Université d'Evry-Val-D'Essonne
Statistique et Génome / Exalead

Thèse

présentée en première version en vu d'obtenir le grade de Docteur,
spécialité Mathématiques Appliquées

par

Hugo Zanghi

Approches modèles pour la structuration du Web vu comme un graphe

Thèse soutenue le 25 juin 2010 devant le jury composé de :

M.	Gérard Govaert	Université de Technologie de Compiègne	(Rapporteur)
M.	Fabrice Rossi	TELECOM ParisTech, Paris	(Rapporteur)
M	Patrick Gallinari	Université Pierre et Marie Curie, Paris	(Jury)
M.	François Bourdoncle	Exalead, Paris	(Jury)
M.	Franck Ghitalla	INIST, Paris	(Jury)
M.	Christophe Ambroise	Université d'Evry-Val-D'Essonne	(Directeur)

Rien ne m'est sûr que la chose incertaine
Obscur, fors ce qui est tout évident
Doute ne fais, fors en chose certaine
Science tiens à soudain accident.
François Villon, Ballade du concours de Blois (extrait, 1458).

Remerciements

A tout seigneur tout honneur, je voudrais exprimer mes plus profonds remerciements à mon directeur de thèse, Christophe Ambroise. Ses intuitions, sa rigueur scientifique, sa disponibilité, son enthousiasme communicatif et son amitié ont été le terreau des fruits de cette thèse.

Que Gérard Govaert et Fabrice Rossi reçoivent toute l'expression de ma reconnaissance pour m'avoir fait l'honneur de rapporter ce travail. Je remercie également Patrick Gallinari, qui a accepté de présider mon jury. Je suis reconnaissant à François Bourdoncle pour sa confiance et sa participation à ce jury. Je tiens aussi à exprimer toute ma gratitude à Franck Ghitalla pour m'avoir fait prendre conscience en premier lieu de l'utilité de ces travaux.

Je remercie vivement la société Exalead de m'avoir proposé cette thèse CIFRE et de m'avoir soutenu en m'offrant un cadre propice à ce type de travail. Mes pensées vont à mes collègues de bureau: Sébastien Richard, Jim Ferenczi, Tristant Chapel, Morgan Champenois, Philippe David et Damien Lefortier.

Je tiens à remercier chaleureusement Franck Picard, Vincent Miele et Stevenn Volant pour leur collaboration scientifique ainsi que l'équipe Linkfluence, en particulier Guilhem Fouetillou et Camille Maussang, pour m'avoir fourni des jeux de données illustrant l'intérêt de mon travail.

Mes remerciements aux membres, passés ou présents, du laboratoire Statistique et Génome pour leur accueil. Merci à Bernard Prum, Catherine Matias[1], Julien Chiquet, Pierre Latouche, Etienne Birmelé, Maurice Baudry, Gilles Grasseau et Michèle Ilbert pour leurs conseils et leur sympathie.

Plus personnel, un grand merci à mes amis, en particulier Jérémie, Hadrien, Felix, Benjamin et l'Arnold team avec qui j'ai toujours autant de plaisir à me muscler les doigts et le foie.

Merci enfin à ma famille pour leur soutien inconditionnel. Je leur dédie cette thèse .

Je conclurai en remerciant de tout cœur la fille de la rue Barsacq pour le soutien moral, l'anglais soutenu et "tout le reste", c'est peu dire.

[1] Ouf, j'ai eu chaud !

Paris, le 30 aout.

Contents

Contents **vii**

Preface **1**

Préambule **5**

1 Graph clustering Survey **9**
- 1.1 Introduction . . . 11
- 1.2 Real-networks and communities . . . 13
- 1.3 Terminology and definitions . . . 16
  - 1.3.1 Graph theory . . . 17
  - 1.3.2 Markov chains . . . 19
  - 1.3.3 Computational complexity . . . 20
- 1.4 Graph Clustering . . . 21
  - 1.4.1 Communities . . . 21
  - 1.4.2 Partitions . . . 25
- 1.5 Algorithms for graph clustering . . . 26
  - 1.5.1 Traditional method . . . 26
  - 1.5.2 Betweenness-based methods . . . 28
  - 1.5.3 Modularity-based methods . . . 30
  - 1.5.4 Spectral algorithms . . . 31
  - 1.5.5 Random Walk algorithms . . . 32
  - 1.5.6 Model-based methods . . . 34
- Conclusion . . . 35

2 Strategies for Online Inference of Model-Based Clustering in large Networks **37**
- 2.1 Introduction . . . 39
- 2.2 Mixture Model and the EM algorithm . . . 40
  - 2.2.1 Finite Mixture Model . . . 40
  - 2.2.2 The EM algorithm . . . 43
- 2.3 A Mixture Model for Networks . . . 47
  - 2.3.1 Model and Notation . . . 47
  - 2.3.2 Sufficient statistics and Online Recursion . . . 51
  - 2.3.3 Likelihoods and online inference . . . 52
- 2.4 Classification EM algorithm . . . 53
  - 2.4.1 Classification log-likelihood . . . 53
  - 2.4.2 Online Estimation . . . 54
  - 2.4.3 Bernoulli affiliation model . . . 56
  - 2.4.4 Initialization and online supervised classification . . . 57
- 2.5 Stochastic Approximation EM for Network Mixture . . . 57

2.5.1 A short presentation of SAEM . . . 57
2.5.2 Simulation of $\Pr(\mathbf{Z}|\mathbf{X})$ in the online context . . . 58
2.5.3 Computing $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ in the online context . . . 58
2.5.4 Maximizing $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$, and parameters update . . . 59
2.6 Application of online algorithm to Variational EM methods . . . 59
2.6.1 Online variational step . . . 60
2.6.2 Maximization/Update step . . . 61
2.7 Choosing the number of clusters . . . 61
2.8 Experiments using simulation . . . 62
2.8.1 Comparison of partitions . . . 63
2.8.2 Comparison of algorithms . . . 63
2.9 Application . . . 68
2.9.1 French Political Blogosphere network . . . 68
2.9.2 The 2008 U.S. Presidential WebSphere . . . 71
Conclusion . . . 77

3 Model based graph clustering using both graph structure and vertex features **79**
3.1 Introduction . . . 81
3.2 A Mixture of Networks with covariates . . . 82
3.2.1 Connectivity Model . . . 83
3.2.2 Remarks about Vertex Features Model . . . 83
3.2.3 Models with additional Matrix Information . . . 84
3.2.4 CoshMix^3: Model with additional Vertex features . . . 87
3.3 Variational EM algorithm for CohsMix^{1,2,3} . . . 88
3.3.1 Estimation of the latent structure . . . 89
3.3.2 Estimation of the parameters . . . 89
3.3.3 Model selection: ICL algorithm . . . 91
3.4 Experiments . . . 93
3.4.1 Parameters estimation . . . 94
3.4.2 Comparison of algorithms . . . 95
3.4.3 Real data . . . 96
Conclusion . . . 100

4 A Web Application: *Constellations* **101**
4.1 Introduction . . . 103
4.2 The World Wide Web . . . 104
4.2.1 The advent of the Web . . . 104
4.2.2 The structure of the Web . . . 106
4.3 *Constellations* Resources Building . . . 109
4.3.1 Introduction to Web Search Engines . . . 110
4.3.2 Crawling and Indexing the Web . . . 111
4.3.3 Ranking the Web . . . 112
4.3.4 Reverse Web graph using a MapReduce-like framework . . . 114
4.3.5 BigGraph . . . 116
4.3.6 Graph simplification . . . 118
4.4 *Constellations* Back-end . . . 119
4.4.1 Handled Queries . . . 120
4.4.2 Graph Search . . . 122

4.4.3 Layout Algorithm . . . 123
4.4.4 Nodes attributes . . . 125
4.4.5 *XML* response of the back-end . . . 126
4.5 *Constellations* Front-end . . . 127
4.6 Services derived from *Constellations* . . . 129
Conclusion . . . 130

Conclusion **131**

A Annexes **133**
A.1 Some properties of the MixNet Model in the Bernoulli Erdös-Rényi mixture . . . 135
A.1.1 Distribution of the degrees . . . 135
A.1.2 Between-group connectivity . . . 135
A.2 Complete log-likelihood of MixNet model . . . 136
A.3 Complete log-likelihood of the Erdős-Rényi Mixture . 136
A.3.1 Directed with self-loops . . . 137
A.3.2 Directed without self-loops . . . 137
A.3.3 Undirected without self-loops . . . 137
A.3.4 Parameters update in the Bernoulli and Poisson cases for the online SAEM . . . 138
A.3.5 Parameters update in the Bernoulli and the Poisson cases for the online variational algorithm . . . 138
A.4 More experiments using simulation . . . 139
A.5 The 2008 U.S. Presidential WebSphere . . . 141
A.5.1 Connectivity matrix . . . 141
A.5.2 Alexa Traffic . . . 141
A.6 Lower bounds $\mathcal{J}$ for the CohsMix^{1,2,3} models . . . 141
A.6.1 Lower bound $\mathcal{J}$ for CohsMix^1 . . . 141
A.6.2 Lower bound $\mathcal{J}$ for CohsMix^2 . . . 144
A.6.3 Lower bound $\mathcal{J}$ for CohsMix^3 . . . 144
A.7 Optimal parameters for all CohsMix^{1,2,3} models . . . 144
A.7.1 Commun optimal parameters for CohsMix^{1,2,3} models . . 144
A.7.2 Optimal parameters for CohsMix^1 model . . . 145
A.7.3 Optimal parameters for CohsMix^2 model . . . 146
A.8 Software . . . 148
A.8.1 MixNet and MixeR . . . 148
A.8.2 CohsMix . . . 149

Bibliography **151**

List of Figures **162**

List of Tables **165**

Notation **167**

Introduction

In many scientific fields, graphs have become extremely useful as representations of a wide variety of systems. World Wide Web, gene interactions, social networks, authors citations, are examples of fields where graph representation can be used to understand the features of these complex networks. As for the Web, nodes represent websites or webpages, and each edge represents a hyperlink relating two nodes.

The latter example is not neutral and constitutes the main application field of this thesis which has been carried out in collaboration with *Exalead*, a global software provider in the enterprise and Web search markets, based in Paris since 2000. Among their activities, search engines regularly handle the Web graph representation to address many of their page processing including ranking, spam detection, search, etc. An advanced understanding of graphs allows to explain the underlying phenomenon which they are associated with. Such studies yielded the emergence of mathematical descriptions such as small-world graphs or power-law graphs that are common properties of real networks. A global overview of a network and a deeper understanding of the nature of its nodes interactions can be obtained by the study of its structure or topology.

Many strategies have been developed for this purpose and model-based clustering has provided an efficient way to summarize complex networks structures. The basic idea of these strategies is to model the distribution of connections in the network, considering that nodes are spread among an unknown number of connectivity classes which are themselves unknown. However, the main drawback of these methods is that they suffer from relatively slow estimation procedures since dependencies are complex. This forbids to deal with networks composed of thousands of nodes, if not more, with a reasonable speed of execution. To circumvent this problem, online algorithms are en efficient alternative to classical batch algorithms. The adaptation of online estimation strategies, originally developed for the Expectation-Maximisation (EM) algorithm, to graph models are then an interesting optimization approach to the statistical analysis of large networks.

Most techniques for clustering graph vertices only use the topology of connections, while ignoring information about the features of vertices. Using statistical models with a latent structure characterizing each vertex both by its connectivity and by a vector of features, one can use both these elements to cluster the data and estimate the model parameter. For the Web graph where each vertex represents a page containing occurrences of certain words, structure can either be then described in terms of the hyperlinks between web pages, or by the words occurring in the web page.

Successful experiences on the Web graph structure encourage us to create an online application which could extract and visually explore the

connectivity information induced by hyperlinks. This application could then offer a new way to browse search engine results.

In summary, our contributions are the following,

- Online EM algorithms for random graphs which are based on a mixture of distributions;
- Statistical models with a latent structure characterizing each vertex both by its connectivity and by a vector of features (or an additional information) and their related estimation procedure;
- *Constellations*, an online application intended for the *Exalead* search engine, which is able to reveal the connectivity information induced by the hyperlinks of a user request.

All of these contributions are developed in the four chapters of this thesis.

The *first chapter* outlines the problem of graph clustering which consists in assigning the vertices of a graph into clusters taking its edge structure into account. Since graph clustering literature is very large, this chapter does not claim to give a comprehensive overview of all the exiting methods, but rather a general description of the approaches commonly applied.

The *second chapter* describes in detail a statistical model, called MixNet, which is used, throughout this thesis, for the description of the network topology. Since efficient computational procedures are required in order for structures to be learned, this chapter adapts online estimation strategies. Our work focuses on three methods, the first based on the CEM algorithm, the second on the SAEM algorithm, and the third on variational methods. We perform a simulation to compare these three algorithms with existing approaches and we use these methods to decipher the structure of the French and US political websites network. We show that our online EM-based algorithms offer a good trade-off between precision and speed when estimating parameters.

In the *third chapter*, we propose clustering algorithms that harness both the topology of connections and the information about the features of vertices. Simulations are carried out to compare our algorithms with existing approaches. Once again, hypertext documents are used as real data sets. We find that our algorithms successfully exploit whatever information is found both in the connectivity pattern and in the features.

The *fourth chapter* is devoted to the online *Constellations* application. Since web content authors tend to link to pages with similar topics or points of view, this service based on the hyperlink topology offers a new way to browse the *Exalead* search results. Indeed, in this application, the connectivity information induced by hyperlinks between the first hits of a given search request is extracted, visually explored and analysed *via* the MixNet algorithms. Since the Web is an open and large scale hypertext system with billions of nodes and links, this chapter will also focus on how to deal with a huge amount data and produce an online service capable of responding in a very short time.

This thesis has been the subject of various publications: three pub-

lished papers (Zanghi et al. 2008; 2010a;b) and three conference proceedings.

Furthermore, the *Constellations* application, currently available at `http://constellations.labs.exalead.com` allows to promote certain aspects of this thesis.

Préambule

Dans de nombreux domaines scientifiques, les graphes sont des outils incontournables de représentation d'une grande variété de systèmes. Le Web, les interactions entre les gènes, les réseaux sociaux, les cocitations entre les articles scientifiques sont des exemples pour lesquels cette représentation peut être utilisée dans le but de comprendre les caractéristiques de ces réseaux complexes. Dans le cas du Web, les nœuds représentent ainsi des sites Web ou pages, et chaque arête représente un lien hypertexte reliant deux nœuds.

Ce dernier exemple n'est pas anodin et constitue le principal contexte applicatif de cette thèse réalisée en collaboration avec *Exalead*, fournisseur de logiciels de recherche et d'accès à l'information en entreprise et sur le Web, basé à Paris depuis 2000. Parmi leurs nombreuses opérations, les moteurs de recherche matérialisent régulièrement le Web sous forme de graphe pour appliquer divers traitements aux pages y compris la classification, la détection de spams ou encore la recherche. Une compréhension avancée de ces graphes permet d'expliquer les phénomènes sous-jacents auxquels ils sont associés. Ces études ont permis l'émergence de descriptions mathématiques telles que les phénomènes de "petit monde" ou de distribution en loi de puissance qui sont des propriétés communes aux réseaux réels. L'étude de la structure ou de la topologie d'un réseau en donne une vision globale qui permet une connaissance approfondie de la nature des interactions entre ses nœuds.

Les techniques de classification automatique, dites de *clustering*, basées sur l'utilisation de modèles statistiques ont fourni un moyen efficace de résumer les structures complexes des réseaux. L'idée de base de ces stratégies est de modéliser la distribution des connexions dans le réseau, en considérant que les nœuds sont répartis parmi un nombre inconnu de classes de connectivités, elles-mêmes inconnues. Cependant, l'inconvénient principal de ces méthodes concerne les procédures d'estimation relativement lentes dûes aux dépendances complexes. Ceci empêche de traiter des réseaux composés de milliers de nœuds, si ce n'est plus, avec une rapidité d'exécution raisonnable. Les algorithmes incrémentaux où les données arrivent au fur et à mesure sont des alternatives efficaces aux algorithmes classiques qui traitent dans la globalité un lot de données. L'adaptation des stratégies d'estimation incrémentales, initialement développées pour l'algorithme EM, aux modèles de graphes sont ainsi des approches d'optimisation intéressantes aux analyses statisitiques de grands réseaux.

La plupart des techniques de *clustering* des nœuds d'un graphe utilisent uniquement la topologie de leurs connexions, en omettant l'information que ces nœuds pourraient contenir. En utilisant des modèles

statistiques à structure latente qui caractérisent chaque nœud à la fois par sa connectivité et par un vecteur de caractéristiques (ou un complément d'information), nous pouvons utiliser ces deux éléments pour grouper les données et estimer les paramètres du modèle. Pour le graphe du Web où chaque sommet représente une page contenant des occurrences de certains mots, la structure peut alors être définie par rapport aux hyperliens entre les pages Web ou par rapport à la similarité des mots employés entre celles-ci.

Les expériences réussies sur l'analyse de la structure du graphe Web nous ont encouragées à créer une application en ligne qui pourrait extraire et explorer visuellement l'information de la connectivité induite par les hyperliens. Cette application proposerait ainsi une nouvelle manière de parcourir les résultats d'un moteur de recherche.

En résumé, nos contributions concernent,

- Des algorithmes EM incrémentaux pour les graphes aléatoires qui sont basés sur des mélanges de distributions;
- Des modèles statistiques avec une structure latente caractérisant chaque sommet à la fois par sa connectivité et par un vecteur de caractéristiques (ou un complément d'information) et leurs procédures d'estimations associées;
- *Constellations*, un service en ligne destiné au moteur de recherche d'*Exalead*, qui est en mesure de révéler les informations de connectivité induite par hyperliens entre les résultats d'une requête utilisateur.

L'ensemble de ces contributions est développé au cours des quatre chapitres de cette thèse.

Le *premier chapitre* décrit brièvement le problème de *clustering* de graphe qui consiste à assigner les nœuds dans des grappes en prenant en considération la structure des arêtes. Puisque la littérature autour de ce problème est dense, l'ambition de ce chapitre est d'avantage de proposer une description générale des approches couramment appliquées que de prétendre donner un aperçu complet de toutes les méthodes existentes.

Le *deuxième chapitre* décrit en détail le modèle statistique MixNet qui est utilisé tout au long de cette thèse pour la description de la structure des réseaux. Etant donné que des procédures efficaces de calcul sont nécessaires pour appréhender ces structures, ce chapitre s'intéresse aux stratégies d'estimation incrémentales. Notre travail se concentre sur trois méthodes, la première basée sur l'algorithme de CEM, le deuxième sur SAEM, et le troisième sur les méthodes variationnelles. Nous réalisons des simulations afin de comparer ces trois algorithmes avec les approches existantes, et nous les utilisons pour déchiffrer la structure des réseaux créés par les sites politiques français et américains. Nous montrons que nos algorithmes EM incrémentaux offrent un bon compromis entre précision et vitesse lors de l'estimation des paramètres.

Dans le *troisième chapitre*, nous proposons des algorithmes de *clustering* qui utilisent la topologie des connexions ainsi que des informations sur les caractéristiques des nœuds. Nous réalisons des simulations pour

comparer les algorithmes existants avec nos propres algorithmes. Nous évaluons également ces derniers avec des données réelles basées sur des documents hypertextes. Nous constatons que nos algorithmes exploitent les informations avec succès quel que soit l'endroit où elles se trouvent: la connectivité des arêtes ou les caractéristiques des sommets.

Le *quatrième chapitre* est consacré au service en ligne *Constellations*. Étant donné que les auteurs de contenu Web ont tendance à lier des pages avec des sujets ou points de vue similaires, ce service, basé sur la topologie hypertexte, offre une nouvelle façon de naviguer dans les résultats de recherche d'*Exalead*. En effet, dans cette application, la connectivité induite par des hyperliens entre les premiers résultats suite à une requête donnée est extraite, explorée et analysée visuellement *via* les algorithmes présentés dans cette thèse. Composé de milliards de nœuds et de liens, le Web est un système hypertexte vaste et ouvert. Ce chapitre se concentrera également sur la façon de traiter une telle quantité de données dans le but de fournir un service en ligne capable d'opérer en un temps très court.

Cette thèse a fait l'objet de divers travaux écrits: trois articles publiés (Zanghi et al. 2008; 2010a;b) et trois conférences.

De plus, l'application *Constellations*, actuellement disponible à l'adresse `http://constellations.labs.exalead.com` permet de promouvoir certains aspects de cette thèse.

Graph clustering Survey

1

Contents

2.1 Introduction . . . 39
2.2 Mixture Model and the EM algorithm . . . 40
2.2.1 Finite Mixture Model . . . 40
2.2.2 The EM algorithm . . . 43
2.3 A Mixture Model for Networks . . . 47
2.3.1 Model and Notation . . . 47
2.3.2 Sufficient statistics and Online Recursion . . . 51
2.3.3 Likelihoods and online inference . . . 52
2.4 Classification EM algorithm . . . 53
2.4.1 Classification log-likelihood . . . 53
2.4.2 Online Estimation . . . 54
2.4.3 Bernoulli affiliation model . . . 56
2.4.4 Initialization and online supervised classification . . . 57
2.5 Stochastic Approximation EM for Network Mixture . . . 57
2.5.1 A short presentation of SAEM . . . 57
2.5.2 Simulation of $\Pr(\mathbf{Z}|\mathbf{X})$ in the online context . . . 58
2.5.3 Computing $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ in the online context . . . 58
2.5.4 Maximizing $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$, and parameters update . . . 59
2.6 Application of online algorithm to Variational EM methods . . . 59
2.6.1 Online variational step . . . 60
2.6.2 Maximization/Update step . . . 61
2.7 Choosing the number of clusters . . . 61
2.8 Experiments using simulation . . . 62
2.8.1 Comparison of partitions . . . 63
2.8.2 Comparison of algorithms . . . 63
2.9 Application . . . 68
2.9.1 French Political Blogosphere network . . . 68
2.9.2 The 2008 U.S. Presidential WebSphere . . . 71
Conclusion . . . 77

The task of assigning a set of objects into groups such that they are composed by similar objects is called *clustering*. Common technique for statistical data analysis, clustering belongs to *unsupervised learning* methods and is used in many field including machine learning, data mining, pattern recognition, image analysis and bio informatics. In general, the clustering is based on some similarity measure defined for data elements.

Graphs are mathematical structures formed by a collection of *vertices* (also called *nodes*) and a collection of edges that are connections between pairs of vertices. The task of *graph clustering* consists in assigning the vertices of the graph into clusters taking into consideration the edge structure of the graph and it is the subject of this survey (See Figure 1.1).

1.1 Introduction

The problem of graph clustering, sometimes confined to community detection in graphs, has a long tradition and it has appeared in various forms given the scientific fields. It should not be confused with the clustering of sets of graphs based on structural similarity. Such clustering of graphs as well as measures of graph similarity is addressed in other literature, for instance Robles-Kelly and Hancock (2005) where the technique used is closely related to the task of detecting clusters within a given graph.

The first study of community structure might be dated back to the work of Rice (1927) where using similarity of voting pattern, people are grouped in political clusters. From there, regrouping vertices according to their mutual similarity, many traditional techniques to find communities in social networks have been proposed. Such techniques like hierarchical or partitional clustering are explained in this Chapter 1.5.1.

Nevertheless, the first algorithm especially designed for detecting community structure might be associated to Weiss and Jacobson (1955). Interested by the detection of work groups within a government agency, the authors examined the matrix of working relationships between members of the agency, which were identified by means of private interviews. Briefly, they proposed to remove the members working with people of different groups in order to identify the work clusters. Particularly intuitive, this approach of cutting connectors between groups is now involved in numerous modern algorithms and Section 1.5.2 presents the key points of such a strategy.

Initially designed for social networks, the identification of structure in graphs is also a popular topic in computer science. For instance, a classical parallel computing problem consists of dividing the computer networks into clusters, such that the clusters are of about same size of processors and there exists very few connections between these clusters. Explained in Section 1.5.1, this graph partitioning allows to minimize the communications between the clusters and increase the use of CPUs. The first algorithms for graph partitioning were proposed in the early 1970's.

In 2002, Girvan and Newman proposed in a seminal paper (Girvan and Newman 2002) a new algorithm, aiming at the identification of edges lying between communities and their successive removal, a procedure that after a few iterations leads to the isolation of the communities. In this approach, the authors detect the inter-community edges according to the values of a centrality measure, the *edge betweenness*. This measure expresses the importance of the role of the edges in processes where signals are transmitted across the graph following paths of minimal length. Following this paper, a large amount of methods has been proposed in the last years including spin models, random walks, etc. This chapter focuses on a few elements of understanding of these tools and technique (Section 1.5.5). During the last years, the field has also taken advantage of concepts and methods from computer science, biology, sociology, discrete mathematics.

Although no single definition of "community" is universally accepted, *modularity* is one of the most popular function which quantifies the quality of a division of a network into modules. The basic idea of this concept is to compare the number of edges inside a cluster with the expected number

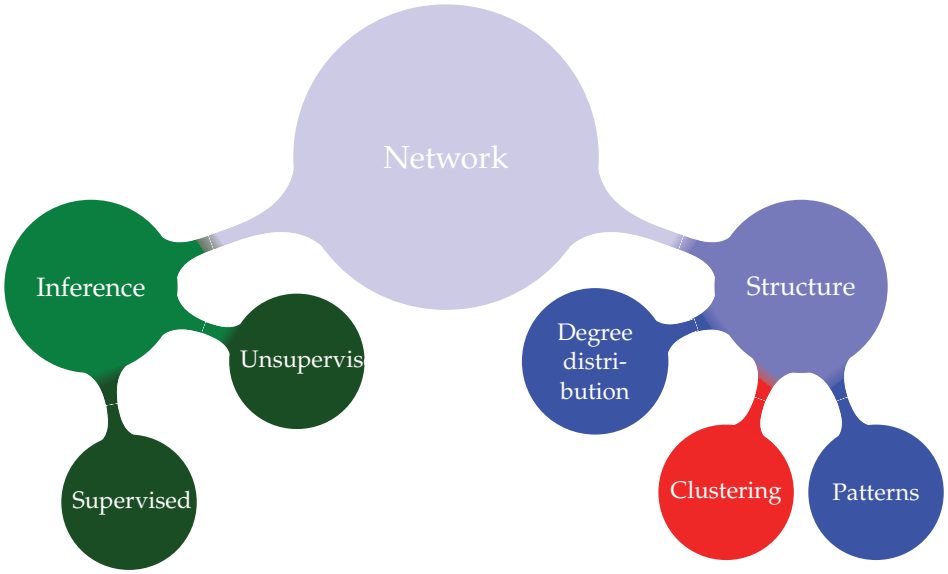


Figure 1.1 – *Network studied problems*

of edges that one would find in the cluster if the network were a random network with the same number of nodes and where each node keeps its degree, but edges are otherwise randomly attached. Many algorithms, like Girvan and Newman (2002), optimize this function to perform a clustering of vertices.

Transformation of the adjacency matrix are often done to detect communities. One might cite methods related to spectral clustering (Ng et al. 2002) and variants (Boulet et al. 2008) which take benefit of the Laplacian matrix or methods related to random walk where the probabilities of the transitions matrix are the basis of the approach (Dongen 2000).

A distinction can be made between model-free (Newman 2006a, Ng et al. 2002) and model-based methods. Among model-based methods, model-based clustering has provided an efficient way to summarize complex networks structures. The basic idea of these strategies is to model the distribution of connections in the network, considering that nodes are spread among an unknown number of connectivity classes which are themselves unknown. This generalizes model-based clustering to network data, and various modeling strategies have been considered (Nowicki and Snijders 2001, Daudin et al. 2008, Airoldi et al. 2008, Handcock et al. 2006). EM like algorithms constitute a common core of the estimation strategy of these graph modelings (Snijders and Nowicki 1997). Since EM is known to be relatively slow, Chapter 2 will focus particularly on a model-based modeling (Daudin et al. 2008) and its online estimation strategies.

Since the graph clustering field has has grown quite popular and the number of published proposals for clustering algorithms is high, this chapter do not even pretend to be able to give a comprehensive overview of all the exiting methods, but rather an explanation of the methodologies commonly applied and pointers to some of the essential publications related to each research branch. More complete graph clustering reviews could be found in Schaeffer (2007) and Fortunato (2009).

Outline of the chapter We first begin this chapter by providing illustration of graph structures in real networks and it is supposed to exhibit the problem and its relevance to many different fields. Next, graph terminology and basic definitions are provided to prepare the reader to the description of the graph clustering algorithms. Then, we begin our algorithm review with traditional clustering methods, i. e. graph partitioning, hierarchical and partitional clustering. A review of methods is divided into sections based on approaches on which they refer. Since many algorithms may enter into several categories, the classification is based on what we believe is their main feature/purpose, even if other aspects may be present. Finally, we finish this chapter by discussing the model-based methods which provides the basis of the future algorithms described in this thesis.

1.2 Real-networks and communities

In many scientific fields, systems can be modeled using networks to represent data relationships. World-wide-web, gene interactions, social networks, authors citations, are examples of fields where graph representation helps interpreting relationships between the nodes. In this section, some striking examples of real-networks with community structure are illustrated. The underlying idea is to make the reader understand the usefulness of automatically detecting these structures. After these examples, we also recall some properties that real-networks share together.

Community structure Firstly, note that the word "community" itself refers to a social context. People naturally tend to form groups, within their work environment, family, friends. However, it will sometimes be employed in completely different contexts to describe any grouping of vertices. Besides, it appears that as social networks any real-networks share the property of owning a high clustering coefficient: my neighbours have a high probability to be neighbours themselves. This property creates important aggregative trend of a graph which takes a special meaning depending on the context: biological, social, etc.

Thereby, we begin the illustration of real-networks along community structure in the social context with Figure 1.2. This first example concerns the Zachary karate club network which is a well-known dataset often used as a benchmark to test graph clustering algorithms. It represents the friendships between 34 members of a karate club at a US university in the 1970s who were observed during a period of three years. Edges connect members who were observed to interact outside the activities of the club. A clash between the president of the club and the instructor led to the fission of the club in two opposed groups: one supporting the instructor and the other the president, respectively (indicated by squares and circles). This fission can be observed on Figure 1.2 where we can distinguish two aggregations. The first one is around vertices 33 and 34 (the president), and the other is around vertex 1 (the instructor). From such an observed network, the purpose of a community detection algorithm is to suggest the split into two groups. We can also note that the two

main structures are linked by several vertices (like 3,9,10) which might be misclassified by graph clustering methods.

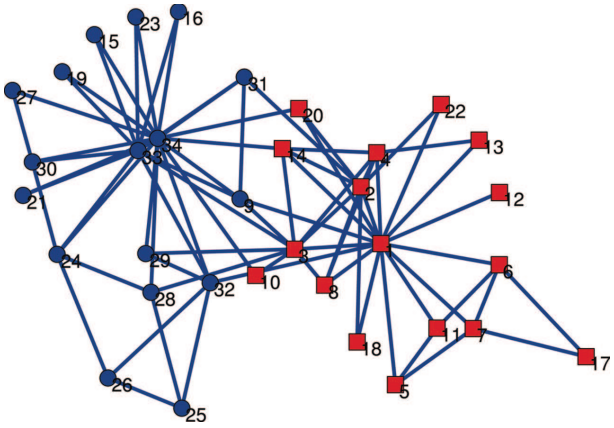


Figure 1.2 – *Zachary's karate club. The colors correspond to the partition found by optimizing the modularity of Newman and Girvan for two groups.*

Next, we propose an example of information flows within the brain which is subject of intense investigations in biology and bio-informatics. Since the interactions between cortical regions are fundamental for each brain function, studying the way cortical regions interact may offer new research perspectives. Figure 1.3 illustrates 47 brain cortical regions connected by 505 inter-regional pathways in the Macaque Cortex (Sporns et al. 2007). It seems that particular brain regions may act differently. Indeed, we can observe that some regions are particularly linked to others causing "central" and more "peripherical" parts in the network. In biological terms, since a lot of information will pass through "central" zone identifying them can be crucial, as their lesion may compromise the integrity of the whole network (Picard et al. 2009). In this example, the authors also note that communities can be related to geographic areas in the cortex. This can be explained by the geographic organization of the connections within the brain.

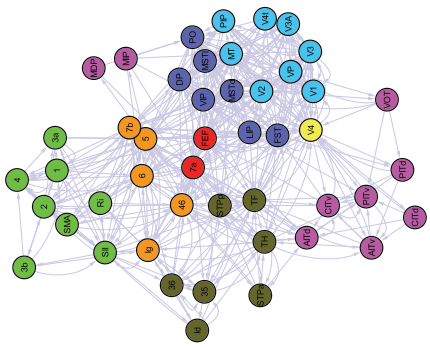


Figure 1.3 – *Macaque cortex network displayed with colors for each MixNet class (8 groups) from the paper of Daudin et al. (2008).*

Relationships between elements of a system are not necessary reciprocal. As an example, the direction of the relationships in predator-prey food webs have to be precise in order to understand the system as a whole. Another example extracted from the computer science fields is proposed

in Figure 1.4. The considered system is the World Wide Web, which can be seen as a directed graph by representing Web pages as vertices and the hyperlinks, that allow users to navigate among pages, as directed edges (Kleinberg et al. 1999). Empirically, we can observe that less than 10% of the hyperlinks are reciprocal. Communities of the Web graph are groups of pages having topical similarities. Chapter 4 discusses in detail an application based on the Web pages.

Characterized by symmetrical matrices (adjacency matrix, Laplacian, etc.), all structure detection algorithms can not be easily extended from the undirected to the directed case. Thus, it is preferable to properly formulate the problem to solve before designing a strategy.

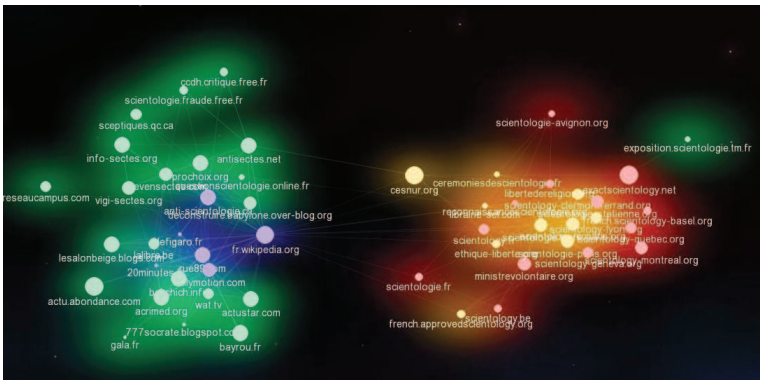


Figure 1.4 – *Community structure in a hypertext document network. Groups, indicated by the colors, were detected with the CEM algorithm of Zanghi et al. (2008)*

In the previous examples, one might be interested in weighting. For instance, the edges of Zachary's network of karate (Figure 1.2) could be weighted by the number of times individuals interacted outside the activities of the club. Similarly, the edges of the co-authorship network in Figure 1.5 could be weighted by the number of papers coauthored by pairs of scientists. Note that in this example, the collaboration dataset is based on the largest connected component (379 nodes) of the network of scientists (1,589 total nodes), determined by coauthorship of papers listed in few articles (for instance (Boccaletti et al. 2006)).

Weights are truly valuable information for edge structure of the graph and should be considered by the community detection algorithms. Thus, algorithms handling weighted graph are generally preferred. However, in many cases methods working on unweighted graphs can be simply extended to the weighted case. Besides, vertices can also embed crucial informations which can be combined with the graph structure to build coherent groups. Chapter 3 will propose algorithms for clustering data sets with a graph structure embedding vertex features.

Last but not least, we can cite the problem of overlapping communities which is not considered in this thesis. In such situation, we have to introduces a further variable, the membership of vertices in different communities that enormously increases the number of possible covers with respect to standard partitions. Therefore, searching for overlapping communities is much more computationally demanding than detecting standard partitions (Latouche et al. 2009).

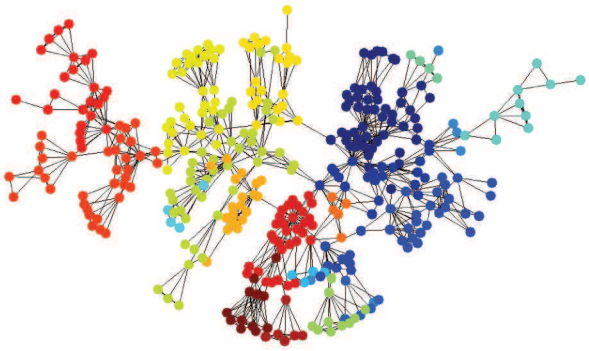


Figure 1.5 – *Co-authorship network : Each of the nodes in the network, which we depict using a Kamada-Kawai visualization(Kamada and Kawai 1989), is colored according to its community assignment using a spectral method algorithm.*

Thereby, besides the well-illustrated property of high clustering coefficient, others properties that are shared by real-networks can be listed.

Other properties of real-networks Although these properties are not in the scope of our work, we briefly recall some properties shared by these so-called real-networks. Some of them will be detailed for the World Wide Web graph dataset when we will describe our application in Chapter 4.

- **Tree-likeness:** The number of edges is linear in the number of nodes.
- **Giant component:** Connected subgraph that contains a majority of the entire graph's nodes
- **Scale-freeness:** The distribution of nodes in the network follows a "power law" distribution (a few nodes have many links, and many nodes have few links).
- **Preferential attachment:** New nodes are free to associate with any nodes, but "prefer" to associate with well-connected nodes (nodes that already have many connections).
- **Small World:** Most nodes are close to each other (six degrees of separation in social networks).

These properties are well-studied in the literature and details can be found in Amaral et al. (2000), Newman (2003), Barabasi and Crandall (2003), Watts (2004)

1.3 Terminology and definitions

We first propose a review of the graph terminology in order to facilitate future discussion in the rest of this chapter. Some of the basic definitions of

computational complexity, approximation algorithms, graph theory, and Markov chains are provided. Obviously, we advise readers familiar with the graph terminology to proceed directly to Section 1.4 page 21.

1.3.1 Graph theory

A graph G is a combination of sets $G = (V, E)$ where V is the set of vertices with the number of vertices $n = |V|$ and E is the set that contains the edges of the graph. As mentioned in the real-networks examples, the relationships direction can be crucial. In this case, each edge is an ordered pair $\{u, v\}$ and the set of edges form a *directed graph* or (also called a *digraph*). In the other case, they are unordered pairs and form a *undirected graph*. The size of the graph is determined by the edge count $|E| = m$. In a *weighted graph*, a weight function $w : E \rightarrow \mathbb{R}$ is defined that assigns a weight on each edge. A *planar* graph is a graph that can be drawn on the plane in such a way that its edges intersect only at their *endpoints*.

A *dense* graph is a graph in which the number of edges is close to the maximal number of edges. The opposite, a graph with only a few edges, is a *sparse* graph. Thereby, we define the *density* of a graph $G = (V, E)$ as the ratio of the number of edges present over the maximum possible,

$$\delta(G) = \frac{2.m}{n.(n-1)} \tag{1.1}$$

Since, the maximum number of edges is $m = 1/2.n.(n-1)$, the maximal density is 1 (for *complete graphs*) and the minimal density is 0 (Coleman and Moré 1984).

In graph theory, vertex u is a neighbour of vertex v if edge $\{u, v\} \in E$. The *neighbourhood* for a given vertex v is defined by the its set of neighbours and is denoted by $N_G(v)$. Note that a vertex v is a member of its own neighbourhood $N_G(v)$ if and only if the graph contains a reflexive edge $\{v, v\}$ (also called a *self-loop*).

A *path* from vertex v to vertex u in a graph $G = (V, E)$ is a sequence of edges in E starting at vertex $v_0 = v$ and ending at vertex $v_{k+1} = u$;

$$\{v, v_1\}, \{v_1, v_2\}, \cdots, \{v_{k-1}, v_k\}, \{v_k, u\}.$$

A path with no repeated vertices is called a *simple path* and a *cycle* corresponds to a path such that the first node of the path corresponds to the last. A graph with no cycle is defined as *acyclic* (also called a *forest*). A connected *acyclic* graph is called a *tree*.

The *length* of a path is the number of edges on it, and the *distance* between vertices v and u is the length of the *shortest* path connecting them. Note that the distance from a vertex to itself is zero. If there exist paths between all pairs of vertices then the graph is *connected*. Otherwise, the graph is *disconnected*. Given a graph G, the minimum number of edges that would need to be removed in order to make it disconnected is defined as the *edge-connectivity* of the graph.

The *degree* of a given vertex v is derived from its number of incident edges and is denoted by $deg(v)$. When every vertex of a given graph have

the same degree, the graph is called *regular*. A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k.

A partition of the vertices V of a graph $G = (V, E)$ into two disjoint subsets S and $V \setminus S$ is called a *cut* and is denoted by $(S, V \setminus S)$. As the sets S and $V \setminus S$ define the same cut, it is often preferred to denote by S the *smaller* set, hence requiring $|S| \leq \frac{n}{2}$. The *size* (or weight) of a cut $(S, V \setminus S)$ is the number of edges crossing the cut and is defined by

$$c(S, V \setminus S) = |\{v, u\} \in E | u \in S, v \in V \setminus S|.$$

Note that in a weighted graph, the same term is defined by the sum of the weights of the edges crossing the cut.

A *subgraph* $G^S = (S, E_S)$ of a graph $G = (V, E)$ is a graph whose vertex set $S \subseteq V$ is a subset of that of G, and whose edge set $E_S \subseteq E$ is a subset of that of G restricted to this subset. In such a context, the graph G is a *supergraph* of G^S

An *induced subgraph* of a graph $G = (V, E)$ is a subset of the vertices $S \subseteq V$ with an edge set $E(S)$ that includes all such edges $\{v, u\}$ in E with both of the vertices v and u included in the set S:

$$E(S) = \{\{v, u\} | v \in S, u \in S, \{v, u\} \in E\}.$$

We denote the subgraph induced by the vertex subset S by $G(S)$ (or by G^S where it is clear that the subgraph is an induced subgraph). A *complete graph* is a graph in which each pair of graph vertices is connected by an edge. A *clique* is a complete induced subgraph. According to Equation 1.1, the *local density* of an induced subgraph in $G = (V, E)$ is defined by

$$\delta(G(S)) = \frac{|E(S)|}{\binom{|S|}{2}}$$

An *isomorphism* of graphs $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$ is a bijection between the vertex sets $f : V_i \rightarrow V_j$ such that $\{v, w\} \in E_i$ if and only if $\{f(v), f(w)\} \in E_j$. With such a one-to-one correspondence, the two graphs G_i and G_j are said to be *isomorphics*

The *adjacency matrix* is frequently used to represent which vertices of a graph are adjacent to which other vertices. The *adjacency matrix* of a given graph $G = (V, E)$ on n vertices is the $n \times n$ matrix denoted $\mathbf{A}_G = (a^G_{v,u})$ where

$$a^G_{v,u} = \begin{cases} 1, & \text{if } \{v, u\} \in E \\ 0, & \text{otherwise.} \end{cases}$$

The set of graph *eigenvalues* of the adjacency matrix $\mathbf{A}_G$ is called the *spectrum* of the graph. Although spectral properties can be computed for both undirected and directed graphs, as well as unweighted, this section provides elements of the spectral graph theory involved in the easiest case (undirected and unweighted simple graphs). Studying the eigenvalues of the *Laplacian* matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}_G$ is often more convenient than those of $\mathbf{A}_G$ itself (Chung 1997). The *normalized* Laplacian is defined as

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_G\mathbf{D}^{-\frac{1}{2}} \tag{1.2}$$

where $\mathbf{I}$ is a $n \times n$ *identity matrix* (with ones on the diagonal, other elements being zero) and $\mathbf{D}$ is the $n \times n$ *degree matrix* of graph G (with vertex degree on the diagonal, other elements being zero). A normalized version of the Laplacian matrix is similarly defined by

$$\mathcal{L}_{uv} = \begin{cases} 1, & \text{if } u = v \text{ and } deg(v) > 0, \\ -\frac{1}{\sqrt{deg(u).deg(v)}}, & \text{if } u \in \Gamma(v), \\ 0, & \text{otherwise.} \end{cases}$$

Eigenvalues of such symmetrical matrices are real and non-negative. Using the normalized Laplacian is convenient as all the eigenvalues of $\mathcal{L}$ lie the in the interval $[0, 2]$. As the matrix is singular, the smallest eigenvalue is always zero, and the corresponding eigenvector is simply a vector with each element being the square-root of the degree of the corresponding vertex. A comprehensive introduction to this field can be found in Chung (1997) and application of spectra in real-world graphs in Farkas et al. (2001)

1.3.2 Markov chains

A *Markov chain* is a random process where all information about the future is contained in the present state, *i.e.* the past is not examined to determine the future which only depend on the current state. The changes of states are called *transition* and the associated probabilities form the *transition matrix* of the Markov chain. A convenient way to represent such matrix is to use a weighted directed graph where each state corresponds to a vertex and each edge weight corresponds to the probability (nonzero) of the transition between both concerned states.

When considering an unweighted graph, the transition probability for moving from one vertex v to any one its neighbours is chosen uniformly at random with the probability $1/deg(v)$. This means that the probability for moving from vertex v to w is simply

$$p_{v,w} = \begin{cases} \frac{1}{deg(v)}, & \text{if } w \in N_G(v), \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

Note that transition matrix $\mathbf{P}$ that results from this construction can also be seen as a *normalized* adjacency matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}_G$ of the graph G. Such transition matrix is generally called *random walk*. However, there exists many possible definitions of walk in graphs and (Lovasz 1993) proposes an interesting survey about them (and theirs interpretations).

The *first passage time* is the time taken for a random walker to reach a specified target and the *mixing time* is the time until the Markov chain is "close" to its *stationary distribution*. The latter defines for each state the probability that the walk is at that state if a single observation is made after the walk has been run for a sufficiently long time. This distribution can be obtained by computing the *dominant* eigenvector corresponding to the *largest* eigenvalue of the transition matrix. The primary eigenvalue λ_1 of any transition matrix is one, as is the case for any stochastic matrix.

To get more insight to the mathematics see for example Behrends (1999), Grimmett and Stirzaker (2001). As one application, we shall describe in Chapter 4 how the stationary distribution can be used to rank the pages of the the World Wide Web.

1.3.3 Computational complexity

Classifying computational problems according to their inherent difficulty is related to the computer science called *computational complexity*. Its interest is in characterizing how the running time and memory consumption of the problem grow when input of size n grows. Except in rare situations, the complexity of an algorithm is usually taken to be its worst-case complexity. Let n be a positive integer, to show a upper bound $\Theta(g(n))$ of the problem solving, one needs to show only that there is a particular algorithm $f(n)$ with running time such that $|f(n)| \leq \Theta(g(n))$ for all sufficiently large values of n. Then, to show a lower bound of $\Omega(g(n))$ for a problem requires showing that no algorithm $f(n)$ can have time complexity $|f(n)| \geq \Omega(g(n))$ for all sufficiently large values of n. Finally, we write $f(n) = O(g(n))$ if both $f(n) = \Theta(g(n))$ and $f(n) = \Omega(g(n))$ hold. The big Oh notation allows to round the function with hiding constant factors and smaller terms. For more insight to the complexity, the related definition and notations, we recommend the basic textbook on algorithms by (Cormen et al. 2001).

A *decision problem* is a problem in a formal system with a "yes" or "no" answer, depending on the values of some input parameters. A problem is said to be in class **P** (for ***P**olynomial*), if it has an algorithm able to answer with time complexity bounded by some polynomial of the input size n. Unfortunately, there exist many important problems which have no known polynomial-time algorithms. Nevertheless, such a problem may still have a polynomial-time verification algorithm in that can check a feasible solution a problem instance of size n. Precisely, such proofs have to be verifiable in polynomial time by a deterministic Turing machine(Papadimitriou and Papadimitriou 1994). Practically, this means that problems with polynomial-time verification algorithms form the class **NP** (for ***N**on-deterministic **P**olynomial*) which contains class **P**.

Besides, a decision problem S which is reducible to a decision problem T using a polynomial-time reduction f such that for any $n \in S, f(n) \in T$, is denoted by $S \neq_m^p T$. A problem T is **NP**-hard if $S \neq_m^p T$ for all problems $S \in$ **NP** (any **NP**-problem can be translated into this problem). A problem which is both **NP** and **NP**-hard is said to be **NP**-complete. Further detail of on **NP**-completeness and the complexity class could be found in Papadimitriou and Papadimitriou (1994) .

As many clustering algorithms are **NP**-hard, exact algorithms can only be employed in very small systems. In such context or to accelerate **P** algorithms, *approximation algorithms* are regularly preferred. Delivering an approximate solution, with the advantage of a lower complexity, they are often *non-deterministic* and commonly used for optimization problems, in which one wants to find the maximum or minimum value of a given cost function over a large set of possible system configurations.

1.4 Graph Clustering

Intuitive at first sight, the problem of graph clustering is actually not well defined. As the the concepts of *community* and *partition* are not rigorously defined, there are many equally legitimate ways of resolving this problem. A primary cause of these ambiguities is that the identification of communities is possible using the concepts of edge density (for example inside versus outside the community), or using the concepts of similarity between each pair of nodes (related to data clustering). Thereby, classical methods for data clustering, sometimes adopted for graph clustering will be discussed later in the review as well as edges related methods. However, before attempting an ordered exposition of the fundamental algorithms of community detection, we shall review the notions of community and partition.

1.4.1 Communities

The first problem in graph clustering is that no single definition of "community" is universally accepted. Indeed, most of the time, communities are algorithmically defined, that means they are just the final result of the algorithm, without a precise *a priori* definition. Firstly, each community should be connected. However, the most common and intuitive definition consists in finding more edges "inside" the community than edges linking vertices of the community with the rest of the graph. Formally, this definition can be tackled as follows.

Let us start with a subgraph C of a graph G , with $|C| = n_c$ and $|G| = n$ vertices. Let $\delta_{int}(C)$ be the intra-cluster density of the subgraph C defined by

$$\delta_{int}(C) = \frac{\text{\# internal edges of } C}{n_c(n_c - 1)/2},$$

which corresponds to the ratio between the number of internal edges of C and the number of all possible internal edges. Similarly, let $\delta_{ext}(C)$ be the inter-cluster density defined by

$$\delta_{ext}(C) = \frac{\text{\# inter-cluster edges of } C}{n_c(n - n_c)},$$

which corresponds to the ratio between the number of edges running from the vertices of C and the rest of the graph and the maximum number of inter-cluster edges possible.

For C to be a community, $\delta_{int}(C)$ has to be larger than the average link density $\delta(G) = m/n.(n-1)$ of G (with m the number edges in G), and $\delta_{ext}(C)$ has to be smaller than $\delta(G)$.

Obviously, each community C should be connected: there should be at least one, preferably several paths connecting each pair of vertices. Thus, on a disconnected graph with several known components, the clustering should usually be conducted on each component separately.

After a recall of this basic notion, we now aim at introducing the main definitions of community which have been proposed along the years by social network analysts, computer scientists and physicists. Actually, three

classes of definitions can be distinguished: *local*, *global* and based on *vertex similarity*. This review is not exhaustive and some other definitions will be introduced with their associated algorithms during this chapter.

Local Definitions

As communities are expected to share more relationships with themselves than with the whole graph, it is coherent to propose local definitions of community where the immediate neighbourhood of the subgraph under study is considered, contrary to the rest of the graph.

The most acknowledged definitions come from social network analysis and Wasserman et al. (1994) proposes a classification based on the identified criteria : *complete mutuality*, *reachability*, *vertex degree* and the *comparison of internal versus external cohesion*.

We start with the complete mutuality definition which actually corresponds to a *clique*, *i.e.* a subset whose vertices are all adjacent to each other (See Figure 1.6). In a social context, It insists that every member or a sub-group have a direct tie with each and every other member (Luce and Perry 1949).

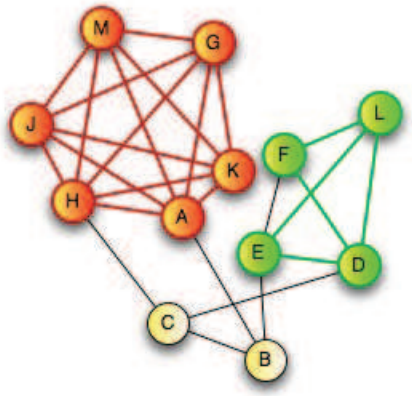


Figure 1.6 – *Example of cliques: the green clique is the maximal clique over the set of vertices,* $\{F, D, L\}$*. There is no larger possible clique in the graph containing those three vertices than the green 4-clique. But the maximal clique in G is actually the 6-clique containing the vertices* $\{A, G, H, J, K, M\}$*.*

Finding whether there is a clique of a given size in a graph (also called the clique problem) is an **NP**-complete problem (Bomze et al. 1999) and the method proposed in Bron and Kerbosch (1973) runs in a time that grows exponentially with the size of the graphs. Besides, to the computational difficulty of finding cliques in large graphs, we shall also consider the strictness of the condition which invalidates this approach . For instance, a subgraph with all possible internal edges except one would be an extremely cohesive subgroup, but it would not be considered a community under this definition.

Nevertheless, there exist several ways to relax the notion of complete mutuality to try to make it more helpful and general. One of them is related to reachability and defines an n-clique as the maximal subgraph such that the distance of each pair of its vertices is not larger than n (Luce 1950). Note that for $n = 1$, this definition is strictly equivalent to a clique.

The main problem of n-clique is that even a 2-clique is not necessary very cohesive by finding long and stringy groupings rather than the tight and discrete ones of the maximal approach. Alternatives, called n-clan and n-club related to the maximal diameter of the subgraph (the greatest distance between any pair of vertices) have been proposed by Mokken (1971) to avoid this problem.

Another alternative way of relaxing the strong assumptions of the clique is to allow that vertices may be members of a clique even if they have ties to all but k other members. One can read, in the literature on social network analysis, two complementary ways of expressing this notion: k-plex and k-core. The first one defines a maximal subgraph in which each vertex is adjacent to all other vertices of the subgraph except at most k of them (Seidman and Foster (1978)). Similarly, a k-core is a maximal subgraph in which each vertex is adjacent to at least k other vertices of the subgraph (Seidman (1983)). By imposing conditions on the minimal number of absent or present edges clusters are more cohesive than n-cliques.

More cohesive than k-plexes and k-cores are LS-sets. A key property of such criteria is that every node in the subgroup has higher edge connectivity with its other members than with any non-member. Using this criterion for defining a cohesive subset, the Lambda-set is defined as a subgraph such that any pair of vertices of the subgraph has a larger edge connectivity than any pair formed by one vertex of the subgraph and one outside the subgraph. More details can be found in Borgatti et al. (1990).

Last but not least, we can mention methods that identify community using fitness measures on the considered subgraph. For instance in Garey and Johnson (1979), the authors tackled the problem of finding subgraph C with k vertices and $\delta_{int}(C)$ is larger than a threshold. This problem is **NP**-complete as it is equivalent with the **NP**-complete clique problem when the threshold is equal to 1. Note that this problem has variants where the focused quantity to optimize is the number of internal edges of the subgraph (Asahiro et al. 2002, Feige et al. 2001).

Global Definitions

As previously mentioned, definitions of communities can also be proposed with respect to the graph as a whole. The literature is very vast and this section does not intend to present a panorama of all the available definitions. However, an outline can be introduced here and will become rather detailed with their algorithms in Section 1.5.

The first class of global definitions is based on the idea that a graph has a community structure if it differs from a random graph. We shall remind that a Erdös-Rényi random graph is not expected to have community structure, as each pair of nodes have the same probability to be adjacent. Thereby, the consists in defining a null model which matches the original graph in some of its structural features, but which is otherwise a random graph. The latter is then used to examine if the graph under study displays community structure or not. One of the most known null model has been by proposed by Newman and Girvan (2004) and consists of a randomized version of the original graph, where edges are rewired at random, under the constraint that each vertex keeps its degree. In the

next, we will see that the null model is the key concept behind the definition of *modularity*: a function which quantifies the quality of a division of a network into communities (also called modules). In the standard formulation of modularity, a subgraph is a community if the number of edges inside the subgraph exceeds the expected number of internal edges that the same subgraph would have in the null model. This expected number is an average over all possible realizations of the null model.

A second class which can be mentioned concerns *statistical model*-based methods. The outline of such methods, for instance the proposal of Daudin et al. (2008), is to model the distribution of connections in the network, considering that nodes are spread among an unknown number of connectivity classes which are themselves unknown. The maximisation of the global criteria represented by the *likelihood* leads to a fitting between the models and the data and allows to estimate both the community of the graph the model parameters. Chapter 2 will discuss the optimization strategies of the so-called likelihood.

Definitions based on vertex similarity

In the last class of community definitions we shall introduce those that are related to vertex similarities *i.e.* a community is a group of vertices which are similar to each other. Therefore, defining a similarity measure based on local and/or global attributes that express the similarity between each pairs of vertices connected or not, allow to mobilize more classical clustering method, like hierarchical clustering (see Section 1.5.1). Although it is impossible to review all the applicable measures, we discuss the most popular used in the literature.

A very common technique consists in embedding the graph vertices in an n-dimensional Euclidean space. Then, the assigned position of these vertices, can be used to compute the distance between a pair of vertices as a measure of their similarity. Let $A = (a_1, a_2, \cdots, a_n)$ and $B = (b_1, b_2, \cdots, b_n)$ be two data points. Vertex similarities can be computed using any norm, like the *Euclidean distance* (L2 -norm),

$$D_{AB}^{E} = \sum_{k=1}^{n} \sqrt{(a_k - b_k)^2},$$

the *Manhattan distance* (L1-norm)

$$D_{AB}^{M} = \sum_{k=1}^{n} |a_k - b_k|,$$

and the L_∞-norm

$$D_{AB}^{\infty} = \max_{k \in [1,n]} |a_k - b_k|.$$

Another popular spatial measure is the *cosine similarity*, defined as

$$\rho_{AB} = \arccos \frac{\mathbf{a}.\mathbf{b}}{\sqrt{\sum_{k=1}^{n} a_k^2}\sqrt{\sum_{k=1}^{n} b_k^2}}.$$

This non-Euclidean measure is often used in document clustering where textual data (words) of each document are summarized in a vector representation.

If the graph cannot be embedded in space, Wasserman et al. (1994) proposes a similarity measure based on the concept of structural equivalence (Lorrain and White 1977) and defined as

$$d_{ij} = \sqrt{\sum_{\substack{k \neq i \\ k \neq j}} (A_{ik} - A_{jk})^2}$$

where **A** is the adjacency matrix. Note that two vertices that the same neighbours and not necessary adjacent themselves are structurally equivalent.

We can also cite methods based on the number of paths running between two vertices to measure their similarity. The computation of such quantity is related to the maximum flow problem (Elias et al. 1956).

1.4.2 Partitions

A standard *partition* is a division of the graph in which each vertex is assigned to a community. However, in real systems vertices can be shared between communities and the detection of their associated partitions refer to overlapping communities (also called cover) algorithm (Latouche et al. 2009). Another class of partitions called *hierarchical* may be defined where each top-level community is composed of communities and so forth. Such graph partitioning can be naturally represented using a dendrogram. Figure 1.7 illustrates the dendrogram of a graph with seven vertices in which each "leave" represents a vertex.

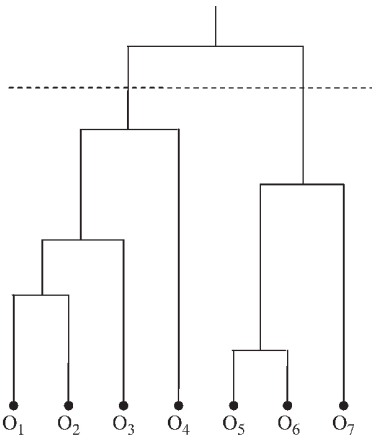


Figure 1.7 – *A dendrogram or a hierarchical tree. Horizontal cuts corresponds to partitions of the graph in communities.*

Vertices are aggregated in communities by moving upward. Logically the uppermost level represents the whole graph as a single community.

Merges of the communities are represented using horizontal lines and a cut of the dendrogram, at a certain height (dashed line), displays one partition of the graph. By construction the dendrogram is hierarchical: each community belonging to a level is included in a community at a higher level.

Last but not least, the number of possible partitions exponentially grows with the size of the graph. As all possible partitions are not equally good with respect to the community concepts, the future proposed algorithms will have to possess a quantitative criterion to assess the global goodness of a graph partition.

1.5 Algorithms for graph clustering

Keeping in mind the previous community and partition definitions, we can now introduce the essential publications related to each research branch.

1.5.1 Traditional method

Graph partitioning

The graph partitioning problem consists in dividing the graph G into Q disjoint groups of predefined size, such that the number of edges connecting between the groups (also called cut size) is minimal.

Graph partitioning problem is known to be **NP**-complete, but can be solved in polynomial time for a bisection (a partition of two groups) of the graph (Garey and Johnson 1979). Fast heuristics work well in practice and we propose to give the outline of the well known Kernighan and Lin (1970) algorithm.

The latter proposes an optimization strategy of a benefit function $\mathcal{Q}$, which is the difference between the number of edges inside the modules and the number of edges connecting between them. The algorithm starts with a balanced initial partition of the n vertices into sets A and B. Then, the algorithm attempts to find an optimal series of interchange operations between elements of A and B which maximizes function $\mathcal{Q}$ producing a partition of G. The algorithm is quite fast, scaling as $O(n^2 logn)$, if only a constant number of interchanges are performed at each iteration. As the found partitions by such procedure strongly depend on the initial partition, it is preferable to start with a good candidate. Therefore the method is typically used to improve on the partitions found through other techniques, by using them as starting configurations for the algorithm.

Then, partitions into more than two clusters are usually obtained by iterative bi-sectioning. For instance, in Suaris and Kedem (1988), the authors propose an adaptation of the Kernighan-Lin algorithm to extract partitions in any number of groups.

Other popular methods for graph partitioning include spectral bisection, level-structure partitioning, geometric algorithm, multilevel algorithms, etc. A review of these algorithms can be found in Pothen (1997).

Nevertheless, from a practical point of view, algorithms for graph partitioning are not suitable for community detection, since both the number

of groups and their size inputs are required and are most of the time unknown.

Partitional clustering

Another popular class of methods to find graph cluster is related to partitional clustering. In such methods, the number of clusters Q is predefined. Embedding graph vertices in a metric space allow them to be represented as a set of data points. Then, a proximity between each pairs of points can be computed in this space with respect to a defined distance measure. Thus, the proximity of points measures the similarity between vertices. Partitional clustering algorithms aim at assigning the data points in Q clusters such to maximize a given cost function based on distances between points and/or from points to centroids.

One of the most popular method in the literature is probably the k-means clustering algorithm (MacQueen et al. 1966). In this method, the cost function is equivalent to the total intra-cluster variance

$$\sum_{i=1}^{Q} \sum_{\mathbf{x_j} \in S_i} ||\mathbf{x_j} - \mathbf{c_i}||^2;$$

where S_i is the subset of points of the i-th cluster and $\mathbf{c_i}$ its centroid. The Lloyd (1982) algorithm which solves this problem can be described as follow. Starting from Q initial "means" points randomly selected from the data set, the Q clusters are formed by assigning every point (vertex) to the nearest mean. Then, the centers of mass of the Q clusters are computed and become a new "means", which allows for a new classification of the vertices, and so on. The convergence is reached after a small number of iterations. Note that the obtained partition is not necessary optimal and strongly depends on the initial means. Generally, to avoid initialisation issues, the algorithm is started with multiple initializations points and the solution which yields the minimum value of the total intra-cluster distance is chosen. There also exists a popular *fuzzy* method of the k-means algorithm (Dunn 1973) where each point has a degree of belonging to clusters rather than belonging completely to just one cluster.

In the same spirit as the k-means, we can cite in the following some of the most commonly used functions to optimize in partitional clustering:

- *Minimum k-clustering:* The cost function is related to the diameter of a cluster *i.e.* the largest distance between two points of a cluster. As the method classifies points such that the largest of the Q cluster diameters is the smallest possible, it leads to very "compact" clusters.
- *k-clustering sum:* Similar to the minimum k-clustering, but the diameter is replaced by the average distance between all pairs of points of a cluster.
- *k-center:* For each cluster i, a reference point x_i (also called *centroid*) is defined and d_i is the maximum distances between the i cluster point from the centroid. The clusters and centroids are self-consistently chosen in order to minimize the largest value of d_i.

- *k-median:* Similar to *k*-center algorithm, but the maximum distance from the centroid is replaced by the average distance.

Although the number of groups is again required for partitional clustering algorithms, the main criticism we can provide to such methods is that the embedding of graphs in a metric space can appears artificial and not directly related to the structure of the graph.

Hierarchical clustering

Hierarchical clustering creates a hierarchy of clusters which may be represented, as previously mentioned, in a tree structure called a dendrogram. Starting once again, from a similarity $n \times n$ matrix $\mathbf{X}$ formed by a distance measure between each pair of vertices, the hierarchical clustering algorithms aim at identifying groups of vertices with high similarity, and can be classified in two categories:

- *Agglomerative algorithms:* starts from the leaves (singleton cluster) and successively merges clusters together.
- *Divisive algorithms:* starts from the root and recursively splits the cluster.

Then, we shall concentrate on agglomerative algorithms which is the most common approach in the literature. Since clusters are merged on the basis of their mutual similarity, it is essential to define a measure that estimates how similar are the clusters. One can mention several prescription defined as follow.

In single linkage (or nearest-neighbour) clustering (Sibson 1973), the similarity between two groups is given by the minimum element X_{ij} where i and j belongs to different groups. On the contrary, the maximum element X_{ij} is used in the procedure of complete linkage (or furthest-neighbour) clustering (Sokal and Michener 1975). In average linkage clustering (Sørensen 1948) one has to compute the average of the X_{ij}. The Algorithm 1, with an $O(n^2)$ computational complexity, illustrates how the single linkage clustering works.

Let $\mathbf{X}$ be the $n \times n$ similarity matrix. The clusters are indexed with a sequence numbers $0, 1, ..., (n-1)$. We denote by (m) the cluster of nodes with sequence number m and $X_{(u),(v)}$ the similarity between clusters (u) and (v). The level of the q-th clustering is denoted by $L(q)$.

The stopping condition is not necessarily the classification of all the elements and it can be imposed by an alternative criterion like a given number of clusters.

The advantage of hierarchical clustering methods is that no preliminary knowledge is required on the number and size of the clusters. However, the solutions of such methods depend on the adopted similarity measure.

1.5.2 Betweenness-based methods

In a seminal paper, Girvan and Newman (2002) proposes an intuitive detection community algorithm which detects the edges that connect vertices

Algorithm 1: Single linkage clustering

Input: **X**, the similarity matrix.
$m = 0$
$L(0) = 0$

while *!(If all objects are in one cluster)* **do**
/* Find the least dissimilar pair of clusters in the current clustering */
$X_{(u),(v)} = min(X_{(i),(j)})$
/* Merge clusters (u) and (v) into a single cluster to form the next clustering m. */
$m = m + 1$
$L(m) = X_{(u),(v)}$
/* Update the proximity between the new cluster (u, v) and old cluster (q) */
$X_{(q),(r,s)} = min(X_{(q),(r)}, X_{(q),(s)})$

Result: A hierarchical clustering of **X** based on a single linkage criterion

of different communities and remove them in order to disconnect the communities from each other. This paper has strongly influenced the field of community detection. In such algorithm, edges are selected according to the values of measures of edge centrality, estimating the importance of edges according to some property or process running on the graph.

The algorithm can be defined by the following steps:

1. Computation of the centrality for all edges;
2. Removal of edge with largest centrality: in case of ties with other edges, one of them is picked at random;
3. Recalculation of centralities on the running graph;
4. Iteration of the cycle from step 2.

The edge centrality can be computed in different way and the authors have been focused on the concept of betweenness. They propose two alternative definitions:

- *edge betweenness:* the number of shortest paths between all vertex pairs that run along the edge. In Newman and Girvan (2004), the authors show that using techniques based on breadth-first-search, betweenness of all edges of the graph can be calculated in a time that scales as $O(mn)$, or $O(n^2)$ on a sparse graph
- *random-walk betweenness:* the frequency of the passages across the edge of a random walker running on the graph (Newman 2005). The computation of such measure requires the inversion of an $n \times n$ matrix (once), followed by obtaining and averaging the flows for all pairs of nodes. The first task requires a time $O(n^3)$, the second

$O(mn^2)$, for a total complexity $O((m+n)n^2)$, or $O(n^3)$ for a sparse matrix.

Since many shortest paths connecting vertices of different communities will pass through them, it is straightforward to see that inter-community edges have a larger value of the edge betweenness (see Figure 1.8), than intra-community edges.

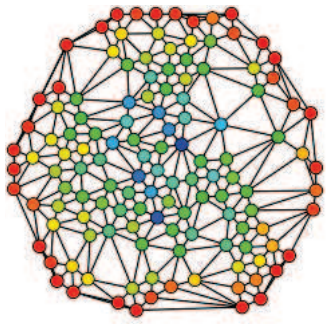


Figure 1.8 – *Hue (from red=o to blue=max) shows the node betweenness.*

Thus, by removing the edge with the largest value and recalculating the betweenness for the remaining edges, the groups are separated from one another and the underlying community structure of the network is revealed. The re-computation of the betweenness is necessary because the removal of an edge can cause a previously low-traffic edge to have much higher traffic. Note that calculating edge betweenness is much faster than random walk betweenness ($O(n^2)$ versus $O(n^3)$ on sparse graphs. Then, we have to repeat these previous steps until we obtain a set of isolated nodes. The end result of the algorithm is a dendrogram (produced from the top down).

Nevertheless, although the betweenness-based methods are especially intuitive, it tends to produce relatively poor results for dense networks, and may appear too slow for many large networks.

1.5.3 Modularity-based methods

The *modularity* proposed in Girvan and Newman (2002) is probably the most popular function which quantifies the quality of a division of a network into modules. Since random graphs are not expected to have a cluster structure, the possible existence of clusters is revealed by the comparison between the actual density of edges in a subgraph and the expected density of a subgraph where edges are attached regardless of community structure. The expected edge density is defined by a *null model* which is a copy of the original graph keeping some of its structural properties but without community structure.

Let P_{ij} be the expected number of edges between vertices i and j in the null model and let δ be the function which yields 1 vertices i and j belong to the same cluster, 0 otherwise. Then, modularity can then be expressed as follows

$$\mathcal{Q} = \frac{1}{2m}\sum_{ij}((A_{ij} - P_{ij})\delta(C_i, C_j), \tag{1.4}$$

where A is the adjacency matrix, m the total number of edges of the graph. Although, the choice of the null model graph is in principle arbitrary, the most common version of such model imposes that the expected degree sequence preserve the actual degree sequence of the graph. In this case, the expected number of edges falling between two vertices i and j following randomization is $k_i k_j / 2m$ where k_i is the degree of vertex i, and hence the previous Equation 1.4 of the modularity can finally be expressed as

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} ((A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j). \tag{1.5}$$

The value of the modularity lies in the range $[-1, 1]$ and is positive if the number of edges within groups exceeds the number expected on the basis of chance.

The modularity maximization method detects communities by searching over possible divisions of a network for one or more that have particularly high modularity. Since exhaustive search over all possible divisions is usually intractable, practical algorithms are based on approximate optimization methods such as greedy algorithms (Clauset et al. (2004), Newman (2004)), simulated annealing (Guimera and Amaral (2005), or spectral optimization (Newman and Girvan (2004), Newman (2006a), with different approaches offering different balances between speed and accuracy. In fact, greedy algorithms are fast heuristics intended to be applied to networks with millions of nodes or more. Recently, the autors of Blondel et al. (2008) have proposed a heuristic method based on modularity optimization which outperforms all other known community detection methods in terms of computation time. Other methods provide more sophisticated but slower means to identify high-modularity partitions.

Importantly, many modularity-maximization techniques are easy to generalize for use with other related quality functions because it is far from clear that modularity is the best function to optimize. For example, modularity has a known resolution limit [1] that might cause one to miss important communities (Fortunato and Barthélemy 2007). A few alternatives to modularity have been considered Hofman and Wiggins (2008), Lancichinetti et al. (2009).

1.5.4 Spectral algorithms

Spectral clustering methods involves information obtained from the spectrum (eigenvalues and eigenvectors) of a matrix describing the graph to group its vertices. Generally, the considered matrix that represent the graph is a combination of the weight and the degree matrix called Laplacian matrix L. Such matrix has components $L_{ij} = k_i \delta(i, j) - A_{ij}$, where k_i is the degree of node i (or, in a weighted network, its strength), and $\delta(i, j)$ is the Kronecker delta (*i.e.*, $\delta(i, j) = 1$ if $i = j$, and 0 otherwise). First studied in the context of graph theory (Fiedler 1973), spectral methods

[1] In its original formulation the modularity misses communities that are smaller than a certain threshold size that depends on the size of the network and the extent of interconnectedness of its communities

have become quite popular in the machine learning community especially after Schölkopf et al. (1998), Shi and Malik (2000), Ng et al. (2002). Since spectral clustering does not make assumptions on the form of the cluster, it can solve problems such as intertwined spirals. Besides, spectral clustering can be implemented efficiently for large sparse graphs. Several of these algorithms are summarized in Verma and Meila (2003) and can be categorized given the number of eigenvectors they use. Indeed, one can cite algorithms which work on the k largest eigenvectors in combination with clustering algorithms, such as k-means Ng et al. (2002) or algorithms which recursively use a single eigenvector to cluster the data (Shi and Malik 2000, Newman 2006b).

Next, one can remark that modularity can also be involved to obtain spectral partitioning Newman (2006a). By reformulating the scalar quantity of modularity in terms of a *modularity matrix* B, with components

$$B_{ij} = A_{ij} - P_{ij}, \tag{1.6}$$

spectral partitioning can be directly applied to heuristically optimize the modularity

$$\mathcal{Q} = \frac{1}{2m} \sum_{i,j} B_{ij} \delta(C_i C_j), \tag{1.7}$$

where similarly to Section 1.5.3 components of null model is defined by $P_{ij} = \frac{k_i k_j}{2m}$ and $\delta(C_i C_j)$ indicates that the B_{ij} components are only summed over cases in which nodes i and j are classified in the same community.

As for Laplacian based methods, one can use several eigenvectors of B for clustering the vertices, but it is effective (and simpler) to recursively subdivide a network using only the "leading eigenvector"(Newman 2006b). Bi-partition of vertices is repeated until the modularity can no longer be increased with additional subdivisions. Note that this method can be generalized by considering different quality functions, allowing steps that decrease global quality in order to further subdivide the communities.

1.5.5 Random Walk algorithms

Random walks can be envolved to detect community structure. Indeed, a random walker tends to be trapped in dense part of a network corresponding to communities. In the following, we describe the most popular clustering methods based on random walks. All of them can be trivially extended to the case of weighted graphs.

A first interesting approach is based on a distance measure defined between pairs of vertices (Zhou 2003). Here the distance d_{ij} between vertices i and j refers to the average number of edges that a random walker has to cross to reach j starting from i. Defining a *global attractor* of a vertex i as the closest vertex of i and a *local attractor* of i as its closest neighbour, two types of communities are can be defined, according to local or global attractors *i.e.* a vertex i has to be put in the same community of its attractor

and of all other vertices for which i is an attractor. The found communities must be minimal subgraphs which means that they cannot include smaller subgraphs which are communities according to the chosen criterion. There exists variants of this method where vertex i is associated to its attractor j only with a probability proportional to $exp(-\beta d_{ij})$, β being a sort of inverse temperature or where random walker is biased moving preferentially towards vertices which share a large number of neighbours with the starting vertex. Concerning the computational complexity of the procedure, the distance matrix is obtained in $O(n^3)$.

In Pons and Latapy (2005), the authors have introduced a different distance measure also based on random walks. Here, the distance d_{ij} between two vertices i and j is equivalent to the probability that the random walker moves from a i to j in a fixed number of steps. As the number of steps has to be large enough to explore a significant portion of the graph but but not too long to not depend of the whole graph and its stationary distribution, a compromise has to be chosen. Then, an agglomerative hierarchical clustering technique based on Ward's method (Ward Jr 1963) is applied on the distance matrix to cluster the vertices. The best partition of the dendrogram is selected using the modularity. The computational complexity of the algorithm [2] is $O(n^2 d)$ on a sparse graph, where d is the depth of the dendrogram.

The last method which concludes this section is known as Markov Cluster Algorithm (MCL)(Dongen 2000). It is based on simulations of flow diffusion in a graph. Starting from the transition matrix $\mathbf{R}$ of the graph, where element R_{ij} of the stochastic matrix indicates the probability that a random walker, sitting at vertex i, moves to j, the author proposes an iterative algorithm where each iteration is composed by two steps. The first step is called *expansion* and corresponds to computing random walks of higher length transforming $\mathbf{R}$ into $\mathbf{M}$ (usually $p = 1$ matrix multiplication of $\mathbf{R}$). Thus, the entry M_{ij} gives the probability that a random walker, starting from vertex i, reaches j in $p + 1$ steps (diffusion flow). The *inflation* refers to the second step which consists in raising each single entry of the matrix $\mathbf{M}$ to some power α, where α is now real-valued. Since this operation enhances the weights between pairs of vertices with large values of the diffusion flow, it have the effect of boosting the probabilities of intra-cluster walks and will demote inter-cluster walks without any *a priori* knowledge of cluster structure. Eventually, iterating expansion and inflation results in the separation of the graph into different segments. There are no longer any paths between these segments and the collection of resulting segments is simply interpreted as a clustering. Figure 1.9 illustrates this algorithm where code can be downloaded from `http://www.micans.org/mcl/`.

Since the expansion step requires a matrix multiplication, the computational complexity scale as $O(n^3)$, even if the graph is sparse. Note that several optimizations are considered such that keeping reference to a maximum number k of non-zero elements per column reducing the complexity to $O(nk^2)$ on sparse graphs.

The main problem of this method is that the found partition is sensitive

[2] The software of the algorithm can be found at `http://www-rp.lip6.fr/~latapy/PP/walktrap.html`.

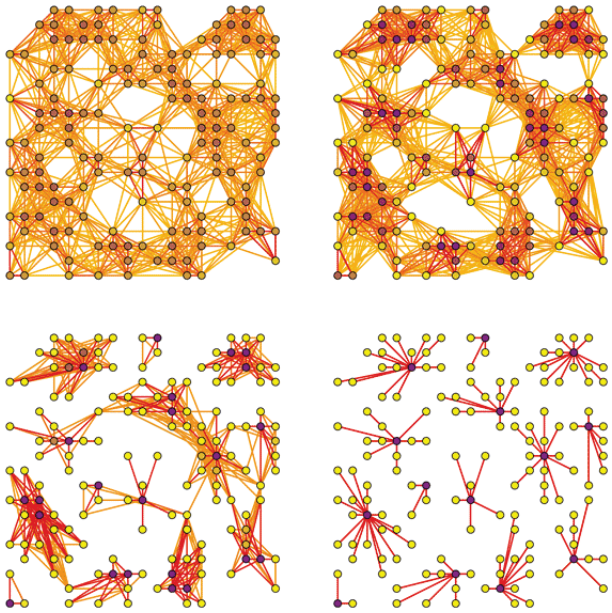


Figure 1.9 – MCL *algorithm. The color of a bond between two nodes indicates the maximum amount of flow taken over the two directions. The color of a node indicates the total amount of incoming flow. A dark bond between a light node and a dark node thus indicates that all flow is going from the lighter node in the direction of the darker node. It is seen that flow is eventually separated into different regions, yielding a cluster interpretation of the initial graph.*

to the parameter α used in the inflation step. Thereby, several partitions can be proposed, and it is not clear which are the most meaningful.

1.5.6 Model-based methods

Among model-based methods, model-based clustering has provided an efficient way to summarize complex networks structures. Due to the interpretability of the results, clustering procedures based on probability models are naturally preferred over the previous heuristic methods. The basic idea of these strategies is to model the distribution of connections in the network, considering that nodes are spread among an unknown number of connectivity classes which are themselves unknown. This generalizes model-based clustering to network data, and various modeling strategies have been considered. Nowicki and Snijders (2001) propose a mixture model on dyads that belong to some relational alphabet, Daudin et al. (2008) propose a mixture on edges, Airoldi et al. (2008) propose mixed labels for a given node, and Handcock et al. (2006) consider continuous hidden variables.

Indeed, even if the modeling strategies are diverse, EM like algorithms constitute a common core of the estimation strategy (Snijders and Nowicki 1997), and this algorithm is known to be slow to convergence and to be very sensitive to the size of the dataset.

This issue should be put into perspective with a new challenge: the analysis of large network datasets. Then the development of optimization strategies, with a reasonable speed of execution, are necessary to deal with networks composed of tens of thousands of nodes, if not more.

To this extent Bayesian strategies are limited, as they may not handle networks with more than a few hundred nodes (Snijders and Nowicki 1997, Nowicki and Snijders 2001), and heuristic-based algorithms may not be satisfactory from the statistical point of view (Newman and Leicht 2007). Variational strategies have been proposed as well (Daudin et al. 2008), but they are concerned by the same limitations as EM. Thus, performing efficient model-based clustering from a computational point of view on very large networks is the topic of the following Chapter 2.

Chapter Conclusion

This chapter has attempted to give a comprehensive overview of all the existing graph clustering methods. Since this field has grown quite popular and the number of published proposals for clustering algorithms is huge, we have chosen to provide key publications related to each research branch. The reader may easily extend its knowledge with more detailed reviews like the ones of Schaeffer (2007), Fortunato (2009).

2 Strategies for Online Inference of Model-Based Clustering in large Networks

Contents

3.1 Introduction 81
3.2 A Mixture of Networks with covariates 82
3.2.1 Connectivity Model 83
3.2.2 Remarks about Vertex Features Model 83
3.2.3 Models with additional Matrix Information 84
3.2.4 CoshMix^3: Model with additional Vertex features 87
3.3 Variational EM algorithm for CohsMix^{1,2,3} 88
3.3.1 Estimation of the latent structure 89
3.3.2 Estimation of the parameters 89
3.3.3 Model selection: ICL algorithm 91
3.4 Experiments 93
3.4.1 Parameters estimation 94
3.4.2 Comparison of algorithms 95
3.4.3 Real data 96
Conclusion 100

The statistical analysis of complex networks is a challenging task, given that appropriate statistical models and efficient computational procedures are required in order for structures to be learned. One line of research has aimed at developing mixture models for random graphs, and this strategy has been successful in revealing structures in social and biological networks. The principle of these models is to assume that the distribution of the edge values follows a parametric distribution, conditionally on a latent structure which is used to detect connectivity patterns. However, these methods suffer from relatively slow estimation

procedures, since dependencies are complex and do not necessarily allow for computational simplifications. In this chapter we adapt online estimation strategies, originally developed for the EM algorithm, to the case of models for which the probability of the missing data conditionally on the available observations is not tractable. Our work focuses on three methods, the first based on the CEM algorithm, the second on the SAEM algorithm, and the third on variational methods. We perform a simulation to compare these three algorithms with existing approaches and we use the method to decipher the structure of the French and US political websites network. We show that our online EM-based algorithms offer a good trade-off between precision and speed, when estimating parameters for mixture distributions in the context of random graphs.

The present chapter is collaborative work with Franck Picard and Vincent Miele.

2.1 Introduction

As mentioned in Chapter 1, analyzing networks has become an essential part of a number of scientific fields. In this chapter, we focus on a type of network for which nodes represent web sites or web pages, and each edge represents a hyperlink relating two nodes. In particular, we consider two one-day snapshot of the Web taken during the 2008 U.S.and 2007 French Presidential campaigns.

Many strategies have been developed for studying a network's structure and the previous chapter lists the main approaches (see Section 1.5). Among model-based methods, model-based clustering has provided an efficient way to summarize complex networks structures. The basic idea of these strategies is to model the distribution of connections in the network, considering that nodes are spread among an unknown number of connectivity classes which are themselves unknown. This generalizes model-based clustering to network data, and various modeling strategies have been considered. Nowicki and Snijders (2001) propose a mixture model on dyads that belong to some relational alphabet, Daudin et al. (2008) propose a mixture on edges, Airoldi et al. (2008) propose mixed labels for a given node, and Handcock et al. (2006) consider continuous hidden variables. In this chapter, our concern is not to assess neither to compare the appropriateness of these different models, but we focus on a computational issue that is shared by most of them. Indeed, even if the modeling strategies are diverse, EM like algorithms constitute a common core of the estimation strategy (Snijders and Nowicki 1997), and this algorithm is known to be slow to convergence and to be very sensitive to the size of the data set.

This issue should be put into perspective with a new challenge that is inherent to the analysis of network data sets is the development of optimization strategies with a reasonable speed of execution, and which can deal with networks composed of tens of thousands of nodes, if not more. To this extent Bayesian strategies are limited, as they may not handle networks with more than a few hundred nodes (Snijders and Nowicki 1997, Nowicki and Snijders 2001), and heuristic-based algorithms may not be satisfactory from the statistical point of view (Newman and Leicht 2007). Variational strategies have been proposed as well (Daudin et al. 2008), but they are concerned by the same limitations as EM. Thus the new question we assess in this work is "how to perform efficient model-based clustering from a computational point of view on very large networks ?".

Online algorithms constitute an efficient alternative to classical batch algorithms when the data set grows over time. The application of such strategies to mixture models has been studied by many authors (Titterington 1984, Wang and Zhao 2002). Typical clustering algorithms include the online k-means algorithm (MacQueen 1967). More recently, Liu et al. (2006) modeled Internet traffic using a recursive EM algorithm for the estimation of Poisson mixture models. However, an additional difficulty of mixture models for random graphs is that the computation of $\Pr(\mathbf{Z}|\mathbf{X})$, the distribution of the hidden label variables $\mathbf{Z}$ conditionally on the observation $\mathbf{X}$ cannot be factorized due to conditional dependency (Daudin et al. 2008). In this work, we consider three alternative strategies to deal with

this issue. The first is based on Classification EM strategy (CEM), where only the prediction of $\mathbf{Z}$ is considered, leaving apart the problem of computing $\Pr(\mathbf{Z}|\mathbf{X})$ (Zanghi et al. 2008). The second one is based on the Monte Carlo simulation of $\Pr(\mathbf{Z}|\mathbf{X})$, leading to a SAEM algorithm (Delyon et al. 1999) and the third one is the variational method proposed by Daudin et al. (2008) which consists in a mean-field approximation of $\Pr(\mathbf{Z}|\mathbf{X})$. The latter strategy has also been proposed by Latouche et al. (2008) and by Airoldi et al. (2008) in the Bayesian framework.

Outline of the chapter In this chapter, we apply online algorithm to the MixNet model that has been proposed by Daudin et al. (2008). First of all, we recall, in the first section, the basic principles of parameter estimation for mixture models using the well-known EM algorithm. In the second section, we present the MixNet model in its general form, where the conditional distribution of the connections is assumed to belong to the exponential family. Then we derive *online* strategies for CEM, SAEM and variational methods in the context of random graphs. We use simulations to show that online methods are very effective in terms of computation time, parameter estimation and clustering efficiency. Finally, we illustrate the method to uncover the connectivity structure of the 2008 U.S. and 2007 French Presidential Webspheres, and we show the gain of using model-based clustering instead of module search algorithms. We provide a computer software that is detailed in Appendix A.8. We also propose a web search engine which takes advantage of the online MixNet algorithm to reveal the connectivity information induced by hyperlinks. It is available at `http://constellations.labs.exalead.com/`, and is fully described in the Chapter 4.

2.2 Mixture Model and the EM algorithm

2.2.1 Finite Mixture Model

The first uses of finite mixtures of distributions can be dated back to the work of Newcomb in 1886 for the detection of outlier points, and then by Pearson (1894) to identify two separate populations of crabs. Since, they have been applied to model a wide variety of random phenomenon. These models assume that measurements are carried out on a set of individuals where each of them belong to a particular subpopulation which is unknown. For instance, in the Pearson's crab data, we know the ratio of forehead to body of each crab, but not the subspecies. Therefore, mixture models are a useful tool to handle the heterogeneity of a population, and are especially well suited to the problem of clustering. Although the literature of this scientific field is vast and ever expanding the McLachlan and Peel's book McLachlan and Peel (2000) is a highly detailed reference when addressing the problem of mixture models. First, we briefly recall the model and the problems of estimating its parameters.

The model

Let the data $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a sample of n independent observations of a random variable $\mathbf{X}$ in $\mathbb{R}^p$ from a finite mixture of Q components. The density can be expressed as

$$f(\mathbf{x}_i) = \sum_{q=1}^{Q} \alpha_q f_q(\mathbf{x}_i), \forall i \in I \tag{2.1}$$

where f_q are the density of each component and α_q are the mixture proportions ($\alpha_q \in]0,1[\forall q$ and $\sum_q \alpha_q = 1$). Mixture model is a generative model: it assumes, given the proportions $\alpha_1, \ldots, \alpha_Q$ and the distributions f_q of each class, that the data are generated according to the following mechanism:

- $\mathbf{z}$: each individual is allocated to a class according to a multinomial distribution with parameters $\alpha_1, \ldots, \alpha_Q$;
- $\mathbf{x}$: each $\mathbf{x}_i$ is assumed to arise from a random vector with probability density function f_q.

In addition, it is usually assumed that the component density f_q belongs to a parametric family of densities $f(., \theta_q)$. The density of the mixture can therefore be written as

$$f(\mathbf{x}_i, \Phi) = \sum_{q=1}^{Q} \alpha_q f(\mathbf{x}_i; \theta_q), \forall i \in I \tag{2.2}$$

where $\Phi = (\alpha_1, \ldots, \alpha_Q, \theta_1, \ldots, \theta_Q)$ is the parameter of the model.

For example, the density of a mixture model for two univariate Gaussian distributions of variance 1 in $\mathbb{R}$ is expressed as

$$f(x_i; \alpha, \mu_1, \mu_2) = \alpha\phi(x_i; \mu_1, 1) + (1 - \alpha)\phi(x_i; \mu_2, 1) \tag{2.3}$$

where $\phi(.; \mu, \sigma^2)$ is the density of the univariate Gaussian distribution of mean μ and variance σ^2.

Figures 2.1 uses the density obtained from a mixture of two Gaussian components in $\mathbb{R}$ to illustrate this concept of a probability mixture.

Estimation of parameters

Since the work of Pearson (1894) which uses the method of moments to estimate the five parameters $(\mu_1, \mu_2, \theta_1, \theta_2, \alpha)$ of a univariate Gaussian mixture model with two components (the method requires to solve polynomial equations of degree 9), a variety of approaches have been envisaged to estimate parameters of mixture distributions (McLachlan and Basford 1988, Titterington et al. 1985). Indeed, from the initial method of moments, they include graphical methods, minimum-distance methods, maximum-likelihood and Bayesian approaches. It is regularly argued that a reason for the huge literature on estimation methodology for mixtures is due to the non-existence of explicit formulas to parameter estimates. For instance, in the Gaussian case, the maximum-likelihood method, which is

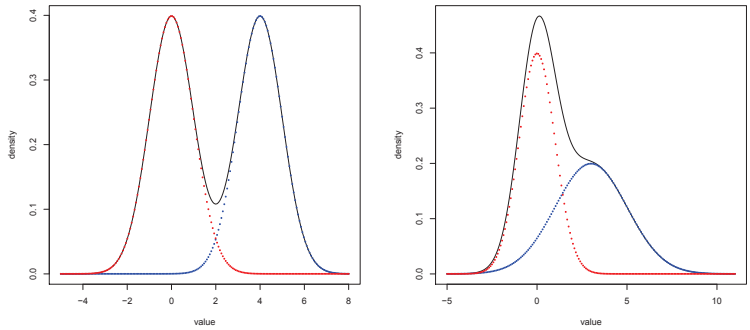


Figure 2.1 – *Densities of Gaussian mixture models. On the left a well separated (black line) case where* $\mu_1 = 0$ *(red points),* $\mu_2 = 4$ *(blue points)and* $\sigma^2 = 1$*. On the right a more intertwined case where* $\mu_1 = 0$*,* $\sigma_1^2 = 1$*,* $\mu_2 = 2$ *and* $\sigma_2^2 = 2$*.*

the most widely used, for the mixing proportions and component means and variances/covariances cannot be written in closed form and have to be computed iteratively. In this chapter we shall restrict ourselves to this method using the EM algorithm. However, before examining this method in Section 2.2.2, we first recall to the reader two difficulties which the estimation of parameters of a mixture model present:

- **Identifiability:** The density of the mixture needs to be identifiable. A number of studies have addressed this problem and several difficulties can be observed. A mixture is identifiable when equality in density implies equality in parameters. Let $\mathcal{M} = \{\mathbb{P}_\phi, \phi \in \Phi\}$ be a latent class model of parameter ϕ. The identifiability may hold, only up to label swapping, if for any ϕ, $\phi' \in \Phi$,

$$\mathbb{P}_\phi = \mathbb{P}_{\phi'} \Rightarrow \phi \sim \phi'.$$

  Indeed, we can notice a difficulty due to the numbering of the classes where any permutation of cluster indices give rise to the same model. For example, in the case of a mixture with two components, the parameters $(\alpha_1, \alpha_2), (\theta_1, \theta_2)$ and $(\alpha_2, \alpha_1), (\theta_2, \theta_1)$, although different, obviously yield the same probability density function (pdf): a mixture is consequently never identifiable. Depending on the estimation algorithms, this difficulty is more or less problematic. In the Bayesian framework it does cause a major problem known as the label-switching problem while in the case of the EM algorithm that does not matter. The second difficulty, considerably more tricky, may arise from the very nature of component pdf. Indeed, although mixtures of Gaussian, exponential and Poisson distributions are identifiable, it may easily be established that a mixture of uniform or binomial distributions is not.

- **Number of components:** Sometimes in practice, using physical facts, the number of components have a precise understanding and may be fully determined. However, most of the time, the number of components is unknown and must itself be estimated. Consider-

ing the number of components as an additional parameter, the mixture model may be seen as a semi-parametric compromise between a classical parametric estimation problem when the number of components corresponds to a fixed constant, and a non-parametric estimation problem when the number of components is equal to the size of the sample. In the following, we will assume that the number Q of components is known to estimate the mixtures and also propose solutions to achieve this difficult choice.

2.2.2 The EM algorithm

Missing information principle. The *Expectation-Maximization* EM algorithm is especially designed for finding maximum likelihood estimates (MLE) of parameters in probabilistic models, where the model depends on unobserved latent variables. Explained in a well-known seminal paper Dempster et al. (1977), the principle of this iterative method is to express the likelihood considering that the observed data $\mathbf{X}$ correspond to a partial knowledge of unknown data which are termed *complete data*. The complete data might for instance be of the form $\mathbf{Y} = (\mathbf{X}, \mathbf{Z})$, in which case $\mathbf{Z}$ is known as *missing information*.

Complete data and complete-data likelihood. The log-likelihood $\mathcal{L}_c(\Phi)$ calculated from these complete data is termed *complete-data likelihood* or, in the case of the mixture model, *classification log-likelihood*. Starting from the relation $f((\mathbf{X}, \mathbf{Z}); \Phi) = f(\mathbf{X}; \Phi) f(\mathbf{Z}|\mathbf{X}; \Phi)$ between the densities, we obtain the likelihood relation

$$\mathcal{L}_c(\Phi) = \mathcal{L}(\Phi) + \log f(\mathbf{Z}|\mathbf{X}; \Phi) \tag{2.4}$$
$$\mathcal{L}(\Phi) = \mathcal{L}_c(\Phi) - \log f(\mathbf{Z}|\mathbf{X}; \Phi) \tag{2.5}$$

between the initial log-likelihood $\mathcal{L}(\Phi)$ and the complete-data log-likelihood $\mathcal{L}_c(\Phi)$.

Alternating optimization algorithm. The EM algorithm assumes that maximizing the complete-data likelihood is a simple task. Since this likelihood cannot be calculated - $\mathbf{Z}$ is unknown - an iterative procedure based on the conditional expectation of the log-likelihood for a value of the current parameter Φ' is used as follows. First, calculating the conditional expectation for the two members of relation 2.4 respectively $Q(\Phi, \Phi') = \mathbb{E}(\mathcal{L}_c(\Phi)|\mathbf{X}, \Phi')$ and $H(\Phi, \Phi') = \mathbb{E}(\log f(\mathbf{Z}|\mathbf{X}; \Phi)|\mathbf{X}, \Phi')$, we obtain the fundamental relation

$$\mathcal{L}(\Phi) = Q(\Phi, \Phi') - H(\Phi, \Phi'). \tag{2.6}$$

Introducing the parameter Φ' allows us to define an iterative algorithm to increase the likelihood. Using Jenssen's inequality (Dempster et al. 1977) it can be shown that, for fixed Φ' , the function $H(\Phi, \Phi')$ is maximum for $\Phi = \Phi'$. The value Φ which maximizes $Q(\Phi, \Phi')$ therefore satisfies the relation

$$\mathcal{L}(\Phi) \geq \mathcal{L}(\Phi'). \tag{2.7}$$

The EM algorithm involves constructing, from an initial solution $\Phi^{(0)}$, the sequence $\Phi^{(p)}$ satisfying $\Phi^{(p+1)} = \text{Argmax}\, Q(\Phi, \Phi^{(p)})$ and converges to a fixed point using iteratively the two following steps:

- **Estimation Step:** Compute $Q(\Phi, \Phi') = \mathbb{E}(\mathcal{L}_c(\Phi)|\mathbf{X}, \Phi')$
- **Maximization Step:** Compute $\Phi^{(p+1)} = \text{Argmax}\, Q(\Phi, \Phi^{(p)})$

Relation 2.7 shows that this sequence causes the criterion $\mathcal{L}(\Phi)$to grow.

Application to mixture models. An early version of the EM algorithm dedicated to mixture was already proposed by Wolfe (1970). Considering a mixture model, the complete data are obtained by adding the original component $\mathbf{z}_i$ to each individual member of the sample:

$$\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n) = ((\mathbf{x}_1, \mathbf{z}_1), \ldots, (\mathbf{x}_n, \mathbf{z}_n)). \tag{2.8}$$

Coding $\mathbf{z}_i = (z_{i1}, \ldots, z_{iQ})$ where z_{iq} equals 1 if object i belongs to class q and 0 otherwise, we obtain the relations:

$$f(\mathbf{Y}; \Phi) = \prod_{i=1}^{n} f(\mathbf{y}_i; \Phi) = \prod_{i=1}^{n} \sum_{q=1}^{Q} z_{iq} \alpha_q f(\mathbf{x}_i; \theta_q), \tag{2.9}$$

$$\mathcal{L}_c(\Phi) = \log(f(\mathbf{Y}, \Phi)) = \sum_{i=1}^{n} \sum_{q=1}^{Q} z_{iq} log(\alpha_q f(\mathbf{x}_i; \theta_q)) \tag{2.10}$$

and

$$Q(\Phi|\Phi') = \sum_{i=1}^{n} \sum_{q=1}^{Q} \mathbb{E}(z_{iq}|\mathbf{X}, \Phi') \log(\alpha_q f(\mathbf{x}_i; \theta_q)). \tag{2.11}$$

Let τ_{iq} be the probability $\Pr(z_{iq} = 1|\mathbf{X}, \Phi')$ that object i belongs to class q, the EM algorithm takes the following form 2:

The parameters θ_q are obtained by solving the likelihood equations that depend on the mixture model employed. For instance, in a Gaussian mixture, the maximization of $Q(\Phi|\Phi')$ provides the optimal value of the parameters summarized in the following relations:

$$\mu_q^{(m+1)} = \frac{1}{\sum_{i=1}^{N} \tau_{iq}^{(m)}} \sum_{i=1}^{N} \tau_{iq}^{(m)} \mathbf{x}_i \tag{2.12}$$

and

$$\mathbf{\Sigma}_q^{(m+1)} = \frac{1}{\sum_{i=1}^{N} \tau_{iq}^{(m)}} \sum_{i=1}^{N} \tau_{iq}^{(m)} (\mathbf{x}_i - \mu_q^{(m+1)})^t (\mathbf{x}_i - \mu_k^{(m+1)}) \tag{2.13}$$

Algorithm 2: EM Algorithm for mixture model

Data: **X**, data matrix

/* Arbitrarily initialization of the parameters */

$\Phi^{(0)} = (\alpha_1^{(0)}, \ldots, \alpha_Q^{(0)}, \theta_1^{(0)}, \ldots, \theta_Q^{(0)})$

$m = 0$

while *not convergence* **do**

/* Estimation step */

/* Compute τ_{ik} the probabilities of i belonging to the classes, conditionally on the current parameter:*/

foreach $i \in \{1, ..., N\}$ **do**

foreach $q \in \{1, ..., Q\}$ **do**

$\tau_{iq}^{(m+1)} = \frac{\alpha_q^{(m)} f(\mathbf{x}_i, \theta_q^{(m)})}{\sum_{l=1}^{Q} \alpha_l^{(m)} f(\mathbf{x}_i, \theta_l^{(m)})}$

/* Maximization step: maximize the log-likelihood conditionally on $\tau_{iq}^{(m)}$ */

/* re-estimate the distribution parameters to maximize the likelihood of the data */

foreach $q \in \{1, ..., Q\}$ **do**

$\alpha_q^{(m+1)} = \sum_{i=1}^{N} \tau_{iq}^{(m)} / N$

$\theta_q^{(m+1)} = \text{Argmax}_{\theta_q} \, Q(\Phi | \Phi^{(m)})$

m = m +1

Result: Estimated parameters $\hat{\Phi}$ and posterior probabilities τ_{iq}

Convergence properties of EM algorithm. Under certain conditions of regularity, it has been established that the EM algorithm converges to a local likelihood maximum. It shows good practical behavior, but may nevertheless be quite slow in some situations. This is the case, for instance, when classes are very mixed.

Relation to variational Bayes methods. In order to obtain a full Bayesian version of the EM algorithm, we simply have to provide a probability distribution over Φ as well as the latent variables (Friedman 1998). In this context, the optimization is realized over each k latent variables (including Φ), one at the time. Thus, the distinction between the E and M step disappears and gives rise to k optimization steps per iteration.

Extensions of the EM algorithm. The EM algorithm has spawned numerous variants in the context mixture models. Many of them can be found in a McLachlan and Krishnan's book (McLachlan and Krishnan 1997). One of them is of particular interest because it allows connections between the EM algorithm for mixture model algorithms and k-means: it is the Classification EM algorithm (CEM) (Celeux and Govaert 1992). This algorithm incorporates a classification step between the E and M steps of EM. This step transforms the posterior probabilities τ_{iq} calculated in Step E for all individuals by distributions requiring a certain mass of 1 on the assumption most probable. This is equivalent to modifying the $\tau_{iq}^{(m)}$ by replacing them with the nearest 1 or 0 values. This algorithm is also an alternating optimization algorithm, but the criterion used is not a likelihood; it leads to biased estimates parameters of models. However, when the entropy of the distribution *a posteriori* on hidden variables is low, the bias remains small and this algorithm has the advantage of converging faster than the classic EM algorithm. Moreover, its interpretation is quite natural when the objective is not the estimation of parameters but the search for a optimal classification data. More details could be found in Section 2.4.

Other extensions of the EM algorithms focused on the problem of its local convergence. To avoid this problem, SEM algorithm developed by Celeux and Diebolt (1985) incorporates a simulation step S (for stochastic) between the E and M steps of EM. In this step, a class is drawn for each individual using the posterior probabilities calculated in Step E. This step aims at preventing the algorithm getting trapped in a local minimum of the likelihood. In practice, when the sample of observed data is small, it appears that stochastic perturbations in SEM can involve important variance of the parameters (Celeux et al. 1995) and provide results less efficient than a classical EM. This is the reason why Celeux and Diebolt (1991) have introduced the SAEM (for *Simulated Annealing EM*) algorithm. The main idea is to keep the stochastic nature of the algorithm going to from pure SEM at the beginning towards pure EM at the end. A confusion is not uncommon concerning this designation. Indeed, SAEM also refers to the *Stochastic Approximation* EM algorithm of Delyon et al. (1999). This approach approximates the expectation of the complete data log-likelihood using Monte Carlo simulations and is described in Section 2.5.

2.3 A Mixture Model for Networks

In this section we present the MixNet model. In its classical form, the model deals with networks without valuation. Here, we introduce the model in a more general form extending it to weighted graph.

2.3.1 Model and Notation

Let us define a random graph G, where $\mathcal{V}$ denotes the set of fixed vertices. We suppose that nodes are spread among Q hidden classes and we denote by Z_{iq} the indicator variable such that $\{Z_{iq} = 1\}$ if node i belongs to class q. We denote by $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_n)$ the vector of random independent label variables such that

$$\mathbf{Z}_i \sim \mathcal{M}(1, \boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_Q\}),$$

follows a multinomial distribution with $\boldsymbol{\alpha}$ the vector of proportions for classes.

Conditional distribution. Random edges are described by a set of random variables $\mathbf{X} = \{X_{ij}, (i,j) \in \mathcal{V}^2\}$, $|\mathcal{V}| = n$, $\mathbf{X}$ coding for the nature of connection between nodes i and j. MixNet is defined using the conditional distribution of edges given the label of the nodes. Knowing group q of node i and group l of node j, X_{ij} is distributed as $f(., \eta_{ql}) := f_{ql}(.)$ where $f_{\eta_{ql}}$ is a probability distribution known up to a finite dimensional parameter η_{ql}:

$$X_{ij} | i \in q, j \in l \sim f(., \eta_{ql}) := f_{ql}(.).$$

X_{ij}s are supposed to be conditionally independent:

$$\Pr(\mathbf{X}|\mathbf{Z}; \boldsymbol{\eta}) = \prod_{ij} \prod_{q,l} \Pr(X_{ij}|Z_{iq}Z_{jl} = 1; \eta_{ql})^{Z_{iq}Z_{jl}}.$$

and $\Pr(X_{ij}|Z_{iq}Z_{jl} = 1; \eta_{ql})$ is supposed to belong to the regular exponential family, with natural parameter η_{ql}:

$$\log \Pr(X_{ij}|Z_{iq}Z_{jl} = 1; \eta_{ql}) = \eta_{ql}^t h(X_{ij}) - a(\eta_{ql}) + b(X_{ij}),$$

where $h(X_{ij})$ is the vector of sufficient statistics, a a normalizing constant and b a given function. Consequently, the conditional distribution of the graph is also from the exponential family (See Figure 2.2):

$$\log \Pr(\mathbf{X}|\mathbf{Z}; \boldsymbol{\eta}) = \sum_{ij,ql} Z_{iq}Z_{jl}\eta_{ql}^t h(X_{ij}) - \sum_{ij,ql} Z_{iq}Z_{jl} a(\eta_{ql}) + \sum_{ij} b(X_{ij}).$$

Up to a relabeling of the classes, the model is identifiable (Allman et al. 2009) and completely specified by both the mixture proportion $\boldsymbol{\alpha}$ and the connectivity matrix $\boldsymbol{\eta} = (\eta_{ql})_{q,l=1...Q}$. We denote $\boldsymbol{\beta} = (\boldsymbol{\alpha}, \boldsymbol{\eta})$ the parameters of the model.

In the following, formulas are derived in the case of directed networks, but could be easily generalized to the non directed case ($X_{ij} = X_{ji}$).

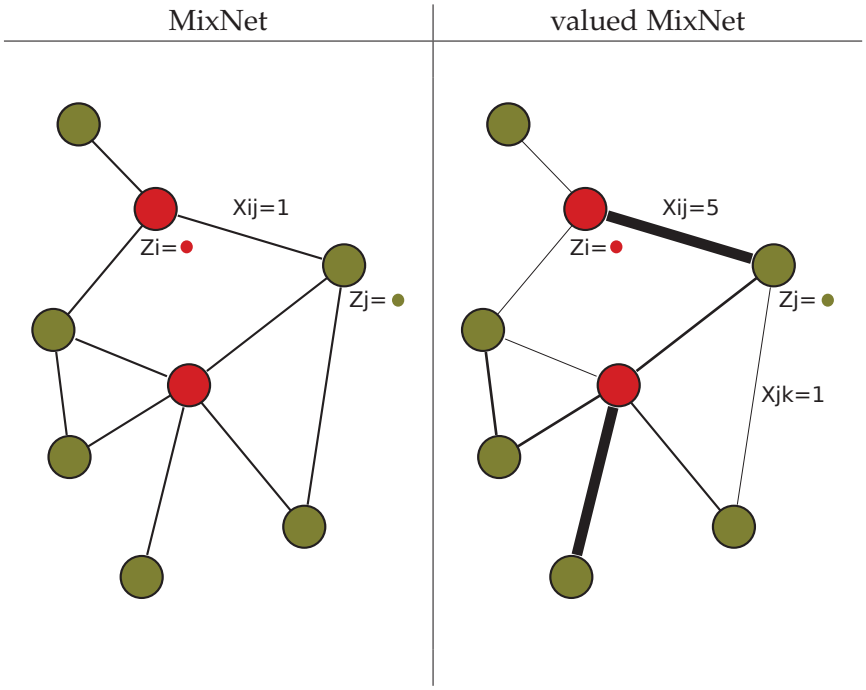
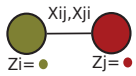
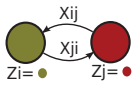
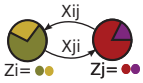
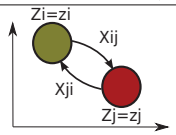


Figure 2.2 – *MixNet and its exponential family generalization.*

Models comparison. The MixNet model is related to other clustering models for networks (Figure 2.3.1). The stochastic blockstructure model (Snijders and Nowicki 1997, Nowicki and Snijders 2001) considers a mixture on dyads (X_{ij}, X_{ji}) whereas we consider a model on the edges of the network. This implicitly assumes the independence of X_{ij} and X_{ji} conditionally on the latent structure. Each edge X_{ij} is considered as a random variable which can be discrete or real. As for the comparison with the Mixed membership model (Airoldi et al. 2008), the hidden variables of their model can stand for more than one group for one node, whereas MixNet only considers one label per node. Airoldi et al. (2008) also model the sparsity of the network, which could be done also with MixNet as well introducing a Dirac mass on zero for the conditional distribution of edges. Here we consider that the conditional distribution belongs to the exponential family. It allows us to model both discrete and real valued relational data while being able to estimate the parameters via an EM algorithm.

Blocks Nowicki and Snijders (2001)	MixNet Daudin et al. (2008)
	
dyads	edges
Mixed-Memb. **Airoldi et al. (2008)**	**Ergm** **Handcock et al. (2006)**
	
mixed **Z**	continuous **Z**

Examples of distributions. Numerous classical distributions fit into this framework (Mariadassou and Robin 2007). For example, when the only available information is the presence or the absence of an edge, then X_{ij} is assumed to follow a Bernoulli distribution:

$$X_{ij}|Z_{iq}Z_{jl} = 1 \sim \mathcal{B}(\pi_{ql}) \begin{cases} \eta_{ql} = \log \frac{\pi_{ql}}{1-\pi_{ql}}, \\ h(X_{ij}) = X_{ij}, \\ a(\eta_{ql}) = \log(1 - \pi_{ql}), \\ b(X_{ij}) = 0. \end{cases} \tag{2.14}$$

Figure 2.3 illustrates a graphical representation of the MixNet Model.

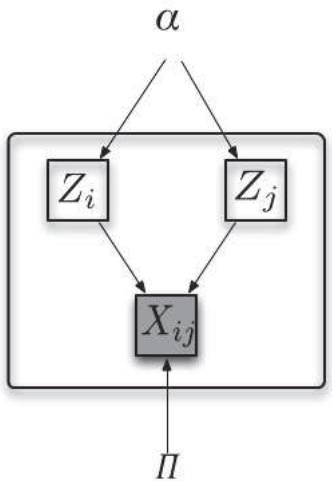


Figure 2.3 – *Graphical representation of the MixNet Model with a Bernoulli distribution of edges. The squares represent discrete random variables.*

Some properties concerning this Bernoulli Erdös-Rényi mixture model, like the distribution of degrees or the between group connectivity are available in the appendix A.1. An interesting property which can be

quoted is that this model allows to explicitly compute the distribution of the degree K_i of node i with approximately a Poisson mixture:

$$K_i \sim \sum_q \alpha_q \mathcal{B}(n-1, \bar{\pi}_q) \approx \sum_q \alpha_q \mathcal{P}(\lambda_q) \tag{2.15}$$

where $\bar{\pi}_q = \sum \alpha_l \pi_{ql}$ and $\lambda_q = (n-1)\bar{\pi}_q$. Such a distrubution allows a better fit to the degree distributions of real-networks.

If additional information is available to describe the connections between vertices, it may be integrated into the model. For example, the Poisson distribution might describe the intensity of the traffic between nodes. A typical example in web access log mining is the number of users going from a page i to a page j. Another example is provided by co-authorship networks, for which valuation may describe the number of articles commonly published by the authors of the network. In those cases, we have

$$X_{ij}|Z_{iq}Z_{jl} = 1 \sim \mathcal{P}(\lambda_{ql}) \begin{cases} \eta_{ql} = \log \lambda_{ql}, \\ h(X_{ij}) = X_{ij}, \\ a(\eta_{ql}) = -\lambda_{ql}, \\ b(X_{ij}) = X_{ij}! \end{cases} \tag{2.16}$$

However, we will only consider Bernoulli Erdös-Rényi mixture models in the experiment and application sections of this chapter (respectively Section 2.8 page 62 and Section 2.9 page 68).

Joint distribution. Since MixNet is defined by its conditional distribution, we first check that the joint distribution also belongs to the exponential family. Using notation

$$\begin{cases} N_q & = \sum_i Z_{iq}, \\ H_{ql}(\mathbf{X}, \mathbf{Z}) & = \sum_{ij} Z_{iq} Z_{jl} h(X_{ij}), \\ G_{ql}(\mathbf{Z}) & = \sum_{ij} Z_{iq} Z_{jl} = N_q N_l, \\ \alpha_q & = \exp(\omega_q) / \sum_l \exp(\omega_l), \end{cases} \tag{2.17}$$

and

$$\begin{cases} T(\mathbf{X}, \mathbf{Z}) & = \left(\{N_q\}, \{H_{ql}(\mathbf{X}, \mathbf{Z})\}, \{G_{ql}(\mathbf{Z})\}\right), \\ \boldsymbol{\beta} & = \left(\{\omega_q\}, \{\eta_{ql}\}, \{-a(\eta_{ql})\}\right), \\ A(\boldsymbol{\beta}) & = n \log \sum_l \exp \omega_l, \\ B(\mathbf{X}) & = \sum_{ij} b(X_{ij}), \end{cases} \tag{2.18}$$

we have the factorization $\log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta}) = \boldsymbol{\beta}^t T(\mathbf{X}, \mathbf{Z}) - A(\boldsymbol{\beta}) + B(\mathbf{X})$, which proves the claim. Details of this factorization are given in Appendix A.2. The sufficient statistics $T(\mathbf{X}, \mathbf{Z})$ of the complete-data model are the number of nodes in the classes (N_q), the characteristics of the between-group links (H_{ql} through function h) and G_{ql} the product of frequencies between classes. In other words, if the considered graph is directed, undirected, has or has not self-loops, valued or not, the MixNet model can be fully described using these sufficient statistics. An explicit detail of

the joint distribution of MixNet (in Bernoulli Erdös-Rényi mixture case) via the complete log-likelihood computation is given in appendix A.3 and can provide an example of this factorization. In the following we aim at estimating $\boldsymbol{\beta}$.

2.3.2 Sufficient statistics and Online Recursion

Online algorithms (Figure 2.4) are incremental algorithms which recursively) update parameters, using current parameters and new observations. We introduce the following notation. Let us denote by $\mathbf{X}^{[n]} = \left(X_{ij}\right)_{i,j=1}^{n}$, the adjacency matrix of the data, when n nodes are present, and by $\mathbf{Z}^{[n]}$ the associated labels. A convenient notation in this context is $\mathbf{X}_{i,\bullet} = \{X_{ij},\, j \in \mathcal{V}\}$, which denotes all the edges related to node i. Note that the addition of one node leads to the addition of $n+1$ potential connections.

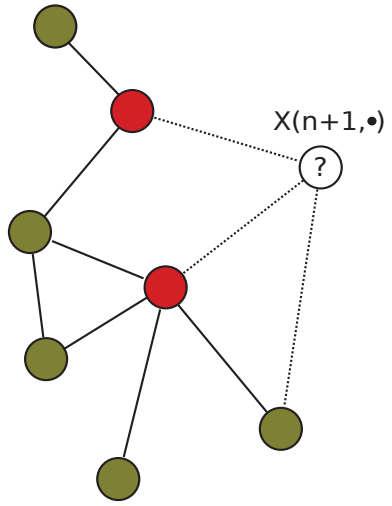


Figure 2.4 – *Online algorithms are incremental algorithms which recursively update parameters using current parameters and additional information provided by new observations*

The use of online methods is based on the additivity of the sufficient statistics regarding the addition of a new node. We can show that:

$$\begin{cases} N_q^{[n+1]} & = N_q^{[n]} + Z_{n+1,q}, \\ H_{ql}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}) & = H_{ql}(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}) + \xi_{ql}^{[n+1]}, \\ G_{ql}(\mathbf{Z}^{[n+1]}) & = G_{ql}(\mathbf{Z}^{[n]}) + \zeta_{ql}^{[n+1]}, \end{cases} \tag{2.19}$$

with

$$\begin{aligned} \xi_{ql}^{[n+1]} &= Z_{n+1,q} \sum_{j=1}^{n} Z_{jl} h(X_{n+1,j}) + Z_{n+1,l} \sum_{i=1}^{n} Z_{iq} h(X_{i,n+1}) \\ \zeta_{ql}^{[n+1]} &= Z_{n+1,q} N_l^{[n]} + Z_{n+1,l} N_q^{[n]}. \end{aligned}$$

Then if we define $T(\mathbf{X}_{n+1,\bullet}, \mathbf{Z}^{[n+1]}) = \left(Z_{n+1,q}, \{\xi_{ql}^{[n+1]}\}, \{\zeta_{ql}^{[n+1]}\}\right)$ which shows the following proposition:

Proposition 2.1

$$T(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}) \quad = \quad T(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}) + T(\mathbf{X}_{n+1,\bullet}, \mathbf{Z}^{[n+1]}).$$

Those equations will be used for parameter updates in the online algorithms.

2.3.3 Likelihoods and online inference

Inspired by the criterion of Scott and Symons (1971), the log-likelihood of the complete data set is

$$\mathcal{L}_c(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta}) = \log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta}) = \sum_i \sum_q Z_{iq} log \alpha_q + \sum_{i,j} \sum_{q,l} Z_{iq} Z_{jl} log f_{ql}(X_{i,j})$$

Further details (like directed/undirected graph and self-loops) concerning the derivation of the expression of the complete log-likelihood in a Erdös-Rényi mixture are given in the appendix A.3.

The likelihood of the incomplete data set can be obtained by summing $\Pr(\mathbf{X}, \mathbf{Z})$ over all possible $\mathbf{Z}$'s: $\Pr(\mathbf{X}) = \sum_{\mathbf{Z}} \Pr(\mathbf{X}, \mathbf{Z})$. This summation involves Q^n terms and quickly becomes intractable. Existing estimation strategies are based on maximum likelihood, and algorithms related to EM are used for optimization purposes. The aim is to maximize the conditional expectation of the complete-data log-likelihood

$$\mathcal{Q}(\boldsymbol{\beta}|\boldsymbol{\beta}') \quad = \quad \sum_{\mathbf{Z}} \Pr(\mathbf{Z}|\mathbf{X}; \boldsymbol{\beta}') \log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta}),$$

and the main difficulty is that $\Pr(\mathbf{Z}|\mathbf{X}; \boldsymbol{\beta}')$ can not be factorized and needs to be approximated (Daudin et al. 2008).

In this chapter, we will propose three approximation strategies to estimate the model parameters:

- **CEM:** A first interesting approach is to consider a Classification EM-based strategy (CEM), where only the prediction of $\mathbf{Z}$ is considered, leaving apart the problem of computing $\Pr(\mathbf{Z}|\mathbf{X})$. Optimizing directly the Classification log-likelihood $\mathcal{L}_c$, this approach is known to give biased estimates. However, it is very efficient from a computational point of view. Section 2.4 and (Zanghi et al. 2008) are concerned with this online approach.

- **SAEM:** The second approach is based on the Stochastic Approximation EM approach (Delyon et al. 1999) which approximates $\Pr(\mathbf{Z}|\mathbf{X})$ using Monte Carlo simulations. The stochastic nature of this method can also avoid local convergences of the estimation. Section 2.5 explains this strategy.

- **VEM:** The last strategy (See Section 2.6), called variational approach, consists in approximating $\Pr(\mathbf{Z}|\mathbf{X})$ by a more tractable distribution on the hidden variables.

In their online versions, while nodes are added, the CEM algorithms directly maximizes the criterion $\mathcal{L}_c(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta})$, whereas SAEM and VEM algorithms optimize $\mathcal{Q}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ sequentially. To this extent, we introduce notation:

$$\mathcal{Q}_{n+1}(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]}) = \sum_{\mathbf{Z}^{[n+1]}} \Pr(\mathbf{Z}^{[n+1]}|\mathbf{X}^{[n+1]}; \boldsymbol{\beta}^{[n]}) \log \Pr(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}; \boldsymbol{\beta}),$$

with $[n+1]$ being either the number of nodes or the increment of the algorithm, which are identical in the online context.

2.4 Classification EM algorithm

The Classification EM (CEM) algorithm is an iterative clustering algorithm which simultaneously yields the parameters and the classification. From a practical point of view, this algorithm is faster than EM algorithm and converges in a few iterations. In the mixture model context the CEM algorithm can be regarded as a classification version of the EM algorithm, involving a classification step between the E-step and the M-step. The paper of Celeux and Govaert (1992), showed that each iteration of the CEM algorithm increases the classification log-likelihood criterion $\mathcal{L}_c$ and its convergence is reached after a finite number of iterations. It is noticeable that the CEM algorithm exactly corresponds to the k-means algorithm, when assuming Gaussian mixtures with equal proportions and covariance matrices equal to the identify matrix. Therefore, this algorithm can be regarded as a generalization of the k-means algorithm ables to handle non-spherical covariances matrices and non-uniform proportions.

2.4.1 Classification log-likelihood

According to the previous notation where $[n+1]$ corresponds to either the number of nodes and the increment of the algorithm, we can write the classification log-likelihood as

$$\mathcal{L}_c(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}; \boldsymbol{\beta}) = \log \Pr(\mathbf{Z}^{[n]}; \boldsymbol{\beta}) + \log \Pr(\mathbf{X}^{[n]}|\mathbf{Z}^{[n]}; \boldsymbol{\beta}).$$

For the sake of illustration, further details concerning the derivation of the expression of the classification log-likelihood in both Bernoulli and Poisson distributions are given in the appendix A.3.

We can remark that a maximization of the log-likelihood criterion according to $(\mathbf{Z}, \boldsymbol{\beta})$ is equivalent to a maximization of the criterion

$$\mathcal{L}_c(\mathbf{X}^{[n]}, \boldsymbol{\beta}) = \max_{\mathbf{Z}^{[n]}} [\mathcal{L}_c(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}; \boldsymbol{\beta})].$$

This alternative formulation of the log-likelihood criterion allows us to consider the CEM algorithm as a parameter estimation algorithm like EM algorithm.

2.4.2 Online Estimation

This section describes an original incremental Classification version of the EM algorithm. As previously argued, incremental algorithms recursively update parameters, using current parameters and new observations. Using the previous notation, we note $\mathbf{X}^{[n]}$ the adjacency matrix of a graph with n nodes and $\mathbf{Z}^{[n]}(\boldsymbol{\beta})$, the classification matrix verifying:

$$\mathbf{Z}^{[n]}(\boldsymbol{\beta}) = \arg\max_{\mathbf{Z}} \mathcal{L}_c(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}, \boldsymbol{\beta}).$$

Let $\boldsymbol{\beta}^{[n]}$ be the parameter vector maximizing $\mathcal{L}_c^n(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}(\boldsymbol{\beta}^{[n-1]}), \boldsymbol{\beta})$ the complete log-likelihood expressed in function of n nodes. When a new node and all attached edges $\mathbf{X}_{n+1,\bullet}$ become available, the new complete log-likelihood of $\boldsymbol{\beta}$ is expressed as the sum of the previous complete log-likelihood and a new term function of the edges between the new node and the existing network:

$$\mathcal{L}_c^{n+1}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}(\boldsymbol{\beta}^{[n]}), \boldsymbol{\beta}) = \mathcal{L}_c^n(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}, \boldsymbol{\beta}) + \mathcal{L}_c(\mathbf{X}_{n+1,\bullet}, \mathbf{Z}^{[n+1]}, \boldsymbol{\beta}).$$

The principle of the recursive algorithm consists in computing the parameter $\boldsymbol{\beta}^{[n+1]}$ maximizing $\mathcal{L}_c^{n+1}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}(\boldsymbol{\beta}^{[n]}), \boldsymbol{\beta})$ and exploiting the fact that the new estimates are function of the old ones. But prior to compute the new estimate of the parameter vector, it is necessary to find the partition $\mathbf{Z}^{[n+1]}(\boldsymbol{\beta}^{[n]})$ which maximizes $\mathcal{L}_c^{n+1}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}(\boldsymbol{\beta}^{[n]}), \boldsymbol{\beta})$.

A suboptimal solution, which increases the complete log-likelihood, consists of assigning to the new node the most probable class (Maximum A Posteriori strategy) without changing the class of all the other nodes. Once the new classification matrix is estimated, the maximisation of the complete log-likelihood is straightforward and leads to new estimates of the parameter vector.

The recursive algorithm is then described by the two following steps each time a new (n)th node (and corresponding vertices) is considered:

- **Classification step:** assign each new node n to the class q^* which maximizes

  $$\mathcal{L}_c(\mathbf{X}_{n,\bullet}, q; \boldsymbol{\beta}) = \log \alpha_q + \sum_l \sum_{j \neq n} Z_{jl} \log f_{ql}(\mathbf{X}_{n,j}).$$

  Thus set Z_{nq} equal to 1 if $q = q^*$, 0 otherwise.

- **Estimation step:**
  - Update the sufficient statistics according to equation 2.3.2.
  - Update the parameters for all classes.

  $$\begin{cases} \alpha_q^{[n+1]} & = N_q^{[n+1]}/(n+1), \\ \psi_{ql}^{[n+1]} & = H_{ql}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]})/G_{ql}(\mathbf{Z}^{[n+1]}), \end{cases}$$

with the notation $\psi_{ql} = \frac{\partial a(\eta_{ql})}{\partial \eta_{ql}}$.

This algorithm increases the complete-log likelihood at each step, and requires at most as many iterations as the number of nodes. It can be considered as a special case of online reinforcement learning Likas (1999) where all clusters are strongly dependent.

Parameters update in the Bernoulli and Poisson cases. The connectivity estimator ψ_{ql} can be expressed as:

$$\psi_{ql}^{[n+1]} = \frac{H_{ql}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]})}{G_{ql}(\mathbf{Z}^{[n+1]}))} = \frac{H_{ql}(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}) + \xi_{ql}^{[n+1]}}{G_{ql}(\mathbf{Z}^{[n]})) + \zeta_{ql}^{[n+1]}}, \tag{2.20}$$

using the notation of Equation 2.19. Dividing the numerator and denominator by $G_{ql}(\mathbf{Z}^{[n]}))$ and using relation $G_{ql}(\mathbf{Z}^{[n]}) = G_{ql}(\mathbf{Z}^{[n+1]})) - \zeta_{ql}^{[n+1]}$ leads to

$$\psi_{ql}^{[n+1]} = \left(\psi_{ql}^{[n]} + \underbrace{\frac{\xi_{ql}^{[n+1]}}{G_{ql}(\mathbf{Z}^{[n+1]})) - \zeta_{ql}^{[n+1]}}}_{\mathbf{A}} \right) \left(\underbrace{\frac{G_{ql}(\mathbf{Z}^{[n+1]}) - \zeta_{ql}^{[n+1]}}{G_{ql}(\mathbf{Z}^{[n+1]})}}_{\mathbf{B}} \right).$$

Using the notation $\alpha^{[n+1]} = \frac{\zeta_{ql}^{[n+1]}}{G_{ql}(\mathbf{Z}^{[n+1]})}$, we can express the **A** term of the previous equation as,

$$\frac{\xi_{ql}^{[n+1]}}{G_{ql}(\mathbf{Z}^{[n+1]}) - \zeta_{ql}^{[n+1]}} = \frac{\xi_{ql}^{[n+1]}}{\zeta_{ql}^{[n+1]}} \frac{\alpha^{[n+1]}}{1 - \alpha^{[n+1]}},$$

and the **B** term simply as,

$$\frac{G_{ql}(\mathbf{Z}^{[n+1]}) - \zeta_{ql}^{[n+1]}}{G_{ql}(\mathbf{Z}^{[n+1]})} = 1 - \alpha^{[n+1]}.$$

Finally, the relation $\gamma^{[n+1]} = 1 - \alpha^{[n+1]}$ allows to express Equation 2.20 in the following form:

$$\psi_{ql}^{[n+1]} = \gamma^{[n+1]} \psi_{ql}^{[n]} + (1 - \gamma^{[n+1]}) \frac{\xi_{ql}^{[n+1]}}{\zeta_{ql}^{[n+1]}}. \tag{2.21}$$

Revisiting the nodes. Still the parameters can be further improved and the complete log-likelihood further increased by revisiting each node a few times. When m the number of iterations is greater than n the size of the network, it is possible to apply the above described recursive principle, and the algorithm can be continued as follows:

- **Classification step:** find a node $\mathbf{X}_{i,\bullet}$, whose class change improves the classification log-likelihood.
- **Estimation step:**
  - Update the sufficient statistics according to equation 2.3.2 taking into account the class change of node i.
  - Update the parameters for all classes with Equation 2.21.

When the classification step does not improve the classification log-likelihood, the algorithm can be stopped. Notice that this second phase of the algorithm can be related to relaxation labeling Rosenfeld et al. (1976), Hancock and Kittler (1990). In this context, the label of node i is the class vector. Using the notation $\mathbf{Z}_{\setminus i}$ which stands for the class of all nodes except node i, the label of this node is changed to the class q maximizing the posterior probability

$$\begin{aligned}\Pr(Z_{iq}=1|\mathbf{X},\mathbf{Z}_{\setminus i}) &= \frac{\Pr(Z_{iq}=1,\mathbf{Z}_{\setminus i},\mathbf{X})}{\Pr(\mathbf{Z}_{\setminus i},\mathbf{X})},\\ &= \frac{\Pr(Z_{iq}=1,\mathbf{Z}_{\setminus i},\mathbf{X})}{\sum_{q=1}^{Q}\Pr(Z_{iq}=1,\mathbf{Z}_{\setminus i},\mathbf{X})},\\ &= \frac{e^{\mathcal{L}_c(\mathbf{x}_{i,\bullet},q;\boldsymbol{\beta})}}{\sum_{\ell=1}^{Q}e^{\mathcal{L}_c(\mathbf{x}_{i,\bullet},\ell;\boldsymbol{\beta})}},\end{aligned}$$

and then fixed, before relaxing another node label.

2.4.3 Bernoulli affiliation model

One might reduce the complexity of the model by reducing the number of parameters. An interesting approach is to consider an simple affiliation model where two types of edges exist: edges between nodes of the same class and edges between nodes of different classes. Thus, each type of edge has a given probability: $\pi_{qq}=\lambda$ and $\pi_{ql}=\epsilon$ $(q\neq l)$. Self-loops are not taken into account (see for example Figure 2.5 page 62).

When considering this model (in an undirected graph without self-loops), the parameters are $\boldsymbol{\beta}=\{\alpha_1,...,\alpha_{Q-1},\lambda,\epsilon\}$. Observing that $Z_{iq}Z_{jl}\log\left(\pi_{ql}^{X_{ij}}(1-\pi_{ql})^{1-X_{ij}}\right)$ takes four different values according to the class of i and j, and their neighbourhood relationship, the classification likelihood can be expressed as:

$$\begin{aligned}\mathcal{L}_c(\mathbf{X},\mathbf{Z};\boldsymbol{\beta}) &= \sum_q n_q\log\alpha_q + n_\lambda\log\lambda + n_{1-\lambda}\log(1-\lambda)\\ &\quad + n_\epsilon\log\epsilon + n_{1-\epsilon}\log(1-\epsilon),\end{aligned}$$

where n_λ, $n_{1-\lambda}$, n_ϵ and $n_{1-\epsilon}$ are the number of node pairs defined by class and neighbourhood relationship (see Table 2.1).

	neighbours	not neighbours
same class	$n_\lambda=\sum_q n_{qq}$	$n_{1-\lambda}=\sum_q\frac{n_q(n_q-1)}{2}-n_\lambda$
different class	$n_\epsilon=\sum_{q\neq l}n_{ql}$	$n_{1-\epsilon}=\frac{n(n-1)}{2}-n_\lambda-n_{1-\lambda}-n_\epsilon$

Table 2.1 – *Statistics of the affiliation model computed from $n_{ql}=\sum_{i>j}x_{ij}z_{iq}z_{jl}$, the number of edges having nodes in class q and l, and $n_q=\sum_i z_{iq}$, the number of nodes of class q.*

Using Equation 2.21, the parameters $\hat{\boldsymbol{\beta}}$ which maximize this criterion for a given partition in an undirected Bernoulli affiliation model are as follows:

$$\begin{cases}\hat{\alpha}_q=\frac{n_q}{n},,\\ \hat{\lambda}=\frac{n_\lambda}{n_\lambda+n_{1-\lambda}},\\ \hat{\epsilon}=\frac{n_\epsilon}{n_\epsilon+n_{1-\epsilon}}.\end{cases}$$

The adaptation of the proposed algorithm to the affiliation model is straightforward and requires only the computation of the four statistics of the affiliation model (see Table 2.1) from the $n_{ql}^{[m]}$. This approach accelerates the estimation procedure in comparision with complete model involving π_{ql} parameters to estimate.

A last remark about this affiliation model is that it allows to explicitly compute the distribution of the degree K_i of node i, which is approximately a mixture of Poisson distribution $\mathcal{P}((n-1)[(1-\alpha_q)\epsilon + \alpha_q\lambda])$.

2.4.4 Initialization and online supervised classification

If one is mainly interested in finding clusters of nodes which have strong interconnection and weak between-connection, the above described algorithm can be, once again, simplified and accelerated by working with fixed parameter values. An example based on a real data set extracted from the Web is provided in Section 2.9.1 page 68. A possible interesting choice for clustering consists in choosing a high value for the probability λ of within-connection and a small one for the probability ϵ of between-group connection (*i.e.* $\lambda = 0.8$ and $\epsilon = 0.05$). This choice implicitly assumes the existence of clique-like clusters. Concerning the proportions, without any additional knowledge, it seems reasonable to consider clusters of the same size $\alpha_1 = ... = \alpha_Q = \frac{1}{Q}$.

Repeating step 2, with a given value of the parameter vector, produces a partition in a finite number of iterations. Starting from a random partition $\mathbf{z}^0$, each iteration considers a randomly chosen node and assigns this node to the cluster which results in the greatest increase of the classification log-likelihood. It is obvious this relaxation procedure increases the classification log-likelihood at each iteration and converges toward a local maximum since the criterion is upper-bounded.

This minimal clustering algorithm can be used as an effective initialization strategy for the previously described online MixNet algorithm. Notice that as all local optimization algorithms, the proposed online MixNet estimation strategy depends strongly on the initialization. A common way to circumvent this problem consists in testing multiple initialization points and selecting the best partition and parameters in terms of likelihood.

2.5 Stochastic Approximation EM for Network Mixture

2.5.1 A short presentation of SAEM

An original way of estimating the parameters of the MixNet model is to approximate the expectation of the complete data log-likelihood using Monte Carlo simulations corresponding to the Stochastic Approximation EM algorithm (Delyon et al. 1999). In situations where maximizing $\mathcal{Q}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ is not in a simple closed form, the SAEM algorithm maximizes an approximation $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ computed using standard stochastic approximation theory such that

$$\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')^{[k]} = \widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')^{[k-1]} + \rho_k\Big(\widetilde{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}') - \widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')^{[k-1]}\Big), \qquad (2.22)$$

where k is an iteration index, $\{\rho_k\}_{k\geq 1}$ a sequence of positive step size and where $\widetilde{Q}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ is obtained by Monte Carlo integration. This is a simulation of the expectation of the complete log-likelihood using the posterior $\Pr(\mathbf{Z}|\mathbf{X}\}$. Each iteration k of the algorithm is broken down into three steps:

Simulation of the missing data. This can be achieved using Gibbs Sampling of the posterior $\Pr(\mathbf{Z}|\mathbf{X})$. The result at iteration number k is $m(k)$ realizations of the latent class data $\mathbf{Z}$: $(\mathbf{Z}(1), ..., \mathbf{Z}(m(k)))$.

Stochastic Approximation of $\mathcal{Q}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ using Eq. 2.22, with

$$\widetilde{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}') = \frac{1}{m(k)} \sum_{s=1}^{m(k)} \log \Pr(\mathbf{X}, \mathbf{Z}(s); \boldsymbol{\beta}) \tag{2.23}$$

Maximization of $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')^{[k]}$ according to $\boldsymbol{\beta}$.

As regards the online version of the algorithm, the number of iterations k usually coincides with $n+1$, the number of nodes of the network.

2.5.2 Simulation of $\Pr(\mathbf{Z}|\mathbf{X})$ in the online context

We use Gibbs sampling which is applicable when the joint distribution is not known explicitly, but the conditional distribution of each variable is known. Here we generate a sequence of $\mathbf{Z}$ approaching $\Pr(\mathbf{Z}|\mathbf{X})$ using $\Pr(Z_{iq} = 1|\mathbf{X}, \mathbf{Z}_{\setminus i}\}$ where $\mathbf{Z}_{\setminus i}$ stands for the class of all nodes except node i. The sequence of samples is a Markov chain, and the stationary distribution of this Markov chain corresponds precisely to the joint distribution we wish to obtain. In the online context, we consider only one simulation to simulate the class of the last incoming node using

$$\begin{aligned}
\Pr(Z_{n+1,q} = 1|\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n]}) &= \frac{\Pr(Z_{n+1,q} = 1, \mathbf{Z}^{[n]}, \mathbf{X}^{[n+1]})}{\sum_{\ell=1}^{Q} \Pr(Z_{n+1,\ell} = 1, \mathbf{Z}^{[n]}, \mathbf{X}^{[n+1]})}. \\
&= \frac{\exp\left\{\boldsymbol{\beta}^t T(\mathbf{X}_{n+1,\bullet}, \mathbf{Z}^{[n]}, Z_{n+1,q})\right\}}{\sum_{\ell=1}^{Q} \exp\left\{\boldsymbol{\beta}^t T(\mathbf{X}_{n+1,\bullet}, \mathbf{Z}^{[n]}, Z_{n+1,\ell})\right\}} \\
&\propto \exp\left(\omega_q + \sum_{\ell=1}^{Q} \eta_{q\ell} \sum_{j=1}^{n} Z_{j\ell} h(X_{n+1,j}) + \sum_{\ell=1}^{Q} N_{\ell}^{[n]} a(\eta_{q\ell})\right).
\end{aligned}$$

2.5.3 Computing $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$ in the online context

As regards the online version of the SAEM algorithm, the difference between the old and the new complete-data log-likelihood may be expressed as:

$$\log \Pr(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}, \boldsymbol{\beta}) - \log \Pr(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}, \boldsymbol{\beta}) = \log \alpha_q + \sum_{l, i<n+1} Z_{il} \log \Pr(X_{n+1,i}|Z_{n+1,q} Z_{il}),$$

where the added simulated vertex label is equal to q ($Z_{n+1,q} = 1$).

Recall that in the online framework, the label of the new node has been sampled from the Gibbs sampler described in Section 2.5.2. Consequently

only one possible label is considered in this equation. Then a natural way to adapt Equation 2.22 to the online context is to approximate

$$\widetilde{\mathcal{Q}}_{n+1}(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]}) - \widehat{\mathcal{Q}}_n(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]})$$

by

$$\log \Pr(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}, \boldsymbol{\beta}) - \log \Pr(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]}, \boldsymbol{\beta}).$$

Indeed, this quantity corresponds to the difference between the log-likelihood of the original network and log-likelihood of the new network including the additional node. Notice that the larger the network, the larger its associated complete expected log-likelihood. Thus $\log \Pr(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}, \boldsymbol{\beta})$ becomes smaller and smaller compared to $Q(\boldsymbol{\beta}|\boldsymbol{\beta}')$ as n increases. The decreasing step ρ_n is thus set to one in this online context. We propose the following update equation for stochastic online EM computation of MixNet conditional expectation:

$$\widehat{\mathcal{Q}}_{n+1}(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]}) = \widehat{\mathcal{Q}}_n(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]}) + \log \alpha_q + \sum_{l,i<n+1} Z_{il} \log \Pr(X_{n+1,i}|Z_{n+1,q}Z_{il}),$$

where $\mathbf{Z}_{n+1}$ is drawn from the Gibbs sampler.

2.5.4 Maximizing $\widehat{\mathcal{Q}}(\boldsymbol{\beta}|\boldsymbol{\beta}')$, and parameters update

The principle of online algorithms is to modify the current parameter estimation using the information added by a new available $[n+1]$ node and its corresponding connections $\mathbf{X}_{n+1,\bullet}$ to the already existing network. Maximizing $\widehat{\mathcal{Q}}_{n+1}(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]})$ according to $\boldsymbol{\beta}$ is straightforward and produces the maximum likelihood estimates for iteration $[n+1]$. Here we have proposed a simple version of the algorithm by setting the number of simulation to one ($m(k) = 1$). In this context, the difference between $\widehat{\mathcal{Q}}_n(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]})$ and $\widehat{\mathcal{Q}}_{n+1}(\boldsymbol{\beta}|\boldsymbol{\beta}^{[n]})$ implies only the terms of the complete log-likelihood which are a function of node $n+1$. Using notation $\psi_{ql} = \frac{\partial a(\eta_{ql})}{\partial \eta_{ql}}$, we get

$$\begin{cases} \alpha_q^{[n+1]} & = N_q^{[n+1]}/(n+1), \\ \psi_{ql}^{[n+1]} & = H_{ql}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]})/G_{ql}(\mathbf{Z}^{[n+1]}), \end{cases}$$

where (ξ_{ql}, ζ_{ql}) were defined in the previous Section. Notice that updating the function ψ_{ql} of the parameter of interest is often more convenient in online context than directly considering this parameter of interest. An example of parameter update is given for the Bernoulli and Poisson cases in the Appendix A.3.4.

Once all the nodes in the network have been visited (or are known), similarly to the online CEM algorithm, the parameters can be further improved and the complete log-likelihood better approximated by continuing with the SAEM algorithm described above.

2.6 Application of online algorithm to Variational EM methods

Variational EM (VEM) methods constitute an alternative to SAEM. Their principle is to approximate the untractable distribution $\Pr(\mathbf{Z}|\mathbf{X};\boldsymbol{\beta})$ by a

newly introduced distribution on $\mathbf{Z}$ denoted by $\mathcal{R}$. Then this new distribution is used to optimize $\mathcal{J}(\mathbf{X}, \mathcal{R}(\mathbf{Z}); \boldsymbol{\beta})$, an approximation (lower bound) of the incomplete-data log-likelihood $\log \Pr(\mathbf{X}; \boldsymbol{\beta})$, defined such that

$$\mathcal{J}(\mathbf{X}, \mathcal{R}(\mathbf{Z}); \boldsymbol{\beta}) \quad = \quad \log \Pr(\mathbf{X}; \boldsymbol{\beta}) - \mathrm{D}_{KL}(\mathcal{R}(\mathbf{Z}) | \Pr(\mathbf{Z}|\mathbf{X}; \boldsymbol{\beta})),$$

with $\mathrm{D}_{KL}(\bullet|\bullet)$ being the Kullback-Leibler divergence between probability distributions (Jordan et al. 1999). Then one must choose the form of $\mathcal{R}$, and the product of Multinomial distributions is natural in the case of MixNet, with $\log \mathcal{R}(\mathbf{Z}) = \sum_i \sum_q Z_{iq} \log \tau_{iq}$, and the constraint $\sum_q \tau_{iq} = 1$. In this case, the form of $\mathcal{J}(\mathbf{X}, \mathcal{R}(\mathbf{Z}); \boldsymbol{\beta})$ is:

$$\begin{aligned} \mathcal{J}(\mathbf{X}, \mathcal{R}(\mathbf{Z}); \boldsymbol{\beta}) &= \sum_{\mathbf{Z}} \mathcal{R}(\mathbf{Z}; \boldsymbol{\tau}) \log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta}) - \sum_{\mathbf{Z}} \mathcal{R}(\mathbf{Z}; \boldsymbol{\tau}) \log \mathcal{R}(\mathbf{Z}; \boldsymbol{\tau}), \\ &= \mathcal{Q}(\boldsymbol{\tau}, \boldsymbol{\beta}) + \mathcal{H}(\mathcal{R}(\mathbf{Z}; \boldsymbol{\tau})), \end{aligned}$$

with $\mathcal{Q}(\boldsymbol{\tau}, \boldsymbol{\beta})$ an approximation of the conditional expectation of the complete-data log-likelihood, and $\mathcal{H}(\mathcal{R}(\mathbf{Z}; \boldsymbol{\tau}))$ the entropy of the approximate *posterior* distribution of $\mathbf{Z}$.

The implementation of variational methods in online algorithms relies on the additivity property of $\mathcal{J}(\mathbf{X}, \mathcal{R}(\mathbf{Z}); \boldsymbol{\beta})$ when nodes are added. This property is straightforward: $\mathcal{Q}(\boldsymbol{\tau}, \boldsymbol{\beta})$ is additive thanks to Proposition 2.1 (because $\mathcal{R}(\mathbf{Z})$ is factorized), and $\mathcal{H}(\mathcal{R}(\mathbf{Z}; \boldsymbol{\tau}))$ is also additive, since the hidden variables are supposed independent under $\mathcal{R}$ and the entropy of independent variables is additive. The variational algorithm is very similar to an EM algorithm, with the E-step being replaced by a variational step which aims at updating variational parameters. Then a standard M-step follows. In the following, we give the details of these two steps in the case of a variational online algorithm.

2.6.1 Online variational step

When a new node is added it is necessary to compute its associated variational parameters $\{\tau_{n+1,q}\}_q$. If we consider all the other τ_{iq} for $i < n+1$ as known, the $\{\tau_{n+1,q}\}_q$ are obtained by differentiating the criterion

$$\mathcal{J}\left(\mathbf{X}^{[n+1]}, \mathcal{R}(\mathbf{Z}^{[n+1]}); \boldsymbol{\beta}\right) + \sum_{i=1}^{n+1} \Lambda_i \left(\sum_{q=1}^{Q} \tau_{iq} - 1 \right),$$

where the Λ_i are the Lagrangian parameters. Since function $\mathcal{J}$ is additive according to the nodes, the calculation of its derivative according to $\tau_{n+1,q}$ gives:

$$\omega_q^{[n]} + \sum_{l=1}^{Q} \sum_{j=1}^{n} \tau_{jl}^{[n]} \left(\eta_{ql}^{[n]} h(X_{n+1,j}) + a(\eta_{ql}^{[n]}) \right) - \log \tau_{n+1,q} + 1 + \Lambda_{n+1} = 0$$

This leads to

$$\tau_{n+1,q} \propto \alpha_q^{[n]} \exp \left\{ \sum_{l=1}^{Q} \sum_{j=1}^{n} \tau_{jl}^{[n]} \left(\eta_{ql}^{[n]} h(X_{n+1,j}) + a(\eta_{ql}^{[n]}) \right) \right\}, \forall q \in \{1, ..., Q\}. \tag{2.24}$$

2.6.2 Maximization/Update step

To maximize the approximated expectation of the complete log-likelihood according to $\boldsymbol{\beta}$, we solve

$$\frac{\partial \mathcal{Q}_{n+1}(\boldsymbol{\tau}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbb{E}_{\mathcal{R}^{[n]}}\left(\frac{\partial \log \Pr(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]}; \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right) = 0. \tag{2.25}$$

Differentiating Equation 2.25 with respect to parameters $\{\omega_q\}$ gives the following update equation:

$$\alpha_q^{[n+1]} = \frac{1}{n+1}\left(\sum_{i=1}^{n} \tau_{iq}^{[n]} + \tau_{n+1,q}\right).$$

The other update equation is obtained by considering parameters $\{\eta_{ql}\}$, and using notation ψ_{ql}, which gives:

$$\psi_{ql}^{n+1} = \frac{\mathbb{E}_{\mathcal{R}^{[n]}}\left(H_{ql}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]})\right)}{\mathbb{E}_{\mathcal{R}^{[n]}}\left(G_{ql}(\mathbf{Z}^{[n+1]})\right)},$$

Thanks to proposition 2.1, which gives the relationships between sufficient statistics at two successive iterations, parameters can be computed recursively using the update of the expectation of the sufficient statistics, such that

$$\begin{aligned}
\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n+1]}\right) &= \mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n]}\right) + \mathbb{E}_{\mathcal{R}^{[n]}}\left(Z_{n+1,q}\right), \\
\mathbb{E}_{\mathcal{R}^{[n]}}\left(H_{ql}(\mathbf{X}^{[n+1]}, \mathbf{Z}^{[n+1]})\right) &= \mathbb{E}_{\mathcal{R}^{[n]}}\left(H_{ql}(\mathbf{X}^{[n]}, \mathbf{Z}^{[n]})\right) + \mathbb{E}_{\mathcal{R}^{[n]}}\left(\xi_{ql}^{[n+1]}\right), \\
\mathbb{E}_{\mathcal{R}^{[n]}}\left(G_{ql}(\mathbf{Z}^{[n+1]})\right) &= \mathbb{E}_{\mathcal{R}^{[n]}}\left(G_{ql}(\mathbf{Z}^{[n]})\right) + \mathbb{E}_{\mathcal{R}^{[n]}}\left(\zeta_{ql}^{[n+1]}\right).
\end{aligned}$$

An example of parameters update is given in Appendix A.3.5 for both the Bernoulli and the Poisson distributions. Note the similarity of the formula compared with the SAEM strategy. Hidden variables $\mathbf{Z}$ are either simulated or replaced by their approximated conditional expectation (variational parameters).

2.7 Choosing the number of clusters

As the algorithm relies on a statistical model, it is possible to use the Integrated Classification Likelihood (ICL) to choose the optimal number of classes Biernacki et al. (2000). This choice is done by running our online algorithm concurrently for models from 2 to Q classes and selecting the solution which maximizes the ICL criterion.

In our situation, following Daudin Daudin et al. (2008), the ICL criterion can be written as:

$$ICL(Q) = \underbrace{-2\mathcal{L}_c(\mathbf{X}, \boldsymbol{\beta})}_{\text{A}} + \underbrace{(Q-1)\log(n) + Q^2 \log(n(n-1))}_{\text{B}}$$

where n is the number of nodes, A is related to the classification log-likelihood, B to the free number of parameters. The first term of B weights

the information of Q proportions α_qs involved by n data and second term ponders Q^2 probabilities η_{ql}s involved by $n(n-1)$ terms (number of edges). The ICL criterion is essentially the ordinary BIC considering the complete log-likelihood instead of the log-likelihood. Consider for example Figure 2.6 which exhibits this strategy for synthetic data of Figure 2.5.

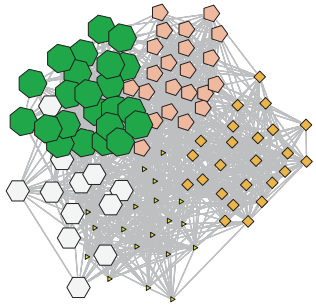


Figure 2.5 – *Simulation of a 100 nodes graph with 5 classes according to an affiliation model (see Section 2.4.3 page 2.4.3).*

The ICL criterion computed for models with 2 to 16 clusters shows a clear maximum for 5 clusters, which is the real number used for data simulation.

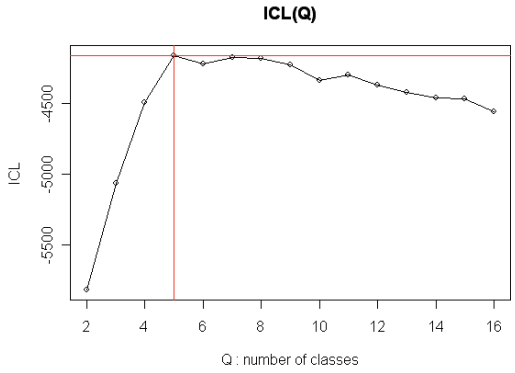


Figure 2.6 – *Integrated Classification Likelihood Criterion in function of the number of clusters computed for the simulated graph of Figure 2.5.*

2.8 Experiments using simulation

Experiments are carried out to assess the trade-off established by online algorithms in terms of quality of estimation and speed of execution. We propose a two-step simulation study. We first report simulation experiments using synthetic data generated according to the assumed random graph model. In this first experiment we use simple affiliation models to check precisely the quality of the estimations given by ours online algorithms. Results are compared to the batch variational EM proposed by

Daudin et al. (2008) to assess the effect of the online framework on the estimation quality and on the speed of execution. An ANSI C++ implementation of the algorithms is available, as well as an *R* package named MixeR , along with public data sets. Further details concerning the software are given in Appendix A.8.

2.8.1 Comparison of partitions

Comparing the estimated partition with the true partition is not as straightforward as comparing the parameter estimates. In order to evaluate the agreement between these two partitions, we use the adjusted Rand index Hubert and Arabie (1985) which lies between 0 and 1. The computation of this index is based on a ratio between the number of node pairs belonging to the same and to different classes when considering the true partition and the estimated partition.

Definition 2.1 *Given a set of n elements* $\mathbf{E} = \{E_1, \ldots, E_n\}$ *and two partitions of* $\mathbf{E}$ *to compare,* $X = \{x_1, \ldots, x_r\}$ *and* $Y = \{y_1, \ldots, y_s\}$*, we define the following:*

- n_1*: the number of pairs of elements in* $\mathbf{E}$ *that are in the same set in* X *and in the same set in* Y*,*
- n_2*: the number of pairs of elements in* $\mathbf{E}$ *that are in different sets in* X *and in different sets in* Y*,*
- n_3*: the number of pairs of elements in* $\mathbf{E}$ *that are in the same set in* X *and in different sets in* Y*,*
- n_4*: the number of pairs of elements in* $\mathbf{E}$ *that are in different sets in* X *and in the same set in* Y*.*

The Adjusted Rand index, AR, is:

$$AR = \frac{2(n_1 n_2 - n_3 n_4)}{((n_1 + n_4)(n_4 + n_2) + (n_1 + n_3)(n_3 + n_2))}$$

Two identical partitions have an adjusted Rand index equal to 1.

2.8.2 Comparison of algorithms

Simulations set-up. In these experiments, we again assume that edges are Bernoulli distributed. We consider a simple affiliation model where two types of edges exist: edges between nodes of the same class and edges between nodes of different classes. Each type of edge has a given probability, respectively $\pi_{qq} = \lambda$ and $\pi_{ql} = \epsilon$. Five affiliation models are examined (see Table 2.2) with $\lambda = 1 - \epsilon$ to limit the number of varying parameters in the experiment.

The parameter λ controls the complexity of the model. The differences between the five models relate to their modular structure, which varies from no structure (almost the Erdős-Rényi model) to strong modular structure (low inter-module connectivity and strong intra-module connectivity, or strong inter-module connectivity and low intra-module connectivity). Figure 2.7 illustrates three kinds of connectivity which allows to represent graphically models 1, 4 and 5.

Model	ϵ	λ
1	0.3	0.7
2	0.35	0.65
3	0.4	0.6
4	0.5	0.5
5	0.9	0.1

Table 2.2 – *Parameters of the five affiliation models considered in the experimental setting.*

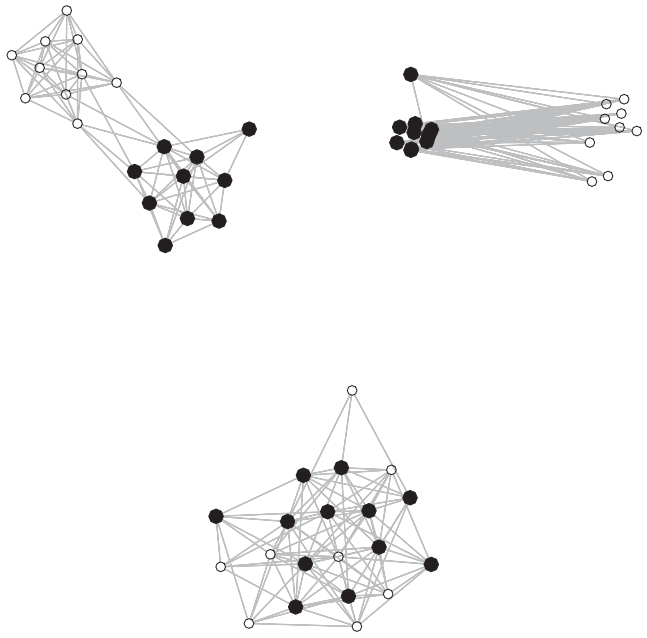


Figure 2.7 – *Top left: low inter-module connectivity and strong intra-module connectivity (model 1), Top right: strong inter-module connectivity and low intra-module connectivity (model 5), Bottom center: Erdős-Rényi model (model 4).*

For each affiliation model, we generate graphs with $Q \in \{2, 5, 20\}$ groups mixed in the same proportions $\alpha_1 = ... = \alpha_Q = \frac{1}{Q}$ and with $n \in \{100, 250, 500, 750, 1000, 2000\}$ nodes. We thus generate a total of 45 graph models, each being simulated 30 times.

Algorithms set-up. In Appendix A.4 and in (Zanghi et al. 2008), we showed that the variational MixNet approach was more accurate than other methods like spectral clustering (Ng et al. 2002) and a *k*-means like algorithm. We also showed that the online CEM MixNet produced similar results to variational MixNet while significantly reducing the computational cost. Consequently, the online SAEM and online variational algorithms developed here will be compared with the online CEM and batch variational (batch MixNet) as references. To avoid initialization issues, each algorithm is started with multiple initialization points and the best result is selected based on its likelihood. Thus, for each simulated network, the algorithm is run 10 times and the number of clusters is chosen using the Integrated Classification Likelihood criterion, as proposed in Section 2.7, which almost never makes a mistake on simulated networks. In these first experiments we use for the online algorithms the same stopping condition than the variational batch MixNet condition which is based on a stabilization of the estimated parameters. The parameter comparison is done at the end of each epoch (one visit of all network nodes).

Simulations results. A first result is that every algorithm shows negligible bias and variance for highly structured models (models 1, 2, 5, Table 2.3). While this is also true for intermediate cases (model 4), this result does not stand for less structured cases (model 3). Even if the Batch MixNet still performs well in this case, the online CEM is the most efficient among online versions (followed by the online variational algorithm).

	online SAEM		online Variational		batch MixNet		online CEM	
Model	$\bar{\epsilon}$	$\bar{\lambda}$	$\bar{\epsilon}$	$\bar{\lambda}$	$\bar{\epsilon}$	$\bar{\lambda}$	$\bar{\epsilon}$	$\bar{\lambda}$
model 1	0.30	0.69	0.30	0.70	0.30	0.70	0.30	0.70
model 2	0.36	0.60	0.35	0.64	0.35	0.65	0.35	0.64
model 3	0.44	0.44	0.44	0.45	0.40	0.60	0.43	0.47
model 4	0.51	0.48	0.50	0.50	0.50	0.50	0.51	0.48
model 5	0.10	0.90	0.10	0.90	0.10	0.90	0.10	0.90

Table 2.3 – *Parameters of the five affiliation models in the experiment. The Q modules are mixed in the same proportion. Each model considers* $n = 500$ *nodes and* $Q = 5$ *groups.*

We also focus on the Rand Index for each algorithm. Indeed, even if poor estimation of λ reveals a small Rand Index (Table 2.4), good estimates do not always lead to correctly estimated partitions. An illustration is given with model 3 for which algorithms produce good estimates with poor Rand Index, due to the non modular structure of the network. Then Figure 2.8 shows that the online CEM and online variational algorithms always perform better than the online SAEM. As expected, the performance increase with the number of nodes (Table 2.5).

	online SAEM		online Variational		batch MixNet		online CEM	
Model	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}
model 1	0.98	0.02	0.98	0.02	0.99	0.02	0.98	0.02
model 2	0.96	0.07	0.97	0.07	0.98	0.01	0.97	0.07
model 3	0.13	0.13	0.10	0.15	0.85	0.14	0.25	0.16
model 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
model 5	1	0.00	1	0.01	1	0.01	1	0.01

Table 2.4 – *Means and standard deviations of the Rand Index for all models with q and n fixed.*

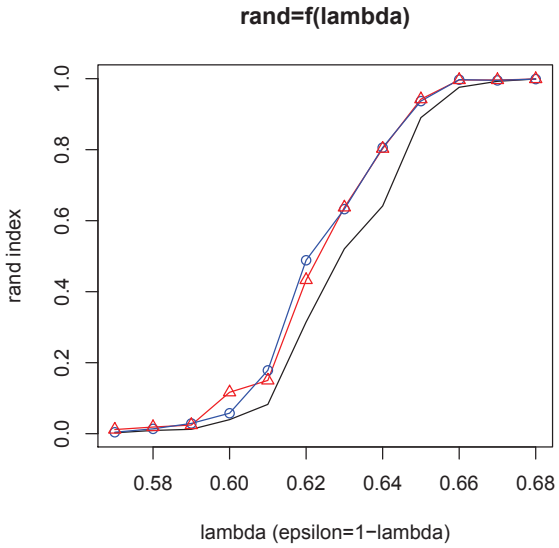


Figure 2.8 – *Rand Index evolution for* $\lambda \in \{0.58, 0.59, \ldots, 0.68\}$. *The plain line represents the online SAEM algorithm, the* $\triangle$ *line represents the online CEM algorithm and the* $\circ$ *line represents the online variational algorithm.*

Since the aim of online methods is to provide computationally efficient algorithms, the performance mentioned above should be put in perspective with the speed of execution of each algorithm. Indeed, Table 2.5 shows the strong gain of speed provided by online methods compared with the batch algorithm, with the online variational algorithm being the fastest. Actually, altought the same computational cost to handle a node is shared by the different algorithms (approximatively $O(nq^2)$), the main gain of speed obtained by the online algorithms is due to the realization of a single epoch to provide the final results (contrary to the batch algorithm that performs 10 iterations). Since the quality of partition estimation remains strong, the online variational algorithm appears very attractive and suitable for large graphs. Note that similary to the other MixNet algorithms, a weakness of this algorithm is that it is not easily parallelizable with a distributed computing framework (see Section 4.3.4 page 114).

	online SAEM		online Variational		batch MixNet		online CEM	
N	$\overline{rand}$	$\overline{time}$	$\overline{rand}$	$\overline{time}$	$\overline{rand}$	$\overline{time}$	$\overline{rand}$	$\overline{time}$
$n = 100$	0.14	0.07s	0.14	0.11s	0.26	0.21s	0.14	0.07s
$n = 250$	0.47	0.76s	0.48	0.77s	0.99	2.08s	0.48	0.74s
$n = 500$	0.64	0.97s	0.67	1.02s	1	25.00s	0.66	0.95s
$n = 750$	0.82	2.20s	0.83	2.36s	1	125.30s	0.83	2.14s
$n = 1000$	0.91	9.44s	0.92	9.60s	1	805.93s	0.92	9.37s
$n = 2000$	0.98	147.67s	0.99	148.31s	1	13136.66s	0.99	147.33s

Table 2.5 – *Means of the Rand Index with speed of the algorithms. $q = 5$, model 2.*

The above results show that a strong case may be made for the online variational algorithm when choosing between alternative clustering methods. Consequently, we shall now compare it with two suitable "rivals" for large networks: a basic spectral clustering algorithm (Ng et al. 2002), and one of the popular community detection algorithms (Newman 2006a). The spectral clustering algorithm searches for a partition in the space spanned by the eigenvectors of the normalized Laplacian, whereas the community detection algorithm looks for modules which are defined by high intra-connectivity and low inter-connectivity.

For our five models with arbitrary fixed parameters $n = 1000$, $Q = 3$, we ran these algorithms and computed the Rand Index for each of them. From Table 2.6 we see that our online variational algorithm always produces the best clustering of nodes.

	Community Detection		Spectral Clustering		online Variational	
Model	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}
1	1.00	0.00	0.97	0.14	1.00	0.00
2	0.99	0.01	0.98	0.00	1.00	0.00
3	0.97	0.02	0.97	0.00	1.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.92	0.19	1.00	0.00

Table 2.6 – *Means and standard deviation of the Rand Index for the five models computed over 30 different runs for graph clustering competitors and Variational algorithms.*

Since generated networks favor MixNet algorithms, these results correspond to what was expected: the online variational algorithm always yields the best node classification. Apart from model 4, it will also be remarked that the spectral algorithm is fairly efficient with a slight bias, and so the spectral clustering algorithm is consistently more accurate than the community algorithm, the latter failing completely when applied to model 5. Although the community algorithm appears less well adapted to these experiments, we shall see in the next section that this algorithm is particularly suitable when partitioning data sets whose nodes are densely interconnected.

2.9 Application

Finally, we propose to illustrate our online EM-like algorithms on real and large data sets extracted from the Web. Taking advantage of the MixNet model allows us to reveal the connectivity information induced by hyperlinks which differs from classical module detection algorithms. In the first experiment, we apply the online CEM "clustering" algorithm using fixed parameters on a sample of the French Political Blogosphere. The agreement between the real and the estimated degrees distribution is also considered. In a second step, we use this data set as a starting point to simulate growing networks with complex structure, and to assess the performance of online methods on this type of networks. Then, a second network representing the 2008 U.S. Presidential WebSphere is studied to analyse the MixNet results and theirs interpretations. We are also interested in the differences between MixNet groups and Newman's modules.

2.9.1 French Political Blogosphere network

Data presentation. The data set consists of a single day snapshot of over 1,100 political blogs automatically extracted the 14 october 2006 and manually classified by the "Observatoire Présidentielle" project: `www.observatoire-presidentielle.fr`. This project is the result of a collaboration between *Linkfluence* [1] and *Exalead* [2] and aims at analyzing the French presidential campaign on the Web.

In this data set, nodes represent hostnames (a hostname contains a set of pages) and edges represent hyperlinks between different hostnames. If several links exist between two different hostnames, we collapse them into a single one. Note that intra domain links can be considered if hostnames are not identical. Finally, in this experimentation we consider that edges are not oriented which is not realistic but which does not affect the interpretation of the groups. This network presents an interesting communities organization due to the existence of several political parties and commentators. We assume that authors of these blogs tend to link, by political affinities, blogs with similar political positions. A sample of 196 blogs of this political blogoshere is provided within the `MixeR` package (see Section A.8) and can be seen on Figure 2.9.

[1] see `http://linkfluence.net/`
[2] see `http://exalead.com/`

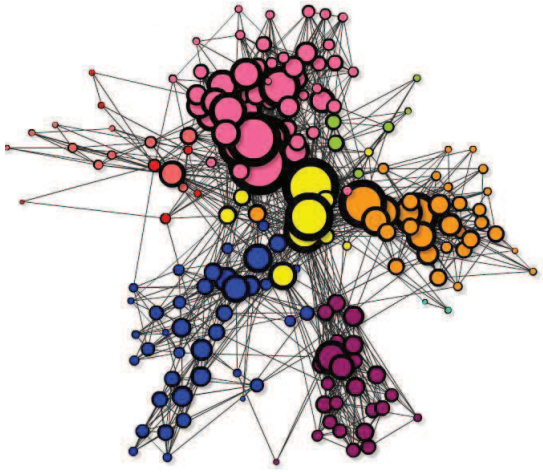


Figure 2.9 – *Network of the blogopole* `www.blogopole.fr`.

Six known communities compose this network: **G**auche ("french democrat"), **D**ivers **C**entre (Moderate party), **D**roite (french republican), **E**cologiste (green), **L**iberal (supporters of economic-liberalism) and finally **A**nalysts. Proportions of blogs in these communities are respectively 0.36, 0.23,0.21, 0.08, 0.08 and 0.04.

Finding clusters. Although the optimal number of clusters is 11 (with the MixNet parameters given in Table 2.8 page 71) according to the penalized likelihood criterion discribed in Section 2.7), we are interested, in this experimentation, in finding six groups of blogs using the online CEM clustering algorithm of Section 2.4.4 for a given pair of ϵ and λ. The number of groups is then fixed to six in order to compare the true partition of blogs with the estimated partition running our algorithm. The "optimal" couple of parameters is obtained using a grid search approach in order to maximize the agreement between the real and estimated partitions. Finally, the pair ($\lambda = 0.55$, $\epsilon = 0.04$) gives the maximal agreement between both partitions with an acceptable Rand index value (0.34) and validates the assumption that political affinities can be detected using the structure of the political blogosphere.

		Estimated					
		DC	A	D	E	G	L
True	DC	172	32	4	18	3	25
	A	2	22	2	5	5	5
	D	3	27	165	11	1	26
	E	1	2	0	80	1	0
	G	5	97	12	66	181	45
	L	1	1	3	1	0	82

Table 2.7 – *Contingency table comparing true and estimated partitions*

Table 2.7 shows a contingency table of the counts of given and estimated blogs classes. Except for the **A** class, we can observe a relative coherence between these two partitions. In fact, the **A** class is a hub class

constituted of blogs which links the other classes in order to analyze and comment the French political news. Our algorithm overestimates the number of blogs contained in this class and generates important classification differences. We can also observe that there are classification errors produced by the political proximity of parties. For example, the estimated **E** class has many blogs belonging to the real **G** class. Finally, we illustrate on Figure 2.10 the density of links using the adjacency matrix projection of the network after reordering by class estimation. We can observe that there is a higher density of intra-group links than inter-groups which validates our MixNet model. Note that classification errors of the **A** class can be observed creating horizontal and vertical bands by linking blogs of other classes.

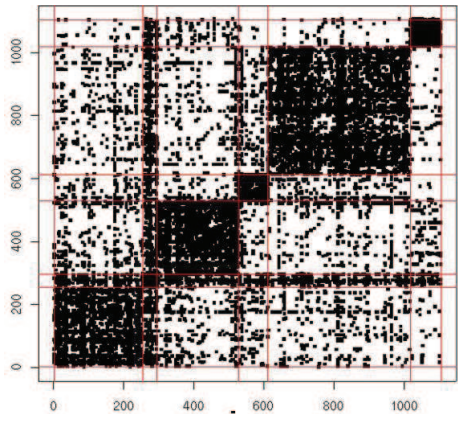


Figure 2.10 – *Adjacency matrix of the blogopole network after reordering according to the estimated partition.*

Degrees distribution. As mentioned in Section 2.4.3, the affiliation model allows to compute the distribution of the degree which is approximately a mixture of Poisson distribution. Figure 2.11 illustrates both the observed and the estimated distributions.

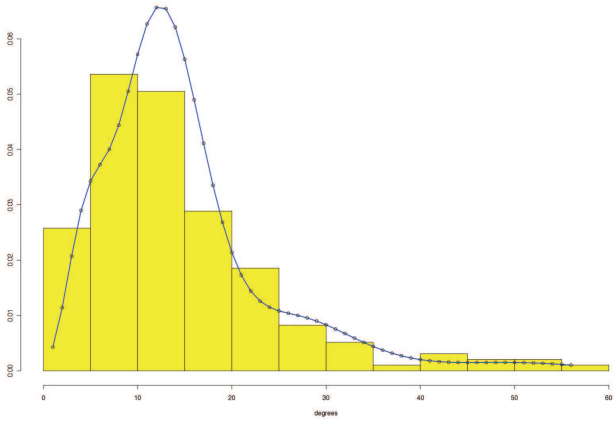


Figure 2.11 – *Degree distribution of the websites of the blogopole network. The histogram and the curve respectively represent the observed and the estimated distributions.*

Realistic networks growthing over time As the algorithm is motivated by large data sets, we are interested in simulating a realistic complex structure and studying its growthing over time. For this purpose, we use the previous sample of the French Political Blogosphere network as a starting point. This complex connectivity pattern of Figure 2.9 is summarized by the MixNet parameters given in Table 2.8 using the algorithm of Daudin et al. (2008).

	1	2	3	4	5	6	7	8	9	10	11
1	8	.	.	13	.	.	15	.	.	.	.
2	.	100	.	.	.	19	89	6	.	66	.
3	.	.	39	83	12	6	10	.	.	.	.
4	13	.	83	100	72	38	67	.	.	.	.
5	.	.	12	72	83	17	20	.	.	.	.
6	.	19	6	38	17	15	60	.	.	.	.
7	15	89	10	67	20	60	100	21	.	19	19
8	.	6	.	.	.	.	21	35	.	.	.
9	.	.	.	.	.	.	.	.	93	.	.
10	.	66	.	.	.	.	19	.	.	22	.
11	.	.	.	.	.	.	19	.	.	.	55

Table 2.8 – *The table corresponds to the probabilities (×100) of connection between the 11 selected clusters (using a penalized likelihood criterion discribed in Daudin et al. (2008)). Dots in the table correspond to connections lower than* 1%.

We generate 200 nodes networks from this model, then we simulate the growth over time of theses networks by adding new nodes according to the same model and we use the online algorithm to update parameters sequentially. The result is striking: even on very large networks with $\sim$ 13,000 nodes and $\sim$ 13,000,000 edges, the online algorithm allows us to estimate mixture parameters with negligible classification error in $\sim$ 6 minutes (Table 2.9). This is the only algorithmic framework that allows to perform model based clustering on networks of that size.

# nodes (previous+new)	ave. # edges	ave. rand	ave. cpu time (s)
200	3131.72	0.94	0.9
200 + 200	50316.32	0.998	0.4
400 + 400	12486.24	0.999	1.4
800 + 800	201009.5	1	5.7
1600 + 1600	803179.6	1	22.8
3200 + 3200	3202196	1	91.9
6400 + 6400	12804008	1	371.1

Table 2.9 – *Quality of the clustering procedure in terms of Rand Index when the network grows over time. Each configuration has been simulated 100 times.*

2.9.2 The 2008 U.S. Presidential WebSphere

Data presentation. As mentioned in Adamic and Glance (2005), the 2004 U.S. Presidential Election was the first where the web and, in particular, blogging played an important role. Although only a small minority of Americans actually used these Weblogs, their influence extended

far beyond their readership, as a result of their interactions with national mainstream media. With the impact of new social network websites like Myspace and Facebook, the web had a stronger influence during the U.S.political campaign in 2008. In this real community extraction experiment, we used a real data set obtained on November 7th 2007 by the French company *Linkfluence* (Information Networks, Territories and Geography) using a specific methodology similar to Fouetillou (2007). This data set consists of a one-day snapshot of over 130,520 links and 1,870 manually classified websites (676 liberal, 1,026 conservative and 168 independent) where nodes are connected if there exists a citation from one to another.

The 2008 U.S. Presidential WebSphere acquisition This data set consists of a one-day snapshot of over two thousand websites, one thousand of which featured in two online directories: `http://wonkosphere.com` and `http://www.politicaltrends.info`. The first site provides a manual classification, and the second an automatic classification based on text analysis. From this seed of a thousand sites, a web crawler (Drugeon 2005) collected a maximum of 100 pages per hostname. External links were examined to check the connectivity with visited and unvisited websites. If websites were still unvisited, and if there existed a minimal path of distance less than two between a hostname which belongs to the seed and these websites, then the web crawler collected them.

Using this seed-extension method, 200,000 websites were collected, and a network of websites was created where nodes represent hostnames (a hostname contains a set of pages) and edges represent hyperlinks between different hostnames. Multiple links between two different hostnames were collapsed into a single link. Intra-domain links were taken into account if hostnames were not similar. For this web network, we computed an authority score (Kleinberg 1999) and keyword score TF/IDF (Salton et al. 1975) on focused words (political entities) in order to identify respectively nodes with high-quality websites (high authority scores) and centered on those topics (on a political corpus). 870 new websites emerged out of these two criteria. They were checked by experts and the validity of the seed was confirmed. The final tally was 130,520 links and 1,870 sites: 676 liberal, 1,026 conservative and 168 independent. Figure 2.12 illustrates a layout of this network using *Gephi*[3] which is described in Bastian et al. (2009).

Comparison with a community detection algorithm. A first step consists in comparing the results of MixNet with the community detection algorithm proposed by Newman (2006a)[4]. If the political classification is used as a reference, the community algorithm produces better agreement with a $randIndex = 0.59$, compared with a $randIndex = 0.25$ for MixNet. However, it appears that this comparison favors Newman, whereas the methods have different objective. Indeed, the community algorithm aims at finding modules which are defined by high intra-connectivity and low

[3] `http://gephi.org/`

[4] Note that the comparison should be almost similar using newer modularity based methods such as Blondel et al. (2008)

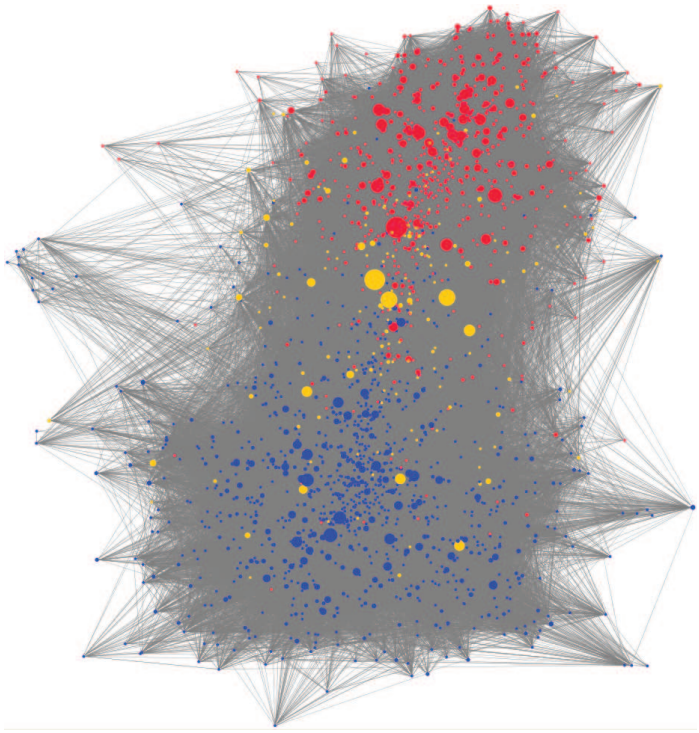


Figure 2.12 – *Graph Layout of the 2008 U.S. Presidential WebSphere using* Gephi. *The red dots represent Conservative websites, the yellow are the Independents and finally the blue dots the Liberal websites.*

inter-connectivity. Given that websites tend to link to one another in line with political affinities, the link topology corresponding to the manual classification naturally favors the community module definition. The objective function can also help to explain the community algorithm's suitability for this data set, since the quality of a partition in terms of Newman's modules can be expressed in terms of the *modularity*, which is maximized. The value of this modularity is a scalar between -1 and 1 and measures the density of links inside communities as compared to links between communities (Newman 2006a). When applying both algorithms on our political network with $Q = 3$, the online variational algorithm yields a $modularity = 0.20$, whereas the community algorithm yields a $modularity = 0.30$, which is close to the manual partition modularity of 0.28. As MixNet classes do not necessarily take the form of modules, one might expect our approach to yield a modularity index that is not "optimal". Nevertheless, the two class definitions are complementary, and both are needed in order to give a global overview of a network: the *community* partition to detect dense node connectivity, and the MixNet partition to analyze nodes with similar connectivity profiles. However, as mentioned by Adamic and Glance (2005), the division between liberal and conservative blogs is "unmistakable", this is why it may be more interesting to uncover the structure of the two communities rather than detecting them.

Interpreting MixNet results. As mentioned by Adamic and Glance (2005), the political websphere is partitioned according to political ori-

	Conservative	Independent	Liberal
cluster 1	734	135	238
cluster 2	290	26	8
cluster 3	2	7	430

Table 2.10 – *Contingency table comparing the political partition and MixNet partition.*

entations. The optimal number of groups found by MixNet is $Q = 21$ (see the Integrated Classification Likelihood Criterion of Figure 2.13).

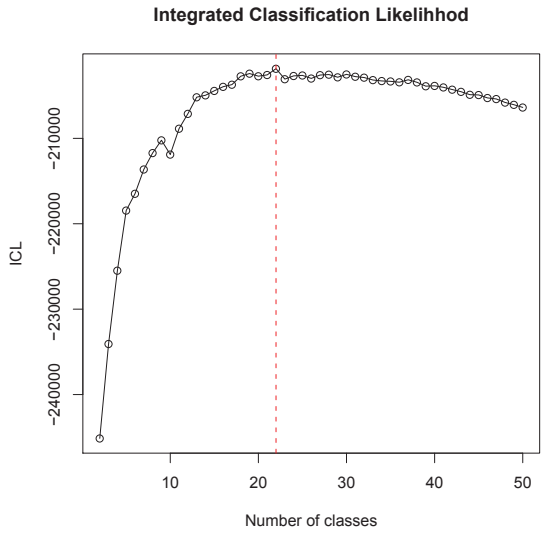


Figure 2.13 – *Integrated Classification Likelihood Criterion for the 2008 U.S. Presidential WebSphere.*

Interestingly when looking at Figure 2.14 that presents MixNet results, one striking structure is that political communities are linked only via main US online portals (C17, made of `nytimes.com`, `washingtonpost.com`, `cnn.com`, `msn.com`). It means that political blogs don't directly cite political oponents, which reinforces the political cyberbalkanization trend that was already observed in 2004. This "mass-media" cluster can be thought as a group of central hubs that make opponent websites communicate. Interestingly, the connections seem to be stronger towards the liberal part of the weblogs (Table A.8).

Then the question is to determine what are the structural characteristics of the liberal and conservative weblogs organization (note that independent sites do not seem to be structured on their own). MixNet reveals three substructures in the liberal part of the network. There is a set of blogs (clusters C7, C8, C12, C13 and C20) that show very strong intra *and* inter group connectivities which nearly forms a clique (Table A.8). Cluster C20 for instance is made of 3 weblogs which appear to have a determinant role in the structuration of the liberal community: `reachm.com`, `mahablog.com`, `juancole.com`. Then this set is connected to two other clusters which do not cite each other: clusters C4 and C6 only communicate via clusters C13 and C20, with very strong connections ($\hat{\pi}_{20,6} = 99\%$ for instance). The last substructure is made of cluster C11, which shows

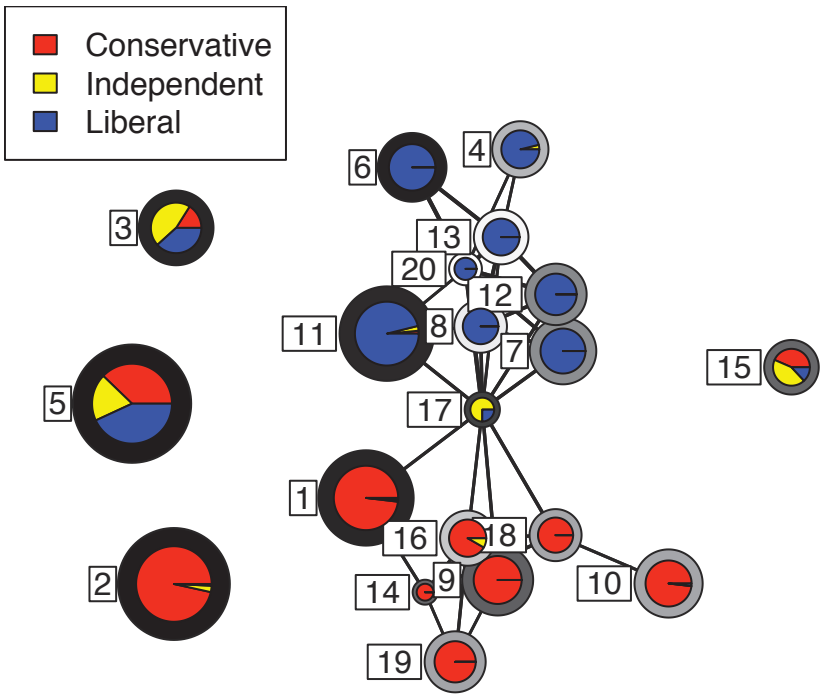


Figure 2.14 – *Network summary of US political websites. Each vertex represents a cluster. Each pie chart gives the proportions of liberal, conservative and independent tagged websites in the cluster. The outer ring color of the vertices is proportional to the intensity of the intra-connectivity: the darker, the weaker. Edges are represented when the inter-connectivity is among the 20% of the largest among all connectivity values.*

intermediate connections with liberal blogs. Actually, this sub-division is also present in the conservative part of the network. Indeed, clusters C1 and C2 are very lightly connected with the conservative blogs, whereas clusters C3, C14, C16, C18, C19 constitute the core of the conservative web-sphere, with very famous websites like `foxnews.com` (C14). The difference lies in the intensity of connection, which is lower for the conservatives.

Overall MixNet emphasizes the basic structures of the political web-sphere. Both communities are characterized by a small set of sites which use the Internet in a very professional and efficient way, with a lot of cross-linking. This results in a core structure to which other sites are linked, these other sites being less efficient in the citations to other websites. This could be explained either by a tendency to ignore other elements in the debate, or by a use of the Internet which is less efficient. Interestingly, this structure is very similar between conservatives and liberals, with the liberal core being more tight. This interpretation is reinforced by the different betweenness centralities of MixNet classes. Betweenness is based on the number of shortest geodesic paths that pass through a vertex. Figure 2.15 shows that MixNet betweenness is higher for MixNet core classes on average in both policital structures, whereas the betweenness patterns of the liberals and conservatives look very similar. Deeper sociological conclusions could be drawn from MixNet results, but would be beyond the scope of the present chapter.

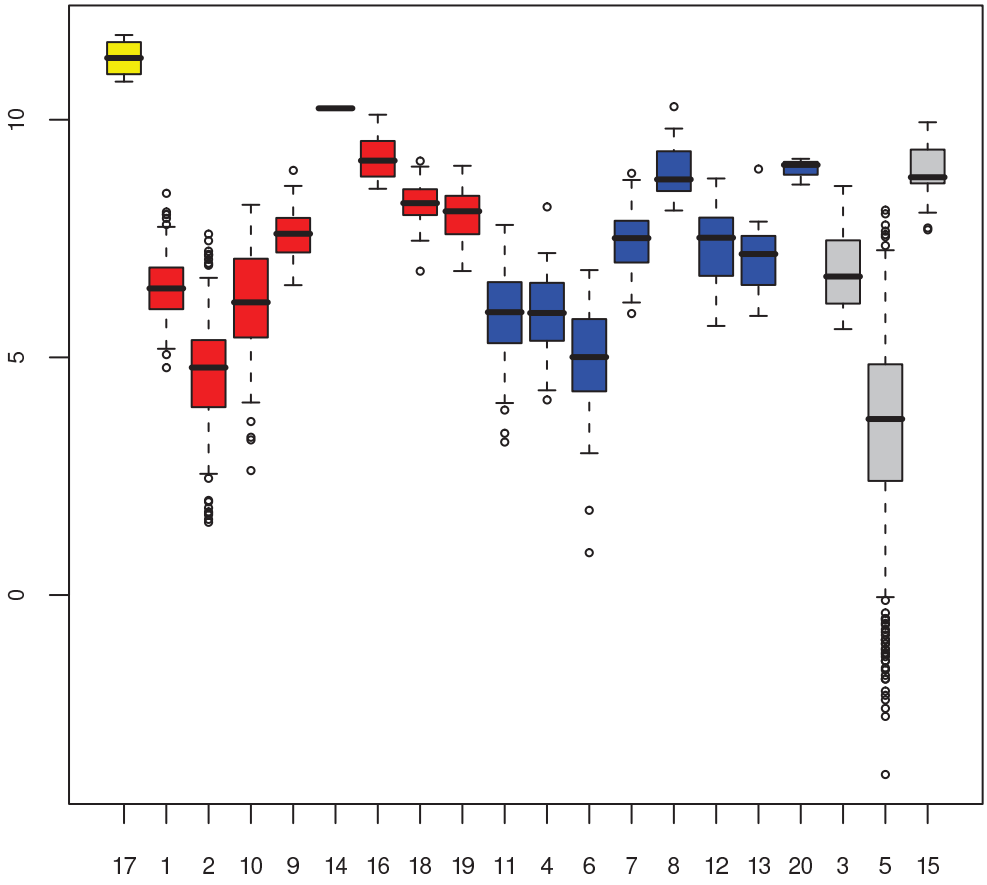


Figure 2.15 – *Boxplot of MixNet classes betweenness (in log).*

Chapter Conclusion

In this chapter we propose an online version of estimation algorithms for random graphs which are based on mixture of distributions. These strategies allow the estimation of model parameters within a reasonable computation time for data sets which can be made up of thousands of nodes. These methods constitute a trade-off between the potential amount of data to process and the quality of the estimations: even if online methods are not as precise as batch methods for estimation, they may represent a solution when the size of the network is too large for any existing estimation strategy. Furthermore, our simulation study shows that the quality of the remaining partition is good when using online methods. In the network of 2008 US political websites we could uncover the structure that makes the political websphere. This structure is very different from classical modules or "communities", which highlights the need for efficient computational strategies to perform model-based clustering on large graphs. The online framework is very flexible, and could be applied to other models such as the block model and the mixed membership model, as the online framework can be adapted to Bayesian algorithms (Opper 1999).

Model based graph clustering using both graph structure and vertex features

3

Contents

4.1 Introduction . . . 103
4.2 The World Wide Web . . . 104
4.2.1 The advent of the Web . . . 104
4.2.2 The structure of the Web . . . 106
4.3 *Constellations* Resources Building . . . 109
4.3.1 Introduction to Web Search Engines . . . 110
4.3.2 Crawling and Indexing the Web . . . 111
4.3.3 Ranking the Web . . . 112
4.3.4 Reverse Web graph using a MapReduce-like framework . 114
4.3.5 BigGraph . . . 116
4.3.6 Graph simplification . . . 118
4.4 *Constellations* Back-end . . . 119
4.4.1 Handled Queries . . . 120
4.4.2 Graph Search . . . 122
4.4.3 Layout Algorithm . . . 123
4.4.4 Nodes attributes . . . 125
4.4.5 *XML* response of the back-end . . . 126
4.5 *Constellations* Front-end . . . 127
4.6 Services derived from *Constellations* . . . 129
Conclusion . . . 130

Large data sets with interactions between objects are common to numerous scientific fields including the social sciences and biology, as well as being a feature of specific phenomena such as the Internet. The interactions naturally define a graph, and a common way of exploring

and summarizing such data sets is graph clustering. Most techniques for clustering graph vertices use only the topology of connections, while ignoring information about the features of vertices. In this chapter we provide clustering algorithms that harness both types of data, based on statistical models with a latent structure characterizing each vertex both by its connectivity and by a vector of features (or an additional information). We perform simulations to compare our algorithms with existing approaches and also evaluate our methods using real data sets based on hypertext documents. We find that our algorithms successfully exploit whatever information is found both in the connectivity pattern and in the features.

The present chapter is collaborative work with Stevenn Volant.

3.1 Introduction

Classical data analysis has been developed for sets of objects with features, but when explicit relationships exist between objects, classical data analysis cannot take these relationships into account. Much recent research has, however, been concerned with analyzing graphs, for example when seeking relationships in the social sciences, studying gene interactions in biology, or analyzing hyperlinks in computer science, which has led to a deeper awareness of the nature of the interactions in these different networks (Schenker et al. 2005, Cook and Holder 2007). Many approaches to graph analysis have been proposed. As mentioned in Chapter 2, model-based approaches, *i.e.*, methods which rely on a statistical model of network edges and vertices, such as those first proposed by Erdös-Rényi , can often provide insights into the structure of networks, enabling deductions to be made regarding their internal properties.

An interesting alternative to using the basic Erdös-Rényi model (often ill-suited to real networks) is to consider a mixture of distributions (Frank and Harary 1982, Snijders and Nowicki 1997, Newman and Leicht 2007, Daudin et al. 2008) where it is assumed that nodes are spread over an unknown number of latent connectivity classes. Conditional on the hidden class label, edges are still independent and Bernoulli distributed, but their marginal distribution is a mixture of Bernoulli distributions with strong dependencies between the edges. Several names have been suggested for this model, and here we have chosen to use the term MixNet, which is closely related to the Block Clustering of Snijders and Nowicki (1997). Details about this model can be found in previous chapter (see Section 2.3).

In addition to the network data used in the methods mentioned above, vertex content will sometimes be available. A typical example is the most famous implementation of Ted Nelson Hypertext (see Figure 3.1) : the World Wide Web. This *information network* where information are related to each other in some fashion, can be described either in terms of the hyperlinks between web pages or by the words occurring in the web pages: each vertex represents a web page containing occurrences of certain words, and each directed edge represents a hyperlink.

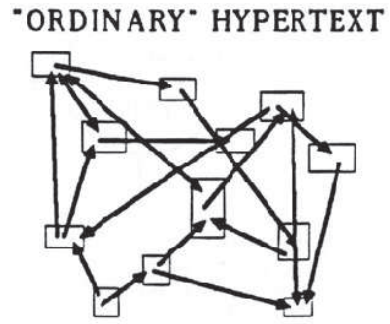


Figure 3.1 – *The hypertext diagram from Ted Nelson (Nelson 1987) which allows to develops complex and dynamic systems of linking and cross-referencing. Hypertext documents can either be static (prepared and stored in advance) or dynamic (continually changing in response to user input). The Web is an implementation of the hypertext paradigm.*

The additional information corresponding to the vertex features is

rarely used in network clustering, but can provide crucial information. Here we combine information from vertex content, traditionally used in classical data analysis, with information inherent in the graph structure, with the aim of clustering objects into coherent groups. This chapter proposes statistical models, CohsMix^i (for ***Co**variates on **h**idden **s**tructure using **Mix**ture models*) which consider dependent structures of the data and the relation with vertex features in order to capture a hidden structure. The upper index i will be used to distinguish the three different proposed models.

Considering spatial or relational data neighbourhoods is not an innovative approach in clustering. For instance, ***H**idden **M**arkov **R**andom **F**ields* (HMRF) are well adapted to handling spatial data and are widely used in image analysis. When the spatial network is not given, it is generally obtained using Delaunay triangulation (Ambroise et al. 1997).

Hoff (2003) proposed a new way of dealing with covariates. He suggested modeling the expected value of the relational ties with a logistic regression. The problem with this method is the dependency between the observations conditional on the regression parameters and the covariates. He therefore proposed incorporating random effect structures in a generalized linear model setting. The distribution of dependencies among the random effects determines the dependencies among the edges.

There are also approaches based on non-statistical frameworks. In particular, there is clearly a strong similarity between multiple view and graph models with covariates. Multiple view learning algorithms (Ruping and Scheffer 2005) consider instances which have multiple representations and make use of these views simultaneously so as to obtain a consensus partition.

Outline of the chapter First, the chapter introduces the proposed CohsMix^i models, which are extensions of the previous MixNet model. Since the models consider a great number of dependencies, the proposed estimation schemes include a variational approach of the EM algorithm, which can deal with larger networks than the Bayesian framework is able to handle. We then introduce practical strategies for initializing and choosing the number of groups. In the third section extensive simulations illustrate the efficiency of the these algorithms, and real data sets dealing with hypertext documents are examined.

3.2 A Mixture of Networks with covariates

This section introduces network models which are able to deal with additional information corresponding to the vertex features (CohsMix^3 model) or to extra relationships between vertices (CohsMix^1 and CohsMix^2 models). First, we explain the different forms that vertex features (or covariates) can take. We then take an interest in representing two kind of dependencies between the additional information and connectivity. The first approach (CohsMix^1 and CohsMix^3 models) assumes that the informations and the edges conditional on the node classes are mutually independent and both can be explained by the class. Thus, this model assumes that

edges have no influence on the values of covariates . Then, we propose a second approach (CohsMix[2]) which deals with dependencies between covariates and edges conditional on the node classes. In the web context, the latter approach considers that the similarity between the occurring words of two web documents, and somehow between their topics, depends on the existence of a hyperlink between them whereas the first approach considers that a given class contains documents that are similar not only in the words they contain, but also in their connectivity patterns with documents inside and outside the class. Although this assumption does not explicitly model the idea that authors tend to link similar topics (words that occur) thus creating a thematic locality (Davison 2000), it nevertheless allows clusters with local themes to be detected. Its simplicity makes it a robust model well suited to a real-network like the Web.

3.2.1 Connectivity Model

Here, we briefly remind MixNet model (see Section 2.3) for the Bernoulli case. Let us define a random graph G, where $\mathcal{V}$ denotes the set of vertices. Our models assumes that $\mathcal{V}$ is partitioned into Q hidden classes. Let us denote by Z_{iq} the indicator variable such that $\{Z_{iq} = 1\}$ if node i belongs to class q. $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_n)$ is the vector of random independent indicator variables such that

$$\mathbf{Z}_i \sim \mathcal{M}(1, \boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_Q\}), \tag{3.1}$$

with $\boldsymbol{\alpha}$ the vector of class proportions. Edges are Bernoulli random variables

$$X_{ij}|Z_{iq}Z_{jl} = 1 \sim \mathcal{B}(\pi_{ql}), \tag{3.2}$$

conditionally independent, given the node classes

$$P(\mathbf{X}|\mathbf{Z}) = \prod_{ij}\prod_{q,l} P(X_{ij}|Z_{iq}Z_{jl} = 1)^{Z_{iq}Z_{jl}}.$$

In this chapter we consider that graphs are directed. We also assume that there are self-loops, *i.e.* a node can be connected to itself (X_{ii} = 1). Nevertheless, the methods can easily be modified to encompass undirected graphs without self-loops.

3.2.2 Remarks about Vertex Features Model

We shall consider n objects described both by their connections and p features. The data can consequently be represented in different forms. One might, for example, wish to characterize each object using a two-part vector, where the first part contains the feature of the object $\mathbf{Y}_i$ and the second part contains a binary vector representing the connection to all $n-1$ other objects $\mathbf{X}_{i,\bullet}$. This form of data representation will be handled by CohsMix[3] (Section 3.2.4). Continuing our example of the World Wide Web, Web pages can be viewed either as a vector of word-occurrences with hyperlinks, or as two matrices, one based on the adjacency matrix describing the topology of the graph generated by the hyperlinks and the other by the features matrix generated by the word-occurrences in each web page.

Another way to represent the data consists in using two squared matrices. One for describing the structure of the graph and the other for describing a valued relation between each pair of nodes (or object features). Concretely, we consider n objects described both by their graph topology and another type of relation. In order to illustrate this data representation, we can cite a previous example of the Chapter 2, where a traffic information of users going from page i to a page j is added to the hypertext connectivity. In this example, it is obvious that link of graph structure have significative influence on covariate's value. The next section explains this kind of representation (CohsMix[1] and CohsMix[2] models) with different dependencies between edges and covariates, then we will describe data with connections and p features.

3.2.3 Models with additional Matrix Information

Here, we consider the second alternative of vertex features model where additional information on edges are added to the graph structure of MixNet model. This additional information are stored in squared matrix $\mathbf{Y}$.

The proposed mixture models are derived from different decomposition of the joint distribution of the graph adjacency matrix $\mathbf{X}$, the information matrix $\mathbf{Y}$ and the hidden structure $\mathbf{Z}$ (Figure 3.2).

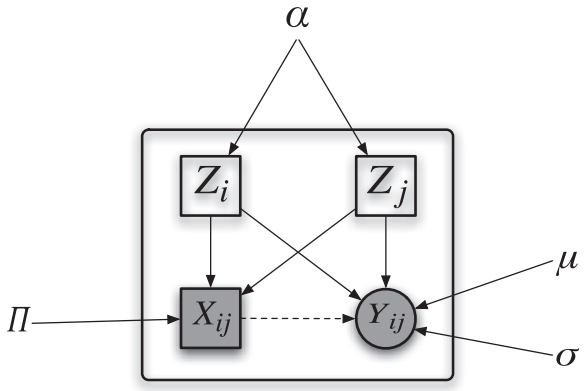


Figure 3.2 – *Graphical representation of the two CoshMix Models. The first one does not take into account the dotted line and assumes independence between* $\mathbf{X}$ *and* $\mathbf{Y}$ *conditional on the hidden structure* $\mathbf{Z}$*. The second model introduces an additional dependence between graph structure and vertex features. The squares represent discrete random variables and circles continuous random variables.*

CoshMix[1]: Assuming independence of X and Y conditional on Z.

Considering the independence between graph structure and additional information, the log-likelihood of the complete data set can be written as:

$$\Pr(\mathbf{X},\mathbf{Y},\mathbf{Z}) = \Pr(\mathbf{Z})\Pr(\mathbf{X},\mathbf{Y}|\mathbf{Z}) \quad = \quad \Pr(\mathbf{Z})\Pr(\mathbf{Y}|\mathbf{Z})\Pr(\mathbf{X}|\mathbf{Z}).$$

We assume that Y_{ij}, the additional information between node i and j, are normally distributed

$$Y_{ij}|Z_{iq}Z_{jl} = 1 \sim \mathcal{N}(\mu_{ql}, \sigma^2) \tag{3.3}$$

and conditionally independent, given the node classes

$$\Pr(\mathbf{Y}|\mathbf{Z}) = \prod_{ij}\prod_{q,l} \Pr(Y_{ij}|Z_{iq}Z_{jl} = 1)^{Z_{iq}Z_{jl}}.$$

Again, for the sake of simplicity we can consider an affiliation model, where the parameter μ_{ql} has two possible values:

$$\begin{cases} \mu_{qq} = \mu_1, & \forall q \in [1, Q], \\ \mu_{ql} = \mu_2, & \forall q, l \in [1, Q] q \neq l. \end{cases}$$

In the following, this model will be called CoshMix[1]. The independence between additional information and graph structure is a strong assumption. Thus, we proposed an alternative model which takes into account a dependency between the information ($\mathbf{Y}$) and edges ($\mathbf{X}$) (see Figure 3.2).

CoshMix[2]: Introducing dependence between X and Y conditional on Z.

It is reasonable to assume that the information $Y_{i,j}$ between vertices i and j depends on the node classes but also on the existence of an edge X_{ij} between those vertices. Thus we propose to model the distribution of informations $\mathbf{Y}$ conditional on both the graph and the latent structure. The following decomposition of the joint distribution is considered:

$$\Pr(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \Pr(\mathbf{Z}) \Pr(\mathbf{X}|\mathbf{Z}) \Pr(\mathbf{Y}|\mathbf{X}, \mathbf{Z}).$$

We preserve the assumption of conditional independence given the node classes

$$\Pr(\mathbf{Y}|\mathbf{X}, \mathbf{Z}) = \prod_{ij}\prod_{q,l} \Pr(Y_{ij}|X_{ij}, Z_{iq}Z_{jl} = 1)^{Z_{iq}Z_{jl}}$$

where the conditional distribution of the Y_{ij}'s are Gaussian. The distribution of $\mathbf{Y}$ will differ depending on existing links of the graph structure between each pair of nodes. In order to write this distribution, we provide two new notations: μ_{ql} and $\tilde{\mu}_{ql}$ respectively the means of two Gaussian distributions which correspond to the presence or absence of an edge. We also assume that variance σ do not vary regardless of the connection between edges. Then, we can write the distribution of $\mathbf{Y}$ as follows:

$$\begin{cases} Y_{ij}|X_{ij} = 1, Z_{iq}Z_{jl} = 1 \sim \mathcal{N}(\mu_{ql}, \sigma^2), \\ Y_{ij}|X_{ij} = 0, Z_{iq}Z_{jl} = 1 \sim \mathcal{N}(\tilde{\mu}_{ql}, \sigma^2). \end{cases}$$

Therefore, the conditional distribution corresponding to the added in-

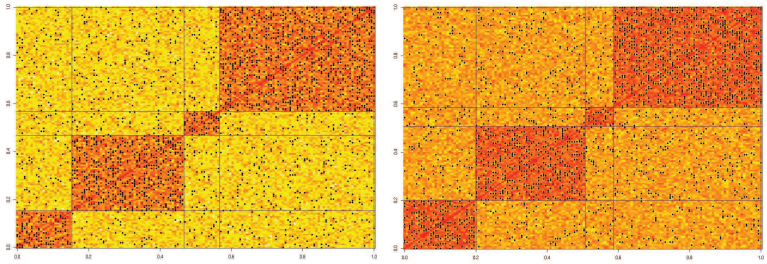


Figure 3.3 – *Simulation of two* 150 *nodes graphs with* 4 *classes according to the parameters 3.1. Edges are represented with black dots and information vary from white (weak) to red (strong). The left figure illustrates the CohsMix*^{1} *model and the right the CohsMix*^{2} *model where in presence of edges we can notice stronger information.*

formation $\mathbf{Y}$ can be written as follows:

$$\begin{aligned}
\log(\Pr(\mathbf{Y}|\mathbf{X},\mathbf{Z})) &= \sum_{i,j}\sum_{q,l} Z_{iq}Z_{jl}X_{ij}\log(\Pr(Y_{ij}|X_{ij})) \\
&\quad + \sum_{i,j}\sum_{q,l} Z_{iq}Z_{jl}(1-X_{ij})\log(\Pr(Y_{ij}|X_{ij})) \\
&= \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l} Z_{iq}Z_{jl}X_{ij}\left(\frac{(Y_{ij}-\tilde{\mu}_{ql})^2}{2\sigma^2} - \frac{(Y_{ij}-\mu_{ql})^2}{2\sigma^2}\right) \\
&\quad - \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l} Z_{iq}Z_{jl}\frac{(Y_{ij}-\tilde{\mu}_{ql})^2}{2\sigma^2} + \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l} Z_{iq}Z_{jl}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right).
\end{aligned}$$

Once again, for the sake of simplicity we can consider an affiliation model, where μ_{ql} and $\tilde{\mu}_{ql}$ are defined by:

$$\begin{cases} \mu_{qq} = \mu_1, & \forall q \in [1,Q], \\ \mu_{ql} = \mu_2, & \forall q,l \in [1,Q], q \neq l, \end{cases} \quad \text{and} \quad \begin{cases} \tilde{\mu}_{qq} = \tilde{\mu}_1, & \forall q \in [1,Q], \\ \tilde{\mu}_{ql} = \tilde{\mu}_2, & \forall q,l \in [1,Q], q \neq l. \end{cases}$$

In the following, this model will be called CoshMix^2. Both models, CoshMix^1 and CoshMix^2 are generative and can thus be used for networks simulation. For instance, simulations of arbitrary CohsMix^{1,2} parameters (see Table 3.1) are represented in Figure 3.3.

Model	Parameter	Value
CohsMix^{1,2}	α	(0.2,0.3,0.1,0.4)
	π_{qq}	0.2
	π_{ql}	0.05
	σ^2	1
	μ_{qq}	2
	μ_{ql}	4
CohsMix^2	$\tilde{\mu}_{qq}$	5
	$\tilde{\mu}_{ql}$	7

Table 3.1 – *Arbitrary parameters of two CohsMix*^{1,2} *models*

In the next section, we describe the data model when each object (or node) is characterized by a vector of p features and its binary connections to all $n-1$ other objects.

3.2.4 CoshMix^3: Model with additional Vertex features

Hereafter, we consider that the n objects are described both by their connections and p features. The data can consequently be represented in different forms. We choose to characterize each object using a two-part vector, where the first part contains the feature of the object $\mathbf{Y}_i$ and the second part contains a binary vector representing the connection to all $n-1$ other objects $\mathbf{X}_{i,\bullet}$.

Then, we consider that the p dimensional feature vector associated to object i is defined by:

$$\mathbf{Y}_i = \begin{pmatrix} Y_i^{(1)} \\ Y_i^{(2)} \\ \vdots \\ Y_i^{(p)} \end{pmatrix}$$

We assume that the feature vectors $\mathbf{Y}_i$ are multivariate normally-distributed

$$\mathbf{Y}_i | Z_{iq} = \mathbf{1} \sim \mathcal{N}(\mu_q, \Sigma_q) \tag{3.4}$$

where

$$\mu_q = \begin{pmatrix} \mu_q^{(1)} \\ \mu_q^{(2)} \\ \vdots \\ \mu_q^{(p)} \end{pmatrix} \text{ and } \mathbf{\Sigma_q} = \sigma I \text{ the covariance matrix is proportional to}$$

the identity.

Notice that this assumption is not systematically supported by the data. As the class structure is not available beforehand, assuming that the data is normally distributed within each class, is difficult to check *a priori*. It thus would be a reasonable practice to check *a posteriori*.

The random feature vectors $\mathbf{Y}_i$ are conditionally independent, given the node classes

$$\Pr(\mathbf{Y}|\mathbf{Z}) = \prod_i \prod_q \Pr(\mathbf{Y}_i | Z_{iq})^{Z_{iq}}.$$

The conditional distribution corresponding to covariate can be written as follows:

$$\begin{aligned} \log \Pr(\mathbf{Y}|\mathbf{Z}) &= \sum_i \sum_q Z_{iq} \log \Pr(\mathbf{Y}_i | Z_{iq}) \\ &= \sum_i \sum_q Z_{iq} \left[\left(\log \frac{1}{2\pi^{\frac{n}{2}} det(\Sigma)^{\frac{1}{2}}} \right) - \frac{1}{2} (\mathbf{Y}_i - \mu_q)^T \sigma^{-1} (\mathbf{Y}_i - \mu_q) \right]. \end{aligned}$$

In this model description, called in the following CoshMix^3, the proposed mixture model assumes an independence of $\mathbf{X}$ and $\mathbf{Y}$ conditional on $\mathbf{Z}$. Therefore, given this independence between edges and covariates, the complete log-likelihood can be written as (Figure 3.4):

$$\Pr(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \Pr(\mathbf{Z}) \Pr(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = \Pr(\mathbf{Z}) \Pr(\mathbf{Y}|\mathbf{Z}) \Pr(\mathbf{X}|\mathbf{Z}).$$

However, it is important to note that as in the previous model (see Section 3.2.3), we could introduce a dependency between the vertex features $\mathbf{Y}$ and the graph structure $\mathbf{X}$ conditional on the node classes $\mathbf{Z}$.

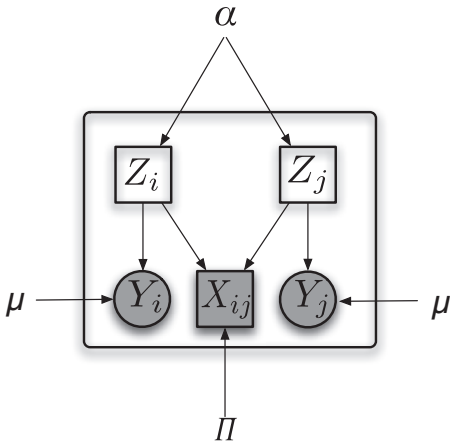


Figure 3.4 – *Graphical representation of the CohsMix[3] Model. The squares represent discrete random variables and circles continuous random variables.*

The following section proposes an estimation scheme for the CohsMix[1,2,3] models.

3.3 Variational EM algorithm for CohsMix[1,2,3]

In the classical EM framework developed by Dempster et al. (1977), where $\mathbf{X}$ and $\mathbf{Y}$ are the available data, inferring the unknown parameters $\boldsymbol{\Theta}$ spread over a latent structure $\mathbf{Z}$ involves the following conditional expectation:

$$\begin{aligned} Q\left(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(m)}\right) &= \mathbb{E}\left\{\log \mathcal{L}_c(\mathbf{X},\mathbf{Y},\mathbf{Z};\boldsymbol{\Theta})|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)}\right\} \\ &= \sum_{\mathbf{Z}\in\mathcal{Z}} \Pr\left(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)}\right) \log \mathcal{L}_c(\mathbf{X},\mathbf{Y},\mathbf{Z};\boldsymbol{\Theta}) \quad (3.5) \end{aligned}$$

where

$$\boldsymbol{\Theta}^{(\mathbf{m+1})} = \underset{\boldsymbol{\Theta}}{\text{Argmax}} Q(\boldsymbol{\Theta},\boldsymbol{\Theta}^{(m)}).$$

The usual EM strategy would be to alternate an E-step computing the conditional expectation (3.5) with an M-step maximizing this quantity over the parameter of interest $\boldsymbol{\Theta}$. Unfortunately, no closed form of $Q\left(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(m)}\right)$ can be formulated in the present case. The technical difficulty lies in the complex dependency structure of the model. Indeed, $\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta})$ cannot be factorized, as argued in Daudin et al. (2008). This makes the direct calculation of $Q\left(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(m)}\right)$ impossible. To tackle this problem we use a variational approach (see, e.g., Jordan et al. 1999, for elementary results on variational methods). In this framework, the conditional distribution of the latent variables $\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)})$ is approximated by a more convenient distribution denoted by $\mathcal{R}(\mathbf{Z})$, which is chosen carefully in order to be tractable. Hence, our EM-like algorithm includes the following approximation of the conditional expectation (3.5)

$$\mathbb{E}_{\mathcal{R}}\left\{\log \mathcal{L}_c(\mathbf{X},\mathbf{Y},\mathbf{Z};\boldsymbol{\Theta})\right\} = \sum_{\mathbf{Z}\in\mathcal{Z}} \mathcal{R}(\mathbf{Z}) \log \mathcal{L}_c(\mathbf{X},\mathbf{Y},\mathbf{Z};\boldsymbol{\Theta}). \quad (3.6)$$

In the following section we develop a variational argument in order to choose an approximation $\mathcal{R}(\mathbf{Z})$ of $\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)})$. This enables us to compute the conditional expectation (3.6) and proceed to the maximization step.

3.3.1 Estimation of the latent structure (E-step)

In this part $\boldsymbol{\Theta}$ is assumed to be known, and we are looking for an approximate distribution $\mathcal{R}(\cdot)$ of the latent variables. The variational approach consists in maximizing a lower bound $\mathcal{J}$ of the log-likelihood $\log\Pr(\mathbf{X},\mathbf{Y};\boldsymbol{\Theta})$, defined as follows:

$$\mathcal{J}(\boldsymbol{\Theta}) = \log\Pr(\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}) - \mathrm{D}_{KL}\left\{\mathcal{R}(\mathbf{Z})\|\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)})\right\} \tag{3.7}$$

where D_{KL} is the Küllback-Leibler divergence. This measures the difference between the probability distribution $\Pr(\cdot|\boldsymbol{\Theta})$ in the underlying model and its approximation $\mathcal{R}(\cdot)$. An intuitively straightforward choice for $\mathcal{R}(\cdot)$ is a completely factorized distribution (see Mariadassou and Robin 2007, Zanghi et al. 2008)

$$\mathcal{R}(\mathbf{Z}) = \prod_{i\in\mathcal{P}} h_{\boldsymbol{\tau}_i}(\mathbf{Z}_i), \tag{3.8}$$

where $h_{\boldsymbol{\tau}_i}$ is the density of the multinomial probability distribution $\mathcal{M}(1;\boldsymbol{\tau}_i)$, and $\boldsymbol{\tau}_i = (\tau_{i1},\ldots,\tau_{iQ})$ is a random vector containing the variational parameters to optimize. The complete set of parameters $\boldsymbol{\tau} = \left\{\tau_{iq}\right\}_{i\in\mathcal{P},q\in\mathcal{Q}}$ is what we are seeking to obtain via the variational inference. In the case in hand the variational approach intuitively operates as follows: each τ_{iq} can be seen as an approximation of the probability that vertex i belongs to cluster q, conditional on the data, that is, τ_{iq} estimates $\Pr(Z_{iq}=1|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta})$, under the constraint $\sum_q \tau_{iq} = 1$. In the ideal case where $\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta})$ can be factorized as $\prod_i \Pr(\mathbf{Z}_i|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta})$ and the parameters τ_{iq} are chosen as $\tau_{iq} = \Pr(Z_{iq}=1|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta})$, the Küllback-Leibler divergence is null and the bound $\mathcal{J}$ reaches the log-likelihood. The Küllback-Leibler distance can be written as

$$\mathrm{D}_{KL}\left\{\mathcal{R}(\mathbf{Z})\|\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)})\right\} = \sum_{Z}\mathcal{R}(\mathbf{Z})\log\left(\frac{\mathcal{R}(\mathbf{Z})}{\Pr(\mathbf{Z}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}^{(m)})}\right)$$

After simplification, the lower bound $\mathcal{J}$ to be maximized in order to estimate $\boldsymbol{\tau}$ can be expressed as

$$\mathcal{J}_{\boldsymbol{\tau}} = \mathbb{E}_{\mathcal{R}(Z)}\left\{\mathcal{J}(\boldsymbol{\Theta})\right\} = \mathbb{E}_{\mathcal{R}(Z)}\{\log(\Pr(\mathbf{X},\mathbf{Y},\mathbf{Z}))|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}\} - \sum_{Z}\mathcal{R}(\mathbf{Z})\log(\mathcal{R}(\mathbf{Z})).$$

The detailed expressions for the different models are given in Appendix A.6. The optimal approximate distribution R and the optimal parameters $\boldsymbol{\Theta}$ are then derived by direct maximization of $\mathcal{J}_{\boldsymbol{\tau}}$.

3.3.2 Estimation of the parameters (M-step)

Concerning the parameters estimation of class proportion and connectivity, it is noticeable that our three CoshMix[1,2,3] models have common optimal formulations. Therefore, only estimators of the normal distribution require an adjustment.

Optimal parameters for all CohsMix^{1,2,3} models The maximization of $\mathcal{J}_\tau$ provides the following proposition which is valid for the *CoshMix*^{1,2,3} models.

Proposition 3.1 *The optimal parameters α_q, π_{ql} i.e. the parameters maximizing $\mathcal{J}_\tau$ satisfy the following relations:*

$$\begin{aligned} \hat{\alpha}_q &= \frac{1}{n}\sum_{i=1}^{n}\tau_{iq}, \\ \hat{\pi}_{ql} &= \frac{\sum_{i\neq j}\tau_{iq}\tau_{jl}X_{ij}}{\sum_{i\neq j}\tau_{iq}\tau_{jl}} \end{aligned} \tag{3.9}$$

Proof. In Appendix A.7.1. □

Optimal parameters for *CohsMix*^{1}

Proposition 3.2 *The optimal parameters μ_{ql} and σ, i.e. the parameters maximizing $\mathcal{J}_\tau$ satisfy the following relations:*

$$\hat{\mu}_{ql} = \frac{\sum_{i\neq j}\tau_{iq}\tau_{jl}Y_{ij}}{\sum_{i\neq j}\tau_{iq}\tau_{jl}} \quad \textit{and} \quad \hat{\sigma} \quad = \frac{\sum_{i\neq j}\sum_{q,l}\tau_{iq}\tau_{jl}(Y_{ij}-\hat{\mu}_{ql})^2}{\sum_{i\neq j}\sum_{q,l}\tau_{iq}\tau_{jl}}. \tag{3.10}$$

Proof. In Appendix A.7.2. □

Proposition 3.3 *Let all the parameters $\hat{\pi}_{ql}, \hat{\alpha}_q, \hat{\mu}_q$ and $\hat{\sigma}$ be known. The following fixed-point relationship holds for the optimal variational parameters $\widehat{\boldsymbol{\tau}} = \arg\max_\tau \mathcal{J}_\tau$.*

$$\begin{aligned} \hat{\tau}_{iq}^{(m+1)} \quad \propto \quad & \alpha_q \prod_{j\neq i}\prod_{l}\left[\hat{\pi}_{ql}^{x_{ij}}(1-\hat{\pi}_{ql})^{1-x_{ij}}\right]^{\tau_{jl}^{(m)}} \\ & \times \left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{1}{2\sigma^2}\left[-(Y_{ij}-\hat{\mu}_{ql})^2\right]\right)\right]^{\tau_{jl}^{(m)}} \end{aligned}$$

Proof. Similar to A.7.3. □

Optimal parameters for *CohsMix*^{2}

Proposition 3.4 *The optimal parameters μ_{ql}, $\tilde{\mu}_{ql}$ and σ, i.e. the parameters maximizing $\mathcal{J}_\tau$ satisfy the following relations:*

$$\begin{aligned} \hat{\mu}_{ql} &= \frac{\sum_{i\neq j}\tau_{iq}\tau_{jl}x_{ij}y_{ij}}{\sum_{i\neq j}\tau_{iq}\tau_{jl}x_{ij}} \quad , \quad \hat{\tilde{\mu}}_{ql} = \frac{\sum_{i\neq j}\tau_{iq}\tau_{jl}(1-x_{ij})y_{ij}}{\sum_{i\neq j}\tau_{iq}\tau_{jl}(1-x_{ij})}, \\ \hat{\sigma} &= \frac{\sum_{i\neq j}\sum_{q,l}\tau_{iq}\tau_{jl}\left[x_{ij}\left((y_{ij}-\hat{\mu}_{ql})^2-(y_{ij}-\hat{\tilde{\mu}}_{ql})^2\right)+(yij-\hat{\tilde{\mu}}_{ql})^2\right]}{\sum_{i\neq j}\sum_{q,l}\tau_{iq}\tau_{jl}} \end{aligned} \tag{3.11}$$

Proof. In Appendix A.7.3. □

Proposition 3.5 *Let all the parameters $\hat{\pi}_{ql}, \hat{\alpha}_q, \hat{\mu}_{ql}$, $\hat{\tilde{\mu}}_{ql}$ and $\hat{\sigma}$ be known. The following fixed-point relationship holds for the optimal variational parameters $\widehat{\boldsymbol{\tau}} = \arg\max_{\boldsymbol{\tau}} \mathcal{J}_{\boldsymbol{\tau}}$.*

$$\begin{aligned}\hat{\tau}_{iq}^{(m+1)} \quad \propto \quad & \alpha_q \prod_{j \neq i} \prod_{l} \left[\hat{\pi}_{ql}^{x_{ij}} (1 - \hat{\pi}_{ql})^{1 - x_{ij}} \right]^{\tau_{jl}^{(m)}} \\ & \times \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(\frac{1}{2\sigma^2} \left[x_{ij} \left(-(y_{ij} - \hat{\mu}_{ql})^2 + (y_{ij} - \hat{\tilde{\mu}}_{ql})^2 \right) - (y_{ij} - \hat{\tilde{\mu}}_{ql})^2 \right] \right]^{\tau_{jl}^{(m)}}\end{aligned}$$

Proof. In Appendix A.7.3. □

Optimal parameters for *CohsMix*[3]

Proposition 3.6 *The optimal parameters μ_q and σ, i.e. the parameters maximizing $\mathcal{J}_{\boldsymbol{\tau}}$ satisfy the following relations:*

$$\hat{\mu}_q = \frac{\sum_i \tau_{iq} \mathbf{Y}_i}{\sum_i \tau_{iq}} \quad \text{and} \quad \hat{\sigma} \quad = \frac{\sum_i \sum_q \tau_{iq} (\mathbf{Y}_i - \hat{\mu}_q)^T (\mathbf{Y}_i - \hat{\mu}_q)}{\sum_i \sum_q \tau_{iq}}. \tag{3.12}$$

Proof. The proof of this proposition is obvious. Given the expression of the normal multivariate distribution in the equation of $\mathcal{J}_{\boldsymbol{\tau}}$, we simply use the known maximum likelihood estimators. □

Proposition 3.7 *Let all the parameters $\hat{\pi}_{ql}, \hat{\alpha}_q, \hat{\mu}_q$ and $\hat{\sigma}$ be known. The following fixed-point relationship holds for the optimal variational parameters $\widehat{\boldsymbol{\tau}} = \arg\max_{\boldsymbol{\tau}} \mathcal{J}_{\boldsymbol{\tau}}$.*

$$\hat{\tau}_{iq}^{(m+1)} \propto \hat{\alpha}_q \prod_{j \neq i} \prod_{l} \left[\hat{\pi}_{ql}^{x_{ij}} (1 - \hat{\pi}_{ql})^{1 - x_{ij}} \right]^{\tau_{jl}^{(m)}} \prod_{k=1}^{p} \left[\exp\left(\frac{1}{2\hat{\sigma}^2} \left(-(Y_i^{(k)} - \hat{\mu}_q^{(k)}) \right)^2 \right) \right]. \tag{3.13}$$

Proof. The proof is similar to Appendix A.7.3. □

For completeness, we summarize the variational EM algorithm for *CohsMix*[3] in the Algorithm 3. Of course, this algorithm can be applied using the correct estimators for *CohsMix*[1] and *CohsMix*[2].

3.3.3 Model selection: ICL algorithm

As the number of clusters is an unknown parameter of our statistical model, it is possible to use the Integrated Classification Likelihood (ICL) to choose the optimal number of classes (Biernacki et al. 2000). The ICL criterion is essentially derived from the ordinary ICL considering the complete log-likelihood instead of the log-likelihood. This optimal number is obtained by running our algorithm concurrently for models from 2 to Q classes and selecting the solution which maximizes the ICL criterion. In our situation where additional covariates are considered, the ICL criterion

Algorithm 3: Variational EM CohsMix^3 Algorithm

Data: Matrices of connectivities $\mathbf{X}$ and similarities $\mathbf{Y}$

/* Initialization of the parameters */

$\boldsymbol{\Theta}^{(0)} = \left(\alpha_1^{(0)}, ..., \alpha_Q^{(0)}, \pi_{11}^{(0)}, ..., \pi_{QQ}^{(0)}, \mu_1^{(0)}, ..., \mu_p^{(0)}, \sigma^{(0)}\right), m = 0$

while *not convergence* **do**

  /* Estimation step */

  /* Compute $\boldsymbol{\tau} = \left\{\tau_{iq}\right\}_{i \in \mathcal{P}, q \in \mathcal{Q}}$ the probabilities that vertex i belong to cluster q finding fix point of $g()$ */

  foreach $i \in \{1, ..., N\}$ **do**

    foreach $q \in \{1, ..., Q\}$ **do**

      $\tau_{iq}^{(m+1)} = g(\boldsymbol{\tau}^{(m)})$ (see Equation 3.13)

    /* normalize posterior probabilities */

    $scale = \sum_{q=1}^{Q} \tau_{iq}$

    $\tau_{iq} = \tau_{iq} \frac{1}{scale}, \forall q \in \{1, ..., Q\}$

  /* Maximization step */

  /* re-estimate the distribution parameters to maximize the likelihood of the data */

  Update parameters according to Equations 3.9 and 3.12

  foreach $q \in \{1, ..., Q\}$ **do**

    $\alpha_q^{(m+1)} = \text{Argmax}_{\alpha_q} \mathcal{J}_{\boldsymbol{\tau}}(\boldsymbol{\Theta})$

    $\mu_q^{(m+1)} = \text{Argmax}_{\mu_q} \mathcal{J}_{\boldsymbol{\tau}}(\boldsymbol{\Theta})$

    foreach $l \in \{1, ..., Q\}$ **do**

      $\pi_{ql}^{(m+1)} = \text{Argmax}_{\pi_{ql}} \mathcal{J}_{\boldsymbol{\tau}}(\boldsymbol{\Theta})$

  m = m +1

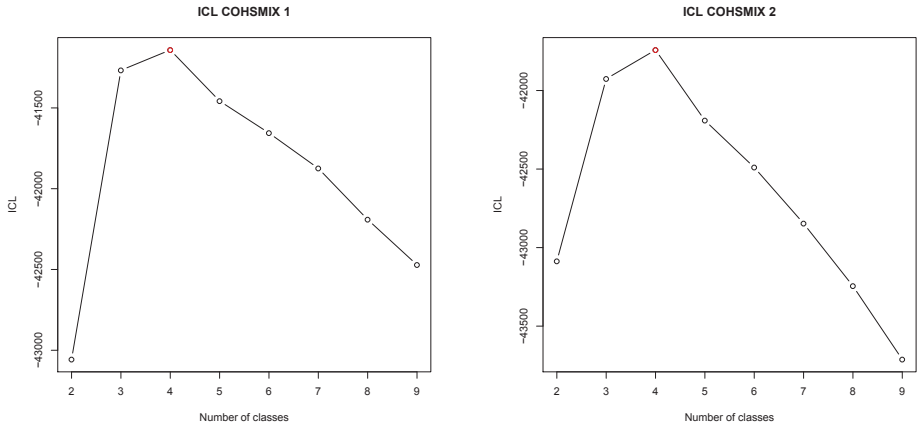
Result: Estimated parameters $\boldsymbol{\Theta}$ and posterior probabilities τ_{iq}

can be written as:

$$ICL(Q) = \max_{\boldsymbol{\Theta}} \log \mathcal{L}(\mathbf{X}, \mathbf{Y}, \mathbf{Z}; \boldsymbol{\Theta}, Q) - \underbrace{Q^2 \log(n(n-1))}_{\text{related to } \pi_{ql}} - \underbrace{(Q-1)\log(n)}_{\text{related to } \alpha_q}$$
$$- \underbrace{p(p-1)\log(n(n-1)) - pQ\log(n(n-1))}_{\text{related to } \mu_q \text{ and } \sigma}$$

This expression of the ICL criterion is based on the method described in Daudin et al. (2008).

As an example, Figure 3.5 illustrates the ICL criterion of two generated networks (See Figure 3.3) for $CohsMix^1$ and $CohsMix^2$ models. In these examples, the optimal BIC values correspond exactly to the real number of simulated classes.



(a) (b)

Figure 3.5 – *Integrated Classification Likelihood Criterion in function of the number of clusters for (a) $CohsMix^1$ and (b) $CohsMix^2$. The associated models are identicals to 3.3*

3.4 Experiments

In this section we report experiments to assess the performances and limitations of the proposed models in a clustering context. We consider both synthetic data generated with respect to the assumed random graph model and real data from the web. Synthetic graphs are useful for evaluating the quality of parameter estimation. In parallel, we also compare classification results with alternative clustering methods using a ground truth. The real data sets consist of hypertext documents retrieved from a web search queries in the *Exalead* search engine.

Notice that for all the experiments, to avoid initialization issues, algorithms are stated with multiple initialization points and the best results are selected based on theirs likelihoods. An R package we have called `CohsMix` is available upon request. Section A.8 add some details about this software.

3.4.1 Parameters estimation

CohsMix[1] and CohsMix[2] First, we have simulated 30 networks for the models proposed in Table 3.1 and run our CohsMix[1]/CohsMix[2] algorithms to estimate the model parameters. Table 3.2 shows that the estimation is very close to the true parameters. The adjusted rand Indexes 2.8.1 are equal to 1.

Model	Parameters	Real Values	Estimated Values	
			CohsMix[1]	*CohsMix*[2]
CohsMix[1,2]	π_{qq}	0.2	0.18	0.18
	π_{ql}	0.05	0.044	0.047
	σ^2	1	0.99	1.003
	μ_{qq}	2	2.04	2.03
	μ_{ql}	4	4.02	3.99
CohsMix[2]	$\tilde{\mu}_{qq}$	5		5.01
	$\tilde{\mu}_{ql}$	7		7.01

Table 3.2 – *Means of the estimated parameters for CohsMix[1,2] models computed over 20 different runs.*

Notice that these models are highly structured with a strong modular structure (low inter-module-connectivity and strong intra-module connectivity) and also a significant distinctness on the added informations $\mathbf{Y}$. Thus, these good results were expected and they confirm the correctness of CohsMix[1]/CohsMix[2] algorithms.

CohsMix[3] After verifying that the proposed algorithm for this model detect the correct number of group, one again we decide to check the accuracy the algorithm analyzing the agreement between the true and estimated parameters. For these simulation, we choose to use the same graph structure model as in the previous experiments and make adjustments on the additional informations. Indeed, we fix a the number of covariates $nbCov = 3$ which means that each vertex disposes of a vector of dimension $p = 3$. Vertex vectors are simulated via a multivariate normal distribution with a covariance matrix equal to the identity (I_3), and mean vectors are defined as

$$\mu = \begin{pmatrix} 7 & 2 & 7 & 9 \\ 8 & 7 & 3 & 3 \\ 2 & 2 & 6 & 6 \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Composed by 4 classes and 3 covariates, we then obtain a matrix $\mu \in \mathcal{M}_{3,4}$. Applying CohsMix[3] on this simulated data set produces the following connectivity estimates

Parameters	Real Value	Estimated Value
$\hat{\pi}_{qq}$	0.2	0.19
$\hat{\pi}_{ql}$	0.05	0.06

and the multivariate estimates of the vertex features distribution

$$\hat{\mu} = \begin{pmatrix} 7.26 & 2.08 & 6.77 & 8.95 \\ 7.75 & 7.19 & 3.21 & 2.81 \\ 2.14 & 1.94 & 5.99 & 5.86 \end{pmatrix} \text{ and } \hat{\Sigma} = \begin{pmatrix} 1.29 & 0.16 & 0.25 \\ 0.16 & 0.77 & 0.06 \\ 0.25 & 0.06 & 1.35 \end{pmatrix}.$$

Therefore, this simple simulation allow us to check the good implementation our algorithm which produces estimates very close to the true parameters and an adjusted rand Index equal to 1.

Finally, whatever the model used (CohsMix^i), the hidden structure and model parameters are estimated accurately. However, the simulations were ideally suited to our models and parameters of different classes were significantly distinct. In the following, we will simulate less obvious structures.

3.4.2 Comparison of algorithms

In these experiments, we choose to focus our attention on CohsMix^3 model studying its behavior with more complex and less conducive simulation networks. We also compare this algorithm with competitors.

Simulation setup We consider simple affiliation models with two parameters defining the probabilities of connection between nodes of the same class and between nodes of different classes, respectively $\pi_{qq} = \lambda$ and $\pi_{ql} = \epsilon$, and equal mixture proportions $\alpha_1 = ... = \alpha_Q = \frac{1}{Q}$. Models have $n = 150$ nodes.

Graph models were generated in order to evaluate the performances of the algorithm as the difficulty of the problem varies. The clustering problem increases in difficulty with the number of classes Q, the number of features $nbCov$, the Euclidean distance $d(\lambda, \epsilon)$ between intra and extra connectivity parameters, and the distance $d(\mu_q, \mu_l)$ between the feature mean vectors of classes. We decided to focus on these parameters to produce data with different levels of structure and used 43 different graph models whose description are summarized in Table 3.3. Each model is simulated 20 times.

Experiments	Q	$nbCov$	$d(\lambda, \epsilon)$	$d(\mu_q^{(j)}, \mu_l^{(j)})$
a	$\{2, ..., 12\}$	3	0.4	4
b	5	$\{2, ..., 15\}$	0.2	4
c	3	3	$\{0, ..., 0.5\}$	4
d	3	3	0	$\{4, ..., 8.5\}$

Table 3.3 – *Parameters of the four different settings which are used to generate the 43 affiliation models considered in the experiments.*

Alternative clustering methods Additionally to the CohsMix^3 algorithm study, we compared it with two "rivals": a multiple view learning algorithm (Ruping and Scheffer 2005, Zhang et al. 2006), and a Hidden Markov Random Fields (Ambroise et al. 1997):

- *Spectral Multiple View Learning (SMVL):* There exits a strong similarity between multiple view and graph models with covariates. Multiple view learning algorithms consider instances which have multiple representations and use these views simultaneously to obtain a consensus partition. This is achieved via spectral clustering on a linear combination of a standard kernel corresponding to the graph structure and a kernel corresponding to vertex proximity.

- *Hidden Markov Random Fields (HMRF):* Hidden Markov Random Fields are commonly used to handle spatial data and are widely used in image analysis. We use a classical Potts model on the latent structure, which encourages spatial smoothing of the cluster. This kind of approach uses the graph structure to smooth the partition of the vertex over the graph, whereas the approach proposed in this paper uses the graph structure directly to estimate the vertex partition.

Simulations results We focus our attention on the Rand Index for each algorithm, inasmuch as a well-estimated partition yields good estimates.

As expected, the performance of the three algorithms deteriorates as the number of groups increases (Figure 3.6 a).

A first interesting result is that where there is a modular structure (Figures 3.6 a,b and c) in the network and weakly-informative features, CohsMix^3 algorithms always perform better than the SMLV and HMRF algorithms.

It is noticeable that the performance of CohsMix^3 improves as the number of features increases, and/or as the distance between mean vectors increases (Figure 3.6 b and d). The HMRF algorithm with a Potts model will generally use the neighbourhood structure for smoothing the partition. A vertex whose neighbours are all in the same given class has a high probability of also being assigned to this class, but HRMF does not take advantage of the graph structure as fully as CohsMix^3 . Our model is thus particularly suited to data sets with an existing graph structure.

When there is no graph structure at all and few informative features (Figure 3.6 d) the CohsMix^3 is no match for HMRF or SMLV. The CohsMix^3 algorithm is more sensitive to the total absence of graph structure than its competitors.

In all other setups, however, the quality of partition estimation remains good with different kind of models, the `CohsMix` algorithm appears very attractive and suitable for structured graphs with vertex features. We shall see in the next section that this algorithm also performs well on real web data sets.

3.4.3 Real data

Exhaustivity is an essential feature for information retrieval systems like Web search engines. However, ambiguous queries tend to produce a huge diversity of responses that can be a real impediment to understanding. A common way of circumventing this problem is to organize search results into groups (clusters), one for each meaning of the query. This has been a

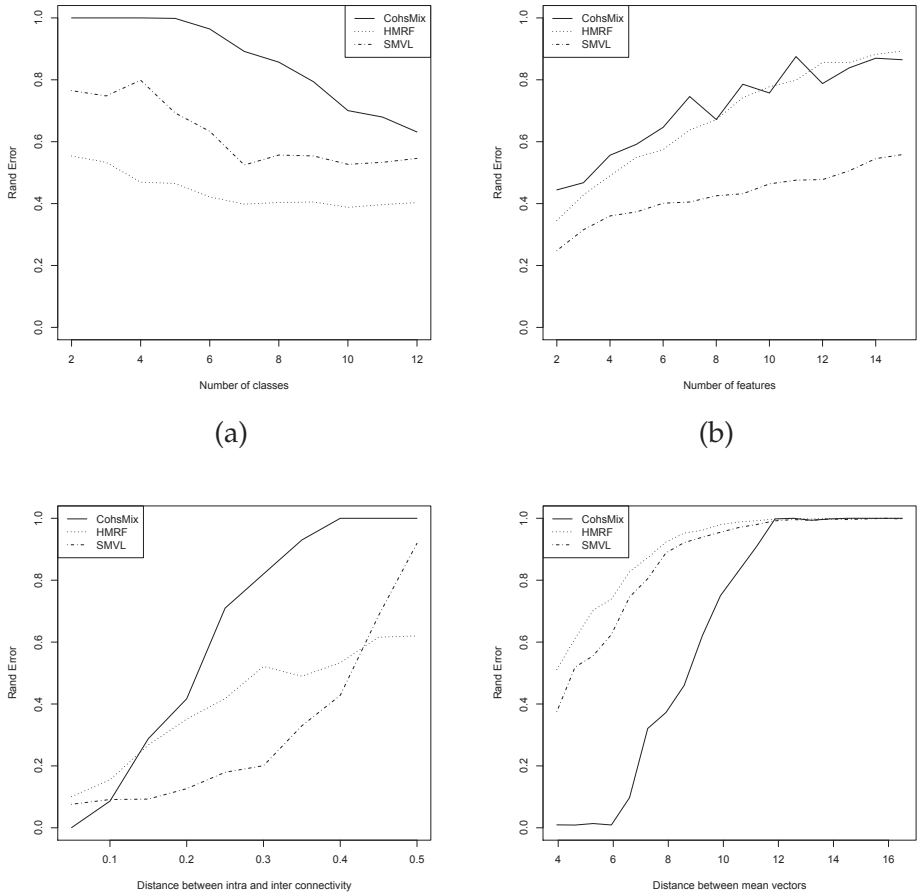


Figure 3.6 – *Comparison of HMRF, Spectral MLV and* `CohsMix`*. (a) Varying Q the number of classes. (b) Varying the number of Features. (c) Varying the distance between intra and inter connectivity parameters. (d) Varying the distance between the mean vector of the classes.*

focal point within the information retrieval community (Hearst and Pedersen 1996, Zamir and Etzioni 1998) since the early days of the Web. More recently, academic (Zeng et al. 2004) and industrial (Bertin and Bourdoncle 2002) (`exalead.com` or `clusty.com`) offerings have made the clustering of search results a common feature for a WWW user. Such approaches do not have to be confused with supervised classification techniques which use training examples to learn a decision function (Denoyer et al. 2001).

The main drawback of many web page clustering methods is that they only take account of the topical similarity between documents in the ranked list, without considering the topology formed by hyperlinks. In competitive or controversial queries (such as "abortion", or "Scientology") such methods fail to reveal community information visible in the link topology: by affinity, authors tend to link to pages with similar topics or points of view, which creates a thematic locality (Davison 2000). In addition, ambiguous queries like "orange" or "jaguar" might also harness link topology so as to produce a more accurate separation of results. Combining topological and topical clustering methods is a proven strategy in

building an effective system. One of the most relevant contributions to the literature is He et al. (2002), which describes a web page clustering system taking into account the hyperlink structure of the Web, considering two web pages to be similar if they are in parent/child or sibling relations in the web graph. A more general multi-agent framework based on the path between each pair of results was proposed by Bekkerman et al. (2007), but these methods, not model-based, use various heuristics and fine tunings.

Data sets setup We use the `exalead.com` search engine in our real data experiments. For each query, we retrieve the first 150 search results in order to build our graph and feature structures. The web is a very sparse graph and thematic subgraphs may amplify this property, creating unconnected components which reduce the feasibility of using classical graph clustering algorithms directly on the observed adjacency matrix. In order to increase the graph density, that is to say the probability of there being a link between two nodes, we propose using the site graph of `exalead.com`, based on the concepts of Raghavan and Garcia-Molina (2003). In this graph, nodes represent websites (a website contains a set of pages) and edges represent hyperlinks between websites. Multiple links between two different websites are collapsed into a single link. Intra-domain links are taken into account if hostnames/websites are not similar. The site graph is previously computed. It will be remarked that this methodology is similar to the Exalead application *Constellations*: `constellations.labs.exalead.com`.

Text features are extracted from the content of the web page returned by the search engine. The features are built using various text-processing techniques including normalization, tokenization, entity-detection, noun-phrase detection and related term detection. Rare features which do not appear more than twice are removed. The resulting feature vectors are approximately of dimension $p = 100$ and summarize the entire text returned pages.

Algorithm results We have selected one ambiguous query ("jaguar") and one controversial query ("Scientology") to illustrate the behavior of our algorithm with real data sets. In Figure 3.7, corresponding to the query "Scientology", we observe a well structured graph which fits our estimated latent partition with an optimal number of classes $Q = 3$. Basically, this partition yields the pro- and anti-Scientology clusters, and identifies a gateway cluster (composed for example by `http://en.wikipedia.org/wiki/Scientology`) bridging the pro- and anti- clusters. We then focus on the most representative text features of each class q. To this end we select the best occurrence-of-term features in the different μ_q. Once again (see 3.7), we notice pro-terms ("self esteem", or "providing real solutions") and anti-terms ("criticism of dianetics", or "truth about Scientology"). The interface class is composed of common terms describing the Church of Scientology. Thus, in a web context, the CohsMix[3] algorithm is enable to name the different partitions obtained, which is very useful for communicating rapidly a global overview of the hidden structure.

Another example of controversial query ("Abortion") is given in Figure 3.8. Similarly to the previous example we observe pro-choice ("med-

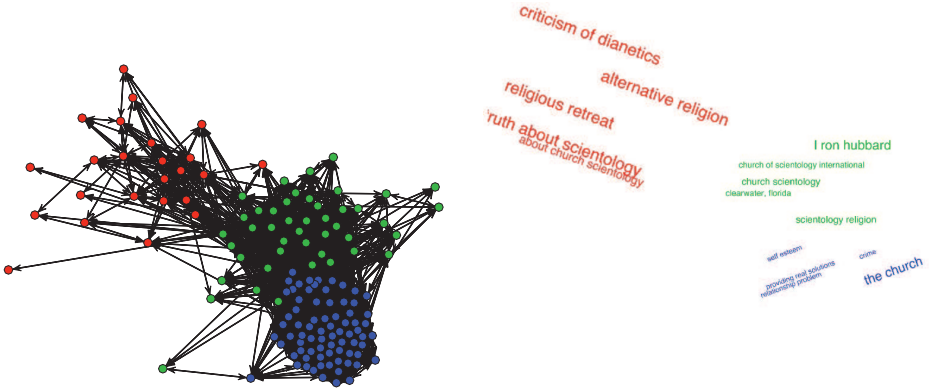


Figure 3.7 – *Representation of the results of a clustering of the webpages returned by the controversial query "Scientology" using CohsMix*^{3} *. The graph structure is represented on the left and on the right are the main features. Colors indicate the CohsMix*^{3} *classification.*

ical abortion", "problem pregnancy") and pro-life ("Jesus", or "abortion trauma"). We also find a class regrouping juridical terms ("supreme court", "states).

Figure 3.8 – *Representation of the results of a clustering of the webpages returned by the ambiguous query "jaguar".*

Finally, the results of the processing of the ambiguous query "jaguar" is represented in Figure 3.9. CohsMix^3 clearly identifies three contexts: computer, animal and car model related web pages.

Figure 3.9 – *Representation of the results of a clustering of the webpages returned by the ambiguous query "jaguar".*

The above results illustrate that our algorithm `CohsMix` seems well adapted to detecting ambiguous or controversial queries by WWW search engine users.

Chapter Conclusion

This chapter has proposed algorithms for clustering data sets that can be modeled with a graph structure embedding vertex features (or additional information on edges). Characterizing each vertex both by its connectivity and by a vector of features (or a matrix information), the CohsMix^i algorithms, based on a variational approach of EM, uses both these elements to cluster the data and estimate the model parameters. Simulation and comparison results show our algorithms to be attractive and competitive for various kind of models. When CohsMix^3 is used to cluster web search results based on hypertextuality and content, the relevance of this approach is amply demonstrated. We find that our algorithm successfully harnesses whatever information is found both in the connectivity pattern and in the features. In the short term we plan to investigate how to focus on one type of information, graph or features, when it becomes predominant.

A Web Application: *Constellations*

4

This chapter describes a new service based on hyperlink topology, called *Constellations*, which offers a new way to browse the *Exalead* search results. It exploits the fact that web content authors tend to link to pages with similar topics or points of view (Kleinberg 1999, Davison 2000). Available at `http://constellations.labs.exalead.com/`, this online application extracts, visually explores and takes advantage of the MixNet algorithm to reveal the connectivity information induced by hyperlinks between the first hits of a given search request. Since the Web is an open and large scale hypertext system with billion of nodes and links, this chapter will also focus on how to deal with a huge amount of data and produce an online service capable of responding in a very short time.

The present chapter is a collaborative work with the engineers of the Exalead team especially with Guillaume Esquevin, Jim Ferenczi, Sébastien Richard and Tristan Chapel.

4.1 Introduction

Since its creation and enhanced by its recent social aspect (*Web 2.0*), the World-Wide-Web is the space where individuals use Internet technologies to talk, discuss and debate. Such space can be seen as a directed graph where the pages and hyperlinks are respectively represented by nodes and edges. From this graph, many studies, like Broder et al. (2000), have been published and Section 4.2.2 recalls the key properties of the Web structure. However, this chapter rather focuses on local studies by considering that the Web is formed by *territories* and *communities* with their own conversation leaders and participants (Ghitalla et al. 2003). Here, we define a territory as a group of websites concerned by the same topic and a community as a group of websites in the same territory which may share the same opinion or the same link connectivity. These communities, revolving around topics (high-tech, finance, etc.), places or common affiliations (associations, political parties, etc.) shape the territories. One usually assumes that the existence of a hyperlink between two pages implies that they are content-related (Kleinberg 1999, Davison 2000), and that this similarity is independent of the hyperlink direction. By exploring the link page exchanges, one can actually draw the borders of web territories and it is the aim our application *Constellations*. The key innovation of this chapter, is to provide a tool which instantaneously extracts, explores and analyses the hyperlink connectivity of the Web, such as the experiments in Section 2.9, according to a user request.

In order to study topological properties of the Web induced by the hyperlinks (also called *Web graph*), the first necessary task consists in retrieving the largest part of the graph. Search Engines whose key tasks are briefly described in the following sections are the more prepared to fulfil these goals. For instance, the ranking of the pages is partly based on the transition matrix of this graph (see Section 4.3.3). Besides, the indexation of the documents allows to obtain the subgraph corresponding to a given territory (or topic). It suffices to only consider the pages (and their links) which match with the query terms.

Since this graph has about a billion nodes today, several billion links, and appears to grow exponentially with time, its storage and its algorithmic manipulation is a real challenge (Bharat et al. 1998, Boldi and Vigna 2004). Even simple operations on such a dataset, like reversing a graph, require many computational resources. This context implies to take advantage of distributed computing. Besides, MapReduce-like framework is an interesting computational paradigm (Dean and Ghemawat 2008) which can be used to build the resources of our application. Furthermore, exploiting the empirical observations and the compression techniques of Boldi and Vigna (2004), this chapter also presents a graph framework named *BigGraph* which allows us to manage very large graphs.

Any real application has to deal with heuristics to handle a whole real system. Our application is not an exception to the rule and some heuristics have to be considered. For instance, as explained in Section 4.3.3, the initial Web graph can be transformed in a website graph to increase the density.

Generally, in software architecture, an application is composed of two main components: the *front-end* which provides a user-friendly inter-

face which provides a visualization solution of the data and the *back-end* which provides the necessary information to the front-end. The latter will have the responsibility of building the subgraphs and computing the group, positions, rank of the returned vertices. The front-end should layout the search results to the user according of the underlying nature of the data and allow interactions. *Constellations* is not a finished tool but it meets the requirements to access the demo page of *Exalead*. To keep informed of its evolution, one can simply follow `http://twitter.com/LookAtTheWeb`.

Last but not least, the built resources and several parts of the *Constellations* service can be exploited for numerous other applications. Some of them, with similar approaches based on named entity recognition, will be mentioned in the following, but others, too far from our application, will unfortunately not be presented here. For instance, Web graph and its structure may help to identify the artificial clusters created by link farms in order to grant pages a higher ranking (Wu and Davison 2005). Detecting these farms links might discourage this unfair practice.

Outline of the chapter. We first begin this chapter explaining the birth and some structural properties of the Web. Then, since *Constellations* is derived from the recent search engine technology, we briefly recall the key tasks necessary to build a search engine. A review of the necessary resources used by our service and their construction is then given. Some heuristics will be also described. Next, we focus our attention on the back-end of the service which is the internal process performed from the initial user query to the front-end input. The latter is described in the last part of this chapter.

4.2 The World Wide Web

4.2.1 The advent of the Web

Before the Web. The World Wide Web was born in the early 1990s by merging ideas that have been formed and evolved throughout the twentieth century. It ensues from two key concepts: hypertext and user interface. Although Otlet (1935) defined what should be the "scholar's workstation" by describing a system of transcription, annotation, and consultation of documents, the majority of the scientific community ascribes authorship of the concept of hypertext to Vannevar Bush. Indeed, at the end of the Second World War, he published an essay, *As We May Think*, where he raised the problematic of the documentary explosion, especially the loss of discoveries within the documentation generated by the past. Efforts must be provided to allow scientists to access to past discoveries more easily. According to Bush, a document must be supplemented, enhanced, classified and searchable in order to be scientifically useful. Thereby, Bush (1945) described the principles of a system called MEMEX for "Memory Extender" which fulfilled the above features. Then, to reflect the functioning of the human mind, he proposed the concept of *associative trails* among the documents which made him one of the fathers of hypertext.

In 1965, Theodor Nelson invented the word "hypertext" for "non-sequential writing". Initially, it was primarily a writing tool which allowed authors to link ideas without being constrained by the linearity of traditional text. In parallel, recovering the concept of associative trail, he launched a digital library project, called XANADU, which aimed to permanently host all the available scientific publications. This project reflected both the project itself and the technological system able to concretise it.

Whereas the previously described works were focused on the ability to produce documents in a non-linear fashion, Douglas Engelbart showed a specific interest in finding tools to facilitate human-computer interactions. It is the inventor of a pointing device which is now universally known : *the mouse*. Focusing on networks and interfaces, it created the first operating system able to deal with hypertext, NLS, for oN Line System. This system allowed the collaborative work using a computer network. In the 1970s and 1980s, some other functional hypertext systems were emerging: HYPERT in 1983, and HYPERCARD in 1987.

Tim Berners-Lee. Computer scientist at CERN in 1989, he wrote an article entitled *Information Management: A Proposal* which drew a bitter criticism on information management, especially on some fundamental principles of data classification,

- *keywords:* About a same topic, two different people would not necessarily associate the same keywords;
- *hierarchical classification (or tree):* Tree structures do not represent the functioning of human thought: two similar concepts will eventually be connected by a common father rather than a bridge between them.

Inspired by the internal organisation of the CERN which was a vast web of researchers linked to each other by different kind of relationships, he proposed to model informations in the same form. The resulting system used hypertext to remedy problems of this period (Berners-Lee 1989). Then, by adding the properties to be an open and decentralized system of documents, *Inquire* was created in the early 1980s. Continuing this work, *WorldWideWeb : Proposal for a HyperText Project* (Berners-Lee and Cailliau 1990) describes what will become the Web. Here, the authors define three basic tools necessary for the operations of a hypertext system built on top of the Internet network :

- URL: an Uniform Resource Locator which specifies where an identified resource is available.
- HTTP: a HyperText Transfer Protocol which transfers hypertext requests and information between servers and browsers.
- HTML: a HyperText Markup Language which codes tags and rules then interpreted by the web browser to format documents in a particular way.

Besides, Berners-Lee and his team also developed a first version of web server and browser.

The Web and its evolution. In 1992, one year after leaving the site of the CERN laboratories, several web browsers and servers appeared. Then, in 1993, CERN opened all its Web related developments to the public domain. This was also the birth of the *Mosaic* browser developed by the NCSA. With the ability to handle images and forms, it will served as the basis for *Netscape Navigator*.

1994 marked a turning point with the founding of the World Wide Web Consortium (also called W3C), which works to harmonize the Web. This was also the birth of the first directory, *Yahoo!*, and *Netscape Navigator* succeeded to *Mosaic*. The hegemony of *Netscape Navigator* only lasted for a year. Indeed, in 1995, *Microsoft* struck back and launched *Microsoft Internet Explorer* (MSIE). Apache, the open-source server, as the first search engine, AltaVista appeared in 1995.

In 1998, *Netscape* had to renounce ahead MSIE and then decided to open its code by creating the *Mozilla* project. The same year, the first search engine of second generation was emerging: *Google*. To become the reference organization, the W3C had to interpose in the browser market. Until 2000, as the look and feel of the browsers were different, web developers had to design several versions of the same site for the different browsers. These behaviors had ceased: browsers implemented the standards carefully and were less permissive. Although the *Mozilla* project is a success, MSIE still owns more than 2/3 of the market.

The term *Web 2.0* appeared in 2003 with the emergence of Web applications that have facilitated interactive information sharing, interoperability, and collaboration on the World Wide Web. It was consolidated in O'Reilly (2005), examples of Web 2.0 include web-based communities, hosted services, web applications, social-networking sites, video-sharing sites, wikis, blogs, mashups and folksonomies. These kind of applications and the search market with its variants are still the most active technology areas of the Web.

4.2.2 The structure of the Web

After nearly a decade of Web growth, Broder et al. (2000) were interested in building a global map of the Web. For this project, the authors used the index of pages and links of *AltaVista* which was one of the largest commercial search engines at the time. Since this study, similar analyses on larger snapshots of the Web have been carried out including an early index of the Google search engine (Bharat et al. 2001) and large research collections of Web pages (Donato et al. 2007). We take advantage of this section to describe the main feature of the Web graph structure. Detail about this emerging research area can be found in Hendler et al. (2008) and Chakrabarti (2003).

Exponential Growth. The Web graph induced by the hyperlinks structure is very large : in July 2000, Murray and Moore (2000) estimated that it contained about 2.1 billions vertices and 15 billions edges. Moreover, about 7.3 millions pages were estimated to be added every day, and many others modified or removed. Recently (2008), Google search engineers affirmed that their system, processing links on the Web to find new content,

hits a milestone: 1 trillion of unique URLs on the Web at once[1]. However, not all of them lead to unique web pages. Many pages have multiple URLs with exactly the same content or very similar of each other (for example : the calendar)(Bar-Yossef et al. 2006). For any search engine, the size of the Web really depends on the definition of what is a useful page, and there is no exact answer. For instance, *Exalead* currently considers that 16 billions of pages should be indexed.

A Giant Strongly Connected Component. As a map of the Web may not resemble the geographical maps, Broder et al. (2000) define an abstract map partitioning the Web into a few large groups, and showing in a stylized way how these groups interact together. Firstly, the authors observed that the Web contains a giant strongly connected component. As mentioned in Section 1.2, many real-world networks have a giant connected component, *i.e.* a single component containing a significant fraction of all the nodes. It means that from major "starting page" sites like search engine or directory-like pages, one can reach the home pages of many of the major commercial, governmental, and non-profit organizations in the world. From here, one can reach most of the pages within each of these large sites. Since many of the pages within these sites link back to the search engines and directory pages themselves, all these pages are mutually reachable, and hence all belong to the same strongly connected component (also called SCC) which is by construction giant. Note that there is almost surely at most one giant SCC, since if there were two giant SCCs, any reciprocal links between them would require them to merge into a single SCC.

The Bow-Tie Structure. The relationships between all the remaining SCCs and the giant one were also studied in Broder et al. (2000). The authors suggest to classify nodes by their ability to reach and be reached from the giant SCC. First, two large sets can be identified and defined by

- IN: contains nodes that can reach the giant SCC but cannot be reached from it. It corresponds to pages that have not been "discovered" by members of the giant SCC.
- OUT: contains nodes that can be reached from the giant SCC but cannot reach it. It corresponds to pages that may receive links from the giant SCC which choose not to link back.

The macroscopic structure of the IN, OUT and the giant SCC relationships is represented in Figure 4.1. Here, the well-known "bow-tie" denomination of the Web structure is clearly illustrated. As the figure comes from the 1999 *AltaVista* search engine, the size of the different sets are long since obsolete but the relevant point is that all three of these sets are very large. One can also observe that there are pages that belong to none of IN, OUT, or the giant SCC. Thus, such pages can neither reach the giant SCC nor be reached from it. The following classification is then proposed:

[1] http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

- *Tendrils*: contains (a) the nodes reachable from IN that cannot reach the giant SCC, and (b) the nodes that can reach OUT but cannot be reached from the giant SCC.
- *Tube*: contains nodes that satisfy both (a) and (b). They travel from IN to OUT without touching the giant SCC.
- *Disconnected*: contains nodes that would not have a path to the giant SCC even if we completely ignored the directions of the edges.

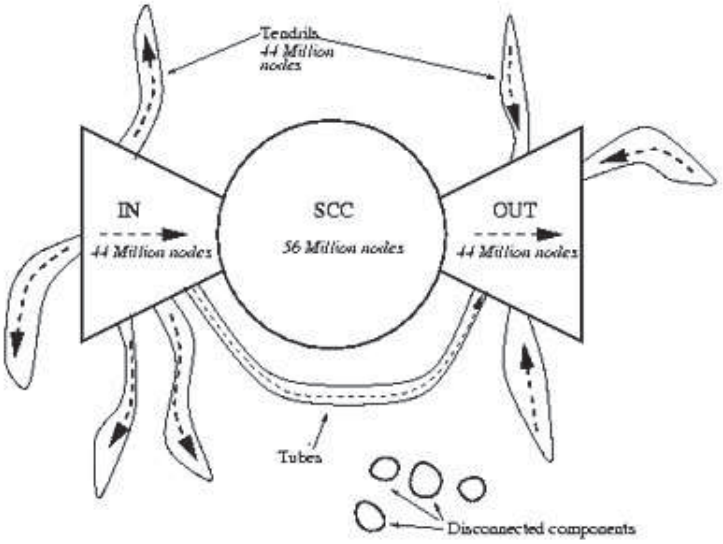


Figure 4.1 – *A schematic picture of the bow-structure of the Web (image from Broder et al. (2000).*

It is important to consider that the boundaries of the sets defined by the above classification are highly dynamic since pages and links constantly appear or disappear. However, subsequent studies suggest that, even as the detailed structure changes continuously, the schematic picture remains relatively stable over time.

While the classification based on page reachability gives us a comprehensive overview of the Web structure, it does not give us insight into the more fine-grained connectivity patterns *i.e.* connections which could serve to highlight important Web pages or communities of thematically related pages. Addressing these latter issues will be provide by the *Constellations* application.

Degree Distribution. Parallel to the Web structure studies, researchers have been interested in measuring and modeling the degree distribution of pages. A first approximation is to consider that both in-degree and out-degree follow the power-law distributions:

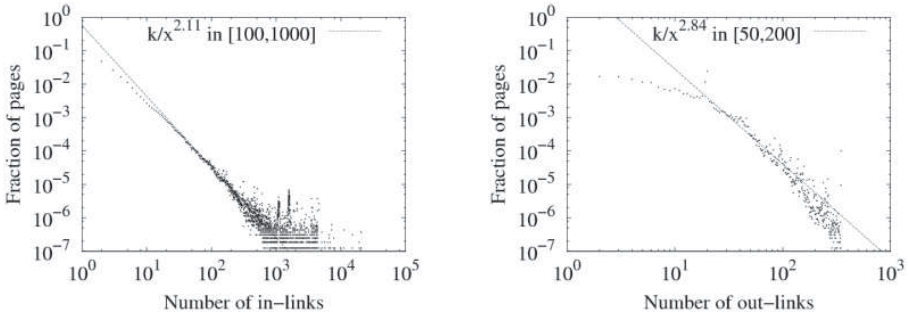
$$\Pr(outdegree = k) \propto 1/k^{a_{out}}$$
$$\Pr(indegree = k) \propto 1/k^{a_{in}}$$

To explain the appearance of the power-law distribution, Albert and Barabasi (2002) proposed mechanism where nodes are continually added

to increase the size of the graph, as it is eminently the case with the Web, with a *preferential attachment*. This key property dictates that a new node is linked to existing nodes not uniformly at random, but with higher probability to existing nodes that already have large degree. Such growing strategy has been called the “winners take all" phenomenon.

Let the Web graph be undirected to simplify the following discussion. Starting with m_0 nodes, at each step (discrete time) a new node u comes with a fixed number of m edges ($m \leq m_0$), which connect to nodes already existing in the graph. Suppose at this time step an existing node v is incident on $deg(v)$ existing edges. One can associate to v, the attachment probability $p(v) = \frac{deg(v)}{\sum_w deg(w)}$, where w ranges over all existing nodes. Then, all the m neighbours v of the node u are chosen with the attachment probability $p(v)$.

If the system runs for a long time ($t \to \infty$), Albert and Barabasi (2002) show that degree distributions follow power-law. Such distributions have been confirmed by a number of other measurements, such as by Broder et al. (2000). Figure 4.2 show that exponents range between $a_{in} = 2.1$ and $a_{out} = 2.5$. However, one can observe that the pure power-law model does not fit well for low values of k. Empirically, it seems that winners does not quite take all the degree distribution of nodes with low value of k creating imperfect power-law.



(a) (b)

Figure 4.2 – *The (a)in- and (b)out-degree of Web nodes closely follow power-law distributions, except at low degrees.*

4.3 *Constellations* Resources Building

After this brief description of the Web history and several of its properties, we now present the resources required to build a Web search service like *Constellations*. Derived from the latest search engines, we first remind the history of their quick evolution. Then, we discuss the main tasks, with their associated technologies, that *Constellations* has to deal with in order to reach, store, manipulate and interrogate the huge quantity of data. Finally, computational and storage issues for the Web graph will be more detailed for it constitutes the core the *Constellations* application.

4.3.1 Introduction to Web Search Engines

Several alternatives are offered to users to discover documents: free navigation (following hyperlinks), bookmarks, directories and search engines. Much of the success of the Web is due to the last two alternatives.

The directories are constructed and maintained by a community of volunteer editors. A short description of the content is indexed and websites are classified in pre-existing categories. Considering the growing size of the Web, it is obviously unthinkable to seek completeness in a directory. Besides, one can note that directories handle pointers to the websites and do not have an internal representation of their contents. Search engines fill the gaps of directories offering automated indexing to the pages contents. Automation is achieved by using crawlers that traverse the World Wide Web and stores, in repositories, copies of the pages downloaded. These pages are then processed in order to optimize and accelerate future searches. Technological details about search engines are given in Section 4.3.2.

First generation. The early search engines, even if they only indexed URLs and titles of pages, were not scalable, *i.e.* they could only operate small sets of pages. The first improvements concerned the adaptation of techniques from content analysis, originally employed in the *information retrieval* field, to the Web and its specificities. Thus, the ranking of the hits was provided according to algorithms based on *Vector Space Model* with TFIDF (Term Frequency-Inverse Document Frequency). Refinements were also proposed, for instance "bold text is more important than regular text", to better adapt to the specificities of Web documents.

Released in 1994, *Lycos* is probably one of the most famous search engines of this generation. Described in Mauldin and Leavitt (1994), this search engine indexed, for each document, the full headers (title, metadata, etc.), the 100 most relevant words (identified through TFIDF) in the first 20 lines, its total number of words and its weight. All operations were achieved in the same process: indexing and ranking were part of the crawling task.

Second generation. By considering only the content of Web pages, the first generation of search engines have ignored an informative type of data: the structure of the Web. The transition to a second generation of search engines is characterized by the addition of specific processings on the hyperlinks structure. Thus, PageRank algorithm of *Google*, described in Section 1.3.2, might be considered as the turning point of this evolution. Indeed, using the graph structure, the PageRank algorithm assigns a numerical weighting to each pages with the purpose of "measuring" its relative importance within the Web. Thereby, when a search is performed, all the relevant documents are ranked according to theirs pagerank scores and then returned to the user. It is important to notice that the topological processing is totally independent of the semantic processing. PageRank is not the only algorithm to take into account the hypertext topology, since the *Teoma* search engine or the *TARENTe* exploring tool (Ghitalla et al. 2004) have also been inspired by the HITS algorithm (Kleinberg 1999).

Moreover, the topological processings are accompanied by several innovations in the way of designing a search engine. For instance, we can notice that crawlers have become scalable using distributed architectures and dissociating the crawl and indexing the phases (see also Section 4.3.2). Concerning model and data structures, as for the first generation of search engines, the literature on the second generation lacks information.

Third generation. This is the current available search engines on the Web. Designed to combine the scalability of previous search engines with new and improved relevance models, the current search engines bring into the equation user preferences, collaboration, collective intelligence, a rich user experience, and many other specialized capabilities that make information more productive. Now, competitive search engines, such as *Exalead*, ensure successful search with user aids like faceted navigation of results, support for natural language queries, related query suggestions, fuzzy matching tools like phonetic, approximate spelling matches and spelling corrections. Besides, the user experience is also improved with seamless mashups and unified search of internal or external content, regardless of source or format (structured like database content, or unstructured like user-generated content, Web content, or multimedia files).

Finally, most of innovative Web 2.0 features are generally integrated like social search, advanced image and video search, maps and visual mapping of results (like *Constellations*), contextual Wiki information, etc. The latest innovations of the search engines may be found at their laboratory pages (see for example: `http://labs.exalead.com/`).

4.3.2 Crawling and Indexing the Web

Crawling the Web. The first task to achieve when studying the Web consists in gathering the largest possible part of it. Note that, since it grows exponentially and a large part of its content is hidden (Bergman 2001) (also called *deep Web*), it is impossible to get the whole Web at a given time. Obtaining parts of it as large as possible is yet a key challenge.

Retrieval of data from the Web is called *crawl* and is performed by computer softwares referred to as crawlers, spiders, robots, etc. (Brin and Page 1998, Heydon and Najork 1999, Najork and Heydon 2001, Boldi et al. 2004, Drugeon 2005). In general, a crawler browses the Web in a automated way which can be viewed as a breadth-first search in a graph: it starts with a list of URLs to visit, called *seeds*, identifies all the hyperlinks in these pages and adds them to the list of URLs to visit (called *crawl frontier*). Then, the process is iterated from the newly visited pages according to a set of policies. Indeed, to avoid overload of Web servers, the frequency of the crawler requests on them have to be limited. Such limitation forbids to perform a real breadth-first search and makes it hard to retrieve huge websites (with millions pages) in less than a month.

As mentioned above, recent crawlers have become more scalable using distributed architectures (Heydon and Najork 1999, Shkapenyuk and Suel 2002) and by dissociating the crawl and the indexing tasks (Najork and Heydon 2001, Brin and Page 1998). The independence of this two tasks is obtained using a *repository*. Its aim is to store downloaded pages by the

crawler. Whereas the first generation of search engines directly processes the pages in memory, the second generation prefers copying them on disk before performing indexation.

Finally, search engines care to offer the freshest content *i.e.* the picture of Web stored in the repository have to be as current as possible. Cho and Garcia-Molina (2000) aim at estimating the frequency of pages updates in order to refresh more regularly the pages that frequently change.

Indexing the Web. After storing the pages in a repository, search engines have to parse and store data in indexes to facilitate fast and accurate information retrieval. Indeed, without indexes, search engines should sequentially scan every document of the corpus, which would require considerable time and computing power. As in Brin and Page (1998), we briefly remind the indexing task that unfolds into three steps:

1. *Parsing:* a process designed to interpret the documents on the entire Web. Speed and robustness are its key features to handle billions of documents and the huge array of possible errors.

2. *Converting Documents:* A process designed to convert, using a *lexicon*, each document into a set wordID. Converted documents are then stored into temporary files.

3. *Sorting:* A process designed to take all the temporary files and sort them by wordID to produce inverted indexes for title, full text, anchor, etc. The sorting phase can be parallelized using a distributed computing framework such one defined in Section 4.3.4.

4.3.3 Ranking the Web

PageRank Application. Google search engine has developed a link analysis algorithm named *PageRank* (Page et al. 1998) that assigns a numerical weighting (also called the PageRank) to each element of a hyperlinked set of documents with the purpose of "measuring" its relative importance within this set. Actually, any collection of entities with references can enter into this algorithm framework.

Using a Markov chain model (see Section 1.3.2) in which the states are pages and the transitions depend on links between pages, the stationary distribution then represents the chance (PageRank value) that the random surfer will visit these pages. In such a random walk on G the transition matrix describing the transition from i to j is given by $\mathbf{P}$ with $P_{ij} = 1/deg(i)$. Figure 4.3 illustrates a step of a random walk.

However, minor changes of $\mathbf{P}$ must be applied in order to be a valid transition probability matrix. Firstly, every vertex (or page) must have at least 1 outgoing transition. Since a huge amount of pages has outdegree 0, this property does not hold for the Web graph. Thus, to transform $\mathbf{P}$ into a valid transition matrix $\mathbf{P}'$, we shall add outgoing transitions to pages with outdegree 0 in the following way:

Let n be the number of nodes in G and $\mathbf{v} = [\frac{1}{n}]$ be the n-dimensional column vector representing a uniform probability distribution over all

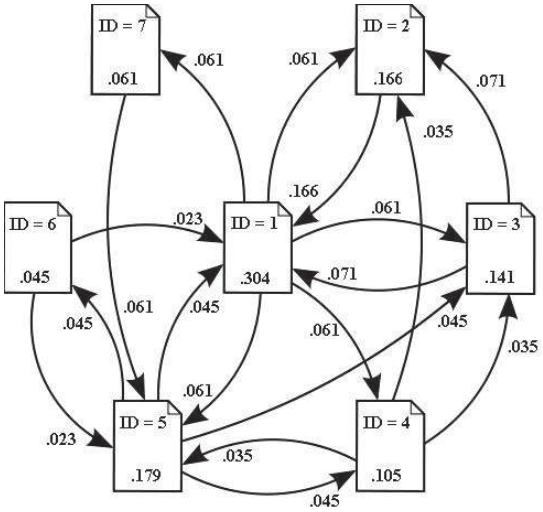


Figure 4.3 – *Graphical representation of PageRank*

nodes. The identification of the nodes with outdegree 0 is performed with a n-dimensional column vector satisfying the relation

$$d_i = \begin{cases} 1 & \text{if } deg(i) = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the new transition matrix $\mathbf{P}^{'}$ can be expressed by the relation $\mathbf{P}^{'} = \mathbf{P} + \mathbf{D}$ where the matrix $\mathbf{D} = \mathbf{d}.\mathbf{v}^T$. In terms of random walk, the effect of $\mathbf{D}$ is as follows: if the random surfer reaches a sink page, it picks another URL at random and continues surfing again.

If matrix $\mathbf{P}^{'}$ is aperiodic and irreducible then it has a unique stationary probability. A standard way to ensure this property is to add, to all nodes, a new set of complete outgoing transitions, with small transition probabilities, creating a complete (and thus strongly connected) transition graph. In matrix notation, the irreducible Markov matrix $\mathbf{P}^{''}$ is defined by the relation:

$$\mathbf{E} = [1]_{n\times 1} \times \mathbf{v}^T$$
$$\mathbf{P}^{''} = c\mathbf{P}^{'} + (1-c)\mathbf{E}$$

The effect of $\mathbf{E}$ on the random walk is related to a surfer teleportation. Indeed, at each time step, a surfer visiting any page will jump with probability $(1-c)$ to random Web page rather than follow an outlink. The destination of the teleportation is chosen according to the probability distribution given in $\mathbf{v}$.

Using notation matrix $\mathbf{A} = (\mathbf{P}^{''})^T$, the column i of $\mathbf{A}$ gives the transition probability for a surfer at node i and the principal eigenvector of $\mathbf{A}$ is exactly the PageRank vector (equivalent to the unique stationary distribution of the Markov chain). At time 0 , we assume that the probability distribution over the surfer's location is given by $\mathbf{x}^{(0)}$. Thereby, at time k, the probability distribution will be $\mathbf{x}^{(k)} = \mathbf{A}^k\mathbf{x}^{(0)}$. In the standard PageRank algorithm, the principal eigenvector is computed by starting with the uniform distribution $\mathbf{x}^{(0)} = \mathbf{v}$ and iterating $\mathbf{x}^{(k)} = \mathbf{A}\mathbf{x}^{(k-1)}$ until convergence. This is known as the *Power Method* and is summarized in the following

algorithm 4. In practice, this iterative method quickly converges ($k = 15$) to the final eigenvector.

Algorithm 4: Power Method for PageRank computation

Input: $A = P''$, the transition matrix associated the transformed Web graph matrix.

```
/* Initialization of the pagerank vector */
```
$\mathbf{x}^{(0)} = \mathbf{v}$
$k = 1$
while $\delta < \epsilon$ **do**
  $\mathbf{x}^{(k)} = A\mathbf{x}^{(k-1)}$
  `/* Compute the` L_1 `norm */`
  $\delta = \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_1$
  $k = k + 1$

Result: $\mathbf{x}$, the pagerank vector

Variants of this algorithm have been proposed in order to combate web spam pages. We can quote Gyöngyi et al. (2004) that have biased the initial PageRank vector by boosting its elements identified by experts as reputable pages.

We may mention that Power Method is also performed in *Hyperlink-Induced Topic Search* (HITS), another link analysis algorithm developed by Kleinberg (1999), to achieve the *Hub* and *Authority* scores computation. Authority and Hub values are defined according to one another in a *mutual recursion*. An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to.

4.3.4 Reverse Web graph using a MapReduce-like framework

In order to study the structure induced by the hyperlinks, we have to transform all the previous stored pages in a more convenient way. Particularly adapted to ranking algorithms, we propose to represent this structure in a *reverse Web graph* fashion.

Definition 4.1 *A **forward Web graph** is a graph that has an edge from node URL_1 to node URL_2 if the web page found at URL_1 has a hyperlink to URL_2. A **reverse Web graph** is the same graph with the edges reversed.*

Nevertheless, reversing a huge graph requires many computational resources. Distributed computing across multiple machines is the only reasonable strategy to compute such a graph. The following describes how to build a reverse Web graph with a distributed computing framework.

Introduction to MapReduce-like frameworks. The MapReduce framework introduced by *Google* (Dean and Ghemawat 2008) is designed for computing distributable problems over many machines. Recently, a wide adoption of this framework has been observed in both industry (*Yahoo!*,

FaceBook, *Exalead*, etc.) and academia[2] becoming *de facto* the standard for large scale data mining. Capable of processing more than 10 petabytes of data per day at *Google* data centers, it can be employed in a wide range of its applications, including: inverted index construction, statistical machine translation(Dyer et al. 2008), machine learning, reversal graph, usage mining, etc.

The name of this framework is inspired by *Map* and *Reduce* functions that interleave parallel and sequential computation. They behave as follows:

- **"Map" step:** The input data is hashed by the master machine which then distributes these sub-datasets to the worker machines. Each worker machine proceeds its sub-dataset, and passes the answer back to its master machine.
- **"Reduce" step:** The master machine then considers the answers of all the sub-datasets and combines them in order to solve the initial problem.

Since all the mapping operations are independent, the Map-step can be performed in parallel. Therefore, in practice, it is only limited by the data source reader and/or the number of CPUs available to process the data.

Another advantage of the MapReduce-like frameworks is that they offer the ability to recover from partial failure of servers or storage during the job. Indeed, since each machine is expected to report back periodically its completed work and status updates, the master machine can reschedule a work to other machines if one falls silent for longer than that expected (assuming the input data is still available).

Mapping and Reducing *(key, value)* pairs. The input to any *Map* and *Reduce* function is exclusively a set of $(key, value)$ pairs. *Map* function transforms one pair of input data with a certain type of domain into a list of pairs with a different domain:

$$Map(k_1, v_1) -> list(k_2, v_2) \tag{4.1}$$

Applying, in parallel, the *Map* function to each single (k_1, v_1) pair of the input data set, produces as output a list of new (k_2, v_2) pairs related to them. Then, all building pairs (k_2, v_2) are analysed and pairs with the same key are grouped together. Each group is then represented by a generated key k_2. The *Reduce* function is then applied to each group, which in turn produces a collection of values in the same domain:

$$Reduce(k_2, list(v_2)) -> list(v_3) \tag{4.2}$$

This behavior highlights one of the sequential aspects of the computation: all the maps need to be completed before the reduce stage can begin. In the reduce step, the parallelism is exploited by observing that reducers operating on the different keys can be executed simultaneously.

Finally, note that one can design an algorithm where many rounds of *Map* and *Reduce* functions are performed one after the other.

[2]see `http://research.yahoo.com/node/2743`

Application to reversal Web graph. As mentioned above, this computational model can be used to build a reverse web graph. According to the model paradigm, the job can be summarized by the two following functions:

- *Map:* outputs $< Target, Source >$ pairs for each link to a target Url found in a document named *Source*.
- *Reduce:* concatenates the list of all source Urls associated with a given target URL and emits the pair $< Target, ListOfSourceUrls >$.

To have a deeper understanding, the hot spots of the reversal Web graph application can be defined by the followings functions:

- Input reader: Reads data from the repository of the stored Web pages, divides them into splits of 128MB (approximately 10000 $(Url, Webpages)$ pairs) and assigns each split to a *Map* function.
- Map function: Takes a set of $(Url, Webpages)$ pairs, proceeds each, and generates zero or more output $< Target, Source >$ pairs for each link to a target URL found in a Web page source.
- Partition function: Allocates the output $< Target, Source >$ of all the maps to a particular reducer by the application's partition function. By default, the index of the reducer is obtained hashing the target URL domain modulo the number of reducers.
- Compare function: Sorts using the application's comparison function where the primary key is the target URL and secondary key the source URL.
- Reduce function: Concatenates the list of all source URLs associated with a given target URL and emits the pair $< Target, ListOfSourceUrls >$.
- Output writer: Writes the output of the reducers in specific graph storage framework named *BigGraph*(see Section 4.3.5).

In practise, using 16 machines each with 4 CPU and 2 gigabytes of ram, this job ($\sim$ 110 To of input data) requires approximately 10 days and 5 terabytes of data storage (including the anchor text of the hyperlinks). Note that this period can be reduced taking advantage of more machines.

Considering that Web pages have been transformed into a *reverse Web graph* by a MapReduce-like job, we now have to store the $< Target, ListOfSourceUrls >$ data ouptut in a efficient way to facilitate fast access to the graph structure.

4.3.5 BigGraph

Exploiting both the empirical observations and the compression techniques of *WebGraph* (Boldi and Vigna 2004), we have developed a graph framework called *BigGraph* which allows us to manage very large graphs such as the Web graph. Usually quoted, the key features of the Web graph

links are the *locality* and the *similarity*. Such features were originally exploited by the Connectivity Server (Bharat et al. 1998) and by the LINK database (Randall et al. 2002). They can be explained by:

- *Locality:* Usually, most of the hyperlinks (80%) are navigational, *i.e.* they point to other pages within the same host. By comparing the source and target Urls, one can observe that they share a long common prefix. This can be exploited if URLs are sorted lexicographically: the index of source and target will be close to each other.
- *Similarity:* Consecutive pages in lexicographical order tend to share many common successors. Actually, many navigational links are the same within the same local cluster of pages, and even non-navigational links are often copied from one page to another within the same host.

Then, as for full-text indexing, graph storage can involve similar compression techniques to handle increasing sequences of integers with small gap (Adler and Mitzenmacher 2001, Randall et al. 2002). Moreover, taking advantage of the similarity features, a successor list of a node can be specified by copying part of a previous list, and adding whatever remains. Such copy can be performed using a list of bits, one for each successor in the referenced list, which tells if the successor should be copied or not, or using other techniques such as explicit deletion lists (Randall et al. 2002).

In practice, *BigGraph* can compress a graph with 4 billions nodes and 40 billions links in 180Go that approximately corresponds to 5 bits per link.

Lazy Iteration. Similarly to Boldi and Vigna (2004), *BigGraph* enumerates successors using lazy iterators: successors lists of nodes are decompressed on the fly, which allows to iterate over very long lists without expanding them into memory. Since graph algorithms could require both sequential and random accesses, one has to consider both ways. Whereas sequential-access is straightforward, random access requires keeping an auxiliary vector of offsets with N entries (one for each node). This vector indicates where the successor list of node starts. However, using 64-bits for each offset suffers from an inherent lack of scalability because it would make the offset array larger than the graph itself. This finding suggested the authors to propose a complex scheme for the offset representation. Thereby, *BigGraph* proposes an interesting trade off between speed and memory consumption.

Actually, our solution allows to perform random accesses with a limited amount of central memory by a partial loading the offsets array. Let G be a fixed natural integer (called gap), only offsets of nodes $1, G, 2G, \cdots$ are loaded in the smaller array of offsets $\mathbf{O}$ of size N/G. The offsets between nodes 1 and G described by $(s_1, \cdots, s_G)$ can be stored in a compressed way as a list of integers with small gaps, as

$$(s_1, \cdots, s_G) = (s_2 - s_1 - 1, \cdots, s_G - s_{G-1} - 1)$$

Memory consumption of the offset list can be drastically reduced using encoding for natural integers such as *alpha coding* or *zeta coding*. Then,

accessing given node x is performed in two steps. We first use the offset relative to node $\mathbf{O}[x/G]$, and then we decompress on the fly the remaining offset until we reach the node. Thus, choosing a small value for G allows to access a node more rapidly than with a larger value. However, it increases the memory consumption.

The *BigGraph* framework provides simple methods to manage very large graphs inspired by the LINK database and the *WebGraph* framework. The library is written in C++ and exposes many other graph representations such as arc-labelling and *BigMap*. The latter provides an efficient method to store Urls of a web graph and allows the mapping between ids and Urls. The whole package is used for accessing, compressing and manage very large graph and is used as a back-end for applications such as the pages ranking or the *Constellations* application.

4.3.6 Graph simplification

A Large part of the hyperlinks are navigational and lead the user to other pages within the same host (example of anchor text: "home", "next", "previous", "up" etc.). Although theses links are essential for the user experience, they are not relevant to detect *territories* or *communities*. A first strategy to circumvent this useless information is to only consider extra-host hyperlinks. However, since the Web graph is already a sparse graph, such an operation amplifies this property and will reduce the ability of using elaborate algorithms. Thus, one can consider a concept of *meta-node* within which a set of nodes is aggregated (Raghavan and Garcia-Molina 2003). Besides, edges are absorbed by the meta-node, keeping those which connect two different meta-node and dropping the others. Multiple links between two different meta-nodes are collapsed into a single valued link. Figure 4.4 illustrates the concept of meta-nodes, aggregating nodes of the same color in the same meta-node. Note that the only considered edges connect pairs of nodes with different colors.

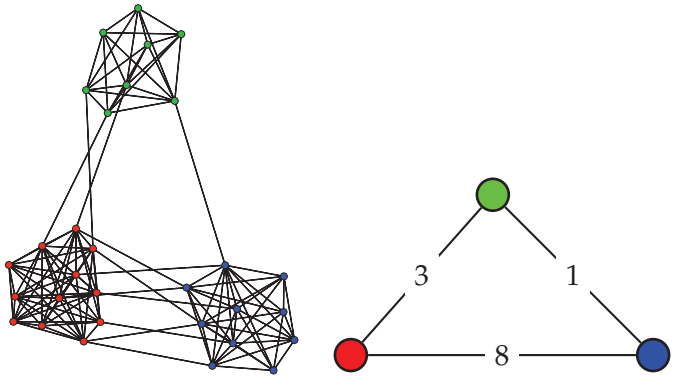


Figure 4.4 – *A schematic example of the graph simplification. Nodes with same color will be associated to the same meta-node.*

At first glance, associating to each URL (node) its hostname (meta-node) is an acceptable strategy and this grouping often produces what is commonly called a *website*. Nevertheless, it is important to notice that there is no strict definition of this concept. Abstractly, a website is defined as a grouping of pages with the same editorial authority. Thus, we can

highlight two phenomenons where our previous hostname grouping is not functioning properly:

- *hosting-server:* For instance, `www.myspace.com/chupachuva` and `www.myspace.com/thebeatles` which belong to the same hostname, have to be considered as two different websites because they do not refer to the same music band (editorial board).
- *multi-host-spammers:* if `www.domain.fr`, `domain.fr`, `www2.domain.fr`, etc. have a replicated or very similar content, they might be considered as the same website. Note that this problem, often named *mirror detection*, can also be considered with different domains (Bharat and Broder 1999).

According to these observations, it is possible to design learning algorithms to transform the graph of pages into a graph of websites, detecting the editorial boundaries (hosting-server and multi-host-spammer). Typical features used by these algorithms are the domain IPs and their connectivities, the distribution of links among the intra-extra domain, the content similarities and finally the lexicographical resemblances of the URLs. In the following, we will name *site-inference* the data structure able to map an Url to its website. The number of websites taken into account in the *Constellations* service is approximately $n = 140,000,000$ with a mean of 66 pages per website. Nodes represent websites (a set of pages) and edges represent hyperlinks between them. Each website is characterized by both an identifier *sid* (for "site *id*") and its neighbourhood identifier (a list of *sid* predecessors). With approximately $m = 1,500,000,000$ edges, the density of this website graph is equal to $\delta(G_{website}) = \frac{2.m}{n.(n-1)} = 1.5 * 10^{-7}$. Since the initial density of the page graph was $\delta(G_{webpage}) = 5 * 10^{-9}$, this transformation significantly increases the density of the considered graph. In the following, we will assume that a website graph, stored in *BigGraph* fashion (see Section 4.3.5) has been built. It approximately requires 3Go of storage and its weights 7Go.

4.4 *Constellations* Back-end

This section highlights the key steps of the *Constellations* back-end. Usually a back-end service interacts and serves the front-end services (user interfaces), by being closer to the required resource or having the capability to communicate with the required resource. Therefore, we will describe the internal process of the service from the initial users query to the front-end input. First of all, we present the variety of queries that the system can handle using the *Exalead Web Search API v1.0*. Then, we explain how the graph which will be returned to the user interface is constructed. This construction consists in retrieving the involved websites/links, computing the relevant position of these nodes by anticipating the drawing of the user interface, computing connectivity groups (like MixNet), and finally returning an understandable answer to the front-end. Obviously, all these operations must be conducted in a very short time which combined should not exceed 10 seconds after the user query.

4.4.1 Handled Queries

Storing a large completeness of the Web has interest if we are unable to perform complex queries on it. Thus, we propose to take advantage of *Exalead* query language which is known for its great expressiveness. This language, understood, by the *Exalead Web Search API* service, allows to create a powerful search service.

Based on the REST approach (for ***Re**presentational **S**tate **T**ransfer*) (Fielding 2000) which uses a client-server architectural style, the search service output of a search query is an *RSS* extended *OpenSearch/1.1* protocol (`http://www.opensearch.org`). Such an output is a standard *XML*-based syndication format for search results. It embeds specific extensions to *Exalead* search engine as the number of search results, pagination information, suggested terms, and thumbnail content. In this section, we briefly describes the *Exalead Web Search API* search protocol including :

- the *HTTP* request and the available parameters
- the *RSS* response and complete element descriptions
- some options for the API's search services.

We can note that *Exalead Web Search API* also supports other search services than the Web, such as image, video, wikipedia or news. This variety of services will allow us to adapt our methodology to various kind of content. The adaptation of the *Constellations* service to wikipedia and news data sets will be presented at the end of this chapter.

HTTP request. A request to the *Exalead Web Search API* consists of a standard *HTTP GET* request including the appropriate parameters. Table 4.1 describes the *HTTP* parameters that are used in our application.

HTTP Parameters	Possible Values	Description
q	Any URL encoded string.	the query terms (detailed in Table 4.2)
nhits	Any positive integer. Default value is '100'.	the number of search results to display
language	All two letter ISO language codes:'en','fr','es', etc. Default value is 'en'.	Boosts documents that match the given language
origcountry	All two letter ISO country codes: 'GB', 'FR', 'DE',etc. Default value is 'US'	Boosts documents that are identified as geolocated in the given country

Table 4.1 – *HTTP parameters of the Exalead Web Search API used for the Constellations services.*

Note that the language and country boosts are a first step towards a more personalized service which focuses on the user preferences.

Query Language. As seen in the *HTTP* parameters, the value of the "q" parameter refers to the query the user wants to execute. The language of this query allows the user to perform Boolean logic with, for example, AND OR operators. The service supports the following common search features (see Table 4.2).

Search feature	Description
Basic search	Only documents that contain all words from the search request are shown. For example, *movie star*.
Exact phrase	Use double quotes to search for an exact phrase. For example, *"to be or not to be"*.
Exact word	Use the exact operator +(plus) to search for the exact word in a document (disable word stemming). For example, *+stars* searches for documents containing "stars" exactly (and not "star").
Exclude terms	Use - to exclude Words from the results. For example, *cow -brown* searches for documents containing "cow" and not "brown".
Logic	Searches can be formulated using Boolean operators such as conjunction (*AND* operator), disjunction (*OR* operator), and negation (*NOT* operator). For example, *((fast OR speed) AND NOT light)*
Optional terms	It is possible to specify optional query terms (*OPT* operator) with specific ranking specifications.
Prefix search	Use * to search for words starting with certain letters. For example, *water** gives results including any word beginning with the letters "water", such as "waterfall" and "waterway".
Proximity search	The *NEXT* operator finds documents where the query terms are next to each other. For example, *movie NEXT star*, would search for documents where "movie" and "star" appear next to each other in that order. The *NEAR* operator finds documents where the query terms are within, by default, 16 words of each other. *NEAR/x* allows to specify the maximum distance of the words.

Table 4.2 – *The Constellations search features.*

Finally, in addition to these common operators, the *Constellations* service supports advanced Exalead search like "approximate spelling search", "site search", "title search", "URL search", "link search", "search after or before a date" or "search language". These optional search syntaxes are described at `http://www.exalead.com/search/web/search-syntax/`.

Refinements. A unique feature of Exalead is to provide navigation refinements associated to the user's query (Bourdoncle 1997). These suggestions allow the user to "navigate" on a semantic graph, focusing, excluding or refining with a particular category or keyword. A refinement always has the same syntax: *refine:type:"value"*. For example: *refine:filetype:"pdf"*.

- *refine:* The keyword to trigger a refinements

- *type:* The type of refinement (language, country, filetype, date, etc...). Refinements types differs depending on the requested service.
- *value:* The effective refining value.

This opportunity to use refinements allows us to retrieve Web pages responding to very complex queries. For instance, one might search documents matching the exact phrase "graph clustering", hosted in blogs, written in french after July 2008. Last but not last, that each service (news,image, etc.) provides specific operators (detailed in advanced search pages of the *Exalead* search engine).

RSS Response. The response from the *Exalead Web Search API* is a *RSS 2.0 Feed* which mainly contains a list of items of hits that have matched the user's search query. An item has the elements described in Table 4.3.

Element	Description
link	The link is the URL of the search result.
title	The title of the search result.
description	A summary of the document with <b> *HTML* bold tags highlighting the words of the user's query.
pubDate	The date of the document (if available).
thumbnail	A small image representation of the document.

Table 4.3 – *Item contained in the* RSS *response which have matched the user's search query.*

From this point, we consider that the 100 most relevant documents (or hits) according to a user query are returned via the *Exalead Web Search API* in an average time of 1 second. Then, we have to construct the underlying website graph formed by theirs hyperlink connectivity.

4.4.2 Graph Search

As argued in Section 4.3.6, a website graph have been previously built where nodes represent websites (a set of pages) and edges represent hyperlinks between them. Although this graph is still huge with more than 140 millions of nodes and 1.5 billions of edges, it is a simplified graph of the Web and it offers an interesting overview of the global connectivity. Then, our task consists in building the underlying website subgraph of the returned hits from a search request. This operation can summarized by the following algorithm:

1. Mapping: For each hit (matching the user search request), we map the Url to its *sid* (related website using the previous *site-inference* data structure). Then, each *sid* is stored in a white list.
2. Neighbourhood: For each element of the white list, we use the *BigGraph* website structure to get all its predecessors (all website *sid* that have a link to it).
3. Filtering: All predecessors have to belong to the white list. Therefore the final considered hyperlinks only connect *sid* belonging to the white list.

Thus, these three steps allow us to build the hyperlink subgraph between website of the returned hit list. Nevertheless, some modifications can be taken into account. For instance, orphan nodes *i.e.* nodes without neighbourhood can be treated separately. In order to reinforce the topology structure, we can also eliminate the edges that have a low valuation (number of links between each pair of websites), or that are not reciprocals. Last but not least, if one wants to increase the size of the subgraph adding websites that do not match the user query, the white list can be extended using a breadth-first search at step 2 of the previous algorithm. The extension begins at the root nodes of step 1 and explores all the neighbouring nodes. Then for each of those nearest nodes, it explores their unexplored neighbour nodes, and so on, until it finds the goal (number of iteration, number of nodes or distance from the roots). All visited nodes are added to the white list which allows us to apply the same filter as in step 3. Once this graph is constructed, we can now draw it using an adapted layout algorithm and detect connectivity group using a clustering algorithm such as MixNet.

4.4.3 Layout Algorithm

Force-based or force-directed algorithms are used for drawing graphs in an aesthetically pleasing way. Their purpose is to layout the nodes of a graph in a two or three dimensional space minimizing the crossing of edges. This is achieved by assigning forces across the sets of nodes and edges. For instance, in the Fruchterman and Reingold (1991) algorithm which is commonly used for graph visualization, the nodes are represented by steel rings and the edges are springs between them. The repulsive force is analogous to the electrical force and the attractive force is analogous to the spring force. The sum of the force vectors determines in which direction a node should move. The step width is a constant which determines how far a node moves in a single step. When the energy of the system is minimized, the system reaches its equilibrium state, *i.e.* the nodes stop moving. Since there is no guarantee that the system will reach equilibrium with a constant step width, the authors have introduced a "global temperature" that controls the step width of node movements and the termination of the algorithm. The step width is proportional to the temperature, so if the temperature is high, the nodes move faster (*i.e.*, a larger distance in each single step). This temperature is the same for all nodes, and cools down at each iteration. For completeness, we summarize in the following the Fruchterman-Reingold algorithm 5.

Although this algorithm have an significant running time ($O(n^2)$) is very useful for visualizing large networks. It provides that topologically close nodes are positioned in the same vicinity, and far nodes are placed far from each other. In the *Constellations* application, we apply 50 iterations of the algorithm with 5 terminal repulsive iterations. The last repulsion steps provides a better node spacing that implies a better readability.

Algorithm 5: FRUCHTERMAN-REINGOLD placement Algorithm

Input: $area = W * L$, W and L are the width and length of the frame
Data: $G = (V, E)$, the vertices are assigned to random initial positions
$k = \sqrt{area/|V|}$

for $i = 1$ *to iterations* **do**
  /* Calculate repulsive force */
  for v *in* V **do**
    /* Each vertex has 2 vectors: *pos* and *disp* which respectively represent the position and the displacement of the vertex. */
    $v_{disp} = 0$
    for $u \in V$ **do**
      if $u \neq v$ **then**
        $\Delta = v_{pos} - u_{pos}$
        $v_{disp} = v_{disp} + (\Delta/\|\Delta\|) * (k^2/\|\Delta\|)$
  /* Calculate attractive force */
  for $e \in E$ **do**
    /* Each edge is and ordered pair of vertices .*v* and .*u*.*/
    $\Delta = e.v_{pos} - e.u_{pos}$
    $e.v_{disp} = e.v_{disp} - (\Delta/\|\Delta\|) * (\|\Delta\|^2/k)$
    $e.u_{disp} = e.u_{disp} + (\Delta/\|\Delta\|) * (\|\Delta\|^2/k)$
  /* Limit the maximum displacement to the temperature t and then prevent from being displaced oustside frame */
  for $v \in V$ **do**
    $v_{pos} = v_{pos} + (v_{disp}/|v_{disp}|) * min(v_{disp}, t)$
    $v_{pos.x} = min(W/2, max(-W/2, v_{pos.x}))$
    $v_{pos.y} = min(L/2, max(-L/2, v_{pos.x}))$
  /* Reduce the temperature as the layout approaches a better configuration */
  $t = cool(t)$

Result: Fruchterman-Reingold placement of G in a two dimensional space.

4.4.4 Nodes attributes

Once the graph has been spatialized, it is important to explore the nodes features as well as the topology. Thus, we choose to add several attributes to each displayed node. First, we begin with those which are not related to the graph topology.

Content attributes Each node is described by the following content attributes:

- *Page/Site URLs:* the URLs of the page and its underlying website.
- *Title:* the title of page hit.
- *Description:* the description of the hit.
- *Thumbnails:* the small image representing the hit.

We can notice that these features are directly derived from the *Exalead Web Search API* response. Now, we focus our attention on attributes related to the graph structure.

topological attributes Each node is described by the following topological attributes:

- *Ranking:* the ranking of the hit.
- *Group:* the connectivity group the hit belong to.

Briefly, the computation of the ranking scores to all the crawled pages is performed using an adaptation of the link analysis algorithm described in Section 4.3.3. Then, the final ranking is provided in the range $[0, \cdots, 12]$ using a mapping function. Schematically, the level scores of such a function are built in order to ensure that each level contains three times more elements than its predecessor. Moreover, we assume that the last level must contain the half crawled web pages.

Concerning the group attribute, interestingly, energy models of pairwise attraction such as the Fruchterman-Reingold method are closely related to maximizing modularity (Noack 2009). Thus, to provide an complementary information to the graph layout, it appears more in interesting to propose clustering based on the connectivity profiles rather than on community. MixNet is then the chosen model to cluster the nodes.

The *Constellations* application uses a *C++* implementation of the online *CEM* algorithm which is the fastest algorithm for the estimation of the parameters. They are estimated as well as a statistical criterion ICL to select the number of classes. In practice, orphan nodes are not considered (they form a connectivity class in itself) and we choose the optimal number of classes running our online *CEM* algorithm concurrently for models by default f from $Q_{min} = 2$ to $Q_{max} = 12$ classes by selecting the solution which maximizes the ICL criterion. However,Q_{min} and Q_{max} can be parametrized keeping $Q_{max} < n$ with n the number of nodes. To avoid initialization issues, the *CEM* algorithm is started with 5 initialization points and the best

result is selected based on its likelihood. Thus, for each query, we run the algorithm $(12-2)*5=50$ times.

Once the graph, its layout and its attributes have been computed, we can propose a user interface capable of rendering them. However, before describing the front-end part, an exchange format between the back-end and front-end has to be defined.

4.4.5 *XML* response of the back-end

The aim of this section is to describe the final *XML* response of the back-end. Using a DTD (Document Type Definition) which allows to define the legal building blocks of an *XML* document, we propose the following structure with a list of legal elements and attributes. First we describe, in Table 4.4, the elements related to the *node* and *edge* elements:

```
<!ELEMENT node>
<!ATTLIST node
          id CDATA #REQUIRED
          title CDATA #REQUIRED
          description CDATA #REQUIRED
          pageurl CDATA #REQUIRED
          siteurl CDATA #REQUIRED
          rank CDATA #REQUIRED
          thumbnail CDATA #REQUIRED
          x CDATA #REQUIRED
          y CDATA #REQUIRED
          group CDATA #REQUIRED >
<!ELEMENT edge EMPTY>
<!ATTLIST edge
          from CDATA #REQUIRED
          to CDATA #REQUIRED>.
```

Table 4.4 – *DTD for the node and edge elements.*

We can observe that from the previous attributes of Section 4.4.4, an identifier and the displayed (x,y) positions are added for each *node*. The *edge* element just makes the relationship between two nodes *id*.

Then, in order to provide the MixNet connectivity results, Table 4.5 proposes the following description, where the element *connectivity* represents the connectivity parameter $\pi_{g1,g2}$. Its Boolean attribute *link* is true if the probability to connect a node belonging to the $g1$ *group* to a node belonging to the $g2$ *group* is high enough in comparison with its distribution. A *group* is described by an identifier and a *ratio* (similar to α_q). The *tag* element allow to provide the most discriminating *word*s in the different classes. In the future, we may consider more relevant approaches to describe the clusters, such as Assali and Zanghi (2006) which automatically builds a metadata hierarchy for a group of websites without the use of any predefined external hierarchies. Then, the element *community* contains these previous elements and provides the optimal number of groups with the attribute *nbGroup*.

Finally, Table 4.6 illustrates the aggregation in the *Graph* element of all

```
< !ELEMENT connectivity EMPTY>
< !ATTLIST connectivity
          g1 CDATA #REQUIRED
          g2 CDATA #REQUIRED
          value CDATA #REQUIRED
          link CDATA #REQUIRED>
< !ELEMENT tag EMPTY>
< !ATTLIST tag
          word CDATA #REQUIRED
          value CDATA #REQUIRED>
< !ELEMENT group (tag+) >
< !ATTLIST group
          id CDATA #REQUIRED
          ratio CDATA #REQUIRED>
< !ELEMENT community (group+,connectivity+)>
< !ATTLIST community
          nbGroup CDATA #REQUIRED>
```

Table 4.5 – *DTD for the MixNet elements.*

the previous elements. Last but not least, the *ConstellationAnswer* element contains *Graph* and *Answer* which is the response from the *Exalead Web Search API* (see Section 4.4.1).

```
< !ELEMENT Graph(community,node+,edge+)>
< !ELEMENT ConstellationAnswer(Graph,Answer)>
```

Table 4.6 – *DTD for the final response*

4.5 *Constellations* Front-end

This section describes a visual solution that allows to explore the data previously built by the remotely-located back-end program. A lot of efforts were put in the graphical representations in order to explore the data and offer the end user a smooth experience.

Layout solution. Due to the underlying nature of the data, we propose to explore them with a graph based approach where the placement of both pages and hyperlinks are given by the Fruchterman-Reingold algorithm. As illustrated in Figure 4.5, the node size depends on the ranking of its Url according to Section 4.3.3 and the color node depends on the MixNet group.

To show that *Constellations* works similar to a conventional search engine, we decide to highlight the query box (and its refines such the language or the type of site, or etc.) at top center of the frame.

Recognizing that this type of interface may destabilize the user accustomed to a search results as a list, we also propose such an openable list on the right side of the frame. A mouseover on a hit of this list or directly on a node, opens an information box of the associated Url. This behavior

can be observed on Figure 4.6. Here, title, thumbnail and snippet of the hit are shown to the user.

To allow a richer user experience, we allow to move and zoom in/out in the graph using both the keyboard and the navigation box at top left of the frame.

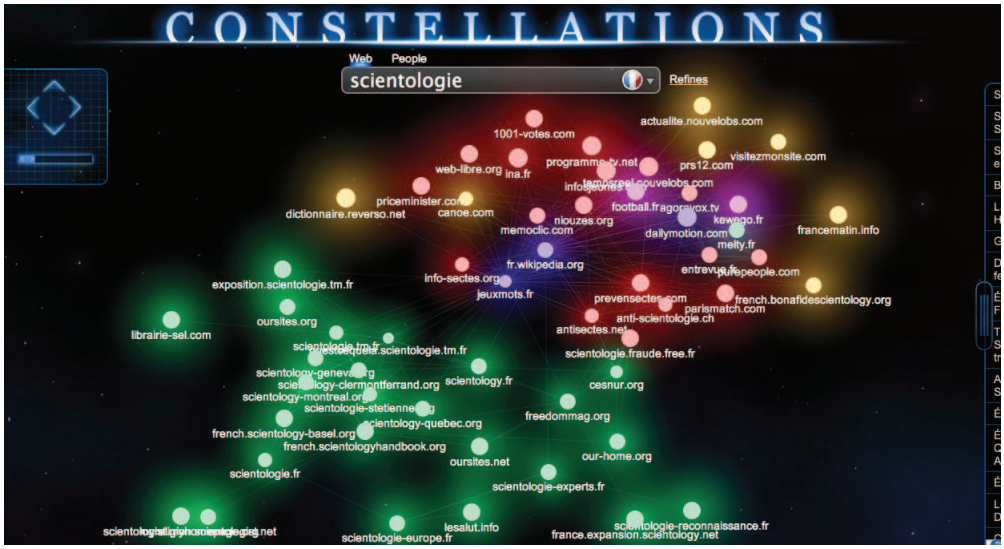


Figure 4.5 – *View of the Constellations application after a query.*

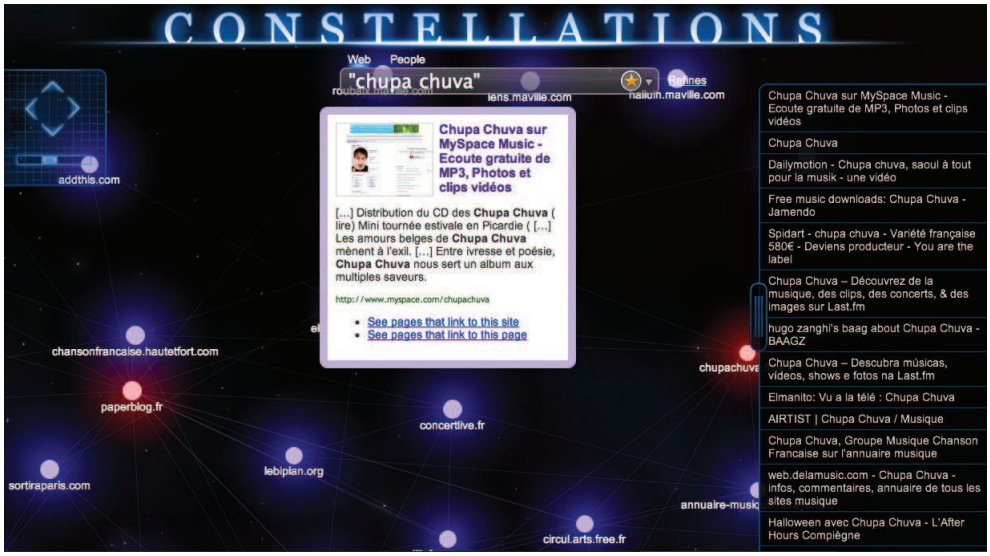


Figure 4.6 – *View of the hits list and an information box for a hit.*

Technological solution. For accessibility reason *Adobe Flash* was avoided in favor of semantically correct HTML (the markup is a simple list of elements). Unfortunately, since HTML has not been created with rendering colorful graphs as "star constellations" this required a bit of *CSS* trickery and *Javascript* behaviors.

Results are displayed as discs composed of the four rounded corners of the collapsed element, allowing them to be gracefully opened from there and positioned in a absolute way. Edges of the graphs are drawn using

the HTML canvas element, as well as the glow behind each results. In order to be able to zoom and avoid aliasing of the canvas element and avoid obvious performance problems when scaling tenfold a one megapixel image, the canvas element only draws the visible part of the graph. It has to be redrawn on every user action such as panning and zooming, but after a lot of different experiments it was the best trade-off. Figure 4.7 illustrates this rendering solution.

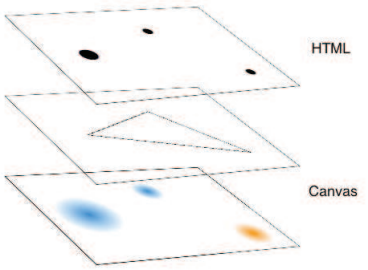


Figure 4.7 – *Technological solution of the Constellations rendering.*

A much simpler version of the rendering has also been developed using SVG which allows a quick and painless development but for which the performance is a real problem in some browsers. In fact, the Webkit based browsers smoothly render the SVG graph while the Gecko based browsers are slow and laggy. To circumvent this problem, we use the *SVGWeb Library* which can automatically replace SVG content with a flash element. Such a render perfectly works for every browsers (including IE).

4.6 Services derived from *Constellations*

Actually, any collection of entities with references can be adapted to the *Constellations* service. We briefly describe approaches and datasets that have been tested and could be further studied in short terms.

Named Entity. Recognition of such entities allows to assign elements in text into predefined categories such as the names of persons, organizations, locations, etc. Considering that one can extract these information from a set of Web pages, one can define a threshold of co-occurring entities in order to detect those which are more related to each other. There by, we consider an edge between two entities if and only if they co-occur in the set of documents more than the threshold. To illustrate this approach, we develop a demo based on the *Wikipedia* dataset, available on the "People tab of the online *Constellations* service.

Scholar. Another dataset studied consists in a corpus of 15 million bibliographic records of documents held in the INIST/CNRS collections and covering all fields of worldwide research in Science, Technology, Medicine, Humanities and Social Sciences. Here, we propose to draw a co-authorships graph with a similar approach to the previous one. An

edge between two authors is added if and only if they have written a minimal number of common papers in the returned set of papers after a given request.

Chapter Conclusion

This chapter has described an online application, called *Constellations*, which offers a new way to browse the web search results. Using the connectivity information to detect ambiguity or different points of views after a search request, one can easily test it at `http://constellations.labs.exalead.com/`. A large part of this chapter is dedicated to the Web and some of its properties. Since, its size is one of the most difficult problematics to build an application, the key points of the used technology are provided. In the short term we plan to investigate how to take advantage of the *CoshMix* models in such an application where the node features could be the text. Besides, we will soon propose, to the researcher community, an API to retrieve the *Constellations* data in a more accessible way.

Conclusion

Analyzing networks has become an essential part of a number of scientific fields. Real networks share some well-studied properties such as small-world phenomenon, power law distribution of the degrees, *etc.* Real networks have also been shown to form groups of vertices and much may be learned from studying their structure or topology. Such study is the main topic of this thesis. Many strategies have been developed for this purpose and the *first chapter* provides a general description of the approaches commonly applied. However, a distinction can be made between model-free (Newman 2006a, Ng et al. 2002) and model-based methods.

The latter has provided an efficient way to summarize complex networks structures. The basic idea of these strategies is to model the distribution of connections in the network, considering that nodes are spread among an unknown number of connectivity classes which are themselves unknown. Although the proposed modeling strategies are diverse (Frank and Harary 1982, Nowicki and Snijders 2001, Handcock et al. 2006, Airoldi et al. 2008, Daudin et al. 2008), EM like algorithms constitute a common core of the estimation strategy (Snijders and Nowicki 1997), and this algorithm is known to be slow to convergence and to be very sensitive to the size of the dataset. The *second chapter* of this thesis proposes original solutions, from a computational point of view, to the statistical analysis of complex and very large networks. The proposed online strategies are the following

- Classification EM algorithm,
- Stochastic Approximation EM algorithm,
- Variational EM algorithm.

The online based EM algorithms, allow the estimation of model parameters within a reasonable computation time for datasets which can be made up of thousands of nodes. These methods constitute a trade-off between the potential amount of data to process and the quality of the estimations: even if online methods are not as precise as batch methods for estimation, they may represent a solution when the size of the network is too large for any existing estimation strategy. Furthermore, our simulation study shows that the quality of the remaining partition is good when using online methods. The structure uncovered by our algorithms is complementary to modules or "communities" which can be detected on large networks with model-free algorithms . This highlights the need for efficient computational strategies to perform model-based clustering on similar networks. The online framework is very flexible, and could be applied to other models such as the block model and the mixed membership

model, as the online framework can be adapted to Bayesian algorithms (Opper 1999).

Sometimes datasets can be modeled with a graph structure embedding vertex features (or additional information on edges). Characterizing each vertex both by its connectivity and by a vector of features (or a matrix information), the CohsMix algorithms proposed in the *third chapter*, use both these elements to cluster the data and estimate the model parameters. Based on a variational approach of EM, simulation and comparison results show our algorithms to be attractive and competitive for various kind of models.

Throughout this thesis, experiments on real datasets are systematically related to the Web graph. Thanks to the collaboration with the *Exalead* search engine, we have succeeded in bypassing technological difficulties and achieving an original corpus studies such as the analysis of the French or US political websites network and query studies with controversial or ambiguous examples. Since the relevance of these experiments were amply demonstrated, we studied the ability to create an online application which could automatically extract, visually explore and reveal the connectivity information induced by the hyperlinks. Thereby, the *fourth chapter* describes *Constellations*, a search engine based service, which allows to browse, in a new way, the results of a given user request. A large part of this chapter is also dedicated to the Web and some of its properties. Since, its size is one of the most difficult problematic to build an application, the key points of the used technology are provided.

Perspectives

Special efforts will be put on models in which data can be modeled with a graph structure embedding vertex features such as *CoshMix*. Continuing to successfully exploit whatever information is found both in the connectivity pattern and in the features, we plan to investigate various areas:

- *Online Approach:* Adapting online estimation strategies to our algorithms
- *Information focusing:* Exploiting one type of information, graph or features, when it becomes predominant.
- *Feature selection:* Identifying a subset of features that are most relevant to cluster the data.
- *Other models:* Providing alternative models to datasets with a graph structure embedding vertex features.

If these developments seems relevant enough, we will integrate them in the new release of *Constellations*. Concerning this service, speed improvements and additional features such as interactivity, captions or map exports are planned in short terms.

Annexes

A

A.1 Some properties of the MixNet Model in the Bernoulli Erdös-Rényi mixture

In this section, we recall the interesting properties of the MixNet model (in the Bernoulli Erdös-Rényi mixture). More details can be found in Daudin et al. (2008).

A.1.1 Distribution of the degrees

Proposition A.1 *Given the label of a vertex, the conditional distribution of the degree of this vertex is Binomial (approximately Poisson);*

$$K_i|\{i \in q\} \sim \mathcal{B}(n-1, \bar{\pi}_q) \approx \mathcal{P}(\lambda_q)$$

where $\bar{\pi}_q = \sum \alpha_l \pi_{ql}$ and $\lambda_q = (n-1)\bar{\pi}_q$

Proof. Conditionally to the belonging of vertices to groups, edges connecting vertex i belonging to group q are independent. The conditional connection probability is:

$$\Pr\{X_{i,j} = 1 | i \in q\} = \sum_l Pr\{X_{i,j} = 1 | i \in q, j \in l\} \Pr\{j \in l\} = \sum \alpha_l \pi_{ql} = \bar{\pi}_q$$

Besides, Poisson distribution can be used as an approximation of the binomial distribution if n is sufficiently large and p is sufficiently small. The result follows. □

A.1.2 Between-group connectivity

Definition A.1 *The connectivity between group q and l is the number of edges connecting a vertex from group q to a group l.*

$$A_{ql} = \sum_i \sum_{j>i} Z_{iq} Z_{jl} X_{ij}$$

A_{qq} is actually the within-connectivity of group q.

Proposition A.2 *The expected connectivity between group q and l is:*

$$\mathbb{E}(A_{ql}) = \frac{n(n-1)\alpha_q \alpha_l \pi_{ql}}{2}$$

Proof. According to Definition A.1 A_{ql} is the sum over $n(n-1)/2$ terms. Conditionally to $\{Z_{iq} Zjl = 1\}$, X_{ij} is a Bernoulli variable with parameter π_{ql}. Thus $\mathbb{E}(Z_{iq} Z_{jl} X_{ij}) = \mathbb{E}(Z_{iq} Z_{jl})\pi_{ql}$. The Z_{iq} are independent, so we have $\mathbb{E}(Z_{iq} Z_{jl}) = \alpha_q \alpha_l$. The result follows.

□

A.2 Complete log-likelihood of MixNet model

The complete log-likelihood of MixNet takes the form:

$$\log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\eta}) = \underbrace{\sum_{i,q} Z_{iq} \alpha_q}_{\log P(\mathbf{Z})} + \underbrace{\sum_{ij,ql} Z_{iq} Z_{jl} \eta_{ql}^t h(X_{ij}) - \sum_{ij,ql} Z_{iq} Z_{jl} a(\eta_{ql}) + \sum_{ij} b(X_{ij})}_{\log P(\mathbf{X}|\mathbf{Z})},$$

where $\alpha_q = \frac{\exp w_q}{\sum_l \exp w_l}$ with $\alpha_1 + \cdots \alpha_Q = 1$. The notations of Equation 2.19 appear expressing the previous likelihood formula as,

$$\begin{aligned} \log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\eta}) &= \sum_{iq} Z_{iq} (w_q - \log \sum_l \exp w_l) + \\ & \sum_{ql} \eta_{ql}^t \underbrace{\sum_{ij} Z_{iq} Z_{jl} h(X_{ij})}_{H_{ql}(\mathbf{X},\mathbf{Z})} - \sum_{ql} a(\eta_{ql}) \underbrace{\sum_{ij} Z_{iq} Z_{jl}}_{G_{ql}(\mathbf{Z})} + \underbrace{\sum_{ij} b(X_{ij})}_{b(\mathbf{X})}. \end{aligned}$$

Let $N_q = \sum_i Z_{iq}$ be the number of nodes in the class q and let $A(\boldsymbol{\beta}) = n \log \sum_l \exp w_l$ be a term which does not depend on the q parameter. Then, Equation can be defined by:

$$\log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\eta}) = \underbrace{\left(\{N_q\}, \{H_{ql}(\mathbf{X}, \mathbf{Z})\}, \{G_{ql}(\mathbf{Z})\}\right)}_{T(\mathbf{X},\mathbf{Z}))} \underbrace{\begin{pmatrix} \{w_q\} \\ \{\eta_{ql}\} \\ \{-a(\eta_{ql})\} \end{pmatrix}}_{\boldsymbol{\beta}^t} + b(\mathbf{X}) - A(\boldsymbol{\beta}).$$

which matches the factorization $\log \Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\beta}) = \boldsymbol{\beta}^t T(\mathbf{X}, \mathbf{Z}) - A(\boldsymbol{\beta}) + B(\mathbf{X})$. Thereby, this proves that the joint distribution $\Pr(\mathbf{X}, \mathbf{Z}; \boldsymbol{\eta})$ also belongs to the exponential family.

A.3 Complete log-likelihood of the Erdős-Rényi Mixture

The complete log-likelihood of the Bernoulli Erdős-Rényi Mixture takes the form:

$$\begin{aligned} \mathcal{L}_c(\mathbf{X}, \mathbf{Z}) &= \underbrace{\sum_i \sum_q Z_{iq} \log \alpha_q}_{\log P(\mathbf{Z})} + \\ & \underbrace{\sum_{ij} \sum_{ql} Z_{iq} Z_{jl} X_{ij} \log \frac{\pi_{ql}}{1 - \pi_{ql}} + \sum_{ij} \sum_{ql} Z_{iq} Z_{jl} \log (1 - \pi_{ql})}_{\log P(\mathbf{X}|\mathbf{Z})}. \end{aligned}$$

If the considered graph is directed, undirected, has or has not self-loops, the complete log-likelihood can be expressed in a relative simple form in function of the $N_q = \sum_i Z_{iq}$, $H_{ql} = \sum_{ij} Z_{iq} Z_{jl} X_{ij}$ and $G_{ql} = N_q N_l$

A.3.1 Directed with self-loops

If the considered graph has self-loops and is directed, we have:

$$\mathcal{L}_c(\mathbf{X},\mathbf{Z}) = \sum_q N_q \log \alpha_q + \sum_{ql} H_{ql} \log \frac{\pi_{ql}}{1-\pi_{ql}} + \sum_{ql} G_{ql} \log(1-\pi_{ql}).$$

To simplify the notation let us use $g(\pi_{ql}) = \log \frac{\pi_{ql}}{1-\pi_{ql}}$ and $h(\pi_{ql}) = \log(1-\pi_{ql})$:

$$\mathcal{L}_c(\mathbf{X},\mathbf{Z}) = \sum_q N_q \log \alpha_q + \sum_{ql} H_{ql} g(\pi_{ql}) + \sum_{ql} G_{ql} h(\pi_{ql})).$$

Deriving according to π_{ql} results in the following estimates:

$$\hat{\pi}_{ql} = \frac{H_{ql}}{G_{ql}}.$$

A.3.2 Directed without self-loops

In order to express, the classification log-likelihood in function of the basic statistics, we express the likelihood as a sum of sums over all indexes $ijql$:

$$\begin{aligned}
\mathcal{L}_c(\mathbf{X},\mathbf{Z}) &= \log P(\mathbf{Z}) + \log P(\mathbf{X}|\mathbf{Z}) \\
&= \sum_q N_q \log \alpha_q + \sum_{i\neq j,ql} Z_{iq} Z_{jl} (X_{ij} g(\pi_{ql}) + h(\pi_{ql})) \\
&\quad + \sum_{iql} Z_{iq} Z_{il} (X_{ii} g(\pi_{ql}) + h(\pi_{ql})) - \sum_{iql} Z_{iq} Z_{il} (X_{ii} g(\pi_{ql}) + h(\pi_{ql})) \\
&= \sum_q N_q \log \alpha_q + \sum_{ij,ql} Z_{iq} Z_{jl} (X_{ij} g(\pi_{ql}) + h(\pi_{ql})) \\
&\quad - \sum_{iql} Z_{iq} Z_{il} (X_{ii} g(\pi_{ql}) + h(\pi_{ql})) \\
&= \sum_q N_q \log \alpha_q + \sum_{ql} H_{ql} g(\pi_{ql}) + \sum_{ql} G_{ql} h(\pi_{ql}) - \sum_q N_q h(\pi_{qq})).
\end{aligned}$$

Deriving according to π_{ql} results in the following estimates:

$$\hat{\pi}_{ql} = \begin{cases} \frac{H_{ql}}{G_{ql}}, \text{if } q \neq l, \\ \frac{H_{qq}}{N_q(N_q-1)}, \text{ otherwise.} \end{cases}$$

A.3.3 Undirected without self-loops

We consider the same kind of computation and try to express the likelihood as a sum of sums over all indexes $ijql$. First decompose the sum of two term (one over $q > l$ and one over $q = q$) and then take into account the fact that $\pi_{ql} = \pi_{lq}$. This leads to:

$$\begin{aligned}
\mathcal{L}_c(\mathbf{X},\mathbf{Z}) &= \log P(\mathbf{Z}) + \log P(\mathbf{X}|\mathbf{Z}) \\
&= \sum_q N_q \log \alpha_q + \sum_{i>j,ql} Z_{iq} Z_{jl} (X_{ij} g(\pi_{ql}) + h(\pi_{ql})) \\
&= \sum_q N_q \log \alpha_q + \sum_{q>l} H_{ql} g(\pi_{ql}) + \sum_{q>l} G_{ql} h(\pi_{ql}) \\
&\quad + \sum_q H_{qq} g(\pi_{qq}) + \frac{1}{2} \left(\sum_q N_q^2 h(\pi_{qq}) - \sum_q N_q h(\pi_{qq}) \right).
\end{aligned}$$

Deriving according to π_{ql} results in the following estimates:

$$\hat{\pi}_{ql} = \begin{cases} \frac{H_{ql}}{G_{ql}}, \text{if } q \neq l, \\ \frac{2H_{qq}}{N_q(N_q-1)}, \text{ otherwise.} \end{cases}$$

A.3.4 Parameters update in the Bernoulli and Poisson cases for the online SAEM

The estimator becomes:

$$\pi_{ql}^{[n+1]} = \gamma_{ql}^{[n+1]}\pi_{ql}^{[n]} + (1-\gamma_{ql}^{[n+1]})\frac{\xi_{ql}^{[n+1]}}{\zeta_{ql}^{[n+1]}},$$

where

$$\begin{aligned} \gamma_{ql}^{[n+1]} &= \frac{N_q^{[n]}N_l^{[n]}}{N_q^{[n]}N_l^{[n]} + Z_{n+1,q}N_l^{[n]} + Z_{n+1,l}N_q^{[n]}}, \\ \xi_{ql}^{[n+1]} &= Z_{n+1,q}\sum_{j=1}^{n} Z_{jl}^{[n]}X_{n+1,j} + Z_{n+1,l}\sum_{i=1}^{n} Z_{iq}^{[n]}X_{i,n+1}, \\ \zeta_{ql}^{[n+1]} &= Z_{n+1,q}N_l^{[n]} + Z_{n+1,l}N_q^{[n]}. \end{aligned}$$

A.3.5 Parameters update in the Bernoulli and the Poisson cases for the online variational algorithm

We get the following update equation:

$$\pi_{ql}^{[n+1]} = \gamma_{ql}^{[n+1]}\pi_{ql}^{[n]} + (1-\gamma_{ql}^{[n+1]})\frac{\mathbb{E}_{\mathcal{R}^{[n]}}\left(\xi_{ql}^{[n+1]}\right)}{\mathbb{E}_{\mathcal{R}^{[n]}}\left(\zeta_{ql}^{[n+1]}\right)},$$

where

$$\begin{aligned} \gamma_{ql}^{[n+1]} &= \frac{\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n]}\right)\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_l^{[n]}\right)}{\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n]}\right)\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_l^{[n]}\right) + \tau_{n+1,q}\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_l^{[n]}\right) + \tau_{n+1,l}\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n]}\right)}, \\ \mathbb{E}_{\mathcal{R}^{[n]}}\left(\xi_{ql}^{[n+1]}\right) &= \tau_{n+1,q}\sum_{j=1}^{n}\tau_{jl}^{[n]}X_{n+1,j} + \tau_{n+1,l}\sum_{i=1}^{n}\tau_{iq}^{[n]}X_{i,n+1}, \\ \mathbb{E}_{\mathcal{R}^{[n]}}\left(\zeta_{ql}^{[n+1]}\right) &= \tau_{n+1,q}\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_l^{[n]}\right) + \tau_{n+1,l}\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n]}\right), \end{aligned}$$

with

$$\mathbb{E}_{\mathcal{R}^{[n]}}\left(N_q^{[n]}\right) = \sum_{i=1}^{n}\tau_{iq}^{[n]}.$$

A.4 More experiments using simulation

In the experiments of Zanghi et al. (2008), we assume that edges are Bernoulli distributed. Four affiliation models have been considered (see Table A.1). The difference among the four models is related to their modular structure, which varies from no structure (almost the Erdős-Renyi model) to strong modular structure (low inter-module connectivity and strong intra-module connectivity).

Model	λ	ϵ	Q
1	0.8	0.02	3
2	0.5	0.05	8
3	0.6	0.25	5
4	0.55	0.35	5

Table A.1 – *Parameters of the four affililation models of the experiment. The Q modules are mixed in the same proportion. Each model consider* $n = 1000$ *nodes.*

Given the number of nodes n and the class proportions (α_q), the color of each node is simulated via a multinomial distribution $\mathcal{M}(1, \alpha_1, ..., \alpha_Q)$. Conditionally to the node colors, edges between two nodes of the same class are drawn according to a probability λ and edges between nodes of different colors are drawn according to a probability ϵ.

We have simulated 30 networks for each model and run the online CEM MixNet algorithm to estimate the model parameters. Figure A.1 shows two boxplots for each experiment: one boxplot for ϵ and one for λ. Notice that for the first model, the highly structured one, the estimation is very close to the true parameters and exhibits no variance. The estimation of the second and third models shows a small downward bias and a small variance.

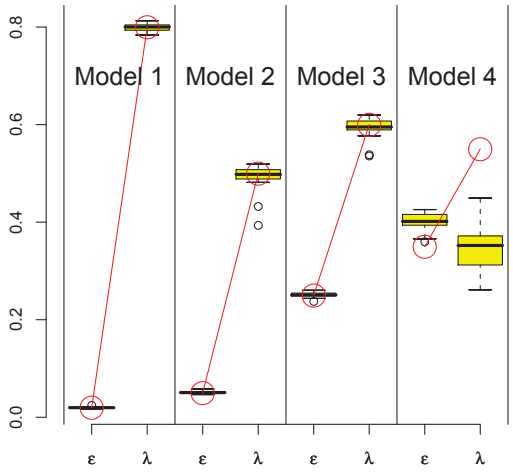


Figure A.1 – *Boxplot of the parameter estimates for 30 estimations of the 4 models. Each model is described by two boxplots, one for the estimations of* ϵ *and the other for the estimations of* λ*. The circles show the true value of the parameters.*

But the fourth model is more difficult to deal with : the algorithm

underestimates λ the probability of within-cluster connection and overestimates ϵ the probablity of between-cluster connection. In summary, the less obvious the structure of the network is, the highest bias we observe in the resulting estimation. Let us also note that in well structured models (except for model 1), the algorithm has a slight tendency to produce biased estimates. This is a phenomenon generally oberved with Classification versions of the EM algorithm. When using Table A.2 to compare the estimates between our online CEM MixNet and MixNet Daudin et al. (2008). we can observe that both estimations are very close. But the online version runs much faster : Although the computationnal complexity of both algorithms is $O(n^2)$ (n being the number of nodes of the network), an important gain of speed can be observed (see Table 2.5).

	MixNet				online MixNet			
Model	$\hat{\lambda}$	σ_λ	$\hat{\epsilon}$	σ_ϵ	$\hat{\lambda}$	σ_λ	$\hat{\epsilon}$	σ_ϵ
1	0.800	0.007	0.020	0.002	0.800	0.007	0.020	0.002
2	0.494	0.024	0.051	0.003	0.493	0.025	0.051	0.003
3	0.593	0.018	0.251	0.005	0.594	0.018	0.251	0.005
4	0.385	0.034	0.390	0.011	0.350	0.046	0.402	0.017

Table A.2 – *Means and standard deviation of the parameter estimates of the four models computed over 30 different runs.*

When considering Table A.3, we can observe that the poor estimation of λ, the probability of within-cluster connection also reveals a small Rand index. This means that the poor estimation of λ makes it impossible to retrieve the modular structure of the network.

	PAM		SC		MixNet		online MixNet	
Model	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}	$\overline{rand}$	σ_{rand}
1	1.000	0.000	0.998	0.008	1.000	0.000	1.000	0.000
2	0.745	0.067	0.988	0.018	0.990	0.027	0.988	0.030
3	0.523	0.111	0.798	0.144	0.967	0.048	0.963	0.055
4	0.083	0.031	0.056	0.043	0.027	0.055	0.045	0.055

Table A.3 – *Means and standard deviation of the Rand index of the four models computed over 30 different runs for graph clustering competitors and MixNet algorithms.*

We also compared the results of the online CEM MixNet algorithm with alternative clustering methods additionnal to the variationnal batch version. We consider as competitors, a basic spectral clustering algorithm Ng et al. (2002), and a k-means like algorithm considering a dissimilarity matrix as input, called PAM (Partitioning Around Medoids).

The spectral clustering algorithm searches for a partition in the space spanned by the eigenvectors of the normalized Laplacian. For the PAM algorithm, we consider a computationally heavy method which builds a dissimilarity matrix based on the shortest paths between all pairs of nodes. Floyd's algorithm is specifically designed to solve this problem. Note that these shortest paths are computed in $O(n^3)$ runtime, thus it does not allow us to use it with huge and dynamic networks.

On the previous generated networks, we have run these algorithms

and have computed the Rand index on each of them. When considering Table A.3, we can observe that both MixNet algorithms always produce the best nodes classification. It is also noticable that PAM and spectral clustering algorithms seem to be rather efficient on obvious models (model 1 & 2), but deteriorate when the network structure becomes weak. The spectral clustering algorithm remains more accurate than the PAM algorithm.

To conclude this section, if the results of MixNet and online CEM MixNet are similar, the latter appears to be the more efficient competitor as far as computational cost is concerned.

A.5 The 2008 U.S. Presidential WebSphere

A.5.1 Connectivity matrix

The connectivity matrix of the rhe 2008 U.S. Presidential WebSphere is given in Table A.8 page 142.

A.5.2 Alexa Traffic

Alexa[1] is a powerful tool used to rank web site traffic. This information comes from the community of Alexa Toolbar users. Each member of the community, in addition to getting a useful tool, is giving back. Simply by using the toolbar each member contributes valuable information about the web, how it is used, what is important and what is not. This information is returned to the community with improved Traffic Rankings and more. Alexa provides API to access the traffic ranking of web sites.

A.6 Lower bounds $\mathcal{J}$ for the CohsMix^{1,2,3} models

The variational approach consists in maximizing a lower bound $\mathcal{J}$ of the log-likelihood $\log \Pr(\mathbf{X}, \mathbf{Y}; \boldsymbol{\Theta})$,

A.6.1 Lower bound $\mathcal{J}$ for CohsMix^1

The detailed expression of the lower bound $\mathcal{J}$ for the CohsMix^1 model can be expressed as

$$\begin{aligned}
\mathcal{J}_{\boldsymbol{\tau}} = \mathbb{E}_{\mathcal{R}(Z)}\{\mathcal{J}(\boldsymbol{\Theta})\} &= \sum_{i=1}^{n}\sum_{q=1}^{Q}\tau_{iq}\log(\alpha_q) + \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}(X_{ij}\log(\pi_{ql}) + (1-X_{ij})\log(1-\pi_{ql})) \\
&\quad - \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}\left[\frac{1}{2}(Y_{ij}-\mu_{ql})^T\sigma^{-1}(Y_{ij}-\mu_{ql})\right] \\
&\quad + \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{Z}\mathcal{R}(Z)\log(\mathcal{R}(Z)).
\end{aligned}$$

[1] http://www.alexa.com/

		Conservative								Liberal										
	C17	C1	C2	C10	C9	C14	C16	C18	C19	C11	C4	C6	C7	C8	C12	C13	C20	C3	C5	C15
C17	.	54	.	.	64	.	52	66	.	62	.	.	67	67	62	43	100	.	.	.
C1	54	.	.	.	.	54	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C10	.	.	.	58	.	.	.	47	.	.	.	.	.	.	.	.	.	.	.	.
C9	64	.	.	.	.	56	61	.	40	.	.	.	.	.	.	.	.	.	.	.
C14	.	54	.	.	56	.	55	72	40	.	.	.	.	.	.	.	.	.	.	.
C16	52	.	.	.	61	55	73	60	56	.	.	.	.	.	.	.	.	.	.	.
C18	66	.	.	47	.	72	60	58	.	.	.	.	.	.	.	.	.	.	.	.
C19	.	.	.	.	40	40	56	.	57	.	.	.	.	.	.	.	.	.	.	.
C11	62	.	.	.	.	.	.	.	.	.	.	.	.	47	.	.	40	.	.	.
C4	.	.	.	.	.	.	.	.	.	.	65	.	.	.	.	42	40	.	.	.
C6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	88	99	.	.	.
C7	67	.	.	.	.	.	.	.	.	.	.	.	47	76	49	.	76	.	.	.
C8	67	.	.	.	.	.	.	.	.	47	.	.	76	90	74	81	98	.	.	.
C12	62	.	.	.	.	.	.	.	.	.	.	.	49	74	45	92	98	.	.	.
C13	43	.	.	.	.	.	.	.	.	.	42	88	.	81	92	95	99	.	.	.
C20	100	.	.	.	.	.	.	.	.	40	42	99	76	98	98	99	100	.	.	.
C3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C15	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
N_q	4	214	407	66	56	1	24	19	36	26	58	207	51	20	37	23	3	81	514	23
$\bar{d}$	649	86	28	149	69	455	335	167	172	192	64	66	66	310	154	170	324	77	18	227

Table A.8 – *Estimated π (in percentage) and number of nodes in each cluster for the US political websphere. $\bar{d}$ represents the estimated mean degree of each group.*

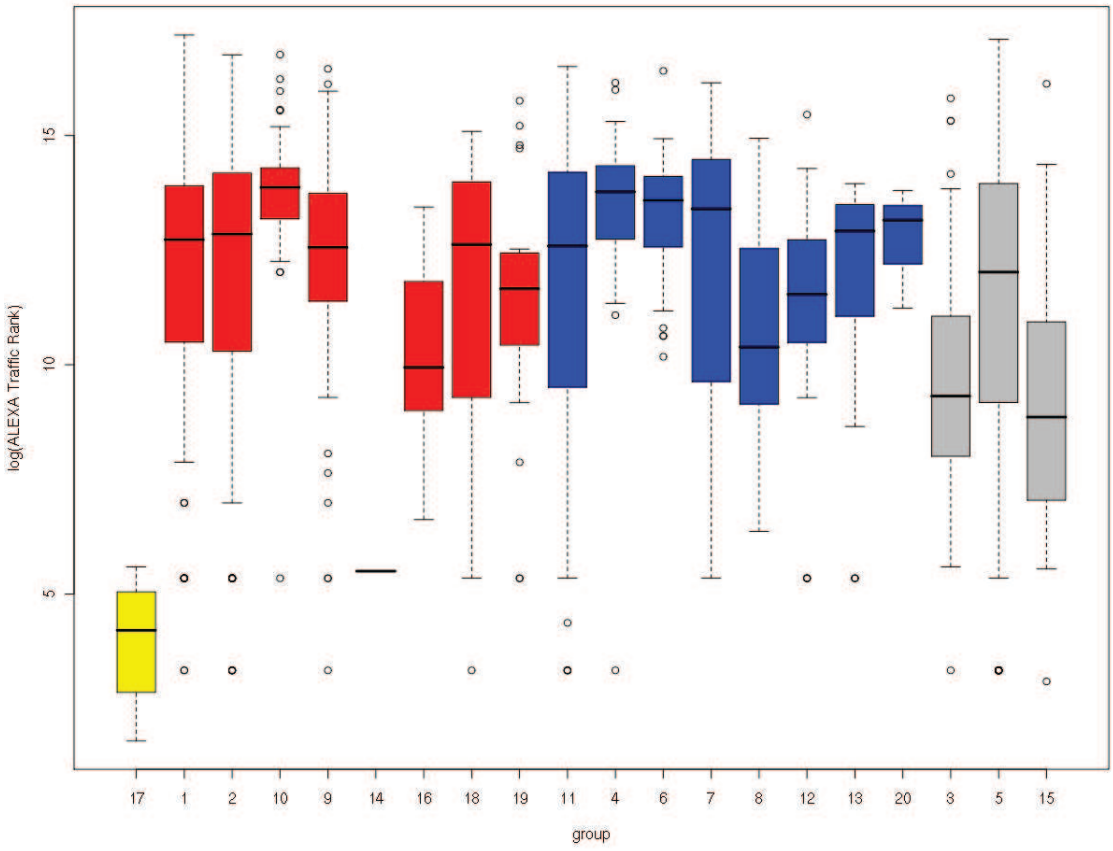


Figure A.2 – *Boxplot of MixNet classes traffic (in log). The lower the Alexa ranking number the more heavily visited the site.*

A.6.2 Lower bound $\mathcal{J}$ for CohsMix^2

The detailled expression of the lower bound $\mathcal{J}$ for the CohsMix^2 model can be expressed as

$$\begin{aligned}\mathcal{J}_{\boldsymbol{\tau}} = \mathbb{E}_{\mathcal{R}(Z)}\{\mathcal{J}(\boldsymbol{\Theta})\} &= \sum_{i=1}^{n}\sum_{q=1}^{Q}\tau_{iq}\log(\alpha_q) + \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}(X_{ij}\log(\pi_{ql}) + (1-X_{ij})\log(1-\pi_{ql})) \\ &+ \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}\left[X_{ij}\left(-\frac{(Y_{ij}-\mu_{ql})^2}{2\sigma^2} + \frac{(Y_{ij}-\tilde{\mu}_{ql})^2}{2\sigma^2}\right) - \frac{(Y_{ij}-\tilde{\mu}_{ql})^2}{2\sigma^2}\right] \\ &+ \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}\log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \sum_{Z}\mathcal{R}(Z)\log(\mathcal{R}(Z)).\end{aligned}$$

A.6.3 Lower bound $\mathcal{J}$ for CohsMix^3

The detailed expression of the lower bound $\mathcal{J}$ for the CohsMix^3 model can be expressed as

$$\begin{aligned}\mathcal{J}_{\boldsymbol{\tau}} = \mathbb{E}_{\mathcal{R}(Z)}\{\mathcal{J}(\boldsymbol{\Theta})\} &= \sum_{i=1}^{n}\sum_{q=1}^{Q}\tau_{iq}\log(\alpha_q) + \sum_{\substack{i,j\\ i\neq j}}\sum_{q,l}\tau_{iq}\tau_{jl}(X_{ij}\log(\pi_{ql}) + (1-X_{ij})\log(1-\pi_{ql})) \\ &+ \sum_{i}\sum_{q}\tau_{iq}\left[\left(\log\frac{1}{2\pi^{\frac{n}{2}}det(\Sigma)^{\frac{1}{2}}}\right) - \frac{1}{2}(\mathbf{Y}_i-\mu_q)^T\sigma^{-1}(\mathbf{Y}_i-\mu_q)\right] \\ &- \sum_{Z}\mathcal{R}(Z)\log(\mathcal{R}(Z)).\end{aligned}$$

A.7 Optimal parameters for all CohsMix^{1,2,3} models

A.7.1 Commun optimal parameters for CohsMix^{1,2,3} models

Proof. Proof of Proposition 3.1 Maximizing the lower bound $\mathcal{J}$ according to parameters α_q and π_{ql} , allows to find the optimal parameters $\hat{\alpha}_q$ and $\hat{\pi}_{ql}$.

Estimation of π_{ql}. Deriving according to π_{ql} leads to

$$\frac{\partial\mathcal{J}_{\boldsymbol{\tau}}}{\partial\pi_{ql}} = \sum_{\substack{i,j\\ i\neq j}}\tau_{iq}\tau_{jl}(\frac{X_{ij}}{\pi_{ql}} + \frac{-(1-X_{ij})}{1-\pi_{ql}})$$

The optimal value of π_{ql} is obtained when we have a derivative at zero

$$\frac{\partial\mathcal{J}_{\boldsymbol{\tau}}}{\partial\pi_{ql}} = 0 \quad\Leftrightarrow\quad \sum_{\substack{i,j\\ i\neq j}}\tau_{iq}\tau_{jl}\pi_{ql} = \sum_{\substack{i,j\\ i\neq j}}\tau_{iq}\tau_{jl}X_{ij}$$

Therefore, the optimal parameter of π_{ql} satisfies the following relation:

$$\hat{\pi}_{ql} = \frac{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}X_{ij}}{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}}$$

Estimation of α_q. In this case, deriving according to α_q is not sufficient because an additional constraint have to be considered. Indeed, $\sum\limits_q \alpha_q$ must be equal to 1. Thus, considering the Lagrangian λ, we can write

$$\frac{\partial\left(\mathcal{J}_{\boldsymbol{\tau}} - \lambda(\sum\limits_q \alpha_q - 1)\right)}{\partial \alpha_q} = \sum_{i=1}^{n} \frac{\tau_{iq}}{\alpha_q} - \lambda$$

The optimal value of α_q is obtained when this derivative is equal to zero.

$$\frac{\partial(\mathcal{J}_{\boldsymbol{\tau}} - \lambda(\sum\limits_q \alpha_q - 1))}{\partial \alpha_q} = 0 \quad \Leftrightarrow \quad \alpha_q = \sum_{i=1}^{n} \frac{\tau_{iq}}{\lambda}$$

We know that the constraint leads to $\sum\limits_q \alpha_q = 1$, thus $\sum\limits_{q=1}^{Q}\sum\limits_{i=1}^{n} \frac{\tau_{iq}}{\lambda} = 1$

Finally we obtain, $\lambda = \sum\limits_{q=1}^{Q}\sum\limits_{i=1}^{n} \tau_{iq} = n$

The optimal parameter of α_q satisfies the following relation:

$$\hat{\alpha}_q = \frac{\sum\limits_{i=1}^{n} \tau_{iq}}{n}$$

□

A.7.2 Optimal parameters for CohsMix^1 model

Proof. Proof of Proposition 3.2

Estimation of μ_{ql}. In this case, we have to derive $\mathcal{J}_{\boldsymbol{\tau}}$ according to μ_{ql} which only takes place in the normal distribution term. Then, we can write:

$$\frac{\partial \mathcal{J}_{\boldsymbol{\tau}}}{\partial \mu_{ql}} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}\frac{(Y_{ij} - \mu_{ql})}{2\sigma^2}$$

The following relation leads to a derivative at zero:

$$\frac{\partial \mathcal{J}_{\boldsymbol{\tau}}}{\partial \mu_{ql}} = 0 \quad \Leftrightarrow \quad \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}\mu_{ql} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}Y_{ij}$$

Thus, the optimal estimator of μ_{ql} is defined by:

$$\hat{\mu}_{ql} = \frac{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}Y_{ij}}{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}}$$

Estimation of σ^2 Once again, parameter σ^2 only takes place in the normal distribution. Derivating $\mathcal{J}_\tau$ according to this parameter leads to:

$$\frac{\partial \mathcal{J}_\tau}{\partial \sigma^2} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}\left[\frac{-(Y_{ij} - \hat{\mu}_{ql})^2}{\sigma^4} + \frac{1}{2\sigma^2}\right]$$

The optimal value ofσ^2 is obtained when this derivative is equal to zero.

$$\frac{\partial \mathcal{J}_\tau}{\partial \sigma^2} = 0 \quad \Leftrightarrow$$

$$\sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}\frac{(Y_{ij} - \hat{\mu}_{ql})^2}{\sigma^4} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}\frac{1}{\sigma^2}$$

Finally, the optimal estimator of σ^2 can be expressed as:

$$\hat{\sigma}^2 = \frac{\sum\limits_{\substack{i,j \\ i \neq j}}\sum\limits_{q,l} \tau_{iq}\tau_{jl}\left[(Y_{ij} - \hat{\mu}_{ql})^2\right]}{\sum\limits_{\substack{i,j \\ i \neq j}}\sum\limits_{q,l} \tau_{iq}\tau_{jl}}$$

□

A.7.3 Optimal parameters for CohsMix^2 model

Proof. Proof of Proposition 3.4

Estimation of μ_{ql}. In this case, we have to derive $\mathcal{J}_\tau$ according to μ_{ql} which only takes place in the normal distribution term. Then, we can write:

$$\frac{\partial \mathcal{J}_\tau}{\partial \mu_{ql}} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}X_{ij}\frac{(-Y_{ij} + \mu_{ql})}{2\sigma^2}$$

The following relation leads to a derivative at zero :

$$\frac{\partial \mathcal{J}_\tau}{\partial \mu_{ql}} = 0 \quad \Leftrightarrow \quad \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}X_{ij}\mu_{ql} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}Y_{ij}X_{ij}$$

Thus, the optimal estimator of μ_{ql} is defined by:

$$\hat{\mu}_{ql} = \frac{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}X_{ij}Y_{ij}}{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}X_{ij}}$$

Estimation of $\tilde{\mu}_{ql}$ The optimal estimator of $\tilde{\mu}_{ql}$ is obtained with a similar approach to μ_{ql} and verify:

$$\hat{\tilde{\mu}}_{ql} = \frac{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}(1 - X_{ij})Y_{ij}}{\sum\limits_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl}(1 - X_{ij})}$$

Estimation of σ^2 Once again, parameter σ^2 only takes place in the normal distribution. Derivating $\mathcal{J}_{\boldsymbol{\tau}}$ according to this parameter leads to:

$$\frac{\partial \mathcal{J}_{\boldsymbol{\tau}}}{\partial \sigma^2} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl} \left[\frac{X_{ij}\left(-(Y_{ij} - \hat{\mu}_{ql})^2 + (Y_{ij} - \hat{\tilde{\mu}}_{ql})^2\right) - (Y_{ij} - \hat{\tilde{\mu}}_{ql})^2}{\sigma^4} + \frac{1}{2\sigma^2} \right]$$

The optimal value ofσ^2 is obtained when this derivative is equal to zero.

$$\frac{\partial \mathcal{J}_{\boldsymbol{\tau}}}{\partial \sigma^2} = 0 \quad \Leftrightarrow$$

$$\sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl} \frac{X_{ij}\left((Y_{ij} - \hat{\mu}_{ql})^2 + (Y_{ij} + \hat{\tilde{\mu}}_{ql})^2\right) - (y_{ij} + \hat{\tilde{\mu}}_{ql})^2}{\sigma^4} = \sum_{\substack{i,j \\ i \neq j}} \tau_{iq}\tau_{jl} \frac{1}{\sigma^2}$$

Finally, the optimal estimator of σ^2 can be expressed as:

$$\hat{\sigma}^2 = \frac{\sum\limits_{\substack{i,j \\ i \neq j}} \sum\limits_{q,l} \tau_{iq}\tau_{jl}\left[X_{ij}\left((Y_{ij} - \hat{\mu}_{ql})^2 - (Y_{ij} - \hat{\tilde{\mu}}_{ql})^2\right) + (Y_{ij} - \hat{\tilde{\mu}}_{ql})^2\right]}{\sum\limits_{\substack{i,j \\ i \neq j}} \sum\limits_{q,l} \tau_{iq}\tau_{jl}}$$

□

Proof. Proof of Proposition 3.5

Estimation of τ_{iq} We obtain the variational parameter derivating $\mathcal{J}_{\boldsymbol{\tau}}$ according to τ_{iq}. Nevertheless, we have to consider an additional constraint which imposes for each node i, the relation $\sum_{q=1}^{Q} \tau_{iq}$ equal to 1.

Then, we have to solve the following equation:

$$\forall i, q \quad \frac{\partial \left(\mathcal{J}_{\boldsymbol{\tau}} - \sum_{i=1} \lambda_i (\sum_q \tau_{iq} - 1))) \right)}{\partial \tau_{iq}} = 0$$

with a constraint: $\forall i, \sum_q \tau_{iq} = 1$

Thus, we obtain:

$$\begin{aligned} \frac{\partial \left(\mathcal{J}_{\boldsymbol{\tau}} - \sum_{i=1} \lambda_i (\sum_q \tau_{iq} - 1))) \right)}{\partial \tau_{iq}} &= \log(\alpha_q) + \sum_{j \neq i} \sum_l \log \left((\pi_{ql}^{X_{ij}} (1 - \pi_{ql})^{1 - X_{ij}})^{\tau_{jl}^{(m)}} \right) \\ &\quad - log(\tau_{iq}) - 1 + \lambda_i \\ &\quad - \frac{1}{2} \sum_{j \neq i} \sum_l \tau_{jl} \left(\frac{(Y_{ij} - \mu_{ql})^2}{\sigma^2} + \log(2\pi\sigma^2) \right) \end{aligned}$$

This equation is equal to 0 and $\forall i, \sum_q \tau_{iq} = 1$ if, and only if, $\hat{\tau}_{iq}$ satisfies:

$$\hat{\tau}_{iq}^{(m+1)} \propto \alpha_q \prod_{j \neq i} \prod_l \left[\frac{\hat{\pi}_{ql}^{X_{ij}} (1 - \hat{\pi}_{ql})^{1 - X_{ij}}}{\sqrt{2\pi\sigma^2}} exp(\frac{1}{2\sigma^2} \left[X_{ij} \left(-(Y_{ij} - \mu_{ql})^2 + (Y_{ij} - \tilde{\mu}_{ql})^2 \right) - (Y_{ij} - \tilde{\mu}_{ql})^2 \right] \right]^{\tau_{jl}^{(m)}}$$

$\exp(\lambda_i - 1)$ stands for normalization constraint. □

A.8 Software

A.8.1 MixNet and MixeR

MixeR is a *R* wrapper for the ANSI $C++$ software package MixNet which also includes Fortran 77 subroutines from the ARPACK library. Erdös-Rényi Mixture Model for Graph (MixNet), which has been proposed by Daudin et al. (2008) with an associated EM estimation algorithm, is not to be confused with Exponential Random Graph Models for Network Data (ERGM) which consider distributions ensuing from the exponential family to model the edge distribution. The MixNet model allows to capture the structure of a network and in particular to detect communities. There exists a strong connection between MixNet and block clustering. Block clustering searches for homogeneous blocks in a data matrix by simultaneous clustering of rows and columns. The proposed estimation strategies deals with undirected graphs. They are of three type:

- variational: which refers to the paper of Daudin et al. (2008). It is the default method.
- classification: which implements the method described in Zanghi et al. (2008). This method is faster than the variational approach and is able to deal with bigger networks but can produce biased estimates.

- bayesian which implements the method described in Latouche et al. (2008). It improves over variational and classification strategies when dealing with small networks (less than 50 nodes). It should not be used with networks having more than 100 nodes.

The two softwares can be downloaded from the Web quiet easily:

- MixNet: `http://stat.genopole.cnrs.fr/software/mixnet/`
- MixeR: `http://cran.r-project.org/web/packages/mixer/`

Compilation and installation are compliant with the GNU standard procedure. Online documentation and man pages are also available. MixNet is licensed under the GNU General Public License.

A.8.2 CohsMix

An R package called *CohsMix* for Covariates in Hidden Structure Using Mixture models provides variational EM algorithms for the three proposed CohsMix[1], 2, 3 models. Theses algorithms allow to detect hidden structure and estimate the model parameters. Generally we can distinguish three features:

- **Simulation:** Simulates data according to the desired model parameters: number of nodes, class proportions, connectivity and covariates values.
- **Estimation:** Estimates the parameters and the hidden structure. If the number of classes of the network is unknown, the ICL algorithm can also be solicited by the estimation function. If the latter is used, the parameter estimates would be provided for all classes. Therefore, the choice of number of classes is recommended but not required.
- **Representation:** Plots the network with the estimated structure. Given the number of group strategy, a representation of the ICL evolution is available.

Yet, *CohsMix* and MixNet are not merged, moreover it may change in the future. *CohsMix R* package is available upon request.

Bibliography

L.A. Adamic and N. Glance. The political blogosphere and the 2004 US election: divided they blog. *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005. (Cited pages 71 and 73.)

M. Adler and M. Mitzenmacher. Towards compressing web graphs. In *Proc. of the IEEE Data Compression Conference (DCC)*, pages 203–212, 2001. (Cited page 117.)

E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E.P. Xing. Mixed-membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008. (Cited pages 12, 34, 39, 40, 48, 49, and 131.)

Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002. (Cited pages 108 and 109.)

E.S. Allman, C. Matias, and J.A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *Ann. Statist*, 37 (6A):3099–3132, 2009. (Cited page 47.)

LAN Amaral, A. Scala, M. Barthelemy, and HE Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States of America*, 97(21):11149, 2000. (Cited page 16.)

C. Ambroise, M. Dang, and G. Govaert. Clustering of spatial data by the EM algorithm. *geoENV I-Geostatistics for Environmental Applications*, 9: 493–504, 1997. (Cited pages 82 and 95.)

Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121(1-3):15–26, 2002. (Cited page 23.)

A.A. Assali and H. Zanghi. Automated Metadata Hierarchy Derivation. *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, 1, 2006. (Cited page 126.)

Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the DUST: different URLs with similar text. In *In Proc. 15th WWW*, pages 1015–1016. ACM New York, NY, USA, 2006. (Cited page 107.)

A.L. Barabasi and RE Crandall. Linked: The new science of networks. *American journal of Physics*, 71:409, 2003. (Cited page 16.)

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. 2009. URL `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`. (Cited page 72.)

E. Behrends. *Introduction to Markov chains: with special emphasis on rapid mixing*. Friedrick Vieweg & Son, October 1999. ISBN 3528069864. (Cited page 20.)

R. Bekkerman, S. Zilberstein, and J. Allan. Web Page Clustering using Heuristic Search in the Web Graph. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2280–2285, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. (Cited page 98.)

M.K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1):07–01, 2001. (Cited page 111.)

Tim Berners-Lee. Information management : A proposal. Technical report, CERN, 1989. (Cited page 105.)

Tim Berners-Lee and Robert Cailliau. Worldwideweb : Proposal for a hypertext project. Technical report, CERN, 1990. (Cited page 105.)

P. Bertin and F. Bourdoncle. Searching tool and process for unified search using categories and keywords, February 27 2002. EP Patent 1,182,581. (Cited page 97.)

K. Bharat and A. Broder. Mirror, mirror on the web: A study of host pairs with replicated content. *Computer Networks-the International Journal of Computer and Telecommunications Networkin*, 31(11):1579–1590, 1999. (Cited page 119.)

K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the web. *Computer networks and ISDN Systems*, 30(1-7):469–477, 1998. (Cited pages 103 and 117.)

K. Bharat, B.W. Chang, M. Henzinger, and M. Ruhl. Who links to whom: Mining linkage between web sites. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, volume 51, page 58. IEEE Computer Society Washington, DC, USA, 2001. (Cited page 106.)

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE PAMI*, 22(7): 719–725, 2000. (Cited pages 61 and 91.)

V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008. (Cited pages 31 and 72.)

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006. (Cited page 15.)

P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pages 595–602. ACM New York, NY, USA, 2004. (Cited pages 103, 116, and 117.)

P. Boldi, B. Codenotti, M. Santini, and S. Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice and Experience*, 34(8): 711–726, 2004. (Cited page 111.)

I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo. The maximum clique problem. *Handbook of combinatorial optimization*, 4(1):1–74, 1999. (Cited page 22.)

S.P. Borgatti, M.G. Everett, and P.R. Shirey. LS sets, lambda sets and other cohesive subsets. *Social Networks*, 12(4):337–357, 1990. (Cited page 23.)

R. Boulet, B. Jouve, F. Rossi, and N. Villa. Batch kernel SOM and related Laplacian methods for social network analysis. *Neurocomputing*, 71(7-9): 1257–1273, 2008. (Cited page 12.)

F. Bourdoncle. Livetopics: Recherche visuelle d'information sur l'internet. *Dossiers de l'Audiovisuel, La Documentation Francaise*, 74:36–38, 1997. (Cited page 121.)

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. (Cited pages 111 and 112.)

A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33(1-6):309–320, 2000. (Cited pages 103, 106, 107, 108, 109, and 165.)

C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973. (Cited page 22.)

Vannevar Bush. As we may think. *Atlantic Monthly*, 1, July 1945. URL `http://www.ps.uni-saarland.de/~duchier/pub/vbush/vbush-all.shtml`. (Cited page 104.)

G. Celeux and J. Diebolt. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2(1):73–82, 1985. (Cited page 46.)

G. Celeux and J. Diebolt. A stochastic approximation type EM algorithm for the mixture problem. Research Report, 1991. (Cited page 46.)

G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational statistics and data analysis*, 14 (3):315–332, 1992. (Cited pages 46 and 53.)

G. Celeux, D. Chauveau, and J. Diebolt. On stochastic versions of the EM algorithm. Technical report, 1995. (Cited page 46.)

Soumen Chakrabarti. *Mining the web*. Morgan Kaufmann Publishers, 2003. ISBN ISBN 1-55860-754-4. (Cited page 106.)

Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *In Proceedings of the Twenty-sixth International Conference on Very Large Databases*, 2000. (Cited page 112.)

F.R.K. Chung. *Spectral graph theory*. Amer Mathematical Society, 1997. (Cited pages 18 and 19.)

A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70, 2004. (Cited page 31.)

T.F. Coleman and J.J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Mathematical Programming*, 28(3):243–270, 1984. (Cited page 17.)

D. Cook and L. Holder, editors. *Mining Graph Data*. Wiley-Interscience, 2007. (Cited page 81.)

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. McGraw-Hill Science / Engineering / Math, 2nd edition, 2001. (Cited page 20.)

J.J. Daudin, F. Picard, and S. Robin. A mixture model for random graph. *Statistics and computing*, 18(2):1–36, 2008. (Cited pages 12, 14, 24, 34, 35, 39, 40, 49, 52, 61, 63, 71, 81, 88, 93, 131, 135, 140, 148, 162, and 166.)

Brian D. Davison. Topical locality in the web. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–279, New York, NY, USA, 2000. ACM. ISBN 1-58113-226-3. doi: http://doi.acm.org/10.1145/345508.345597. (Cited pages 83, 97, 101, and 103.)

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1327452.1327492. (Cited pages 103 and 114.)

B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the em algorithm. *The Annals of Statistics*, 27: 94–128, 1999. (Cited pages 40, 46, 52, and 57.)

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–39, 1977. (Cited pages 43 and 88.)

L. Denoyer, H. Zaragoza, and P. Gallinari. HMM-based passage models for document classification and ranking. In *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*. Citeseer, 2001. (Cited page 97.)

D. Donato, L. Laura, S. Leonardi, and S. Millozzi. The Web as a graph: How far we are. *ACM Transactions on Internet Technology (TOIT)*, 7(1):4, 2007. (Cited page 106.)

S. Dongen. Performance criteria for graph clustering and Markov cluster experiments. Technical report, Amsterdam, The Netherlands, The Netherlands, 2000. (Cited pages 12 and 33.)

T. Drugeon. A technical approach for the french web legal deposit. *5th International Web Archiving Workshop (IWAW05)*, 2005. (Cited pages 72 and 111.)

JC Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Cybernetics and Systems*, 3(3):32–57, 1973. (Cited page 27.)

C. Dyer, A. Cordova, A. Mont, and J. Lin. Fast, easy, and cheap: Construction of statistical machine translation models with MapReduce. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 199–207. Association for Computational Linguistics, 2008. (Cited page 115.)

P. Elias, A. Feinstein, and C. Shannon. A note on the maximum flow through a network. *IEEE Transactions on Information Theory*, 2(4):117–119, 1956. (Cited page 25.)

I.J. Farkas, I. Derenyi, A.L. Barabasi, and T. Vicsek. Spectra of "real-world" graphs: Beyond the semicircle law. *Physical Review E*, 64(2):26704, 2001. (Cited page 19.)

U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001. (Cited page 23.)

M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973. (Cited page 31.)

RT Fielding. Representational state transfer (REST). *Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine*, 2000. (Cited page 120.)

S. Fortunato. Community detection in graphs. *arXiv*, 906, 2009. (Cited pages 12 and 35.)

S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36, 2007. (Cited page 31.)

G. Fouetillou. Le web et le traité constitutionnel européen, écologie d'une localité thématique. *Réseaux*, pages 279–304, 2007. (Cited page 72.)

Ove Frank and Frank Harary. Cluster inference by using transitivity indices in empirical graphs. *Journal of the American Statistical Association*, 77(380):835–840, 1982. ISSN 0162-1459. (Cited pages 81 and 131.)

N. Friedman. The Bayesian structural EM algorithm. In *Proc. UAI*, volume 98. Citeseer, 1998. (Cited page 46.)

T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. (Cited page 123.)

M.R. Garey and D.S. Johnson. *Computers and intractability*. Freeman San Francisco, 1979. (Cited pages 23 and 26.)

F. Ghitalla, D. BOULLIER, P. GKOUSKOU, L. LE DOUARIN, and A. Neau. *L'outre-lecture: manipuler,(s') approprier, interpréter le Web*. Bibliothèque publique d'information Centre Pompidou, 2003. (Cited page 103.)

F. Ghitalla, E. Diemert, C. Maussang, and F. Pfaender. Tarente: an experimental tool for extracting and exploring web aggregates. In *proceedings of IEEE conference ICCTA, Damas, Syrie*, 2004. (Cited page 110.)

M. Girvan and MEJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821, 2002. (Cited pages 11, 12, 28, and 30.)

G. Grimmett and D. Stirzaker. *Probability and random processes*. Oxford University Press, USA, 2001. (Cited page 20.)

R. Guimera and L.A.N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005. (Cited page 31.)

Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, page 587. VLDB Endowment, 2004. (Cited page 114.)

E.R. Hancock and J. Kittler. Discrete relaxation. *Pattern Recognition*, 23(7): 711–733, 1990. (Cited page 56.)

MS. Handcock, A. Raftery, and J.M. Tantrum. Model based clustering for social networks. *JRSS A*, 2006. (Cited pages 12, 34, 39, 49, and 131.)

X. He, H. Zha, C. HQ Ding, and H. D. Simon. Web document clustering using hyperlink structures. *Computational Statistics and Data Analysis*, 41 (1):19–45, 2002. (Cited page 98.)

M.A. Hearst and J.O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, 1996. (Cited page 97.)

J. Hendler, N. Shadbolt, W. Hall, T. Berners-Lee, and D. Weitzner. Web science: an interdisciplinary approach to understanding the web. *Communications of the ACM*, 51(7), 2008. ISSN 0001-0782. (Cited page 106.)

A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, 1999. (Cited page 111.)

P.D. Hoff. Random Effects Models for Network Data. *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*, pages 303–312, 2003. (Cited page 82.)

J.M. Hofman and C.H. Wiggins. Bayesian approach to network modularity. *Physical review letters*, 100(25):258701, 2008. (Cited page 31.)

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2: 193–218, 1985. (Cited page 63.)

M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, 1999. (Cited pages 60 and 88.)

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989. (Cited pages 16 and 162.)

B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970. (Cited page 26.)

J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999. (Cited pages 72, 101, 103, 110, and 114.)

J.M. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A.S. Tomkins. The web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, pages 1–17, 1999. (Cited page 15.)

A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure of complex networks. *New Journal of Physics*, 11(033015):033015, 2009. (Cited page 31.)

P. Latouche, E. Birmele, and C. Ambroise. Bayesian methods for graph clustering. Technical Report 17, SSB, 2008. (Cited pages 40 and 149.)

P. Latouche, E. Birmelé, and C. Ambroise. Overlapping stochastic block models. *preprint SSB*, 2009. (Cited pages 15 and 25.)

A. Likas. A reinforcement learning approach to online clustering. *Neural computation*, 11(8):1915–1932, 1999. (Cited page 54.)

Z. Liu, J. Almhana, V. Choulakian, and R. McGorman. Online em algorithm for mixture with application to internet traffic modeling. *Computational Statistics & Data Analysis*, 50(4):1052–1071, February 2006. (Cited page 39.)

S.P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. (Cited page 27.)

F. Lorrain and H.C. White. Structural equivalence of individuals in social networks. *Social Networks: A Developing Paradigm*, 1:67, 1977. (Cited page 25.)

L. Lovasz. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2:1–46, 1993. (Cited page 19.)

R.D. Luce. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15(2):169–190, 1950. (Cited page 22.)

R.D. Luce and A.D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949. (Cited page 22.)

J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, 1:281–296, 1967. (Cited page 39.)

J.B. MacQueen et al. Some methods for classification and analysis of multivariate observations, 1966. (Cited page 27.)

M. Mariadassou and S. Robin. Uncovering latent structure in valued graphs: a variational approach. Technical Report 10, SSB, october 2007. (Cited pages 49 and 89.)

Michael Mauldin and John Leavitt. Web agent related research at the center for machine translation. In *Proc. 11th AAAI Conference*, pages 91–99, 1994. (Cited page 110.)

G.J. McLachlan and K.E. Basford. *Mixture models. Inference and applications to clustering*. Marcel Dekker, New York, 1988. (Cited page 41.)

G.J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley New York, 1997. (Cited page 46.)

G.J. McLachlan and D. Peel. *Finite mixture models*. Wiley-Interscience, 2000. (Cited page 40.)

R.J. Mokken. *A theory and procedure of scale analysis with applications in political research*. Mouton, 1971. (Cited page 23.)

B.H. Murray and A. Moore. Sizing the internet. *White paper, Cyveillance*, 2000. (Cited page 106.)

M. Najork and A. Heydon. High-performance web crawling. In *SRC-RR 173, Compaq Systems Research Center*, 2001. (Cited page 111.)

T.H Nelson. *Computer lib: Dream machines*. Tempus Books, 1987. (Cited pages 81 and 164.)

M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004. (Cited page 31.)

ME. Newman and EA. Leicht. Mixture models and exploratory analysis in networks. *PNAS*, 104(23):9564–9569, 2007. (Cited pages 35, 39, and 81.)

MEJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006a. (Cited pages 12, 31, 32, 67, 72, 73, and 131.)

MEJ Newman. The structure and function of complex networks. *Arxiv preprint cond-mat/0303516*, 2003. (Cited page 16.)

MEJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005. (Cited page 29.)

MEJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):36104, 2006b. (Cited page 32.)

MEJ Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):26113, 2004. (Cited pages 23, 29, and 31.)

A.Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS 14*, 2002. (Cited pages 12, 32, 65, 67, 131, and 140.)

A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 79(2):26102, 2009. (Cited page 125.)

Krzysztof Nowicki and Tom A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1090, 2001. (Cited pages 12, 34, 35, 39, 48, 49, and 131.)

Manfred Opper. *On-line learning in neural networks*, chapter 16, pages 363–378. Cambridge University Press, 1999. (Cited pages 77 and 132.)

Tim O'Reilly. What is web 2.0? design patterns and business models for the next generation of software. www.oreilly.com, September 2005. (Cited page 106.)

Paul Otlet. Monde. essai d'universalisme : Connaissance du monde, sentiments du monde, action organisée et plan du monde. *Editions Mundaneum*, 1, 1935. (Cited page 104.)

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998. (Cited page 112.)

C.H. Papadimitriou and CH Papadimitriou. *Computational complexity*. Addison-Wesley Reading, MA, 1994. (Cited page 20.)

K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894. (Cited pages 40 and 41.)

F. Picard, V. Miele, J.J. Daudin, L. Cottret, and S. Robin. Deciphering the connectivity structure of biological networks using MixNet. *BMC bioinformatics*, 10(Suppl 6):S17, 2009. (Cited page 14.)

P. Pons and M. Latapy. Computing communities in large networks using random walks. *Lecture notes in computer science*, 3733:284, 2005. (Cited page 33.)

A. Pothen. Graph partitioning algorithms with applications to scientific computing. *ICASE LARC INTERDISCIPLINARY SERIES IN SCIENCE AND ENGINEERING*, 4:323–368, 1997. (Cited page 26.)

S. Raghavan and H. Garcia-Molina. Representing Web graphs. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 405–416, 2003. (Cited pages 98 and 118.)

K. Randall, R. Stata, R. Wickremesinghe, and J.L. Wiener. The link database: Fast access to graphs of the web. In *Proceedings of the Data Compression Conference*, pages 122–131. IEEE Computer Society Washington, DC, USA, 2002. (Cited page 117.)

S.A. Rice. The identification of blocs in small political bodies. *The American Political Science Review*, 21(3):619–627, 1927. (Cited page 11.)

A. Robles-Kelly and E.R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 365–378, 2005. (Cited page 11.)

A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433, 1976. (Cited page 56.)

S. Ruping and T. Scheffer. Learning with multiple views. In *Proc. ICML Workshop on Learning with Multiple Views*, 2005. (Cited pages 82 and 95.)

G. Salton, A. Wong, and CS Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. (Cited page 72.)

S.E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. (Cited pages 12 and 35.)

A. Schenker, H. Bunke, M. Last, and A. Kandel. *Graph-Theoretic Techniques for Web Content Mining*. World Scientific, 2005. (Cited page 81.)

B. Schölkopf, A.J. Smola, P. Knirsch, and C. Burges. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. *Informatik Aktuell*, pages 125–132, 1998. (Cited page 32.)

AJ Scott and MJ Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, 27(2):387–397, 1971. (Cited page 52.)

S.B. Seidman. Network structure and minimum degree. *Social Networks*, 5 (3):269–287, 1983. (Cited page 23.)

S.B. Seidman and B.L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6(1):139–154, 1978. (Cited page 23.)

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. (Cited page 32.)

Vladislav Shkapenyuk and Torsten Suel. Design and implementation of a high-performance distributed web crawler. 2002. (Cited page 111.)

R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973. (Cited page 28.)

T. A. B. Snijders and K. Nowicki. Estimation and prediction for stochastic block-structures for graphs with latent block structure. *Journal of Classification*, 14:75–100, 1997. (Cited pages 12, 34, 35, 39, 48, 81, and 131.)

R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *Multivariate Statistical Methods, Among-groups Covariation*, page 269, 1975. (Cited page 28.)

T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. *Biologiske Skrifter*, 5: 1–34, 1948. (Cited page 28.)

O. Sporns, C.J. Honey, and R. K
"otter. Identification and classification of hubs in brain networks. *PLoS One*, 2(10), 2007. (Cited page 14.)

PR Suaris and G. Kedem. An algorithm for quadrisection and its application to standard cellplacement. *IEEE Transactions on Circuits and Systems*, 35(3):294–303, 1988. (Cited page 26.)

D. M. Titterington. Recursive parameter estimation using incomplete data. *JRSS-B*, 46:257–267, 1984. (Cited page 39.)

DM Titterington, AFM Smith, and UE Makov. *Statistical analysis of finite mixture distributions*. John Wiley & Sons Inc, 1985. (Cited page 41.)

D. Verma and M. Meila. A comparison of spectral clustering algorithms. *University of Washington, Tech. Rep. UW-CSE-03-05-01*, 2003. (Cited page 32.)

S. Wang and Y. Zhao. Almost sure convergence of titterington's recursive estimator for finite mixture models. *IEEE International Symposium on Information Theory IST*, 2002. (Cited page 39.)

J.H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963. (Cited page 33.)

Stanley Wasserman, Katherine Faust, and Dawn Iacobucci. *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, November 1994. (Cited pages 22 and 25.)

D.J. Watts. *Six degrees: The science of a connected age*. WW Norton & Company, 2004. (Cited page 16.)

R.S. Weiss and E. Jacobson. A method for the analysis of the structure of complex organizations. *American Sociological Review*, 20(6):661–668, 1955. (Cited page 11.)

J.H. Wolfe. Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, 5(3):329–350, 1970. (Cited page 44.)

B. Wu and B.D. Davison. Identifying link farm spam pages. In *International World Wide Web Conference*, pages 820–829. ACM New York, NY, USA, 2005. (Cited page 104.)

O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54, 1998. (Cited page 97.)

H. Zanghi, C. Ambroise, and V. Miele. Fast online graph clustering via Erdős–Rényi mixture. *Pattern Recognition*, 41(12):3592–3599, 2008. (Cited pages 3, 7, 15, 40, 52, 65, 89, 139, 148, and 162.)

H. Zanghi, F. Picard, V. Miele, and C. Ambroise. Strategies for Online Inference of Model-Based Clustering in large and growing Networks. *Annals Of Applied Statistics*, 4(2):687–714, 2010a. (Cited pages 3 and 7.)

Hugo Zanghi, Stevenn Volant, and Christophe Ambroise. Clustering based on random graph model embedding vertex features. *Pattern Recognition Letters*, In Press, Corrected Proof, 2010b. ISSN 0167-8655. doi: DOI:10.1016/j.patrec.2010.01.026. (Cited pages 3 and 7.)

H.J. Zeng, Q.C. He, Z. Chen, W.Y. Ma, and J. Ma. Learning to cluster web search results. *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 210–217, 2004. (Cited page 97.)

T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–826. ACM New York, NY, USA, 2006. (Cited page 95.)

H. Zhou. Distance, dissimilarity index, and network community structure. *Physical review e*, 67(6):61901, 2003. (Cited page 32.)

List of Figures

1.1 Network studied problems 12
1.2 Zachary's karate club. The colors correspond to the partition found by optimizing the modularity of Newman and Girvan for two groups. 14
1.3 Macaque cortex network displayed with colors for each MixNet class (8 groups) from the paper of Daudin et al. (2008). 14
1.4 Community structure in a hypertext document network. Groups, indicated by the colors, were detected with the *CEM* algorithm of Zanghi et al. (2008) 15
1.5 Co-authorship network : Each of the nodes in the network, which we depict using a Kamada-Kawai visualization(Kamada and Kawai 1989), is colored according to its community assignment using a spectral method algorithm. 16

1.6 Example of cliques: the green clique is the maximal clique over the set of vertices, $\{F, D, L\}$. There is no larger possible clique in the graph containing those three vertices than the green 4-clique. But the maximal clique in G is actually the 6-clique containing the vertices $\{A, G, H, J, K, M\}$. 22
1.7 A dendrogram or a hierarchical tree. Horizontal cuts corresponds to partitions of the graph in communities. 25
1.8 Hue (from red=0 to blue=max) shows the node betweenness. 30
1.9 MCL algorithm. The color of a bond between two nodes indicates the maximum amount of flow taken over the two directions. The color of a node indicates the total amount of incoming flow. A dark bond between a light node and a dark node thus indicates that all flow is going from the lighter node in the direction of the darker node. It is seen that flow is eventually separated into different regions, yielding a cluster interpretation of the initial graph. 34

2.1 Densities of Gaussian mixture models. On the left a well separated (black line) case where $\mu_1 = 0$ (red points), $\mu_2 = 4$ (blue points)and $\sigma^2 = 1$. On the right a more intertwined case where $\mu_1 = 0$, $\sigma_1^2 = 1$, $\mu_2 = 2$ and $\sigma_2^2 = 2$. 42
2.2 MixNet and its exponential family generalization. 48
2.3 Graphical representation of the MixNet Model with a Bernoulli distribution of edges. The squares represent discrete random variables. 49
2.4 Online algorithms are incremental algorithms which recursively update parameters using current parameters and additional information provided by new observations 51
2.5 Simulation of a 100 nodes graph with 5 classes according to an affiliation model (see Section 2.4.3 page 2.4.3). 62
2.6 Integrated Classification Likelihood Criterion in function of the number of clusters computed for the simulated graph of Figure 2.5. 62
2.7 Top left: low inter-module connectivity and strong intra-module connectivity (model 1), Top right: strong inter-module connectivity and low intra-module connectivity (model 5), Bottom center: Erdős-Rényi model (model 4). . . 64
2.8 Rand Index evolution for $\lambda \in \{0.58, 0.59, \ldots, 0.68\}$. The plain line represents the online SAEM algorithm, the $\triangle$ line represents the online CEM algorithm and the $\circ$ line represents the online variational algorithm. 66
2.9 Network of the blogopole `www.blogopole.fr`. 69
2.10 Adjacency matrix of the blogopole network after reordering according to the estimated partition. 70
2.11 Degree distribution of the websites of the blogopole network. The histogram and the curve respectively represent the observed and the estimated distributions. 70

2.12 Graph Layout of the 2008 U.S. Presidential WebSphere using *Gephi*. The red dots represent Conservative websites, the yellow are the Independents and finally the blue dots the Liberal websites. 73
2.13 Integrated Classification Likelihood Criterion for the 2008 U.S. Presidential WebSphere. 74
2.14 Network summary of US political websites. Each vertex represents a cluster. Each pie chart gives the proportions of liberal, conservative and independent tagged websites in the cluster. The outer ring color of the vertices is proportional to the intensity of the intra-connectivity: the darker, the weaker. Edges are represented when the inter-connectivity is among the 20% of the largest among all connectivity values. 75
2.15 Boxplot of MixNet classes betweenness (in log). 76

3.1 The hypertext diagram from Ted Nelson (Nelson 1987) which allows to develops complex and dynamic systems of linking and cross-referencing. Hypertext documents can either be static (prepared and stored in advance) or dynamic (continually changing in response to user input). The Web is an implementation of the hypertext paradigm. 81
3.2 Graphical representation of the two CoshMix Models. The first one does not take into account the dotted line and assumes independence between $\mathbf{X}$ and $\mathbf{Y}$ conditional on the hidden structure $\mathbf{Z}$. The second model introduces an additional dependence between graph structure and vertex features. The squares represent discrete random variables and circles continuous random variables. 84
3.3 Simulation of two 150 nodes graphs with 4 classes according to the parameters 3.1. Edges are represented with black dots and information vary from white (weak) to red (strong). The left figure illustrates the CohsMix^1 model and the right the CohsMix^2 model where in presence of edges we can notice stronger information. 86
3.4 Graphical representation of the CohsMix^3 Model. The squares represent discrete random variables and circles continuous random variables. 88
3.5 Integrated Classification Likelihood Criterion in function of the number of clusters for (a) $CohsMix^1$ and (b) $CohsMix^2$. The associated models are identicals to 3.3 93
3.6 Comparison of HMRF, Spectral MLV and `CohsMix`. (a) Varying Q the number of classes. (b) Varying the number of Features. (c) Varying the distance between intra and inter connectivity parameters. (d) Varying the distance between the mean vector of the classes. 97

3.7 Representation of the results of a clustering of the webpages returned by the controversial query "Scientology" using CohsMix[3] . The graph structure is represented on the left and on the right are the main features. Colors indicate the CohsMix[3] classification. 99
3.8 Representation of the results of a clustering of the webpages returned by the ambiguous query "jaguar". 99
3.9 Representation of the results of a clustering of the webpages returned by the ambiguous query "jaguar". 99

4.1 A schematic picture of the bow-structure of the Web (image from Broder et al. (2000). 108
4.2 The (a)in- and (b)out-degree of Web nodes closely follow power-law distributions, except at low degrees. 109
4.3 Graphical representation of PageRank 113
4.4 A schematic example of the graph simplification. Nodes with same color will be associated to the same meta-node. . 118
4.5 View of the *Constellations* application after a query. 128
4.6 View of the hits list and an information box for a hit. 128
4.7 Technological solution of the *Constellations* rendering. 129

A.1 Boxplot of the parameter estimates for 30 estimations of the 4 models. Each model is described by two boxplots, one for the estimations of ϵ and the other for the estimations of λ. The circles show the true value of the parameters. 139
A.2 Boxplot of MixNet classes traffic (in log). The lower the Alexa ranking number the more heavily visited the site. . . 143

List of Tables

2.1 Statistics of the affiliation model computed from $n_{ql} = \sum_{i>j} x_{ij} z_{iq} z_{jl}$, the number of edges having nodes in class q and l, and $n_q = \sum_i z_{iq}$, the number of nodes of class q. 56
2.2 Parameters of the five affiliation models considered in the experimental setting. 64
2.3 Parameters of the five affiliation models in the experiment. The Q modules are mixed in the same proportion. Each model considers $n = 500$ nodes and $Q = 5$ groups. 65
2.4 Means and standard deviations of the Rand Index for all models with q and n fixed. 66
2.5 Means of the Rand Index with speed of the algorithms. $q = 5$, model 2. 67
2.6 Means and standard deviation of the Rand Index for the five models computed over 30 different runs for graph clustering competitors and Variational algorithms. 67

2.7 Contingency table comparing true and estimated partitions 69
2.8 The table corresponds to the probabilities ($\times 100$) of connection between the 11 selected clusters (using a penalized likelihood criterion discribed in Daudin et al. (2008)). Dots in the table correspond to connections lower than 1%. 71
2.9 Quality of the clustering procedure in terms of Rand Index when the network grows over time. Each configuration has been simulated 100 times. 71
2.10 Contingency table comparing the political partition and MixNet partition. 74

3.1 Arbitrary parameters of two CohsMix^{1,2} models 86
3.2 Means of the estimated parameters for CohsMix^{1,2} models computed over 20 different runs. 94
3.3 Parameters of the four different settings which are used to generate the 43 affiliation models considered in the experiments. 95

4.1 HTTP parameters of the Exalead Web Search API used for the *Constellations* services. 120
4.2 The *Constellations* search features. 121
4.3 Item contained in the *RSS* response which have matched the user's search query. 122
4.4 DTD for the *node* and *edge* elements. 126
4.5 DTD for the MixNet elements. 127
4.6 DTD for the final response 127

A.1 Parameters of the four affililation models of the experiment. The Q modules are mixed in the same proportion. Each model consider $n = 1000$ nodes. 139
A.2 Means and standard deviation of the parameter estimates of the four models computed over 30 different runs. 140
A.3 Means and standard deviation of the Rand index of the four models computed over 30 different runs for graph clustering competitors and MixNet algorithms. 140
A.8 Estimated π (in percentage) and number of nodes in each cluster for the US political websphere. $\bar{d}$ represents the estimated mean degree of each group. 142

Notation

Mathematic

$\text{Argmax}_t f(t)$	argument that maximizes the function f
$\text{Argmin}_t f(t)$	argument that minimizes the function f
$\mathbf{I}$	identity matrix
$f(t) \propto g(t)$	functions f and g are proportional

Probability

$f(\mathbf{X}; \boldsymbol{\beta})$	density function of $\mathbf{X}$ indexed by the parameter $\boldsymbol{\beta}$
$\mathbb{E}_{\boldsymbol{\beta}}[g(\mathbf{X})]$	expectation of $g(\mathbf{X})$, a function of $\mathbf{X}$, indexed by the parameter $\boldsymbol{\beta}$
$X \sim f(\mathbf{X}; \boldsymbol{\beta})$	X follows a distribution with density function $f(\mathbf{X}; \boldsymbol{\beta})$

Statistic and classification

$(X_1, \ldots, X_N)$	sample of N observations
$\mathcal{L}(\boldsymbol{\beta}; \mathbf{X})$	log-likelihood of parameter $\boldsymbol{\beta}$
$\mathbf{Z}$	classification matrix
α_q	proportion of class q

Title Model Based approaches for uncovering Web structures

Abstract The statistical analysis of complex networks is a challenging task, given that appropriate statistical models and efficient computational procedures are required in order for structures to be learned. The principle of these models is to assume that the distribution of the edge values follows a parametric distribution, conditionally on a latent structure which is used to detect connectivity patterns. However, these methods suffer from relatively slow estimation procedures, since dependencies are complex. In this thesis we adapt online estimation strategies, originally developed for the EM algorithm, to the case of graph models. In addition to the network data used in the methods mentioned above, vertex content will sometimes be available. We then propose algorithms for clustering data sets that can be modeled with a graph structure embedding vertex features. Finally, an online Web application, based on the *Exalead* search engine, allows to promote certain aspects of this thesis.

Keywords graph clustering, finite mixture models, online EM algorithms, Web mining, large-scale Web application.

Titre Approches modèles pour la structuration du Web vu comme un graphe

Résumé L'analyse statistique des réseaux complexes est une tâche difficile, étant donné que des modèles statistiques appropriés et des procédures de calcul efficaces sont nécessaires afin d'apprendre les structures sous-jacentes. Le principe de ces modèles est de supposer que la distribution des valeurs des arêtes suit une distribution paramétrique, conditionnellement à une structure latente qui est utilisée pour détecter les formes de connectivité. Cependant, ces méthodes souffrent de procédures d'estimation relativement lentes, puisque les dépendances sont complexes. Dans cette thèse nous adaptons des stratégies d'estimation incrémentales, développées à l'origine pour l'algorithme EM, aux modèles de graphes. Additionnellement aux données de réseau utilisées dans les méthodes mentionnées ci-dessus, le contenu des nœuds est parfois disponible. Nous proposons ainsi des algorithmes de partitionnement pour les ensembles de données pouvant être modélisés avec une structure de graphe incorporant de l'information au sein des sommets. Finalement, un service Web en ligne, basé sur le moteur de recherche d' *Exalead*, permet de promouvoir certains aspects de cette thèse.

Mots-clés partitionnement de graphe, modèles de mélange finis, algorithmes EM incrémentaux, fouille de données Web, application Web à grande-échelle.

Ce document a été préparé l'aide de l'éditeur de texte GNU Emacs et du logiciel de composition typographique LaTeX 2_ε.