konect., August 2013. Dataset available at `http://konect.uni-koblenz.de/networks/facebook-wosn-wall`.

[2] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2003.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, June 2005.

[4] Fabio Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 273–280, New York, NY, USA, 2013. ACM.

[5] O. Allali, C. Magnien, and M. Latapy. Link prediction in bipartite graphs using internal links and weighted projection. *Proceedings of the third international workshop on Network Science for Communication Networks (NetSciCom)*, 2011.

[6] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. *CoRR*, abs/1306.1547, 2013.

[7] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 913–921, New York, NY, USA, 2007. ACM.

[8] Thomas Aynaud and Jean-Loup Guillaume. Static community detection algorithms for evolving networks. In *WiOpt'10: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 508–514, Avignon, France, 2010.

[9] Thomas Aynaud and Jean-Loup Guillaume. Multi-Step Community Detection and Hierarchical Time Segmentation in Evolving Networks. In *Fifth SNA-KDD Workshop Social Network Mining and Analysis, in conjunction with the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011)*, 2011.

[10] Kernighan B. W. and Lin S. An efficient heuristic procedure for partitionning graphs. In *Bell System Technical Journal*, volume 49, pages 291–308, 1970.

[11] J. P. Bagrow. Evaluating local community methods in networks. In *Journal of Statistical Mechanics*, page 05001, 2008.

[12] Albert-Laszlo Barabasi. *Linked: the new science of networks*. Perseus Pub., Cambridge, Mass., 2002.

[13] Albert-Laszlo Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume, reissue edition, April 2003.

[14] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[15] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 2009.

[16] L. Benamara and C. Magnien. Performance of modularity maximization in practical contexts. *Physical Review E.*, 2010.

[17] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[18] V. D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. In *Journal of Statistical Mechanics: Theory and Experiment*, page 10008, 2008.

[19] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[20] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner. Maximizing modularity is hard, 2006. cite arxiv:physics/0608255 Comment: 10 pages, 1 figure.

[21] Romain Campigotto, Jean-Loup Guillaume, and Massoud Seifi. The power of consensus: random graphs have no communities. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 272–276. ACM, 2013.

[22] Remy Cazabet and Frédéric Amblard. Simulate to detect: A multi-agent system for community detection. In *IAT*, pages 402–408, 2011.

[23] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2):1–26, June 2008.

[25] B. Chapus, F. Fogelman Soulié, E. Marcadé, and J. Sauvage. Mining on social networks. *Statistical Learning and Data Science, edited by Mireille Gettler Summa, Leon Bottou, Bernard Goldfarb and Fionn Murtagh. Computer Science and Data Analysis Series, CRC Press, Chapman & Hall*, 2011.

[26] J. Chen and Y. Saad. Dense subgraph extraction with application to community detection. In *IEEE Transactions on Knowledge and Data Engineering*, volume 99, pages 1–14, 2010.

[27] J. Chen, O. R. Zaiane, and R. Goebel. Detecting communities in large networks by iterative local expansion. In *Proceedings of the 2009 International Conference on Computational Aspects of Social Networks*, pages 105 – 112, 2009.

[28] J. Chen, O. R. Zaiane, and R. Goebel. Local communities identification in social networks. In *ASONAM*, pages 237–242, 2009.

[29] A. Clauset. Finding local community structure in networks. In *Physical Review*, volume 72, page 026132, 2005.

[30] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. In *Physical Review*, volume 70, page 066111, 2004.

[31] W Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[32] Johan Dahlin and Pontus Svenson. Ensemble approaches for improving community detection methods. *CoRR*, abs/1309.0242, 2013.

[33] The Anh Dang and Emmanuel Viennet. Community detection based on structural and attribute similarities. In *International Conference on Digital Society (ICDS)*, pages 7–14, January 2012. ISBN: 978-1-61208-176-2. Best paper award.

[34] The Anh Dang and Emmanuel Viennet. Collaborative filtering in social networks: A community-based approach. In *IEEE ComManTel 2013, Int. Conf. on Computing, Management and Telecommunications*, January 2013.

[35] The Anh Dang and Emmanuel Viennet. Collaborative filtering in social networks: A community-based approach. In *IEEE ComManTel 2013, Int. Conf. on Computing, Management and Telecommunications*, January 2013.

[36] Maximilien Danisch, Jean-Loup Guillaume, and Bénédicte Le Grand. Towards multi-ego-centred communities: a node similarity approach. *IJWBC*, 9(3):299–322, 2013.

[37] K. Dasgupta, R Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. Nanavati, and A. Joshi. Social ties and their relevance to churn in mobile telecom networks. In *EDBT '08: Proceedings of the 11th International Conference on Extending Database Technology*, pages 668–677, 2008.

[38] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[39] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. Toward the next generation of recruitment tools: an online social network-based job recommender system. In *ASONAM*, pages 821–828, 2013.

[40] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2000.

[41] P. Domingos. A general method for making classifiers cost-sensitive. In *International Conference on Knowledge Discovery and Data Mining*, pages 155 – 164, 1999.

[42] David Ediger, Robert McColl, E. Jason Riedy, and David A. Bader. Stinger: High performance data structure for streaming graphs. In *HPEC*, pages 1–5, 2012.

[43] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, February 2011.

[44] Evans and Lambiotte. Line graphs, link partitions and overlapping communities. In *Physical Review*, volume 80, page 016105, 2009.

[45] Gerhard Fischer. External and shareable artifacts as opportunities for social creativity in communities of interest. In *University of Sydney*, pages 67–89, 2001.

[46] F. Fogelman Soulié and Marcadé. Industrial mining of massive data sets. *Mining massive Data Sets for Security Advances in data mining, search, social networks and text mining and their applications to security ». F. Fogelman-Soulié, D. Perrotta, J. Pikorski, R. Steinberger eds. IOS Press. NATO ASI Series*, 2008.

[47] S. Fortunato. Community detection in graphs. In *Physics Reports*, volume 486, pages 75–174, 2010.

[48] S. Fortunato and M Barthélemy. Resolution limit in community detection. In *PNAS*, volume 104, pages 36–41, 2007.

[49] Antonino Freno, C. Garriga, Gemma, and Mikaela Keller. Learning to Recommend Links using Graph Structure and Node Content. In *Neural Information Processing Systems Workshop on Choice Models and Preference Learning*, 2011.

[50] Adrien Friggeri, Guillaume Chelius, and Eric Fleury. Egomunities, exploring socially cohesive person-based communities. *CoRR*, abs/1102.2623, 2011.

[51] Csárdi Gábor and Nepusz Tamás. The igraph software package for complex network research. In *InterJournal Complex Systems*, page 1695, 2006.

[52] L. George. *HBase: The Definitive Guide*. O'Reilly, 2011.

[53] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 2005.

[54] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.

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

[56] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, pages 176–183, Washington, DC, USA, 2010. IEEE Computer Society.

[57] S. Gregory. Finding overlapping communities using disjoint community detection algorithms. In *Complex Networks*, pages 47–61, 2009.

[58] J.-L. Guillaume and M. Latapy. Modularities for bipartite networks. *Information Processing Letters 90(6)*, pages 215–221, 2004.

[59] J. Hadden, A. Tiwari, R. Roy, and D. Ruta. Computer assisted churn management: Stat-of-the-art and futur trends. In *Computers and Operations Research*, volume 34(10), pages 2902–29177, 2007.

[60] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, August 2008.

[61] Greg Hamerly and Charles Elkan. Learning the k in k-means. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 281–288. MIT Press, 2004.

[62] J. A. Hanley and B. J. Mcneil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, September 1983.

[63] Frank E. Harrell. *Regression Modeling Strategies, with Applications to Linear Models, Survival Analysis and Logistic Regression*. Springer, 2001.

[64] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. *Proceedings of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[65] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[66] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[67] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, 2003.

[68] H. Hwang, T. Jung, and E. Suh. An ltv model and customer segmentation based on customer value: a case study on the wireless telecommunication industry. In *Expert Syst. Appl.*, volume 26(2), pages 181–188, 2004.

[69] Vanessa Kamga, Maurice Tchuente, and Emmanuel Viennet. Prévision de lien dans les graphes bipartites avec attributs. *Revue des Nouvelles Technologies de l'Information (RNTI-A6)*, pages 67–85, November 2013.

[70] Vanessa Kamga, Maurice Tchuente, and Emmanuel Viennet. Prévision de liens dans les graphes bipartites avec attributs. *Revue des Nouvelles Technologies de l'Information (RNTI-A6)*, 2013.

[71] U. Kang, C. E. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pages 229–238, Washington, DC, USA, 2009. IEEE Computer Society.

[72] L. Katz. A new status index derived from sociometric analysis. *Psychmetrika 18*, page 39, 1953.

[73] Ivan Keller and Emmanuel Viennet. A characterization of the modular structure of complex networks based on consensual communities. In *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, pages 717–724. IEEE, 2012.

[74] Ivan Keller and Emmanuel Viennet. A characterization of the modular structure of complex networks based on consensual communities. In *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, pages 717–724. IEEE, 2012.

[75] S. Kerthi and C. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. In *Neural Computation*, volume 15, pages 1667–1689, 2003.

[76] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24, January 2010.

[77] Valdis Krebs. Mapping networks of terrorist cells. *J. Amer. Soc. Inform. Sci. 27*, 24(3):43–52, 2002.

[78] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. In *Physical Review*, volume 78, page 046110, 2008.

[79] Andrea Lancichinetti, Filippo Radicchi, José J. Ramasco, and Santo Fortunato. Finding Statistically Significant Communities in Networks. *PLoS ONE*, 6(5), 2011.

[80] Cornelius Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral, 1950.

[81] Matthieu Latapy and Jordan Viard. Complex networks and link streams for the empirical analysis of large software. In *Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings*, pages 40–50, 2014.

[82] J. Leskovec, L. Adamic, and B. Adamic. The dynamics of viral marketing. In *ACM Transactions on the Web (ACM TWEB)*, volume 14, pages 228–237, 2006.

[83] Runpeng Liang, Jun Hua, and Xiaofan Wang. Vcd: A network visualization tool based on community detection. In *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, pages 1221–1226, Oct 2012.

[84] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference*

on Information and knowledge management, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[85] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[86] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[87] László Lovász. Random walks on graphs: A survey, 1993.

[88] Linyuan Lu and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[89] Linyuan Lu and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[90] F. Luo, J. Z. Wang, and E. Promislow. Exploring local community structure in large networks. In *WI'06.*, pages 233–239, 2006.

[91] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[92] Ashwin Machanavajjhala, Aleksandra Korolova, and Atish Das Sarma. Personalized social recommendations: Accurate or private. *Proc. VLDB Endow.*, 4(7):440–450, April 2011.

[93] Sofus A Macskassy and Foster Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. *International conference on intelligence analysis*, 2005.

[94] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A System for Large-scale Graph Processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[95] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[96] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[97] Mohri Mehryar, Rostamizadeh Afshin, and Talwalkar Ameet. Foundations of machine learning, 2012.

[98] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 263–266, New York, NY, USA, 2003. ACM.

[99] Bivas Mitra, Lionel Tabourier, and Camille Roth. Intrinsically dynamic network communities. *CoRR*, abs/1111.2018, 2011.

[100] M. Mitzenmacher. A brief history of lognormal and power law distributions. *Proceedings of the Allerton Conference on Communication, Control, and Computing*, pages 182–191, 2001.

[101] M. Mozer, R. H. Wolniewicz, D. B. Grimes, E. Johnson, and H. Kaushansky. Churn reduction in the wireless industry. In *NIPS*, pages 935–941, 1999.

[102] T. Murata. Modularities for bipartite networks. *In HT '09 : Proceedings of the Twentieth ACM Conference on Hypertext and Hypermedia, New York, NY, USA. ACM.*, 2009.

[103] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E.*, 64:025102, 2001.

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

[105] M. E. J. Newman. Spectral methods for network community detection and graph partitioning. *Physical Review*, 884:042822, 2013.

[106] M.E.J. Newman. The structure and function of complex networks. In *Statistical Mechanics*, volume 45, pages 167–256, 2003.

[107] M.E.J. Newman. Fast algorithm for detecting community structure in networks. In *Physical Review*, volume 69, page 066133, 2004.

[108] M.E.J. Newman. Finding community structure in networks using the eigenvectors of matrices. In *Physical Review*, volume 74, page 036104, 2006.

[109] M.E.J. Newman. Modularity and community structure in networks. In *Proc. Natl. Acad. Sci*, volume 103, pages 8577–8582, 2006.

[110] M.E.J. Newman and M Girvan. Community structure in social and biological networks. In *Proc. Natl. Acad. Sci*, volume 99, pages 7821–7826, 2002.

[111] M.E.J. Newman and M Girvan. Finding and evaluating community structure in networks. In *Phy. Rev.*, volume 69, page 026113, 2004.

[112] B. Ngonmang, M. Tchuente, and E. Viennet. Identification de communautés locales dans les réseaux sociaux. In *AGS, Conférence CAP*, pages 16–27, Mai 2011.

[113] B. Ngonmang and E. Viennet. Toward community dynamic through interactions prediction in complex networks. In *Signal-Image Technology Internet-Based Systems (SITIS), 2013 International Conference on*, pages 462–469, Dec 2013.

[114] B. Ngonmang, E. Viennet, S. Sean, P. Stepniewski, F. Fogelman-Soulié, and R. Kirche. Monetization and services on a real online social network using social network analysis. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 185–193, Dec 2013.

[115] Blaise Ngonmang, Maurice Tchuente, and Emmanuel Viennet. Local communities identification in social networks. *Parallel Processing Letters*, 22(1), March 2012.

[116] Blaise Ngonmang, Emmanuel Viennet, and Maurice Tchuente. Churn prediction in a real online social network using local community analysis. In *IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM'12)*, pages 282–290, August 2012.

[117] N.P. Nguyen, T.N. Dinh, Ying Xuan, and M.T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2282 –2290, april 2011.

[118] Ian Soboroff. Charles Nicholas and Charles K. Nicholas. Combining content and collaboration in text filtering. In *In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering*, pages 86–91, 1999.

[119] G. Palla, I. Derényi, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. In *Nature*, volume 435, pages 814–818, 2005.

[120] Gergely Palla, Albert lászló Barabási, Tamás Vicsek, and Budapest Hungary. Quantifying social group evolution. *Nature*, 446:2007, 2007.

[121] H. Papadakis, C. Panagiotakis, and P. Fragopoulou. Local community finding using synthetic coordinates. In *International Conference On Future Information Technology(FutureTech)*, pages 9–15, 2011.

[122] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[123] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. (We used the Python implementation avaible at https://pypi.python.org/pypi/stemming/1.0).

[124] A. Pothen, H. D. Simon, and K. P. Liou. Partitionning sparse matrices with eigenvectors of graphs. In *SIAM J. Matrix Anal. Appl.*, volume 11, pages 430–452, 1990.

[125] Bruno Pradel, Savaneary Sean, Julien Delporte, Sébastien Guérif, Céline Rouveirol, Nicolas Usunier, Françoise Fogelman-Soulié, and Frédéric Dufau-Joel. A case study in a recommender system based on purchase data. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 377–385, New York, NY, USA, 2011. ACM.

[126] D. J. de S. Price. A general theory of bibliometric and other cumulative advantage processes. *J. Amer. Soc. Inform. Sci. 27*, pages 292–306, 1976.

[127] E. Prud Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[128] R. J. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[129] F. Radicchi, C. Castellano, F. Cecconi, V Loreto, and D. Parisi. Defining and identifying communities in networks. In *PNAS*, volume 101, pages 2658–2663, 2004.

[130] Usha N. Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106+, September 2007.

[131] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[132] S. Ray and R. H. Turi. Determination of number of clusters in K-means clustering and application in colour image segmentation. In *4th International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99)*, pages 137–143, 1999.

[133] Raimundo Real and Juan M. Vargas. The Probabilistic Basis of Jaccard's Index of Similarity. *Systematic Biology*, 45(3):380–385, 1996.

[134] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[135] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[136] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[137] Jorge M. Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, ICANN '09, pages 175–184, Berlin, Heidelberg, 2009. Springer-Verlag.

[138] Satu Elisa Schaeffer, Stefano Marinoni, Mikko Sarela, and Pekka Nikander. Dynamic local clustering for hierarchical ad hoc networks. In *Sensor and Ad Hoc Communications and Networks*, pages 667–672, 2006.

[139] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, pages 291–324, Berlin, Heidelberg, 2007. Springer-Verlag.

[140] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[141] Massoud Seifi, Ivan Junier, Jean-Baptiste Rouquier, Svilen Iskrov, and Jean-Loup Guillaume. Stable Community Cores in Complex Networks. In *Complex Networks*, volume 424 of *Studies in Computational Intelligence*, pages 87–98. Springer Berlin Heidelberg, 2013.

[142] H. A. Simon. On a class of skew distribution functions. *Biometrika 42*, 1955.

[143] Ingve Simonsen, Kasper A. Eriksen, Sergei Maslov, and Kim Sneppen. Diffusion on complex networks: a way to probe their large-scale topological structures. *Physica A: Statistical and Theoretical Physics*, 336(1-2):163–173, May 2004.

[144] K. Suzuki and K. Wakita. Extracting multi-facet community structure from bipartite networks. *In CSE '09 : Proceedings of the 2009 International Conference on Computational Science and Engineering, Washington, DC, USA*, IEEE Computer Society:312–319, 2009.

[145] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[146] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 717–726, New York, NY, USA, 2007. ACM.

[147] Amanda L. Traud, Eric D. Kelsic, Peter J. Mucha, and Mason A. Porter. Comparing community structure to characteristics in online collegiate social networks. In *SIAM Review.*, 2010.

[148] Grout V., Cunningham S., and Picking R. Practical large-scale network design with variable costs for links and switches. In *International Journal of Computer Science and Network Security*, volume 7, pages 113–125, 2007.

[149] Emmanuel Viennet. Recherche de communautés dans les grands réseaux. In *RNTI*, pages 145–160, 2008.

[150] Jiří Šíma and Satu Elisa Schaeffer. On the np-completeness of some graph cluster measures. In *Proceedings of the 32Nd Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM'06, pages 530–537, Berlin, Heidelberg, 2006. Springer-Verlag.

[151] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1(393):440–442, 1998.

[152] T. White. *Hadoop: The Definitive Guide*. Yahoo Press, second edition, October 2010.

[153] X. Wu, V. Kumar, and R. Quilan. The top tens algorithms in datamining. In *Springer*, 2007.

[154] Bo Yang and Jiming Liu. Discovering global network communities based on local centralities. *ACM Trans. Web*, 2(1):1–32, February 2008.

[155] R. Yossi, Y.-T. Elam, and S Noam. Predicting customer churn in mobile networks through analysis of social groups. In *Proceedings of the Tenth SIAM International Conference on Data Mining*, April 2010.

[156] T. Zhou, L. Lü, and Y.-C. Zhang. Predicting missing links via local information. *Eur. Phys. J. B 71 (2009) 623.*, 2009.

[157] H. Zhu and W. Kinzel. Antipredictable sequences: Harder to predict than random sequences. *Neural Computation*, 1998.