THESE DE DOCTORAT DE
L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité

Systèmes Informatiques

Présentée par

M. Oussama Allali

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE PIERRE ET MARIE CURIE

Structure et dynamique des graphes de terrain bipartis : liens internes et prédiction de liens

Soutenance : le 24 Juin 2011

Jury :

Eric FLEURY	**Rapporteur**	**Professeur, ENS de Lyon**
Céline ROUVEIROL	**Rapporteur**	**Professeur, Université Paris-Nord**
Patrick GALLINARI	**Examinateur**	**Professeur, Université Pierre et Marie Curie**
Anne-Marie KERMARREC	**Examinateur**	**Directeur de Recherche INRIA, Rennes**
Christophe PRIEUR	**Examinateur**	**Maître de Conférences, Université Paris-Diderot**
Matthieu LATAPY	**Co-directeur**	**Directeur de Recherche CNRS**
Clémence MAGNIEN	**Co-directeur**	**Chargée de Recherche CNRS**

DOCTOR OF SCIENCE THESIS
PIERRE AND MARIE CURIE UNIVERSITY

Specialization

Computer Science

presented by

M. Oussama Allali

Submitted for the degree of

DOCTOR OF SCIENCE FROM PIERRE AND MARIE CURIE UNIVERSITY

Structure and dynamics of bipartite complex networks: internal links and link prediction

Defence: June, 24^{th} 2011

Committee:

Abstract

Many real-world complex networks, like actor-movie or file-provider relations, have a bipartite nature and are best modeled as bipartite graphs, where nodes are divided into two sets with links between nodes of different sets only. However, there is currently a lack of methods to analyze properly such graphs as most existing measures and methods are suited to classical graphs. A common but limited approach consists in deriving classical graphs (called projections) from the underlying bipartite structure, though it causes important loss of information and data storage issues. We introduce in this thesis *internal links* and *pairs* as a new notion useful for such analysis: it gives insights on the information lost by projecting the bipartite graph. We illustrate the relevance of these concepts on several real-world datasets and illustrate how it enables to discriminate behaviors among various cases, when we compare them to a benchmark of random networks. Then, we show that we can draw benefit from this concept for both modeling complex networks and storing them in a compact format.

Many bipartite real-world complex networks are dynamic: they evolve during time, with node and link additions and removals. Predicting links that will appear in them is one of the main approaches to understand their dynamics. Only few works address the bipartite case, though, despite its high practical interest and the specific challenges it raises. We propose in this thesis a link prediction method based on internal links. We thoroughly describe the method and its variations, and experimentally compare it to a basic collaborative filtering approach. We present results obtained for a typical practical case. We reach the conclusion that our method performs very well, and we study in details how its parameters may influence the obtained results.

Keywords:

complex networks, bipartite graphs, projection, internal links, graph storage, graph analysis, graph dynamics, link prediction.

Table of contents

Chapter 1

Introduction

One may model various objects coming from the real world with graphs. We can cite many examples among computer, social, biological, or linguistic networks, like internet maps, web graphs, data exchanges, authoring, protein interactions or occurrence networks. It appeared recently that most of these objects share nontrivial statistical properties [22,77, 136]. This has showed that it is indeed relevant to consider these objects as a coherent group. For this reason, one designates them by the general term *real-world complex networks*.

Understanding how real-world complex networks are structured, how they evolve over time and what are the events that may occur within them, are key questions of the field. It is structured into four research areas: measurement, analysis, modeling and algorithmic of these networks.

Real-world complex networks are not directly available: collecting data about them requires the use of *measurement* procedures. In most case, these procedures lead to partial views obtained using various and often intricate exploration methods. In general, the measurement procedure cannot capture the whole graph, because of its size and various other constraints. A bias may in addition be introduced by the exploration method. It is crucial to study the influence this may have on the results, and try to correct such bias.

Analysis of a real-word complex networks aims at describing their structure. This is done using statistical notions and/or structural ones, aiming at capturing the key features of a graph. This topic has led to an important stream of studies [9,22,30,39,46,58,59,135,136]. The definition of such notions is however not trivial, as well as the evaluation of their relevance and the interpretation of the obtained description. One method is to compare the behaviors of real-world complex networks to random ones. Notions which make real-world networks different from random networks indicate a specificity of real-world networks.

In order to explain the nature of observations, to provide mathematically rigorous results and to conduct appropriate simulations, it is important to capture the observed proprieties in models of real-world complex networks. *Modeling* a real-world complex network basically consists in producing an *artificial* graph similar to the real one.

Finally, the study of real-world complex networks calls for the design of appropriate *algorithmic* methods. Indeed, their size is generally very large, from a few thousands to a few billions of nodes. Therefore, most classical algorithmic solutions are not applicable (average distance computation for instance). Therefore, it is not always possible to perform exact computations on these graphs, and approximations are often necessary. The context of complex networks also raises new algorithm issues that have not been raised before (community detection for instance).

In addition, most real-world complex networks are dynamic, *i.e.* their structure evolves over time by the addition and/or removal of nodes and/or links. Recently, some works have studied graphs dynamics. They are mostly in-depth studies of specific cases, for instance mobile networks [38, 42, 44, 120, 128], peer-to-peer exchange networks [64, 80, 82], internet topology [76,102,105,106], biological networks [17,110,124,131], citations networks [85], and various social networks [47,121,125]. Other works have developed general methods, relevant to the study of any dynamic graph. We can cite methods to manipulate dynamic graphs [36, 73], event detection [65], community detection or evolution [18,104], link prediction [28,67, 87], and some even more general questions [16, 78, 86, 99].

One of the main approaches developed for studying network dynamics is *link prediction*, which consists in predicting the links that will probably appear in the future, given a snapshot of the considered graph at a given time.

Many real-world complex networks actually have a *bipartite* nature: their nodes may be separated into two classes, with links between nodes of different classes only. Typical examples include actor-movie networks [101, 136] where actors are linked to the movies they played in; authoring networks [94, 95] where authors are linked to the papers they signed, etc.

In this thesis, we focus on bipartite complex networks, and we follow two directions for improving the understanding of these objects: we study their structure by introducing new relevant notions and we study their dynamics using link prediction.

Some notions used in the analysis of classical (non-bipartite) graphs extend directly to the bipartite case, like for instance the size, density, or degree distribution; they may therefore be used in this context. For others, the extension does not make any sense in itself. For instance, the notion of clustering coefficient, since it relies on the enumeration of triangles, does not make sense for bipartite graphs, which cannot contain any triangle. Although there already exist methods for the analysis of bipartite netwoks [26, 34, 41, 54, 55, 113, 115, 123], much remains to be done in this direction.

Similarly, several works study the problem of link prediction on classical (non-bipartite) networks [19, 67, 87, 103]. However, their methods are not directly applicable to, or appropriate for, bipartite graphs. Another research problem is closely related to link prediction in bipartite graphs: the recommendation problem [112]. Recommendation systems are used to suggest items to users, such as products to customers for instance. Notice however that the two problems are quite different: recommendation aims typically at finding a few products of interest for each customer; prediction aims at finding as many links as possible that will appear in the future.

Contribution. The bipartite nature of many real-world complex networks calls for the development of new, specific notions and methods that would not make sense in the classical case. In this perspective, we identify a special kind of links that we call *internal links*, and propose them as an important notion for analyzing bipartite networks. These links are very frequent in practice, and we show that associated statistics are fruitful to point out similarities and differences among real-world complex networks and study their features.

Internal links also have an important role regarding the dynamics of networks. We study this and propose an approach based on internal links for link prediction in bipartite graphs. Our method performs very well, much better than a classical recommendation approach. Moreover, our method is purely structural: it relies on the identification of a specific kind of links which will probably appear in the future; this gives much insight on the properties of the underlying dynamics.

This manuscript is organized as follows. Chapter 2 presents a state of the art regarding notions and methods used for analyzing real-world bipartite complex networks. Chapter 3 describes eight real-world datasets that we use in this manuscript. We use the statistics and notions presented in Chapter 2 for analyzing these eight datasets. Chapter 4 presents the notion of internal links and pairs in bipartite graphs, and shows that this notion is important for analyzing real-world bipartite complex networks [14], and their dynamics. Chapter 5 builds on the observations made in the previous chapters to propose a new method for link prediction in bipartite graphs [12, 13]. Chapter 6 presents our conclusions and some key directions for further work. Appendix A describes a method for measuring activity in a peer-to-peer system developed during this thesis [11].

Chapter 2

Bipartite complex networks

Contents

2.1 Introduction

Many real-world complex networks have a natural bipartite structure and may therefore be modeled as bipartite graphs, *i.e.* graphs with two sets of nodes, and links only between nodes in different sets. Typical examples include actor-movie networks [101, 136] where actors are linked to the movies they played in; authoring networks [94, 95] where authors are linked to the papers they signed; occurrence networks where the words occurring in a book are linked to the sentences of the book they appear in [56]; company board networks where the board members are linked to the companies they lead [27, 114]; peer-to-peer exchange graphs [57,64,133] where peers are linked to the files they provided/searched for; on-line shopping networks where clients are linked to the products they bought [89]; etc.

All these real-world complex networks have been studied in various disciplines such as biology, computer science, social science, marketing, etc. In most of these contexts, however, the authors transform the bipartite nature of network they study into a classical (non-bipartite) graph, where two nodes are linked if they have at least one neighbor in common in the bipartite graph. This process is called *projection*.

For the examples presented above, some projections are: co-starring networks, where two actors are linked if they played together in a movie; co-authoring networks, where two authors are linked if they signed a paper together; peer-to-peer exchange networks, where two peers are linked together if they have provided/searched for a same data; etc.

The projection approach allows the study of bipartite graphs using the powerful tools and notions existing for classical graphs. However it implies much loss of information initially available in the bipartite graphs [79]. For instance, the fact that two authors co-authored many papers is an important information which is present in a bipartite authoring graph but not in the projected co-authoring graph.

To avoid these issues, one may use *weighted* projections in which a weight is associated to each link [23,24,98]. For instance, in a co-authoring graph the weight may be the number of co-authored papers. This approach transforms the problem of studying a bipartite graph into the problem of studying weighted projections, for which much methods exist [23,24,98]. This allows to understand in more details the properties of bipartite graphs. There are several methods for weighting the links [4,82,117,118], each bringing different information about the bipartite graph.

Conversely, one may see the properties of the projected graph as consequences of the projection process itself [62, 63, 101]. Some authors therefore proposed a model based on bipartite graphs and projection. This approach makes it possible to reveal unexpected behaviors, meaning that the underlying bipartite structure has nontrivial properties, and deepening this makes sense.

Finally, some authors study the bipartite data directly. To do so, they extend the notions used for classical graphs, and develop new notions designed for bipartite graphs: overlap [108], clustering [33, 88, 114], centrality [54], redundancy [79] and others [7, 37, 53, 107].

In this chapter, we present the state-of-the-art of notions and methods used for analyzing bipartite graphs, including the notions extended from classical graphs. We first present the basic bipartite and projection definitions in Sections 2.2 and 2.3. We then present statistics specific to bipartite graphs in Section 2.4. We present in Section 2.5 different methods for weighting projected graphs. We present the extension of classical random graphs to the bipartite case in Section 2.6. We finally present the different kinds of dynamics of bipartite graphs in Section 2.7. We discuss our conclusions in Section 2.8.

2.2 Basic definitions

We give here basic definitions and notations which we use throughout this manuscript.

A classical graph $G = (V, E)$ is defined by a set V of nodes and a set $E \subseteq V \times V$ of links. We denote by $N(u) = \{v \in V,\ (u, v) \in E\}$ the neighborhood of a node u in G. The number of nodes in $N(u)$ is the degree of u: $d(u) = |N(u)|$. Note that we consider here undirected graphs, meaning that there is no distinction between links (u, v) and (v, u).

The basic statistics describing such a graph are its size $n = |V|$ and its number of links $m = |E|$. Its average degree is $k = \frac{1}{n}\sum_{u\in V} d(u) = \frac{2m}{n}$; its density is $\delta = \frac{2m}{n.(n-1)}$, *i.e.* the number of links divided by the number of possible links between all pairs of nodes.

A bipartite graph $B = (\bot, \top, L)$ is defined by a set $\bot$ of bottom nodes, a set $\top$ of top nodes and a set $L \subseteq \bot \times \top$ of links. The key point is that links exist only between a node in $\bot$ and one in $\top$. We denote by $N(u) = \{v \in (\bot \cup \top),\ (u, v) \in L\}$ the neighborhood of a node u in B. If $u \in \bot$ then $N(u) \subseteq \top$, and conversely. More generally, given any set of nodes $S \subseteq (\bot \cup \top)$, we denote by $N(S)$ its neighborhood: $N(S) = \bigcup_{u\in S} N(u)$.

The basic statistics of bipartite graphs are direct extensions of the ones on classical graph: $n_\bot = |\bot|$ and $n_\top = |\top|$ are the numbers of top and bottom nodes respectively, and $m = |L|$ is the number of links. The bottom average degree is $k_\bot = \frac{1}{n_\bot}\sum_{u\in\bot} d(u) = \frac{m}{n_\bot}$ and the top average degree is $k_\top = \frac{1}{n_\top}\sum_{u\in\top} d(u) = \frac{m}{n_\top}$. The density is $\delta = \frac{m}{n_\bot n_\top}$.

2.3 Projection

The main approach to study bipartite complex networks consists in turning them into classical (non-bipartite) graphs through a process called *projection*. This approach makes it possible to use the powerful tools and notions provided for classical graphs for the study of bipartite graphs.

The $\bot$-projection of B is the graph $B_\bot = (\bot, L_\bot)$ in which $(u, v) \in L_\bot$ if and only if u and v have at least one neighbor in common in B: $N(u) \cap N(v) \neq \emptyset$. In other words, there is a link between u and v in $L_\bot$ if u and v are linked to a same top node in L (see Figure B.1).

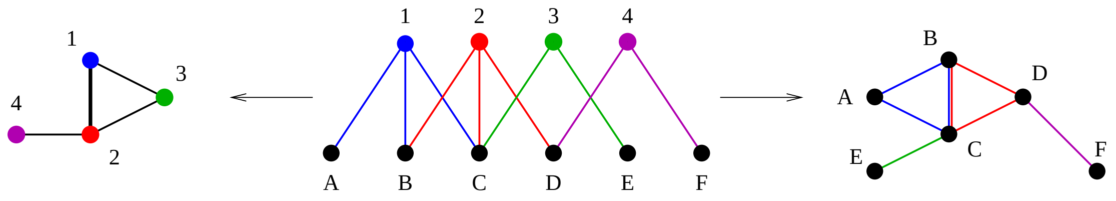


Figure 2.1 – An example of a bipartite graph B (center), together with its $\top$-projection $B_\top$(left), and its $\bot$-projection $B_\bot$(right).

We denote by $\bot(u, v)$ the sets of links induced by a pair of nodes (u, v) in $(\bot \times \top)$: $\bot(u, v) = \{(u, w), w \in N(v) \setminus \{u\}\}$.

In Figure B.1, for instance, $\bot(A, 1) = \{(A, B), (A, C)\}$. Notice that $L_\bot = \bigcup_{(u,v)\in L} \bot(u, v)$: the links of the $\bot$-projection of B are the links induced by all the links of B.

We denote by $\bot_v$ the set of links induced by a node v in $\top$: $\bot_v = \bigcup_{u\in N(v)} \bot(u, v)$. Each top node v induces in $B_\bot$ a clique between its neighbors in B (see Figure B.1). For example, all actors who have played in a same movie will be connected with each other in the projection.

Note that a same link in the projection may be induced by many nodes (for instance the link (B, C) in the $\bot$-projection in Figure B.1 is created by their common neighbors 1 and 2). This is not captured in the projection process.

We denote by $N_\bot(u)$ the neighborhood of a node u in $B_\bot$: $N_\bot(u) = \{v \in \bot,\ (u, v) \in L_\bot\} = N(N(u))$.

The $\top$-projection of B, denoted by $B_\top$, and associated notion, are defined dually.

2.4 Bipartite statistics

We present here the main statistics used for describing directly a bipartite graph without using the projection.

2.4.1 Degree distribution

The degree distribution of a classical graph is, for each integer k, the fraction P_k of nodes of degree k: $P_k = \frac{|\{u \in V : d(u)=k\}|}{n}$. In other words, it is the probability that a randomly chosen node has degree k.

Two degree distributions can naturally be associated to a bipartite graph. The bottom degree distribution $\bot_k = \frac{|\{u \in \bot : d(u)=k\}|}{n_\bot}$ gives the fraction of nodes in $\bot$ having degree k, and the top degree distribution $\top_k = \frac{|\{u \in \top : d(u)=k\}|}{n_\top}$ gives the fraction of nodes in $\top$ having degree k.

Degree distributions play a key role in the analysis of graphs. In particular, one makes in general the distinction between two kinds of degree distributions: *homogeneous* and *heterogeneous* ones.

Homogeneous distributions (such as normal, Gaussian and Poissonian distributions) are such that the degrees of the nodes are very close to the average degree. This means that the average degree gives important information: the average degree is meaningful because it indicates the expected behavior of nodes.

Heterogeneous distributions (such as Zipf and power-law distributions) are such that there are several orders of magnitude between degrees, and most nodes have a degree very different from the average degree. Then, the average value gives little information: it is very different from the degree of most nodes, and randomly chosen nodes will have very different degrees.

2.4.2 Clustering coefficient

We present here the two classical versions of clustering coefficient and their equivalent versions in bipartite graphs.

2.4.2.1 Clustering coefficient of classical graphs

The clustering coefficient in classical graphs is a measure of the local density of nodes. It aims at capturing a notion of overlap: it measures the probability that two nodes are linked together, provided they have a neighbor in common. In other words, it is the probability that any two neighbors of any node are linked together. There are two slightly different versions:

The first one computes the probability, for any given node, that two of its neighbors are linked together. It is defined for any node u of degree at least 2:

$$\mathrm{cc}_\bullet(u) = \frac{|\{(v,w) \in E \ s.t \ v,w \in N(u)\}|}{\frac{d(u).(d(u)-1)}{2}}.$$

The clustering coefficient of the graph itself is the average of this value for all nodes:

$$\mathrm{cc}_\bullet(G) = \frac{\displaystyle\sum_{u \in V,\ d(u) \geq 2} \mathrm{cc}_\bullet(u)}{|\{u \in V, d(u) \geq 2\}|}.$$

The second notion of clustering coefficient of G (sometimes called *transitivity ratio*) applies directly to the whole graph:

$$\mathrm{cc}_\vee(G) = \frac{3N_\triangle}{N_\vee}$$

where $N_\triangle$ denotes the number of triangles, *i.e.* sets of three nodes with three links in G, and $N_\vee$ denotes the number of connected triples, *i.e.* sets of three nodes with at least two links, in G. This notion of clustering is slightly different from the previous one since it gives the probability, when one chooses two links with one extremity in common, that the two other extremities are linked together.

2.4.2.2 Clustering coefficient of bipartite graphs

Whereas there are quite direct extensions of the basic statistics to the bipartite case, the notion of clustering coefficient does not make sense in itself in a bipartite graph. Indeed, it relies on the enumeration of triangles in the graph, and there can be no triangle in a bipartite graph.

Both definitions of classical clustering coefficients capture the fact that when two nodes have something in common (one neighbor) then they are linked together with a probability much higher than two randomly chosen nodes. Conversely, they capture the fact that when two nodes are linked together then they probably have neighbors in common. In other words, they capture correlations between neighborhoods. From this point of view, it is possible to extend the first notion of clustering coefficient to bipartite graphs [34, 79, 82] and capture the overlap between the neighborhoods pairs of nodes:

$$\mathrm{cc}_\bullet(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}.$$

If u and v have no neighbor in common then $\mathrm{cc}_\bullet(u, v) = 0$. If they have the same neighborhood, then $\mathrm{cc}_\bullet(u, v) = 1$. And if their neighborhoods partially overlap then the values are in the interval $]0, 1[$, closer to 1 when the overlap is large compared to their degrees, and inversely. See Figure 2.2 for an illustration.

Capturing the overlap between neighborhoods may however be ambiguous. Suppose that degrees are heterogeneous in the network, and consider two nodes u and v. If one

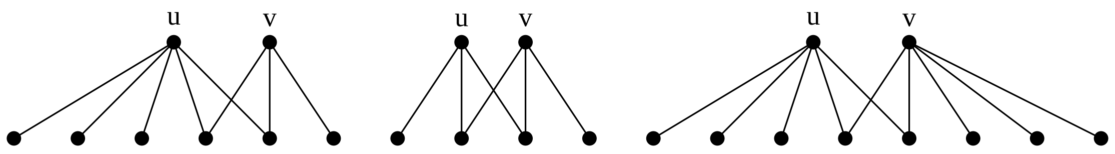


Figure 2.2 – Examples of bipartite clustering coefficient. For each case we give the values of Left: 0.33, 0.66 and 0.4. Center: 0.5, 0.66 and 0.66. Right: 0.25, 0.4 and 0.4.

of these nodes has a high degree and the other has not, then $\mathrm{cc}_{\bullet}(u,v)$ will necessarily be small. This will be true even if one of the neighborhoods is entirely included in the other. This may be captured better using the following definition:

$$\mathrm{cc}_{\underline{\bullet}}(u,v) = \frac{|N(u) \cap N(v)|}{\min(N(u), N(v))}.$$

One may define dually:

$$\mathrm{cc}_{\overline{\bullet}}(u,v) = \frac{|N(u) \cap N(v)|}{\max(N(u), N(v))}.$$

See Figure 2.2 for an illustration. These two notions, called min- and max-clustering, were introduced first in [82]. The first one emphasizes the fact that small neighborhoods may intersect significantly large ones; it is equal to 1 whenever one of the neighborhoods is included in the other. The second one emphasizes on the fact that neighborhoods (both small or large ones) may overlap very significantly: it is 1 only when the two neighborhoods are the same and it tends to decreases rapidly if the degree of one of the involved nodes increases. It captures the fact that nodes with similar degrees have high neighborhood overlaps.

These definitions of clustering coefficients capture the tendency of two particular nodes to have large neighborhood overlaps. One may then define the clustering coefficient of one node as the average of its clustering coefficients with other nodes [79]:

$$\mathrm{cc}_{\bullet}(u) = \frac{\sum_{v \in N(N(u))} \mathrm{cc}_{\bullet}(u,v)}{|N(N(u))|}.$$

The clustering coefficient of all the graph is the average of this value for all nodes:

$$\mathrm{cc}_{\bullet}(B) = \frac{\sum_{u \in (\perp \cup \top)} \mathrm{cc}_{\bullet}(u)}{n_{\perp} + n_{\top}}.$$

Two correlations may be derived from this definition. The bottom correlation of clustering coefficient with node degree: $\mathrm{cc}_{\bullet\perp}(k) = \frac{\sum_{u \in \perp : d(u)=k} \mathrm{cc}_{\bullet}(u)}{|\{u \in \perp : d(u)=k\}|}$ is the average clustering coefficient of all nodes in $\bot$ having degree k, and the top one: $\mathrm{cc}_{\bullet\top}(k) = \frac{\sum_{u\in\top:d(u)=k}\mathrm{cc}_{\bullet}(u)}{|\{u\in\top:d(u)=k\}|}$ is the average clustering coefficient of all nodes in $\top$ having degree k.

These correlation make it possible to observe if the value of the clustering coefficient is related to the degree of nodes. In particular, one may observe if a high (or low) value of the degree implies a high (or low) value of the clustering coefficient.

The extension of the second notion of classical clustering coefficient is proposed in [114]. It measures the probability that, given four nodes with three links, they actually are connected with four links (all the possible bipartite ones):

$$\mathrm{cc}_{\mathrm{N}}(B) = \frac{2N_{\bowtie}}{N_N}.$$

where $N_{\bowtie}$ is the number of quadruplets of nodes with four links in B, and N_N is the number of quadruplets of nodes with at least three links.

2.4.3 Redundancy

The redundancy coefficient is a measure of the importance of nodes regarding the projection process [79].

First notice that links in the $\bot$-projection (resp. $\top$) may be induced by several $\top$ nodes (resp. $\bot$ nodes) during the projection, and the role of nodes cannot be distinguished from one another in the projection. In fact, the links induced by a node may be (all) also induced by others nodes. Removing such a node from the bipartite graph will only change slightly (or not at all) the projection. This may be captured by the redundancy coefficient, which is the fraction of the links induced by u ($d(u) \geq 2$) that are also induced by another node than u:

$$\mathrm{rc}(u) = \frac{|\{\{v,w\} \subseteq N(u), \exists u' \neq u, (u',v) \in L \text{ and } (u',w) \in L\}|}{\frac{d(u).(d(u)-1)}{2}}.$$

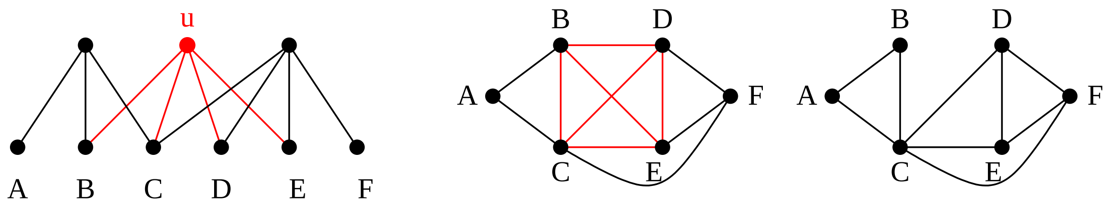


Figure 2.3 – Example of redundancy. From left to right: a bipartite graph, its $\bot$-projection, and the $\bot$-projection obtained if the node u is removed. Only two links disappear, leading to $\mathrm{rc}(u) = \frac{4}{6} = 0.666$.

In other words, the redundancy coefficient of u is the fraction of pairs of neighbors of u linked to another node than u. In the projection, these nodes would be linked together even if u were not there, see Figure 2.3. If it is equal to 1 then the projection would be exactly the same without u; if it is 0 it means that none of its neighbors would be linked together in the projection.

Here again, we can study the distribution of the redundancy coefficient and its correlation with node degree, which we do not detail here.

2.5 Weighted projections

As explained for instance in [79] and above, $B_\perp$ contains much less information than B. In particular, the fact that u and v are linked in $B_\perp$ means that they have *at least* one neighbor in common in B but says nothing on their *number* of common neighbors. Several approaches are used for weighting the links of the $\perp$-projection in order to capture such information. We present the main ones in this section (examples are presented in Figure 2.4).

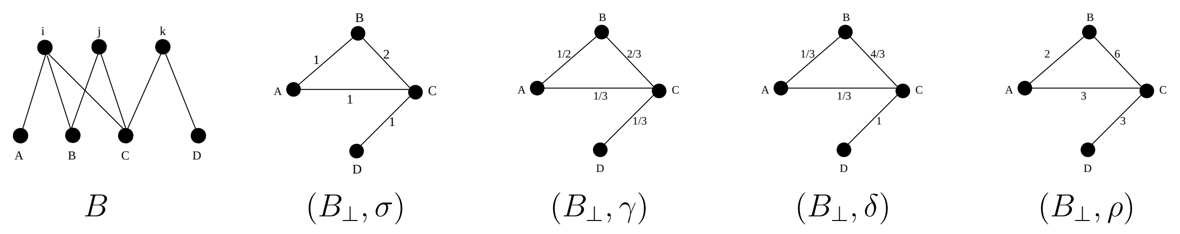


Figure 2.4 – A bipartite graph B (left) and its $\perp$-projection with the different weight functions defined in Section 2.5.

First, the weight of a link (u, v) may be defined as the number of (top) neighbors that u and v have in common in the bipartite graph, called *sum*:

$$\sigma(u, v) = |N(u) \cap N(v)|.$$

The σ weight function has been used for instance to estimate the probability of collaboration between authors [93].

Notice that if u and v both have many neighbors, then $\sigma(u, v)$ will naturally tend to be high. Conversely, if u and v have only few neighbors but these neighbors are the same, then $\sigma(u, v)$ will be low, which does not reflect the fact that the neighborhoods of u and v are very similar. To capture this, one may use the *Jaccard* coefficient:

$$\gamma(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}.$$

This quantity has been used for instance in the context of peer-to-peer exchange analysis to capture similarity between peers [82].

The value of $\gamma(u,v)$ may however be strongly biased if one of the two nodes has many neighbors and the other one only few: the value would then be very low, even if all neighbors of one node are neighbors of the other. To avoid this, variants of the *Jaccard* coefficient, *overlap* and *cosine*, have been proposed in the literature [117, 118]:

$$\gamma^{ove}(u,v) = \frac{|N(u) \cap N(v)|}{\min(|N(u)|, |N(v)|)} \qquad \gamma^{cos}(u,v) = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)| \times |N(v)|}}.$$

From this point of view, though, nodes play an unbalanced role: a $\top$-node x has an influence on the similarity between $\frac{|N(x)| \times (|N(x)|-1)}{2}$ pairs of nodes in the $\bot$-projection. When $N(x)$ is large, this is huge; on the contrary, if a $\top$-node only has two neighbors then it probably indicates a significant similarity between them. To capture this, one may consider that each $\top$-node *votes* for the similarity between its neighbors and that the sum of its votes is only one (it has only one voice to distribute). This leads to the *delta* weight function:

$$\delta(u,v) = \sum_{x \in N(u) \cap N(v)} \frac{2}{|N(x)| \times (|N(x)| - 1)}.$$

In Figure 2.4, for instance, nodes i and j vote respectively $\frac{1}{3}$ and 1 for link (B,C), so $\delta(B,C) = \frac{1}{3} + 1$. A similar quantity has been used in [4] to capture the similarity between two home pages as a function of the features they share.

All weight functions above intuitively capture similarity between nodes. One may also use weight functions to capture other features, like the activity of nodes in the network. This leads for instance to compute the product of the number of neighbors of u and v in the bipartite graph [21, 93], called *attachment*:

$$\rho(u,v) = |N(u)| \times |N(v)|,$$

which reflects the expectation that u and v may have neighbors in common: if links were placed at random then the probability that (u,v) is a link would be proportional to $\rho(u,v)$.

All weighting functions presented above are natural and capture relevant informations about a bipartite graph. Each has its own strengths and weaknesses.

2.6 Random graphs

Various models have been proposed to generate artificial graphs [10,22,31,51,91,92,96, 137]. One key use of such models is to identify non-trivial features of real-world graphs by comparing them to null models. To this regard, the most simple and random models are of highest interest. We present here the case of classical graphs and their equivalent bipartite models.

2.6.1 Random classical graphs

The basic model of random graphs was introduced by Erdös- Rényi [32,52]. There are two variants of the model. The model $\mathcal{G}_{n,p}$ generates a graph with n nodes in which each of the $\frac{n.(n-1)}{2}$ possible links exists with a given probability p. The model $\mathcal{G}_{n,m}$ generates a graph with n nodes and m links chosen at random from the $\frac{n.(n-1)}{2}$ possible links. The two models are equivalent if p and m satisfy $m = p.\frac{n.(n-1)}{2}$.

It has been shown [10,32,50,97,126,137] that this model does not capture some of the main features of real-world complex networks. In particular, the obtained graphs have a clustering coefficient equal to p since the probability that each pair of nodes is connected is independent of their neighbors. This means that the clustering is small when the average degree is small (which is the case in practice for most of the graphs we consider). Moreover, the degree distribution follows a Poisson law: $p_k = e^{-\lambda}\frac{\lambda^k}{k!}$. This distribution is centered on the mean value $\lambda = p.n$, and all nodes have a degree close to this average value.

2.6.1.1 Classical graphs with prescribed degree distribution

The configuration model [91,92] builds random graphs with prescribed degree distribution (a power-law for instance). The first step is to generate the degree distribution desired. It can be explicitly described (distribution of a real network for instance) or implicitly defined (power-law with a given exponent for instance). This allows to construct the degree sequence, *i.e.* the degree of each node. The second step consists in assigning to each node as many connection points as its degree. Finally, one constructs links by choosing random pairs of connection points and connecting them.

The sum of the degrees must be even to allow the connection of all connection points. Note that this algorithm may induce multiple links and loops. There are however very few such links when the graph grows and they are usually neglected in large complex network studies. One may also use algorithms and techniques proposed in [48,96,132] to avoid them.

2.6.1.2 Classical graphs with preferential attachment

The second main model is based on the preferential attachment principle [22, 51]. The aim of this model is to simulate a growing graph. For instance in the Web, when a new Web page is created, it likely connects to a well known Web page rather than a randomly chosen one. However, the most famous Web pages are those that have more links pointing to them. Therefore, the pages with the most incoming links will probably acquire more in the future.

A graph from this model is generated as follows: at each step, a node is added with links to preexisting nodes chosen with a probability proportional to their degree. This leads to a degree distribution that follows a power-law with exponent 3. This model has been studied intensively and is now well known (see [10] for a survey of its properties).

2.6.2 Random bipartite graphs

The purely random model $\mathcal{G}_{n,p}$ may extended to $\mathcal{G}_{n_\perp,n_\top,p}$, a bipartite graph with $n_\perp$ bottom nodes and $n_\top$ top nodes, each of the $n_\perp.n_\top$ possible links existing with a given probability p. Equivalently, one may construct such a bipartite graph by choosing $m = p.n_\perp.n_\top$ links at random (which is an extension of the $\mathcal{G}_{n,m}$ model to $\mathcal{G}_{n_\perp,n_\top,m}$). Like for classical graphs, though, these simple models have Poisson degree distribution, which does not fit real-world cases.

2.6.2.1 Bipartite graphs with prescribed degree distribution

Generating a bipartite graph with prescribed degree distributions may be done by extending the classical configuration mode as follows [90, 101] (see Figure 2.5):

1. generate both top and bottom nodes and assign to each a degree chosen from the given distributions;
2. create for each node as many connection points as its degree;
3. choose random pairs of top and bottom connection points to create links.

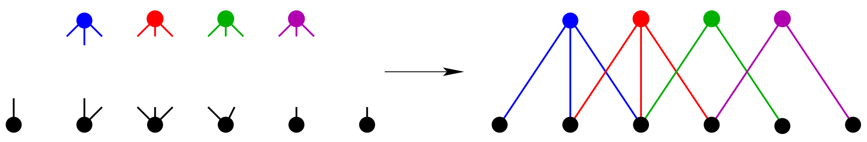


Figure 2.5 – Example of the construction of a random bipartite graph with prescribed degree distributions.

This process generates random bipartite graphs uniformly within the set of bipartite graphs with the given degree distributions. However it cannot be used without taking into account the following constraint: the number of connection points of top and bottom must be equal after the second step. This can be achieved in one of the following ways:

– if the distributions are explicitly given, by taking original distributions for instance, then the constraint is necessarily verified;
– if the distributions are given implicitly then, before sampling the degrees, one must make sure that the distributions are compatible: the product of the number of nodes by their average degree must be the same for top and bottom (in practice, we do not need a strict equality);
– if the two distributions are compatible, and the two sets of degrees experimentally drawn from these distributions are inconsistent (the sums of the degrees are different) then, one can drop one top and one bottom node at random and sample their degree again [90, 101].

Note that, just like with the configuration model, multiple links may appear. Again, there are however very few such links when the graph is large, and simply ignoring them generally has no significant impact on the results.

2.6.2.2 Bipartite graphs with preferential attachment

In order to capture a power-law degree distributions in growing bipartite graphs like for classical graphs, a variant of the preferential attachment has been defined [62, 63]. At each step:

– add a new top node u, and choose its degree k according to the prescribed (top) distribution;
– for each k links of node u, add a link to a preexisting bottom node using preferential attachment (with probability λ), or create a new bottom node (with probability $1-\lambda$) and connect it to u.

The parameter λ is the *overlap ratio*, *i.e.* the proportion of preexisting bottom nodes to which a new top node is connected.

2.7 Dynamics of bipartite graphs

Most real-world complex networks actually are dynamic: they evolve during time, with node and link additions and removals. We deal in this thesis with *growing* graphs only, *i.e.* graphs in which there are no node or links removals, only additions. This is very frequent in practice.

Let us define a dynamic (growing) bipartite graph by a set of n timestamped links $D = \{(t_i, u_i, v_i), i = 1...n\}$ where t_i is the arrival time of link (u_i, v_i). We define $B = (\bot, \top, L)$ as the graph observed from a given instant a to another instant $b > a$: $\bot = \{u, \exists(t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$, $\top = \{v, \exists(t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$ and $L = \{(u, v), \exists(t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$. We call B the graph observed during period $[a, b[$

Now let us consider an instant $c > b$. This induces a set L' of links added to B during period $[b, c[$: $L' = \{(u, v), \exists(t, u, v) \in D \text{ s.t. } b \leqslant t < c\}$.

Notice that, even when we focus on *growing* graphs different kinds of dynamics may occur, in particular:

- new links may appear with new nodes only. For instance, in the actor-movie bipartite graph, growth occurs with addition of new movies only, with links to old actors (existing nodes in B) and to some new actors, playing in a movie for the first time. No link may appear between two nodes already in B.
- new links may appear between pair of previously existing nodes. For instance, in peer-to-peer bipartite graph, an existing peer may start to provide a file already provided by others.

This different types of dynamics call for different approaches for their study. For instance, predicting new links appearing between preexisting nodes is quite different from predicting new node arrivals, and we will deal with the former problem in Chapter 5.

2.8 Conclusion

We have introduced the methods and the notions most used for analyzing bipartite graphs. Although other notions have been introduced, most of them are ad hoc and specific for the case under study, or variants of the ones presented here

Most notions described here are extensions to the bipartite case of the most basic notions used nowadays to analyze large classical graph. These notions go from the very basics (size, degree distributions) to more subtle ones (various clustering coefficients, random bipartite models).

One way to capture the relevance of the notions defined above is to compare their values on a representative set of real-world cases and on random graphs with the same size and degree distributions. The idea behind this is that a notion which behaves similarly on real-world and random networks is just a property of most networks. Instead, one generally looks for properties which are specific to real-world networks and make then different from random networks.

Another approach described above consists in transforming a bipartite graph into a classical one through projection. This approach allows to use the powerful tools developed for classical graphs. However it leads to important loss of information. Although this may be mitigated by the use weighted projection.

The ideal consists in studying graphs directly, without resorting to such tricks. However, much remains to be done in this direction.

Chapter 3

Datasets and their properties

Contents

3.1 Introduction

In this chapter, we describe eight real-world bipartite complex networks that we use in this thesis. We use the statistics and notions described in Chapter 2 for analyzing these eight examples. In order to complete our analysis, we estimate the relevance of these statistics by comparing their values on random bipartite graph with the same degree distributions as our datasets.

We do not detail the way these graphs are obtained. This is explained in the references that we cite for each case. Let us insist on the fact that our aim here is not to derive conclusions on these particular networks: we only discuss their representativity of a wide range of cases.

Also, our aim here is not to discuss in detail the specificities of each statistics used for analyzing these real-world networks, but to give evidence of the fact that these statistics have nontrivial behaviors and capture significant information.

We describe the eight real-world networks and their basic features in Section 3.2. Then, we analyze their degree distributions and correlations (Section 3.3). After that, we study their clustering coefficients (Section 3.4) and redundancy coefficient (Section 3.5). We finally present our conclusions in Section 3.6.

3.2 Datasets and basic statistics

We present in this section the datasets we use and their general features. The networks under consideration all connect people ($\perp$-nodes) to events or similar interests ($\top$-nodes).

We have used the following real-world networks:

Internet Movie Database (IMDB `www.imdb.com`) [22] is an online database of information related to movies, actors and other visual entertainment. We consider the *Imdb-movies* bipartite network where actors connected to the movies they played in. $\perp$ is the set of actors and $\top$ is the set of movies.

Delicious (`www.delicious.com`) [61] is a social bookmarking web service for storing, sharing, and discovering web bookmarks. We consider *Delicious-tags* bipartite network where *Delicious* users connected to the tags they use for indexing their bookmarks. $\perp$ is the set of users and $\top$ is the set of tags.

Flickr (`www.flickr.com`) [109] is a photo publication web site. It allows to the users to share photos, index them, organize them, comment on them, and other features. We consider four bipartite networks:

– *Flickr-tags* where *Flickr* users are connected to the tags they use for indexing their photos. $\perp$ is the set of users and $\top$ is the set of tags.

- *Flickr-comments* where *Flickr* users are connected to the photos they comment. $\perp$ is the set of users and $\top$ is the set of photos.
- *Flickr-favorites* where *Flickr* users are connected to the photos they pick up as favorites. $\perp$ is the set of users and $\top$ is the set of photos.
- *Flickr-groups* where *Flickr* users are connected to the groups they belong to. $\perp$ is the set of users and $\top$ is the set of groups.

Peer-to-Peer the *eDonkey* network [8] is a semi-distributed peer-to-peer file exchange system based on directory servers. We consider the *P2P-files* bipartite network where peers are linked to the files they provide. $\perp$ is the set of peers and $\top$ is the set of files.

PRL Web of Science (`www.isiwebofknowledge.com`) is a database of papers and authors of *Physical Review Letters*. We consider the *PRL-papers* bipartite network where authors are linked to the papers they signed. $\perp$ is the set of authors and $\top$ is the set of papers.

	$n_\perp$	$n_\top$	m	$k_\perp$	$k_\top$	δ
Imdb-movies	127,823	383,640	1,470,418	11.5	3.8	0.00003
Delicious-tags	532,924	2,474,235	37,421,585	70.2	15.1	0.000028
Flickr-tags	319,686	1,607,879	13,336,993	41.7	8.3	0.000025
Flickr-comments	122,561	1,489,485	4,190,415	34.1	2.8	0.000023
Flickr-groups	72,875	381,076	5,662,295	77.7	14.8	0.0002
Flickr-favorites	321,312	6,450,934	17,871,828	55.6	2.7	0.0000086
P2P-files	122,599	1,920,353	4,502,704	36.7	2.3	0.00002
PRL-papers	15,414	41,633	249,474	16.2	6.0	0.0003

Table 3.1 – Basic bipartite statistics of our eight examples of real-world bipartite complex networks.

The basic properties of our eight examples of real-world bipartite complex networks are given in Table 3.1. It appears clearly that all are large networks with small average degrees compared to their size. However, there is a difference of an order of magnitude between the densities. For instance *Flickr-groups* is ten times denser than *P2P-files*.

3.3 Degrees statistics

We present in Figure 3.1 the top and bottom degree distributions of our eight networks. Top and bottom degree distributions are very heterogeneous, and some may be fitted by power laws (of various exponents), for instance *Flickr-favorites* (top) and *Delicious-tags* (top). However, most degree distributions are quite far from power-laws.

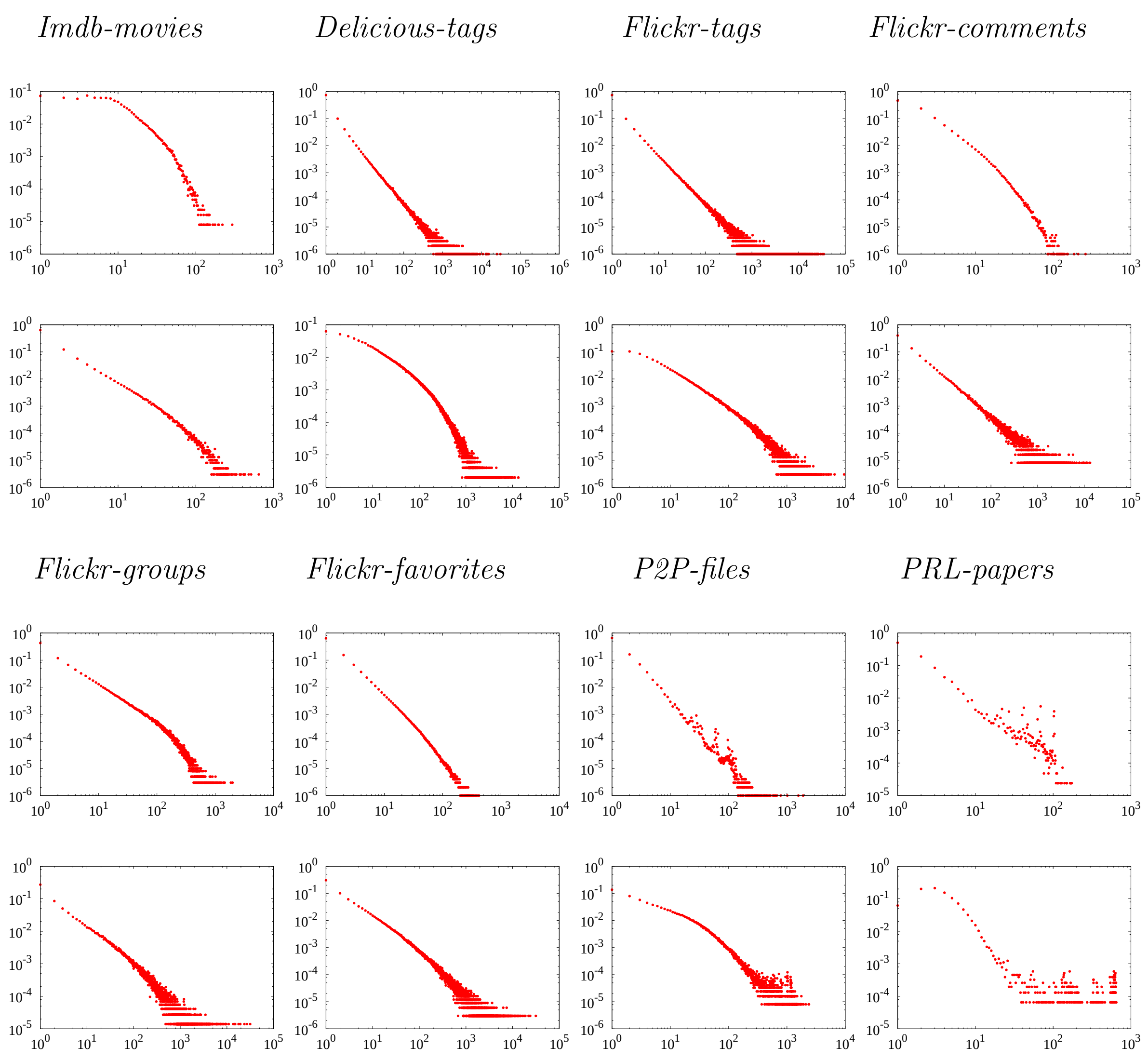


Figure 3.1 – Degree distributions of our eight datasets. First row: top nodes. Second row: bottom nodes.

To study the correlations between top and bottom degrees, we plot in Figure 3.2 for each integer i the average degree of all nodes which are neighbors of a node of degree i. We plot this for top and bottom nodes separately. We plot the same values obtained for random graphs of the same size and same degree distributions.

In all cases, the plots for the random cases are close to horizontal lines, showing that there are no correlations between a node's degree and the average degree of its neighbors: this last value is independent of the node's degree. Real-world cases exhibit significantly different behaviors, thus demonstrating that these behaviors are nontrivial and related to

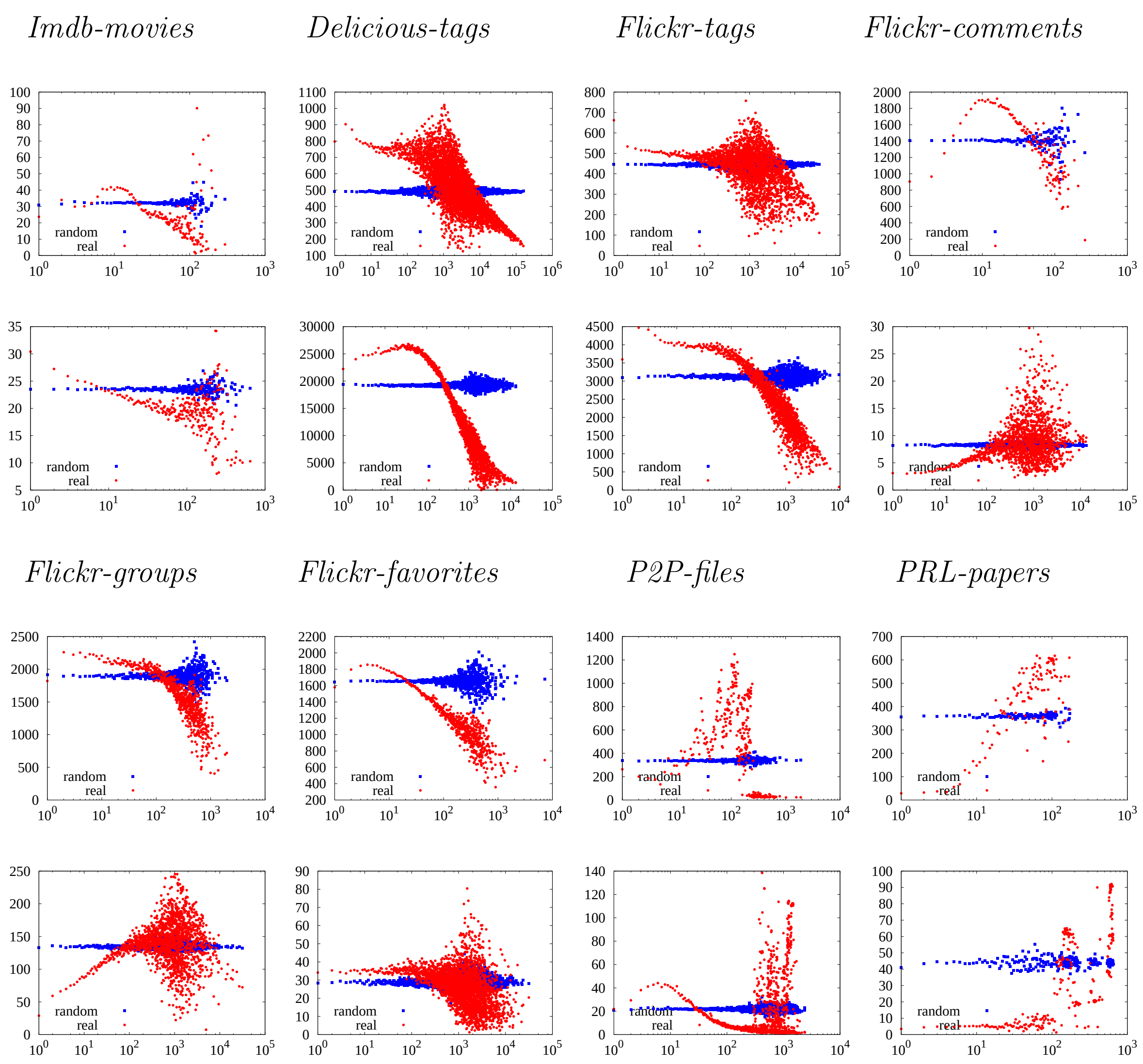


Figure 3.2 – Degree correlations in our eight datasets, and in random bipartite graphs of the same size and same degree distributions. First row: the average degree of the neighbors of top nodes as a function of their degree. Second row: the average degree of the neighbors of bottom nodes as a function of their degree.

intrinsinc properties of the underlying networks. We can see a general tendency: the average degree of neighbors of nodes decreases (or increases) with the node degree. For instance, in the *Imdb-movies* (bottom), the actors playing in many movies tend to play in smaller movies (in terms of the number of involved actors). In the *PRL-paper* (top), the papers cosigned by many authors tend to be cosigned by prolific authors. However, there are no clear tendency in other datasets. For instance, *Flickr-groups* (bottom) and *Flickr-comments* (bottom): point are scattered over a wide range of values, though such a tendency exists for small degrees.

3.4 Clustering statistics

Let us now compare the values of clustering statistics of our real-world networks with random bipartite graphs of the same size and same degree distributions.

We compute the ratio between the clustering coefficients on real networks and the values obtained on random graphs, so that the values correspond to how many times the clustering coefficient in real network is larger than the one in random network. The corresponding results are presented in Table 3.2.

It appears clearly that the obtained values for the different notions of clustering (with the exception of $\mathrm{cc}_{\underline{\bullet}}$) are larger on real-world networks than on random graphs. Moreover, the obtained values for $\mathrm{cc}_N(G)$ are often significantly larger on real-world networks than on random graphs, which shows that it may capture more relevant information in some cases.

	$\frac{\mathrm{cc}_{\bullet}(\bot)}{\mathrm{cc}^*_{\bullet}(\bot)}$	$\frac{\mathrm{cc}_{\bullet}(\top)}{\mathrm{cc}^*_{\bullet}(\top)}$	$\frac{\mathrm{cc}_N(G)}{\mathrm{cc}^*_N(G)}$	$\frac{\mathrm{cc}_{\underline{\bullet}}(\bot)}{\mathrm{cc}^*_{\underline{\bullet}}(\bot)}$	$\frac{\mathrm{cc}_{\underline{\bullet}}(\top)}{\mathrm{cc}^*_{\underline{\bullet}}(\top)}$	$\frac{\mathrm{cc}_{\overline{\bullet}}(\bot)}{\mathrm{cc}^*_{\overline{\bullet}}(\bot)}$	$\frac{\mathrm{cc}_{\overline{\bullet}}(\top)}{\mathrm{cc}^*_{\overline{\bullet}}(\top)}$
Imdb-movies	1.39	1.8	34.16	1.04	1.02	1.42	1.76
Delicious-tags	1.62	3.11	1.43	1.14	1.02	1.57	3.22
Flickr-tags	1.64	1.72	1.25	1.07	1.01	1.61	1.81
Flickr-comments	3.29	1.28	14.16	0.70	0.98	3.38	1.25
Flickr-groups	6.38	3.40	2	0.68	1	5.78	3.2
Flickr-favorites	1.70	1.31	2	0.8	1	1.54	1.3
P2P-files	3.64	1.64	581.08	1.04	1.05	3.52	1.58
PRL-papers	5.64	4	26.29	1.2	1.25	5.19	4.07

Table 3.2 – The ratio between the clustering statistics of real-world networks and the values on random bipartite graphs with the same size and same degree distributions. We denote by $*$ the value for random bipartite graphs.

3.4.1 Clustering distributions

We show in Figure 3.3 the cumulative distributions of $\mathrm{cc}_{\bullet}(u)$, $\mathrm{cc}_{\underline{\bullet}}(u)$ and $\mathrm{cc}_{\overline{\bullet}}(u)$ for our eight datasets: a point with coordinates (x, y) means that a fraction y of all the nodes have a value lower than x for the considered statistics.

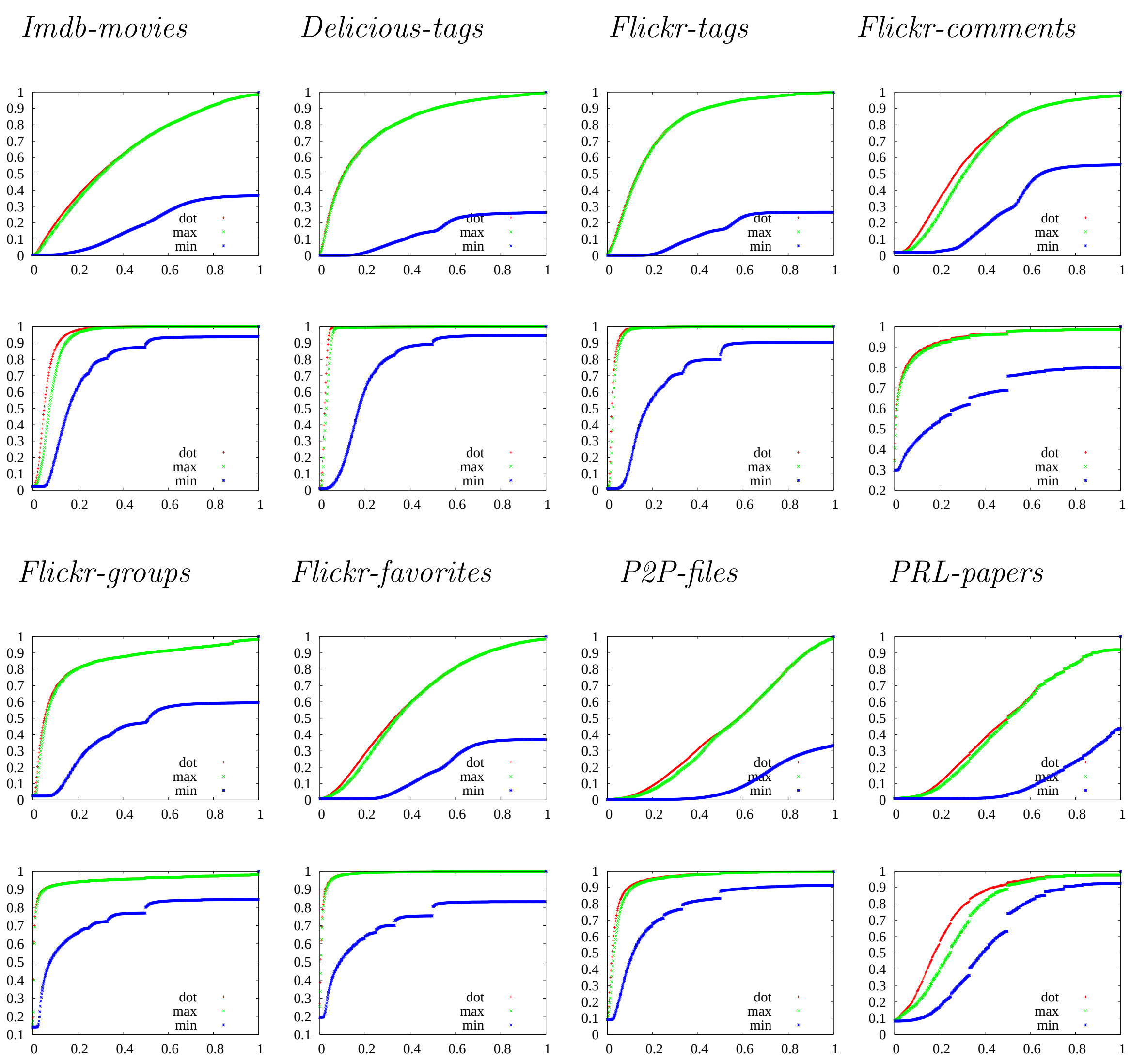


Figure 3.3 – Cumulative distributions of the various clustering coefficients in our eight datasets. First row: for top nodes. Second row: for bottom nodes. Dot, max and min correspond to respectively.

Before entering in the discussion of these plots, notice that, by definition, we have $\mathrm{cc}_{\bullet}(u) \leq \mathrm{cc}_{\bar{\bullet}}(u) \leq \mathrm{cc}_{\underline{\bullet}}(u)$ for any u. Therefore in each case, the lowest plot is the one of $\mathrm{cc}_{\underline{\bullet}}(u)$, the highest is the one of $\mathrm{cc}_{\bullet}(u)$ and the middle one corresponds to $\mathrm{cc}_{\bar{\bullet}}(u)$.

Three main different behaviors may be observed in Figure 3.3:

– The plots of $\mathrm{cc}_{\underline{\bullet}}(u)$, $\mathrm{cc}_{\bullet}(u)$ and $\mathrm{cc}_{\bar{\bullet}}(u)$ grow very rapidly and are close to 1 almost immediately, see for instance the case *Imdb-movies* (bottom). This means that the neighborhoods of nodes have a small intersection, compared to the union of their neighborhoods.

- The plot of $\mathrm{cc}_{\underline{\bullet}}(u)$ grow much less quickly than $\mathrm{cc}_{\bullet}(u)$ and $\mathrm{cc}_{\bar{\bullet}}(u)$. In this case, the plot of $\mathrm{cc}_{\underline{\bullet}}(u)$ remains lower than 1 for a long time and even until the end of the plot, meaning that for an important number of nodes the value of $\mathrm{cc}_{\underline{\bullet}}(u)$ is equal to 1, see for instance the case *Flickr-tags* (top). This means that, although overlaps are in general small compared to their possible value, the neighborhoods of many low-degree nodes significantly or even completely overlaps with other nodes' neighborhoods.
- The plots of $\mathrm{cc}_{\underline{\bullet}}(u)$, $\mathrm{cc}_{\bullet}(u)$ and $\mathrm{cc}_{\bar{\bullet}}(u)$ grow slowly, meaning that a significant number of nodes have a large value of these statistics, see for instance the case *P2P-files* (top). This means that node neighborhoods overlap significantly, and that this is not only a consequence of the fact that low degree nodes have their neighborhoods included in the ones of other nodes.

It is clear from the discussion above that the three notions of clustering captured by $\mathrm{cc}_{\bullet}(u)$, $\mathrm{cc}_{\bar{\bullet}}(u)$ and $\mathrm{cc}_{\underline{\bullet}}(u)$ are different, and give complementary insight on the underlying network properties.

Let us now compare the behaviors of $\mathrm{cc}_{\bullet}(u)$ on real-world cases and on random ones. We present them in Figure 3.4. It appears clearly that, except in some cases (for instance *PRL-papers*), the plots of the real-world values and of the random ones are quite similar. This means that, concerning the values of $\mathrm{cc}_{\bullet}(u)$, real-world graphs are not drastically different from random ones. This is due to the fact that the low degree nodes (which are numerous in our networks) have with high probability their neighbors in common with high degree nodes; by definition, this induces a low value of $\mathrm{cc}_{\bullet}(u)$ for the nodes of high and low degree, since the node with high degree have also their neighbors in common with low degree nodes. This is true by construction for random graphs, and the plots in Figure 3.4 show that this is mostly true for real-world networks also, which was not obvious.

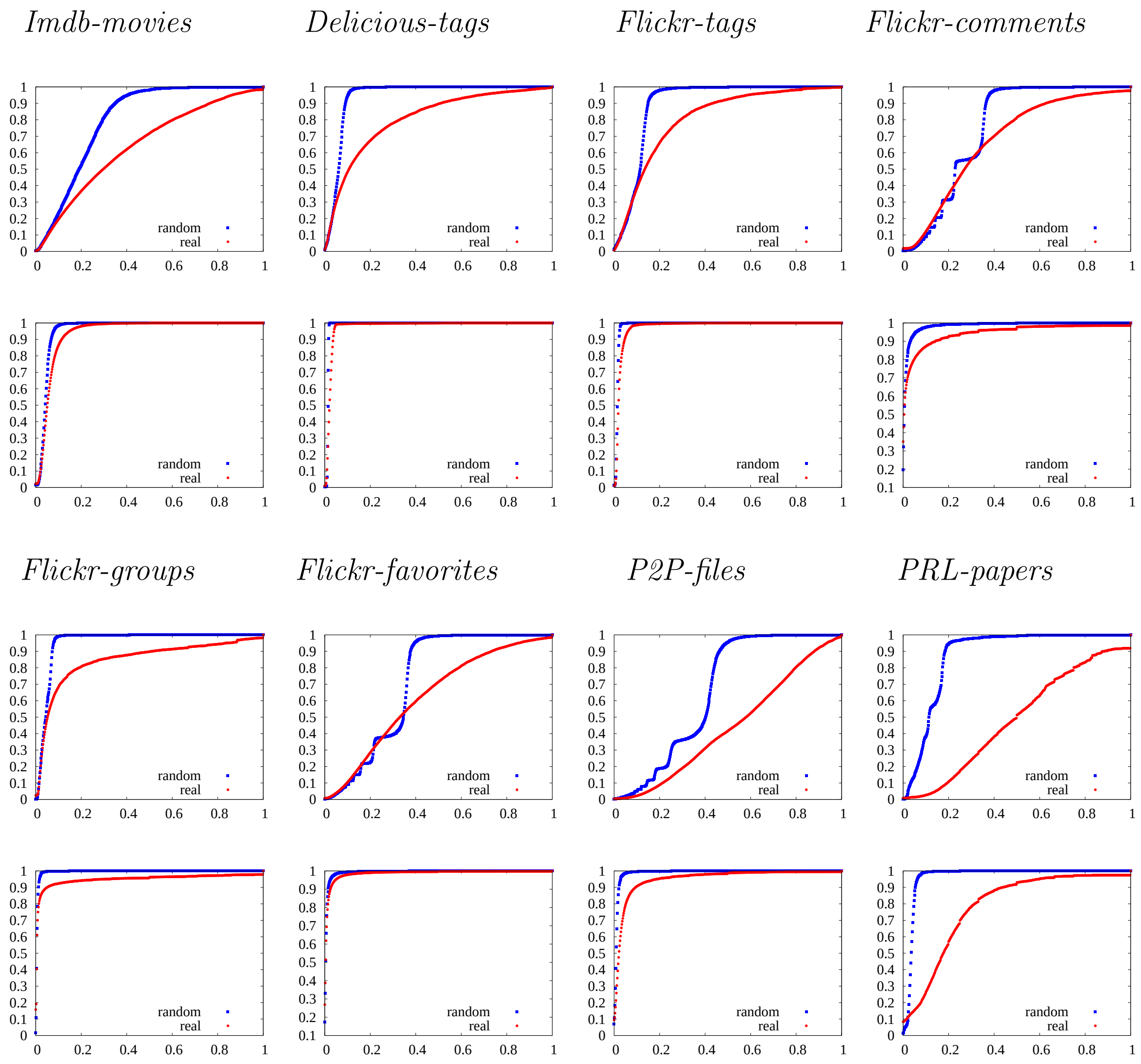


Figure 3.4 – Cumulative distributions of the $cc_\bullet(u)$ clustering coefficient in our eight datasets, and in random bipartite graphs of the same size and same degree distributions. First row: for top nodes. Second row: for bottom nodes.

3.4.2 Clustering correlations

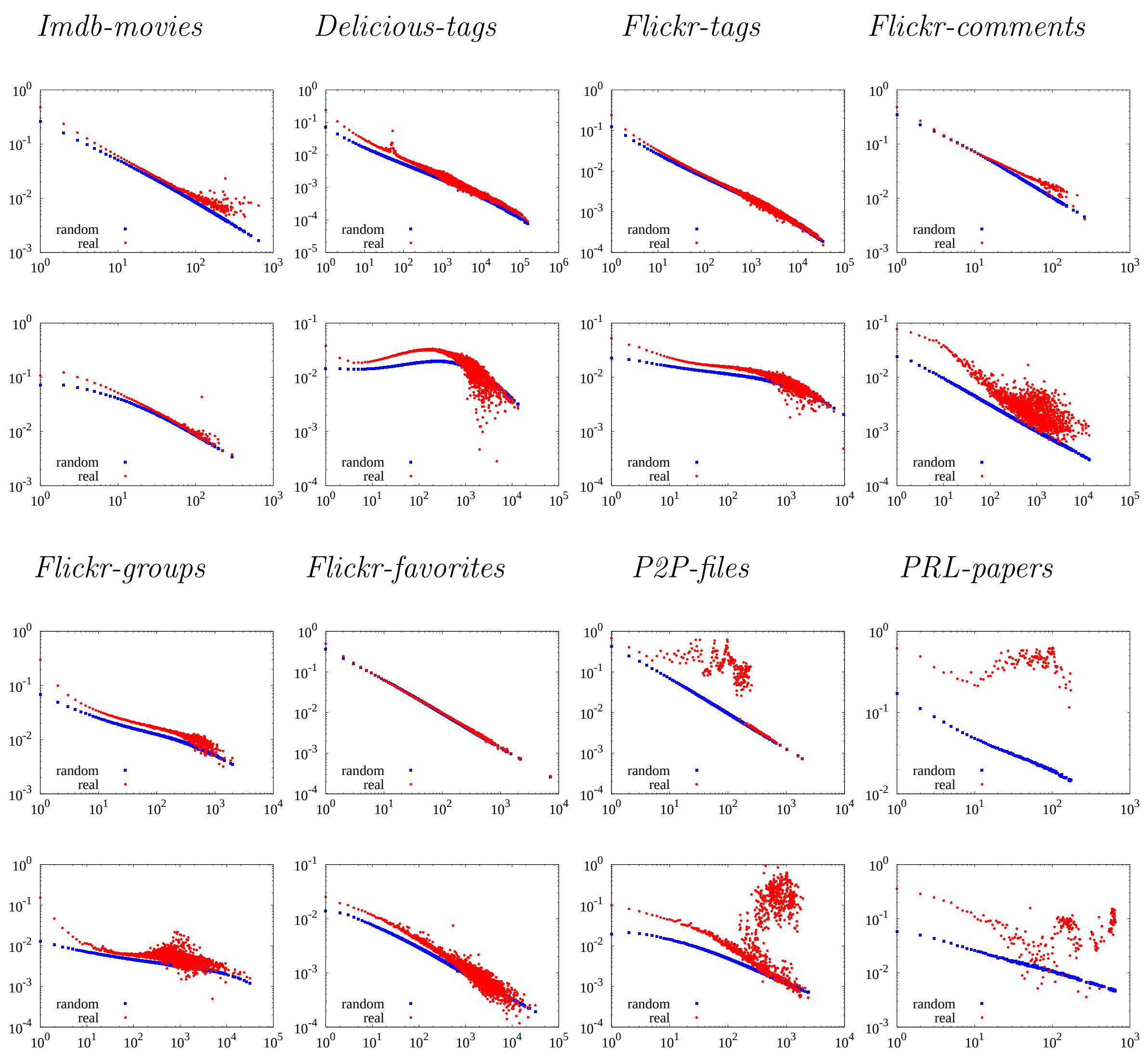


Figure 3.5 – Correlations of the $cc_\bullet(u)$ clustering coefficient with nodes degrees in our eight datasets, and in random bipartite graphs with the same size and degree distributions. First row: for top nodes. Second row: for bottom nodes.

Let us observe the correlations between nodes' degrees and their clustering coefficient. Figure 3.5 presents for each integer i the average value of the clustering coefficients of all nodes which have degree i. We plot this for top and bottom nodes separately. We plot the same values obtained for random graphs of the same size and same degree distributions. Three main observations may be made:

- The values for the random graphs are below the ones for the real-world cases. The value of $\text{cc}_\bullet(u)$ are slightly larger in real-world cases than in random ones, except in *PRL-papers* (top) where the values are significantly larger.
- The values for the random graphs decrease approximately as a power of the degree of u (straight line in log-log scale).
- In general, the clustering coefficient of low degree nodes is quite large, but the one of large degree nodes is very small, like in random graphs. This is due to the fact that the neighbors of nodes with high degree have a small intersection compared to the union of their neighborhoods.

3.5 Redundancy statistics

We have seen in the previous section that bipartite clustering coefficients do not make a huge difference between real-world and random bipartite graphs. This is why the redundancy coefficient was introduced in [79]. We now compare the values of the redundancy statistics of our real-world with comparable random bipartite graphs. We use the same method as in Section 3.4.

Table 3.3 shows clearly that, except in some graphs, the redundancy coefficient is not much larger in real-world networks than in random graphs. This is due to the fact that the projections of these graphs are very dense. The redundancy coefficient therefore is huge, but this is not a specific property of how the neighborhoods overlap: this is a direct consequence of the high density of the projections. In such a case, the redundancy coefficient has similar behaviors in such graphs and in their random equivalent.

	$\frac{\text{rc}_\bullet(\perp)}{\text{rc}^*_\bullet(\perp)}$	$\frac{\text{rc}_\bullet(\top)}{\text{rc}^*_\bullet(\top)}$
Imdb-movies	18.57	25
Delicious-tags	1.05	1.01
Flickr-tags	1.12	1.04
Flickr-comments	8.75	1.72
Flickr-groups	0.92	1.02
Flickr-favorites	3.75	1.41
P2P-files	12	2.36
PRL-papers	0.82	1.10

Table 3.3 – Ratio between the redundancy of real-world networks and the values on random bipartite graphs with the same size and same degree distributions.

3.5.1 Redundancy distributions

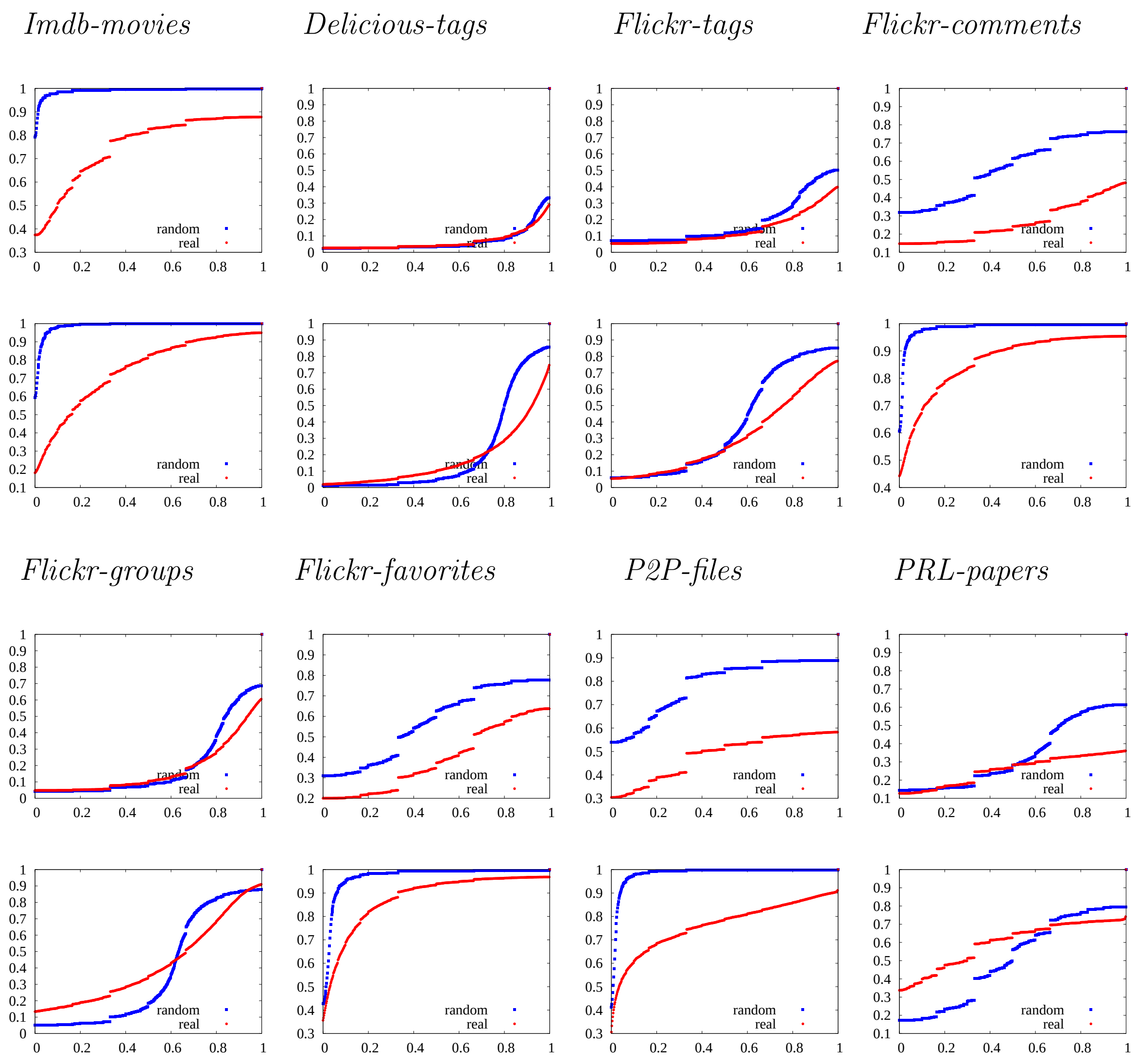


Figure 3.6 – Cumulative distributions of the redundancy coefficient $\mathrm{rc}_{\bullet}(u)$ in our eight datasets, and in random bipartite graphs of the same size and same degree distributions. First row: for top nodes. Second row: for bottom nodes.

Let us now compare the distributions of $\mathrm{rc}_{\bullet}(u)$ for our datasets with the ones of comparable random graphs. Figure 3.6 shows two different behaviors:

– the value of the redundancy coefficient in random graphs is close to 0 for most nodes; instead, in real-world networks it is significantly larger, see for instance case *Imdb-movies*. In these cases, the redundancy coefficient captures a property that makes real-world networks different from random ones.

– the value of the redundancy coefficient is very large for both (real and random) graphs, their plots have very similar shape and equal to 1 for a large portion of the nodes, see for instance case *Flickr-tags*. This is due to the density of the projected graphs (as explained above).

Note that, in all cases, the plot of the redundancy coefficient starts with a nonzero value. This means that there are some nodes with redundancy equal to zero, and so removing these nodes would have a strong impact on the projections. In this

3.5.2 Redundancy correlations

Let us now observe the correlations between nodes' redundancy coefficient and their degree, plotted in Figure 3.7. From these plots, two main observations may be made:

– the plots for random graphs are horizontal, which means that the values of node redundancy in random graphs are independent of the degree.

– the plots for real-world graphs exhibit nontrivial behaviors. The redundancy decreases or increases with the degree, depending on the graph. In several cases, some of the very high degree nodes have a very large redundancy coefficient.

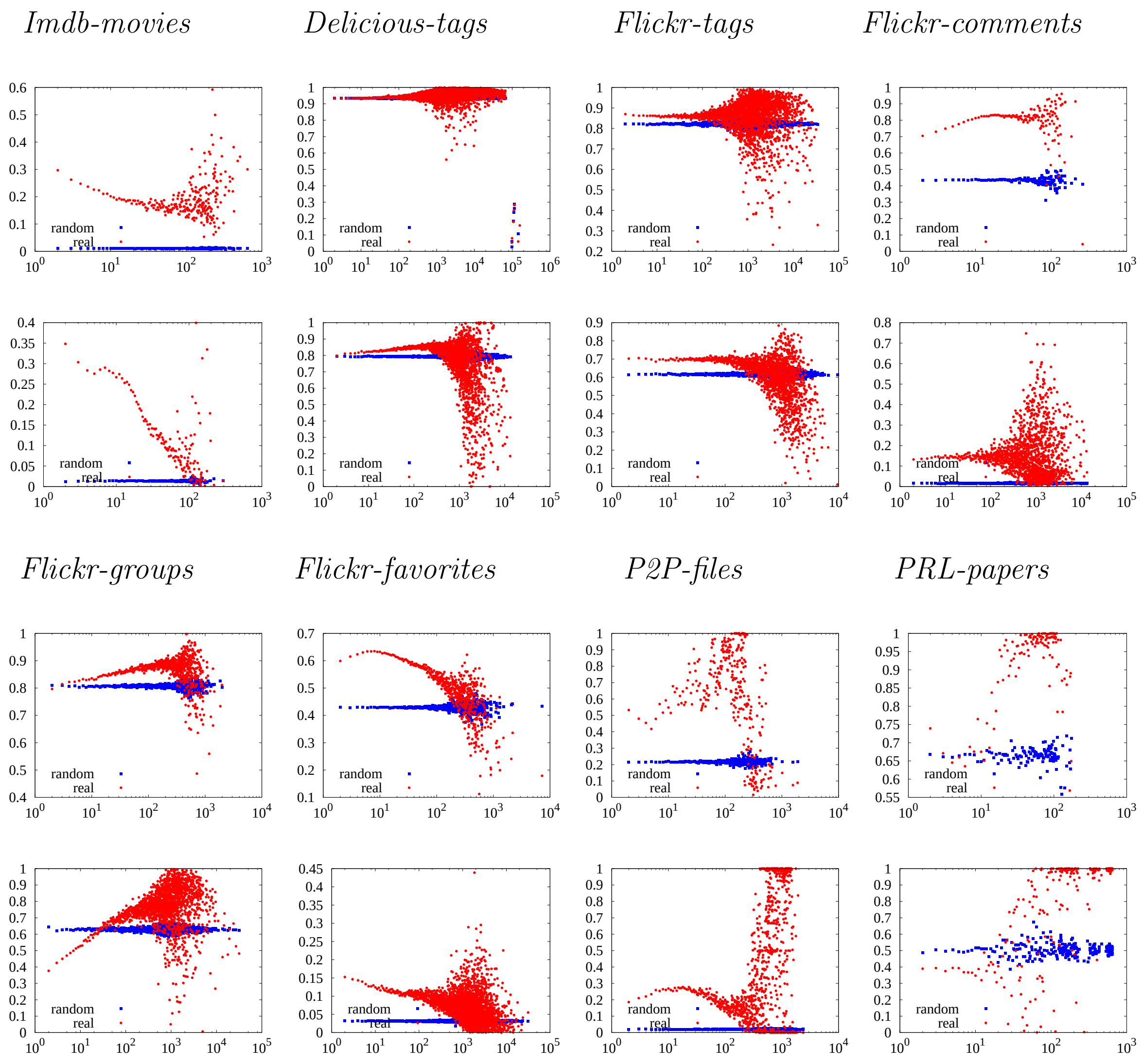


Figure 3.7 – Correlation of the redundancy coefficient with degree in our eight datasets, and in random bipartite graphs of the same size and same degree distributions. First row: for top nodes. Second row: for bottom nodes.

3.6 Conclusion

In this chapter, we have described the real-world bipartite complex networks which we use in the following chapters. The main observations that we made are:

– top and bottom degree distributions are very heterogeneous. The average degree of neighbors of nodes decrease, increases or have more complex behavior with the node degree, depending on the datasets,

– the three notions of clustering coefficient give complementary insight on the datasets properties, and
– the redundancy coefficient is larger in real-world networks than in random graphs. However, this depends on the density of the projection of the graph.

Our real-world networks have the advantage of spanning well the variety of cases met in practice. We have observed that they have different properties. This is an important point that will be useful in the next chapter because they are representative of many different behaviors.

Chapter 4

Internal links and pairs

Contents

4.1 Introduction

Many real-world complex networks are best modeled as bipartite graphs. However, there is currently a lack of methods to analyze properly such graphs. Despite previous efforts to develop such methods [79, 88, 140], much remains to be done in this direction.

We introduce here *internal links* and *pairs* as a new notion useful for such analysis. We illustrate its relevance on several real-world cases and show that it is also interesting for storing and modeling complex networks. We also show that it plays an important role in the dynamics of bipartite networks.

The chapter is organized as follows. We introduce *internal links* and *pairs* in Section 4.2. We then analyze the real-world datasets presented in Chapter 3 with regard to this new notion in Section 4.3. We explore a more algorithmic perspective in Section 4.4, and the role of this notion regarding the dynamics of real-world bipartite networks in Section 4.5. We finally present our conclusions and perspectives in Section 4.6.

4.2 Internal pairs and links

Let us consider a bipartite graph $B = (\perp, \top, L)$. For any pair of nodes $(u, v) \notin L$, we denote by $B + (u, v)$ the graph $B' = (\perp, \top, L \cup \{(u, v)\})$ obtained by adding the new link (u, v) to B. For any link $(u, v) \in L$, we denote by $B - (u, v)$ the graph $B' = (\perp, \top, L \setminus \{(u, v)\})$ obtained by removing the link (u, v) from B.

Definition 1 (internal pairs) *A pair of nodes $(u, v) \in \perp \times \top$ with $(u, v) \notin L$ is a $\perp$-*internal *pair of B if the $\perp$-projection of $B' = B + (u, v)$ is identical to the one of B. We define $\top$-internal pairs dually.*

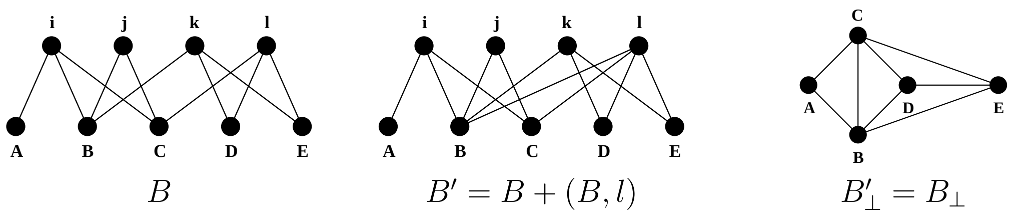


Figure 4.1 – **Example of a $\perp$-internal pair**. Left to right: a bipartite graph B, the bipartite graph B' obtained by adding the link (B, l) to B, and the $\perp$-projection of these two graphs. As $B'_\perp = B_\perp$, (B, l) is a $\perp$-internal pair of B.

Definition 2 (internal links) *A link* $(u, v) \in L$ *is a* $\perp$-internal *link of* B *if the* $\perp$-*projection of* $B' = B - (u, v)$ *is identical to the one of* B*. We define* $\top$*-internal links dually.*

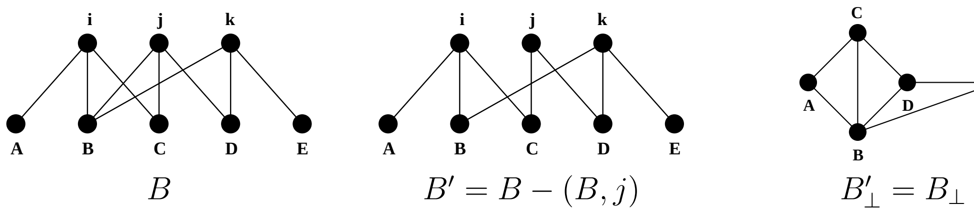


Figure 4.2 – **Example of a $\perp$-internal link**. Left to right: a bipartite graph B, the bipartite graph B' obtained by removing link (B, j) from B, and the $\perp$-projection of these two graphs. As $B'_\perp = B_\perp$, (B, j) is a $\perp$-internal link of B.

In other words, (u, v) is a $\perp$-internal pair of B if adding the new link (u, v) to B does not change its $\perp$-projection; it is a $\perp$-internal link if removing link (u, v) from B does not change its $\perp$-projection. See Figures 4.1 and 4.2 for examples.

The notion of internal link is related to the redundancy of a node [79], defined for any node v as the fraction of pairs in $N(v)$ that are still linked together in the projection of the graph B' obtained from B by removing v and all its links (all these pairs are linked in $B_\perp$). There is however no direct equivalence between the two notions. The redundancy is a node-oriented property: it gives a value for each node, while the notion of internal links and pairs is link-oriented. As illustrated on Figure 4.3, nodes exhibiting the same fraction of internal links may have different redundancies, and conversely two nodes having the same redundancy may correspond to different internal connectivity patterns.

It is possible to classify the links of each node as internal or not; this induces a notion of $\perp$-*internal degree* and $\top$-*internal degree* of a node, which is its number of $\perp$-internal links and $\top$-internal links, respectively.

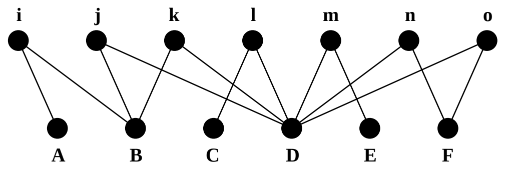


Figure 4.3 – **Redundancy versus internal links.** In this graph, B and D have the same fraction of $\perp$-internal links ($\frac{2}{3}$) while having different redundancies (resp. $\frac{1}{3}$ and $\frac{2}{15}$).

We now give a characterization of internal links which does not explicitly rely on the projection anymore and provides another point of view on this notion.

Lemma 1 *A link (u, v) of B is $\bot$-internal if and only if $N(v) \setminus \{u\} \subseteq N(N(u) \setminus \{v\})$.*

Proof : Let us consider a link $(u, v) \in L$ and let $B' = B - (u, v)$ be the bipartite graph obtained by removing the link (u, v) from B. Then, by definition, $L_\bot = L'_\bot \cup \{(u, x),\ x \in N(v) \setminus \{u\}\}$.

Suppose that (u, v) is a $\bot$-internal link, *i.e.* $L_\bot = L'_\bot$. Then all links (u, x) in the expression above already belong to $L'_\bot$. Therefore, for each $x \in N(v) \setminus \{u\}$, $\exists\ y \neq v \in \top$ such that $y \in N(u) \cap N(x)$. By symmetry, $x \in N(y)$ and $y \in N(u) \setminus \{v\}$ therefore, $x \in N(N(u) \setminus \{v\})$ and so $N(v) \setminus \{u\} \subseteq N(N(u) \setminus \{v\})$.

Suppose now that $N(v) \setminus \{u\} \subseteq N(N(u) \setminus \{v\})$. Then for each node $x \in N(v) \setminus \{u\}$, $\exists\ y \in N(u) \setminus \{v\}$ such that $x \in N(y)$. Thus, by definition of the projection, $(u, x) \in L'_\bot$. Therefore $L_\bot = L'_\bot$ and the link (u, v) is $\bot$-internal.

$\square$

4.3 Analysis of real-world cases

In this section, we use the notions of internal links and pairs introduced in Section 4.2 to describe the real-world cases presented in Chapter 3. Let us insist on the fact that our aim is *not* to provide accurate information on these specific cases, but to illustrate how internal links and pairs may be used to analyze real-world data. We first show that there are many internal links in typical data, then study the number of internal links of each node and the correlation of this number with the node's degree.

Since the links attached to $\top$-nodes (resp. $\bot$-nodes) of degree 1 are all $\bot$-internal (resp. $\top$-internal), and since there may be a large fraction of nodes with degree 1 in real-world graphs, we only study in the sequel links attached to nodes with degree at least 2.

4.3.1 Amount of internal links and pairs

In order to capture how redundant is the bipartite structure, we compute the number of $\top$- and $\bot$-internal pairs and links. The fraction of internal links, denoted f_{E_I} and presented in Table 5.1, seems in general not negligible. A quantitative analysis of these values however requires the definition of a benchmark. That is why we compare the measures to the corresponding amounts on random bipartite graphs with the same sizes and degree distributions, which is a typical random model to evaluate the deviation from an expected behavior – see for example [100, 101]. The measures related to this model will be referred to with the symbol *.

We denote by $\mathcal{P}_I(\perp)$ (resp. $\mathcal{P}_I(\top)$) the set of $\perp$-internal pairs (resp. $\top$-internal pairs) and by $E_I(\perp)$ (resp. $E_I(\top)$) the set of $\perp$-internal links (resp. $\top$-internal links). We normalize the number of internal pairs and links measured on real graphs to the values obtained with the model described above. The corresponding results are also presented in Table 5.1.

	$f_{E_I}(\perp)$	$\frac{\mathcal{P}_I(\perp)}{\mathcal{P}_I^*(\perp)}$	$\frac{E_I(\perp)}{E_I^*(\perp)}$	$f_{E_I}(\top)$	$\frac{\mathcal{P}_I(\top)}{\mathcal{P}_I^*(\top)}$	$\frac{E_I(\top)}{E_I^*(\top)}$
Imdb-movies	0.031	0.441	47.0	0.026	0.491	147
Delicious-tags	0.112	0.972	1.47	0.104	1.823	5.31
Flickr-tags	0.117	0.920	1.51	0.048	1.040	2.50
Flickr-comments	0.398	0.258	4.22	0.002	0.151	22.0
Flickr-groups	0.228	0.491	2.21	0.015	0.249	2.86
Flickr-favorites	0.172	0.574	2.02	0.002	0.704	12.4
P2P-files	0.337	0.082	8.53	0.136	0.092	1430
PRL-papers	0.718	0.033	7.17	0.487	0.001	11.2

Table 4.1 – Fraction of internal links (f_{E_I}), number of internal pairs ($\mathcal{P}_I$) and internal links (E_I) of real-world graphs normalized to the values on random bipartite graphs with the same size and same degree distributions.

We first notice that the behaviors in regards to the amount of internal links are very heterogeneous. Still some general trends can be underlined: in the random case, $\perp$- and $\top$-internal links are underestimated. So, the probability of having nodes sharing the same neighborhood is higher in real graphs than in random ones. We may indeed expect, for instance, that people participating to the same paper have a higher probability to be coauthors of another one than a random pair of authors.

Meanwhile the numbers of internal pairs are generally overestimated in random networks. To understand this effect, let us consider the extreme case where two $\perp$-nodes in a graph have either exactly the same neighborhood, or no common neighbors. Then all links are $\perp$-internal, and the graph does not contain any internal pair. This example suggests that the number of internal pairs is probably anti-correlated to the number of internal links.

In general, there is a correlation between the fact that the number of internal links is underestimated in random graphs and the fact that the number of internal pairs is overestimated, but this correlation does not hold in all cases. Moreover, there is no direct link between these observations and the sizes or average degrees of the considered graphs.

Finally, we observe a specific behavior for the two graphs which correspond to tagging databases, *i.e. Delicious-tags* and *Flickr-tags*. For these graphs we observe the lowest gaps between the real and random cases for $\perp$-internal links and the amounts of $\perp$-internal pairs are very close in the real and random cases. Conversely, they are the only graphs for which the amount of $\top$-internal pairs is underestimated in random graphs.

Since we can observe a wide range of behaviors both for $\top$- and $\bot$-internal links and pairs, we will restrict our analysis in the following to $\bot$-internal links and pairs for the sake of brevity. We will see that this allows enlightening observations.

4.3.2 Distribution of internal links among nodes

The notion of internal links partitions the links of each node into two sets: the internal ones and the others. We now study how the fraction of internal links is distributed among nodes. On Figure 4.4, we plot the complementary cumulative distribution of the fraction of internal links per node for the datasets under study. We also plot the complementary cumulative distribution for random graphs.

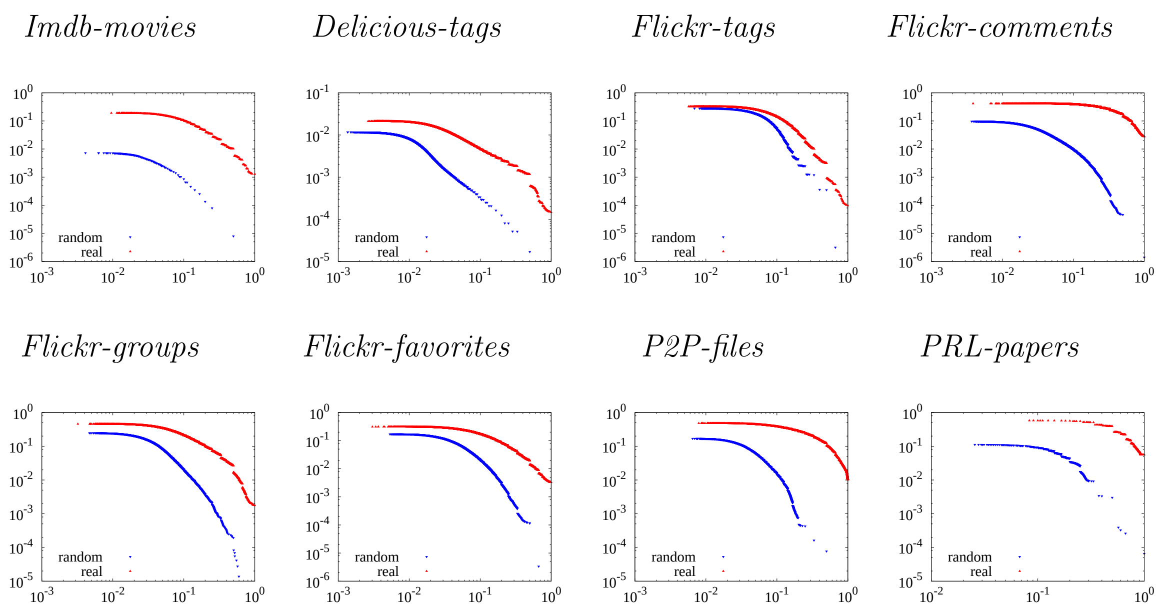


Figure 4.4 – Complementary cumulative distribution of the fraction of internal links per node.

One of the most noticeable differences between both curves lies in the probability of having a node whose links are all internal ($x = 1$): this fraction is indeed much higher in real than in random graphs. We also observe that real graphs exhibit fewer nodes with very low (or null) fractions of internal links (though the fraction of nodes with *no* internal link is high in both cases). In this respect too, the datasets behave differently: for *Imdb-movies* the probability of having a 10^{-2} fraction of internal links is more than one order of magnitude larger in the random than in the real graph, while *Flickr-tags* curves are close to be superimposed at low fractions. Notice that this is not directly related to the fact that the number of internal links is underestimated or not in random graphs: for *Delicious-tags* the ratio between the number of $\perp$-internal links in the real and in the random case is smaller than for *Flickr-tags*, but the difference between the distributions of the fraction of internal links per node are larger for *Delicious-tags* than for *Flickr-tags*.

Finally, the very low fractions that we observe are associated to nodes with high degree: to have a 10^{-4} fraction of internal links, a node has to have a degree of at least 10^4. Therefore, we study in the following the correlation between the degree of a node and its number of internal links.

4.3.3 Correlation of internal links with node degrees

We call the number of internal links of a node its internal degree, its total number of links being its degree. We investigate in this section the relationship between both quantities, plotting on Figure 4.5 the average degree of a node in regards its ($\perp$-)internal degree for the real datasets and the randomized ones.

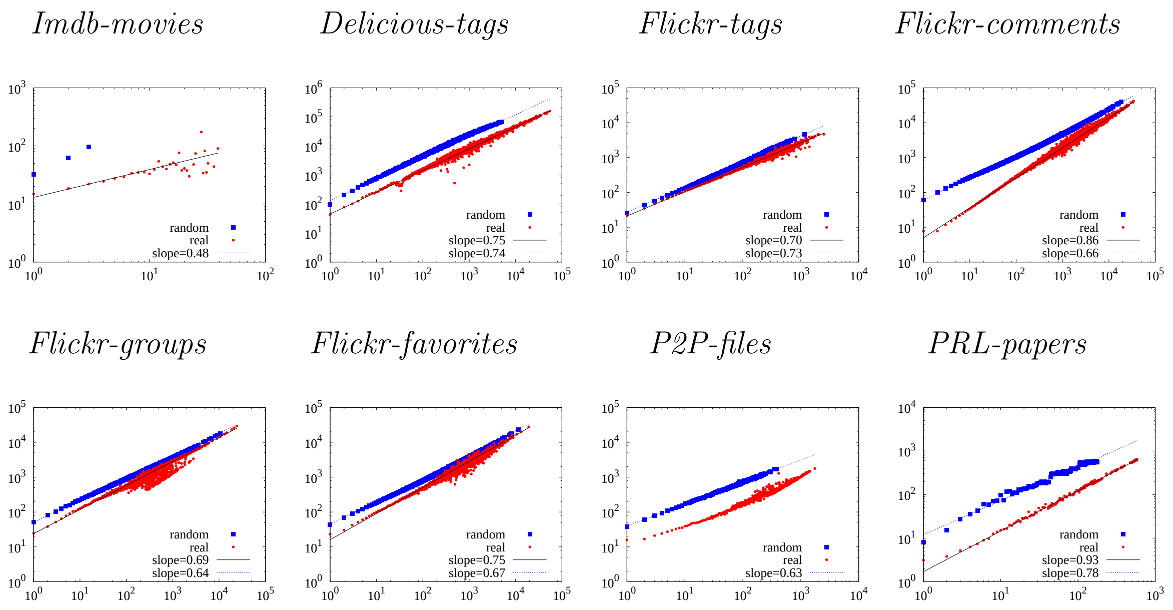


Figure 4.5 – Average degree as a function of the internal degree (for users projection).

We observe that both real and random curves in several cases can be approximated by a sub-linear law on several decades. However, this model is unsatisfactory on *P2P-files* database, and questionable on cases where the values are too rare or too scattered: most noticeably *Imdb-movies* and *Flickr-groups*. The dispersion observed at large degrees is a consequence of the heterogeneous degree distribution, the number of nodes with high degree being low.

If the fact that a given link is internal or not was independent from the node's degree, these curves would be linear. As random graphs have a sublinear behavior, that means that nodes with large degrees have on average a higher fraction of internal links. This effect can be explained qualitatively: increasing the degree of a node u – everything being otherwise unchanged – implies increasing the probability that one of his neighbors v is such that $N(v) \setminus \{u\} \subseteq N(N(u) \setminus \{v\})$.

On the other hand, the slope for real graphs is in most cases larger than for the random ones – again tagging datasets exhibit a different behavior. So there is an additional effect leading high degree nodes to have not as high an internal degree as expected by considering only the degree distributions. This is consistent with previous observations: the real case provides more internal links and fewer nodes with a low (but not null) fraction of such links, which must be high degree nodes. This stems from the fact that if nodes u and v are neighbors, the probability that $N(v) \setminus \{u\} \subseteq N(N(u) \setminus \{v\})$ is all the more important if v has a small degree and u a large one. Therefore we expect that degree-correlated graphs yield larger slopes than degree-anticorrelated ones. Yet, a more quantitative understanding of these phenomena calls for a study of the degree correlations in real-world graphs.

4.4 Removing internal links

When modeling complex networks using bipartite graphs [63, 101], the presence of internal links may be a problem as they are poorly captured by models. To this regard, removing internal links before generating a random bipartite graph may lead to better models. Moreover, internal links are precisely these links in a bipartite graph which may be removed without changing the projection. As the bipartite graph may be seen as a compact encoding of its projection [79], one then obtains an even more compressed encoding. Considering the example of the *P2P-files* dataset, it demands 30 MB if stored as a usual bipartite table of lists, while the corresponding $\bot$-projection (i.e. users) demands 213 MB and the $\top$-projection demands 4.6 GB.

However, removing internal links is not trivial, as removing one specific link (u, v) may change the nature of other links: while they were internal in the initial graph, they may not be internal anymore after the removal of (u, v). See Figure 4.6 for an example. Therefore, in order to obtain a bipartite graph with no internal link but still the same projection (and so a *minimal* graph to this regard), it is not possible in general to delete all initial internal links since this would alter dramatically the structure of the projection. The set of internal links must therefore be updated after each removal. Going further, there may exist removal strategies which maximize the number of removals, whereas others may minimize it.

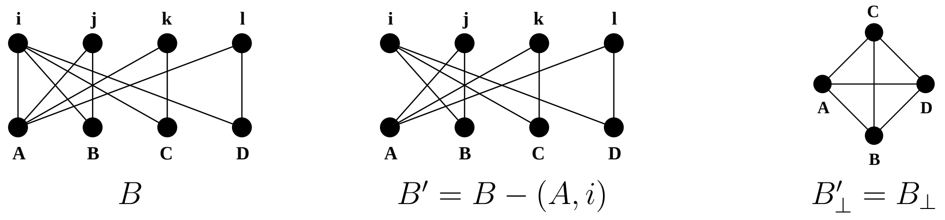


Figure 4.6 – **Influence of the deletion process on internal links.** $\{(A, i), (B, j), (C, k), (D, l)\}$ are $\perp$-internal links of B, yet deleting (A, i) leads to B' where $\{(B, j), (C, k), (D, l)\}$ are no longer $\perp$-internal links, as they are the only links in B' ensuring that A is connected to respectively B, C and D in $B_\perp$.

To explore these questions, let us consider a random removal process, where each step consists in choosing an internal link at random and removing it, and we iterate such steps until no internal link remains. Figure 4.7 presents the number of remaining internal links as a function of the number of internal link removed for typical cases. We also plot the upper bound $E_I - x$ (where x denotes the number of link removals), which represents the hypothetical case where all links initially internal remain internal during the whole process.

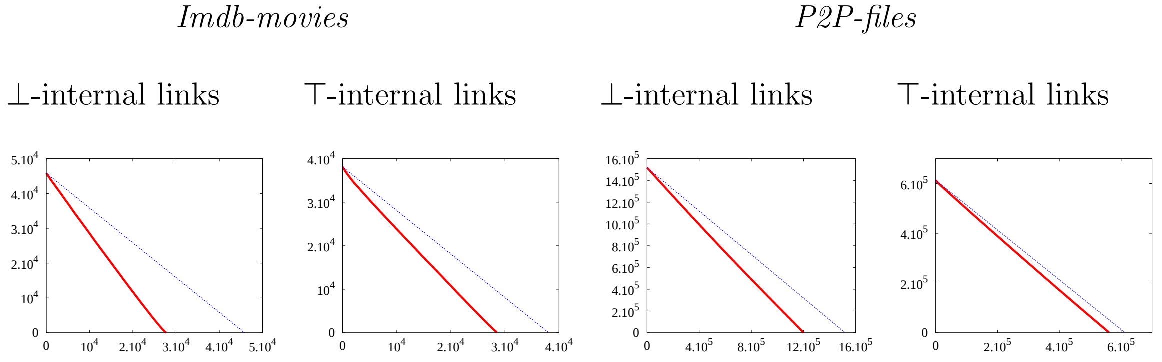


Figure 4.7 – Number of internal links remaining as a function of the number of deletions. Red thick line: random deletion process, blue thin line: theoretical upper bound.

This random deletion process leads to a pruned bipartite graph, containing the information of the 1-mode graph. Going back to the example of the *P2P* dataset, the obtained bipartite storage graph demands 12 MB for the related $\perp$-projection and 22 MB for the $\top$ one, thus enabling a compression to 0.40 (resp. 0.73) when compared to the standard 30 MB bipartite representation of the graph (which is itself a compact encoding of the projection).

To go further, one may seek strategies that remove as many internal links as possible, for instance using a greedy algorithm selecting at each step the internal link leading to the lowest decrease of the number of remaining internal links, but this is out of the scope of the current work.

4.5 Dynamics of real-world cases

Using the formalism described in Section 2.7, let us consider a set $D = \{(t_i, u_i, v_i), i = 1...n\}$ of timestamped links and the reference graph $B = (\bot, \top, L)$ observed during the reference period $[a, b[$. Our aim here is to give an insight of the reference graph's dynamics. In particular, we study whether the new links appearing among nodes of B after date b, during a *future* period $[b, c[$ for a given $c > b$, are *internal pairs* in the reference graph.

We consider only the links between nodes of B (we ignore new nodes appearing in the future period $[b, c[$) which are not present in B: we consider links in $\bot \times \top \setminus L$ only. This leads to the set $L' = \{(u, v), \exists(t, u, v) \in D \text{ s.t. } b \leqslant t < c\} \cap (\bot \times \top \setminus L)$ of new links.

We present the results obtained for *P2P-files*, *Delicious-tags* and *Flickr-comments*.

4.5.1 Impact of the future period

In order to gain more insight on the dynamics of our datasets, let us consider the number $|L'|$ of new links appearing during the future period $[b, c[$, for $c = b+1, b+2..., b+n$. Results are presented in Figure 4.8 (first row). For each dataset, we close a duration b of the reference period $[0, b[$ which is representative of wide range of reference period durations for the dataset. We chose the units by which we increase the future period duration accordingly.

The number of new links grows rapidly with the duration of the future period, showing that many new links appear between nodes of the reference period, even after a long time. As one may expect, though, the number of new links grows faster during the first few days.

The fraction of internal pairs among these new links is presented in Figure 4.8 (second row). It is very high, with a maximal at almost 45%, 21% and 40% for *P2P-files*, *Delicious-tags* and *Flickr-comments* respectively. The fraction is stable or even slightly increasing for *Delicious-tags* and *Flickr-comments* and in *P2P-files* it decreases as the duration of the future period grows, but it remains above 25% for future periods of up to 50 days.

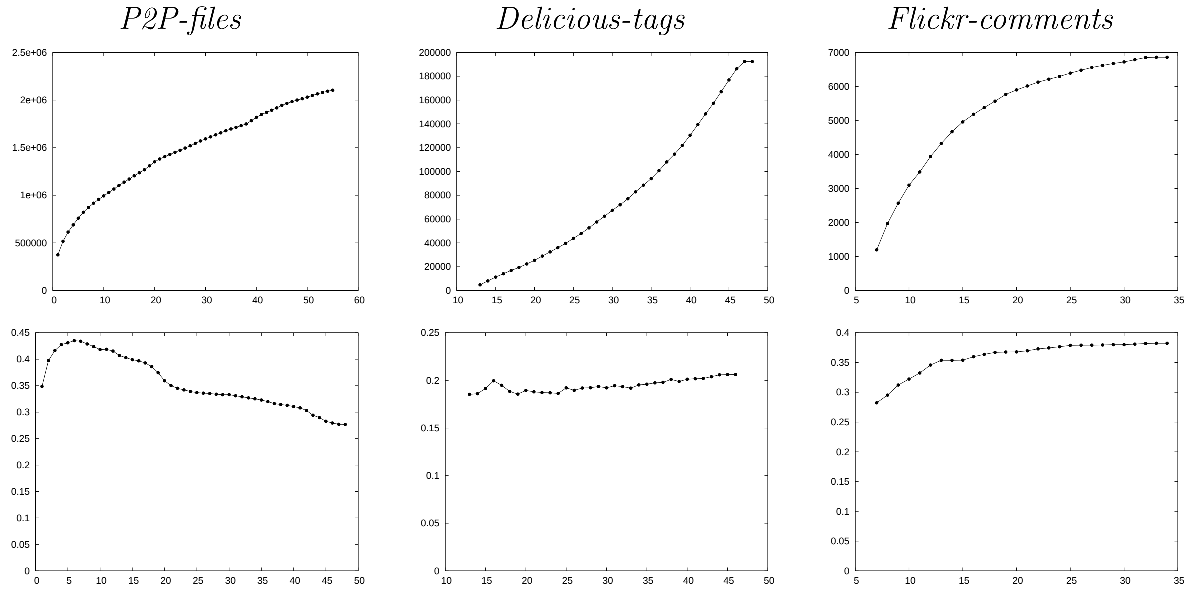


Figure 4.8 – Number of new links (first row) and fraction of internal pairs among them (second row) as functions of the future period duration. **Left**: *P2P-files* with a reference period $[0, 1\text{day}[$ and a future period $[1, x[$, for $x = 2, ..., 55$ days (horizontal axis, in days). **Center**: *Delicious-tags* with a reference period $[0, 12$ months$[$ and a future period $[12, x[$, for $x = 13, ..., 46$ months (horizontal axis, in months). **Right**: *Flickr-comments* with a reference period $[0, 6$ months$[$ and a future period $[6, x[$, for $x = 7, ..., 34$ months (horizontal axis, in months).

4.5.2 Impact of the reference period

Let us now observe how the number of new links $|L'|$ and the fraction of internal pairs among them evolves as the duration of the reference period grows. We consider reference periods $[0, b[$, for $b = 1, 2, ..., n$, and for each b we consider a fixed duration α of the future period $[b, b + \alpha[$ (α depends to the dataset). We present the number of new links in Figure 4.9 (first row). We observe that it grows rapidly with the reference period duration b. The fraction of internal pairs among these new links is presented in Figure 4.9 (second row):

- in *P2P-files*, it increases from 35% to 45% for reference periods from 1 to 6 hours. After this it decreases slowly but remains above 28% for reference periods of up to 48 hours.
- in *Delicious-tags*, it increases slowly from 20% to 27% for reference periods from 1 to 24 months.
- in *Flickr-comments*, it decreases rapidly from 70% to 20% for reference periods from

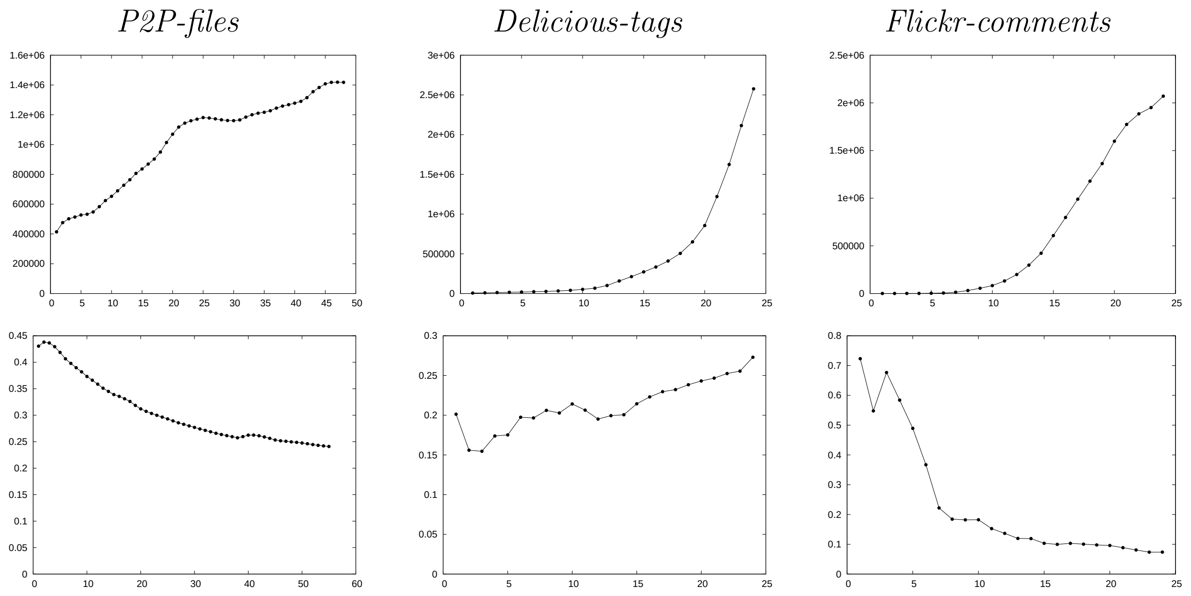


Figure 4.9 – Number of new links (first row) and fraction of internal pairs among them (second row) as functions of the reference period duration, **Left**: *P2P-files* with a reference period $[0, x[$, for $x = 1, 2, ..., 48$ hours and a observation period $[x, x + 15$ days$[$ of 15 days (horizontal axis, in hours). **Center**: *Delicious-tags* with a reference period $[0, x[$, for $x = 1, 2, ..., 24$ months and a observation period $[x, x+24$ months$[$ of 24 months (horizontal axis, in months). **Right**: *Flickr-comments* with a reference period $[0, x[$, for $x = 1, 2, ..., 24$ months and a observation period $[x, x + 12$ months$[$ of 12 months (horizontal axis, in months).

1 to 7 months. After this it decreases slowly.

These statistics show that the fraction of new links which were internal pairs is very high in our datasets, even for long reference and future periods, and therefore that internal pairs play a strong role in the dynamics.

4.6 Conclusion

In this chapter, we introduced the notion of internal links and pairs in bipartite graphs, and proposed it as an important notion for analyzing real-world bipartite complex networks. Using a wide set of real-world examples, we observed that internal links are very frequent in practice, and that associated statistics are fruitful measures to point out similarities and differences among real-world networks. This makes them a relevant tool for analysis of bipartite graphs, which is an important research topic. Moreover, removing internal links may be used to compact bipartite encodings of graphs and to improve their modeling. We finally observed that internal pairs become internal links with high probability in the future evolution of the graph, and that they constitute a significant part of new links. This shows that internal links and pairs are also important for the dynamics of real-world complex networks.

We provided a first step towards the use and understanding of internal links and pairs. Further investigations could bring more precise information about the role of internal links, in particular regarding the dynamics. One may also study the links (and pairs) which are both $\bot$- and $\top$-internal, as they may have a special importance in a graph.

Chapter 5

Link prediction in bipartite graphs

Contents

5.1 Introduction

Many real-world complex networks are dynamic: they evolve during time, with node and link additions and removals. Studying such dynamics is extremely important for our understanding of these objects, but very limited knowledge and ground methodology is available, even for classical (non-bipartite) dynamic graphs.

One of the main approaches developed for studying graph dynamics is *link prediction* [28, 67, 87], which consists in predicting the links that will probably appear in the future, given a snapshot of the considered graph at a given time.

We address here the problem of link prediction in dynamic bipartite graphs. To do so, we propose an approach based on *internal pairs*, introduced in the previous chapter, and compare it to a classical approach. We study the performance of our method on real-world datasets using wide ranges of parameters and detail the results for a P2P-file graph. We show that this method reaches very good performances and that internal pairs play a key role in the dynamics of real-world bipartite graphs.

The chapter is organized as follows. We review related work in Section 5.2. We then formally state the considered problem and its assessment in Section 5.3. We present in Section 5.4 our prediction method and a basic method which we consider for comparison purpose. We finally present our experimental setup in Section 5.5 and the results of our experiments in Sections 5.6 and 5.7. We discuss our conclusions and perspectives in Section 5.8.

5.2 Related work

Link prediction is a key research problem in network dynamic analysis. Several works study this problem on classical (non-bipartite) graphs. Most of them are based on measures of similarity between nodes. For instance, in [87] the authors examine several topological measures (such as Jaccard coefficient, Adamic/Adar coefficient, SimRank, etc.) based on node neighborhoods and the set of all paths between nodes. They use these measures for ranking possible future co-authors collaborations. In [69] the author proposes to use another topological measure called generalized clustering coefficient. In [67, 103] the authors add several non-topological measures based on node attributes (such as keyword match, number of papers, geographic proximity, KL-divergence of two nodes' topic distribution, etc.) and they use a supervised learning algorithm to perform link prediction. In a similar way, the authors of [29] predict co-authoring of publications by using topological measures computed in the co-authoring graph and indirect topological measures computed using the co-author graph (where two papers are linked if they are signed by a same author). The authors of [134] add another measure (local probabilistic model) to estimate the co-occurrence probability of two nodes, and in [129] the authors extend this by incorporating available temporal information. The authors of [19] propose an approach based on supervised random walks. This approach uses the nodes' and links' attributes to guide the random walk for visiting the nodes to which new links will be created in the future. Finally, the authors of [45] use a hierarchical decomposition of a social network for predicting missing links. They generate a set of hierarchical random graphs, and they compute average probability of connection between two nodes within these hierarchical random graphs.

Works presented above deal with classical (non-bipartite) graphs, and are not directly applicable to or appropriate for bipartite graphs.For instance, the methods based on the immediate common neighborhood of two nodes rely on the presence of triangles in the graph. However, there are no triangles in the bipartite graphs. The methods based on the paths between nodes or random walk must be adapted to take into account only paths of odd length, since a link can only appear between two nodes if they are already connected by paths of odd lengths in a bipartite graphs.

Up to our knowledge, only two papers target the problem of link prediction in bipartite graphs [28,70]. In [70], the authors adapt some topological measures used in classical graphs for predicting links in bipartite graphs. For each possible link (u, v), the authors compare the neighbors of u in the bipartite graph and neighbors of neighbors of v. They also study the set of all paths between nodes in the bipartite graph. Going further, the authors of [28] consider two transformations of the bipartite graph into a classical one, then for predicting link (u, v) they consider the classical graph containing u, and they study the topological measures between u and the neighbors of v in the bipartite graph, and conversely. They apply a supervised learning algorithm to obtain the results.

Another research problem is closely related to link prediction in bipartite graphs: the recommendation problem [112]. Recommendation systems are used to suggest items to users, such as products to customers for instance. Notice however that the two problems are quite different: recommendation aims typically at finding a small number of products of interest for each customer; prediction aims at finding links that will appear in the future. Predicting that a given node will have a huge number of new links while many others will not have any is of little interest regarding recommendation but may be a great success regarding prediction.

Various approaches have been developed for recommendation [20,25,74], with collaborative filtering being the most successful and widely used approach [74]. Two main approaches of collaborative filtering have been proposed, both based on the idea that similar users will purchase similar items and that users will purchase items similar to the ones they already purchased. The first approach consists in predicting the rating of a given user for a given item [111]. It relies on known rating data (e.g. explicit users opinions for items, rated on a scale of 1 to 5). The second approach consists in ranking the most relevant items for a given user in order of decreasing interest, and then in recommending the top N items to this user [49, 89]. This approach does not require explicit ratings but only the information of which users adopted which items, and is the most similar to link prediction. We will use such an approach in this chapter for the purpose of comparison with our method, see Section 5.4.2.

5.3 The bipartite link prediction problem

Using the formalism described in Section 2.7, let us consider a set $D = \{(t_i, u_i, v_i), i = 1...n\}$ of timestamped links, the reference graph $B = (\bot, \top, L)$ observed during the reference period $[a, b[$, and the set $L' = \{(u, v), \exists(t, u, v) \in D \text{ s.t. } b \leqslant t < c\} \cap (\bot \times \top \setminus L)$ of links added to B during period $[b, c[$ which we call now the *prediction period*. Notice that we consider only the links between nodes of B (we ignore new nodes appearing in the period $[b, c[$) which are not present in B: we consider links in $\bot \times \top \setminus L$ only.

The goal of a link prediction method is to find a set P of *predicted links* which contains many of the links in L' but only few which are not in L'. Notice that in the extreme case where one predicts *all* possible links, *i.e.* $P = \bot \times \top \setminus L$, then one succeeds in predicting all links of L' but also predicts many links which are not in L'. Conversely, predicting no link at all, *i.e.* $P = \emptyset$, trivially does not predict links not in L' but fails in predicting any link in L'.

Evaluating the performances of a prediction method therefore consists in evaluating its success in reaching a tradeoff regarding these two objectives, which is non-trivial. We present below a classical method to do so [43, 118], which we use in this chapter.

Let us denote by $\overline{P}$ the set of links that the method predicts will not appear: $\overline{P} = (\bot \times \top \setminus L) \setminus P$. Figure 5.1 illustrates the four possible cases which may occur during link prediction: the set $P \cap L'$ of *true positives* is the set of appearing links that the method successfully predicts; the set $\overline{P} \setminus L'$ of *true negatives* is the set of unpredicted links which indeed do not appear; conversely, the *false positives* are the links in $P \setminus L'$, *i.e.* the links which we predicted but do not appear, and the *false negatives* are the links in $\overline{P} \cap L'$.

The aim of a link prediction method is to maximize the number of true positives and negatives while minimizing the number of false positives and negatives. This is captured by two quantities, called *precision* and *recall*.

The *precision* is the fraction of true positives among the predicted links, *i.e.* $\frac{|P \cap L'|}{|P|}$. In other words, it is the probability that the method is right when it predicts that a given link will appear, and therefore is a measure of correctness.

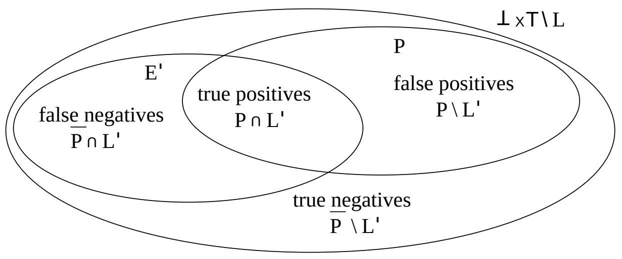


Figure 5.1 – A prediction method divides the set of possible links $\perp \times \top \setminus L$ into four categories: true positives, $P \cap L'$; true negatives, $\overline{P} \setminus L'$; false positives, $P \setminus L'$; and false negatives, $\overline{P} \cap L'$.

The *recall* is the fraction of true positives among the appearing links, *i.e.* $\frac{|P \cap L'|}{|L'|}$. In other words, it is the probability that an appearing link will indeed be predicted by the method, and so is a measure of completeness.

As explained above, there is a tradeoff between precision and recall, as, in general, improving one degrades the other and conversely. In order to capture this in a single value, which often is more convenient, one generally considers the *F-measure*, $\frac{2 \times |P \cap L'|}{|P| + |L'|}$, which is the harmonic mean of precision and recall [130]. The goal of a prediction method then is to maximize the F-measure.

5.4 Bipartite prediction methods

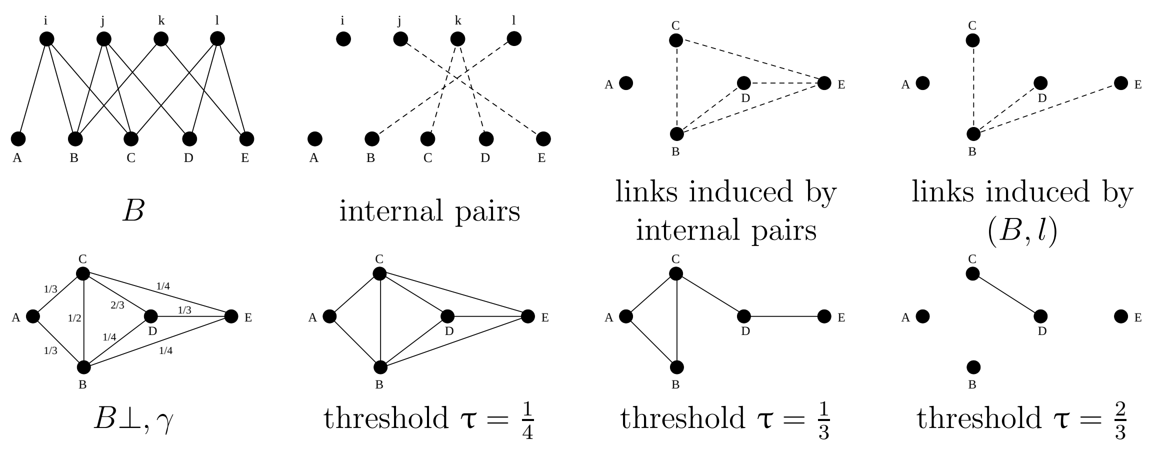
In this section, we introduce our link prediction method for bipartite graphs, which we call *internal link prediction*. We also present a typical collaborative filtering method which we use for comparison in the next sections.

5.4.1 Internal link prediction

The key feature of our prediction method is that it focuses on internal pairs: it predicts only links that are internal pairs in the reference graph. The intuition behind this is that two bottom nodes which already have a common neighbor in B (*i.e.* they are linked in $B_\perp$) will probably acquire more in the future. Instead, if two nodes have no common neighbor in B, then they will probably still have none in the future. The links that can be added to B which fit both criteria are precisely internal pairs.

Going further, two bottom nodes with many common neighbors in B are more likely to have more common new neighbors in the future than nodes which have only one neighbor in common. More generally, all the weight functions presented in Section 2.5 are measures (from different points of view) of our expectation that two nodes having at least one neighbor in common probably will have more in the future. Therefore, we expect that the links that will appear are the internal pairs inducing $\perp$-links with high weights.

This leads to the following prediction method, which we call *internal link prediction.* Let us consider a weight function ω like the ones described in Section 2.5, and a given weight threshold τ. We denote by $L_{\perp\tau} = \{(u,w) \in L_\perp,\ \omega(u,w) \geq \tau\}$ the set of links in the projection that have a weight larger than or equal to τ. We then predict all the internal pairs which induce at least one link in $L_{\perp\tau}$.



i j k l
A B C D E
B
internal pairs
links induced by internal pairs
links induced by (B, l)
1/3
1/4
2/3
1/2
1/3
1/4
1/3
1/4
$B\perp, \gamma$
threshold $\tau = \frac{1}{4}$
threshold $\tau = \frac{1}{3}$
threshold $\tau = \frac{2}{3}$

Figure 5.2 – **Example of internal link prediction**. First row (left to right): a bipartite graph B, the internal pairs of B, the $\perp$-links they induce, and the links of $B_\perp$ induced by the internal pair (B, l). Second row (left to right): the *Jaccard* weighted $\perp$-projection of B $(B_\perp, \gamma)$, and the links $L_{\perp\frac{1}{4}}$, $L_{\perp\frac{1}{3}}$, $L_{\perp\frac{2}{3}}$ obtained using thresholds τ respectively equal to $\frac{1}{4}$, to $\frac{1}{3}$ and to $\frac{2}{3}$.

Figure 5.2 shows an example of internal link prediction using the *Jaccard* weight function, γ. The set of internal links of B is $\{(B,l),(C,k),(D,k),(E,j)\}$; let us focus on the internal pair (B,l). It induces (B,C), (B,D), and (B,E). Given a threshold τ we predict (B,l) if one of these links has weight at least τ. For instance:

- if $\tau = \frac{1}{4}$, all links in the projection have a weight larger than or equal to τ, and so we predict all possible internal pairs in the bipartite graph, including (B,l);
- if $\tau = \frac{1}{3}$, only 5 links in the projection have weight larger than or equal to τ, including (B,C), which is induced by (B,l); we therefore predict (B,l);
- if $\tau = \frac{2}{3}$, only one link has the weight larger than or equal to τ, and it is not a link induced by (B,l); therefore we do not predict (B,l).

Algorithm 1 provides the details of the method useful for implementation, and Theorem 1 shows its complexity.

Algorithm 1: Internal link prediction

```
Input: bipartite graph $B = (\bot, \top, L)$, weight function $\omega$, threshold $\tau$
Output: set $P$ of predicted links
$P \leftarrow \emptyset$
for $u \in \bot$ do
    $Nbot \leftarrow \emptyset$
    for $v \in N(u)$ do
        for $w \in N(v) \setminus \{u\}$ do
            $Nbot \leftarrow Nbot \cup \{w\}$
            compute $\omega(u, w)$
    $I_u \leftarrow \emptyset$
    $d \leftarrow 0, 0, \dots, 0$
    $m \leftarrow \emptyset$
    for $w \in Nbot$ do
        for $v \in N(w)$ do
            if $\omega(u, w) \geqslant \tau$ then
                $m \leftarrow m \cup \{v\}$
            $d[v] \leftarrow d[v] + 1$
    for $v \in m$ do
        if $|N(v)| = d[v]$ then
            $I_u \leftarrow I_u \cup \{v\}$
    $P \leftarrow P \cup \{(u, v), v \in I_u\}$
```

Theorem 1 *Algorithm 1 performs internal link prediction on a bipartite graph $B = (\bot, \top, L)$ in time $\mathcal{O}(\Delta_\bot |L|)$, where $\Delta_\bot = \max_{u \in \bot} |N_\bot(u)|$ is the largest degree in $B_\bot$, and space $\mathcal{O}(|\top| + |\bot|)$ in addition to the space needed for storing B.*

Proof : We first show the termination and correctness of our algorithm. The algorithm consists of imbrications of `for` loops over finite and static sets, so it necessarily terminates.

To show its correctness, we must show that it predicts all the internal pairs that induce links in $L_{\bot\tau}$, *i.e.* links with weight larger than or equal to τ, and only those links.

The algorithm consists in a loop over all nodes u in $\bot$, with three main parts, each consisting in a `for` loop.

The first part computes $N_\bot(u)$, denoted by *Nbot* in the algorithm to emphasize that it is not precomputed nor stored, and the weights of the corresponding links. The nodes w added to *Nbot* are exactly those in $N(N(u))$, which by definition is exactly $N_\bot(u)$. Therefore $Nbot = N_\bot(u)$ at the end of the first part of the loop. We do not detail the weight computation here, because it depends on the weight function considered. We assume that it can be computed without time complexity overhead together with $N_\perp$, which is true for all the weight functions presented in Section 2.5.

The second part does two things. First it stores in a set m the nodes v such that (u, v) induces a link $(u, w) \in L_{\perp\tau}$ (notice that (u, v) is not necessarily an internal pair). Indeed, a node v is added to m as soon as it has a neighbor $w \in \perp$ such that $(u, w) \in L_{\perp\tau}$.

This loop also stores in $d[v]$ the number of neighbors of v which are neighbors of u in $B_\perp$. At the end of the second part of the loop, the nodes v for which $d[v] = |N(v)|$ are exactly the nodes v such that (u, v) is an internal pair. Indeed, $d[v]$ is incremented for each $w \in N(v)$ such that $w \in N_\perp(u)$. Therefore $d[v] = |N(v)|$ if and only if $N(v) \subseteq N_\perp(u) = N(N(v))$, *i.e.* if and only if (u, v) is an internal pair, from Lemma 1.

The third part of the loop then computes the intersection between the nodes v which correspond to internal pairs (those for which $d[v] = |N(v)|$) and the nodes corresponding to links which induce at least one link $(u, w) \in L_{\perp\tau}$ (the nodes in m).

Before entering in the details of the complexity analysis, let us discuss how sets may be efficiently managed in our algorithm. A set s of nodes in $\top$ (resp. $\perp$) may be represented by an array indexed by nodes in $\top$ (or $\perp$), such that $s[v] = 1$ if and only if $v \in s$, together with an array i_s containing the indexes of nodes in s, and an integer n_s representing the number of nodes in s. Therefore listing all nodes in s simply consists in listing the n_s first values of i_s. Adding a node v to s is performed in two steps: if $s[v] = 1$, do nothing; otherwise, set $s[v]$ to 1, increment n_s, and set $i_s[n_s]$ to v. To reinitialize s to $\emptyset$, iterate set $s[i_s[k]] \leftarrow 0$ for all $k \leq n_s$, then set $n_s = 0$. Finally, adding an element to a set requires constant time, and setting a set to $\emptyset$ requires as many operations as $|s|$, which is necessarily bounded by the time needed to populate the set, and therefore does not create any time complexity overhead. In our algorithm, the array d may be managed in a similar way: an additional array stores the indexes of nodes for which $d[v] \neq 0$, which allows to reset d to 0 without iterating over all nodes in $\top$.

This leads to the space complexity of our algorithm. With the encoding above, a set of nodes in $\top$ (resp. $\perp$) requires $\Theta(|\top|)$ (resp. $\Theta(|\perp|)$) space. Therefore, although the number of links in $B_\perp$ is huge in general compared to $|L|$, our algorithm only requires $\Theta(|\top| + |\perp|)$ space in addition to the space needed for storing its input B and its output P (which we only store for clarity but may not be stored).

Let us now study the time complexity of the algorithm. The main loop runs over all nodes in $\perp$ and performs three sets of operations. We will study the complexity of these parts independently.

The first part of the loops performs $\Theta(|N(v)|)$ operations for each node $v \in N(u)$.

Therefore the total number of operations for the first part of the main loop is of the order:

$$\Theta\left(\sum_{u\in\bot}\sum_{v\in N(u)}|N(v)|\right)=\Theta\left(\sum_{v\in\top}|N(v)||N(v)|\right).$$

We have that $|N(v)|\leq\delta_\top$, where $\delta_\top=\max_{v\in\top}|N(v)|$. Therefore the above quantity can be bounded:

$$\Theta\left(\sum_{v\in\top}|N(v)||N(v)|\right)\subseteq\mathcal{O}\left(\delta_\top\sum_{v\in\top}|N(v)|\right)=\mathcal{O}\left(\delta_\top|L|\right).$$

The second part of the loop performs $\Theta(|N(w)|)$ operations for each node $w\in N_\bot(u)$. The total number of operations in this part of the loop is therefore performed in time:

$$\Theta\left(\sum_{u\in\bot}\sum_{w\in N_\bot(u)}|N(w)|\right)=\Theta\left(\sum_{w\in\bot}|N_\bot(w)||N(w)|\right).$$

This expression can be bounded in two ways, by considering that $|N_\bot(w)|\leq\Delta_\bot$ (where $\Delta_\bot=\max_{u\in\bot}|N_\bot(u)|$ is the largest degree in $B_\bot$) or that $|N(w)|\leq\delta_\bot$ (where $\delta_\bot=\max_{v\in\bot}|N(v)|$). In the first case we obtain

$$\Theta\left(\sum_{w\in\bot}|N_\bot(w)||N(w)|\right)\subseteq\mathcal{O}\left(\Delta_\bot\sum_{w\in\bot}|N(w)|\right)=\mathcal{O}\left(\Delta_\bot|L|\right).$$

In the second case we obtain

$$\Theta\left(\sum_{w\in\bot}|N_\bot(w)||N(w)|\right)\subseteq\mathcal{O}\left(\delta_\bot\sum_{w\in\bot}|N_\bot(w)|\right)=\mathcal{O}\left(\delta_\bot|L_\bot|\right).$$

Finally, the third part of the loop iterates over all nodes in m. Since m was computed in the second part of the loop, the number of operations in the third part of the loop is bounded by the one for the second part of the loop, therefore we do not need to evaluate it further.

The overall complexity of the algorithm is therefore in the order of

$$\mathcal{O}\left(\delta_\top|L|+\min(\Delta_\bot|L|,\delta_\bot|L_\bot|)\right)\subseteq\mathcal{O}\left((\delta_\top+\Delta_\bot)|L|\right)\subseteq\mathcal{O}\left(\Delta_\bot|L|\right),$$

because $\delta_\top\in\mathcal{O}(\Delta_\bot)$ since each node in $\top$ with degree d induces a clique of d nodes, each of them having a degree at least $d-1$ in $B_\bot$.

□

5.4.2 Collaborative filtering prediction

As explained in Section 5.2, typical collaborative filtering approaches consist in predicting that $\bot$-nodes tend to create links to the $\top$-neighbors of $\bot$-nodes which are similar to themselves (clients will buy products that similar clients already bought). More precisely, such methods first select for each $\bot$-node u the set of k $\bot$-nodes which are the most similar to u and then the N $\top$-nodes the most strongly linked to these nodes, for given parameters k and N. Here, natural notions of similarity between $\bot$-nodes are provided by the weighted $\bot$-projection.

We will therefore use the following collaborative filtering method [35]: given a weight function ω, we consider for each $\bot$-node u the set $U_k \subseteq N_\bot(u)$ of its k neighbors with largest weight. Then for each $v \in N(U_k) \setminus N(u)$, we compute the score $s(u,v)$ of the link (u,v) as the sum of the weights $\omega(u,w)$, for each $w \in N(v) \cap U_k$:

$$s(u,v) = \sum_{w \in U_k \cap N(v)} \omega(u,w).$$

There are other possible ways to compute the score [35, 40, 60, 68, 138], however in this chapter we restrict ourselves to the formula above which is typical. Finally, the collaborative filtering method predicts for each node the N links with highest scores.

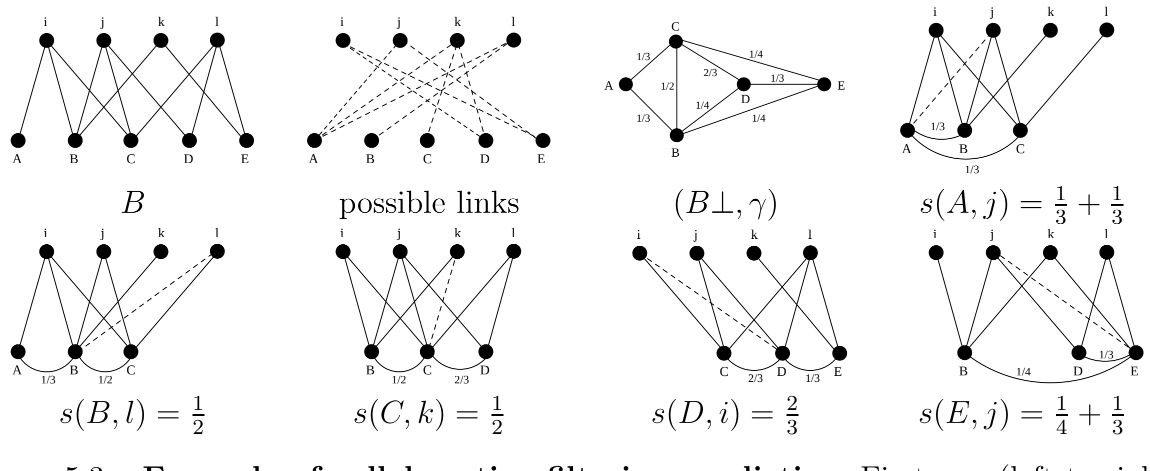


Figure 5.3 – **Example of collaborative filtering prediction**. First row (left to right): an example of a bipartite graph B, the set of all possible links that may appear, the *Jaccard* weighted $\bot$-projection of B $(B\bot, \gamma)$, and the result of the collaborative filtering prediction for node A. Second row (left to right): the result of collaborative filtering prediction for B, C, D and E. In this example, $k = 2$ (we consider 2 most similar neighbors) and $N = 1$ (we predict 1 link with highest score).

Figure 5.3 presents an example of collaborative filtering using the *Jaccard* weight function γ, $k = 2$ and $N = 1$. For instance, node E has three neighbors in the $\perp$-projection: $N_\perp(E) = \{B, C, D\}$. The method then considers the $k = 2$ most similar ones, *i.e.* the ones linked to E with the largest weight in $B_\perp$. (E, D) has the largest weight. As (E, B) and (E, C) have the same weight, the method chooses at random between B and C for the second link. Suppose that $U_2 = \{B, D\}$. The method then computes the score of links (E, v) for all $v \in (N(B) \cup N(D))\backslash N(E)$, leading to $s(E, j) = \gamma(E, B) + \gamma(E, D) = \frac{7}{12}$, and $s(E, i) = \gamma(E, B) = \frac{1}{4}$. Finally, as $N = 1$, it predicts the link with highest score *i.e.* (E, j).

Algorithm 2: Collaborative filtering

```
Input: bipartite graph $B(\perp, \top, L)$, weight function $\omega$, $k$, $N$
Output: set $P$ of predicted links
$P = \emptyset$
for $u \in \perp$ do
    $Nbot \leftarrow \emptyset$
    for $v \in N(u)$ do
        for $w \in N(v)\backslash\{u\}$ do
            $Nbot \leftarrow Nbot \cup \{w\}$
            compute $\omega(u, w)$
    $U \leftarrow$ sort$(Nbot(u), \omega)$
    $U_k \leftarrow$ head$(U, k)$
    $s \leftarrow 0, 0, \ldots, 0$
    for $w \in U_k$ do
        for $v \in N(w)\backslash N(u)$ do
            $s[v] \leftarrow s[v] + \omega(u, w)$
    $P_u \leftarrow$ sort$(s)$
    $P \leftarrow P \cup \{(u, v), v \in$ head$(P_u, N)\}$
```

Let us present the details in Algorithm 2. The first part of the loop computes $N_\perp(u)$. The corresponding time complexity is therefore in the order of $\mathcal{O}(\delta_\top|L|) \subseteq \mathcal{O}(\Delta_\perp|L|)$, as detailed in the proof of Theorem 1 (notations are defined there). The complexity of the second part of the loops depends on parameters k and N. The `sort` and `head` instructions are used to compute the k and N largest values in $N_\perp$ (according to weight function ω) and s, respectively. This can be done in constant time if k and N are constant, but we kept this notation to make the algorithm easier to read.

The loop iterating over all nodes $w \in U_k$ performs $|N(w)|$ operations at each step. This loop is repeated for all nodes $u \in \perp$. The total number of operations performed is bounded by $\mathcal{O}(\sum_{u\in\perp} \sum_{w\in N_\perp(u)} |N(w)|) \subseteq \mathcal{O}(\Delta_\perp|L|)$.

The total time complexity is therefore in the order of $\mathcal{O}(\Delta_\perp|L|)$, as Algorithm 1. The space complexity, besides the space needed to store B, is of the order of $|\top| + |\bot|$, this space being needed for storing $Nbot, U, U_k$, and s. This is the same as Algorithm 1.

5.5 Experimental setup

Evaluating our method in practice requires the availability of large scale bipartite data *with their dynamics*. One natural source for such data might be benchmarks for recommendation systems. However, such datasets often do not contain temporal information or have been filtered to fit recommendation needs (for instance, nodes with a small degree have been removed, as well as large degree ones). This makes them unusable in our context.

We did our best to obtain appropriate data for assessing our method and finally conducted experiments on various datasets, in particular *Delicious-tags*, *P2P-files* and *Flickr-comments* graphs. We detail here results obtained with a *P2P-files* dataset which are representative of all obtained results.

We first describe the real-world datasets. We then discuss appropriate parameters for our experimentation for both our method and the collaborative filtering one. We present the results of our experimentations in Section 5.6.

5.5.1 P2P-files dataset

We focus here on *file-provider* relations, where a set $D = \{(t_i, u_i, v_i)\}$ of triplets indicating that the server pointed peer u_i as a provider for file v_i at time t_i.

Using the formalism described in Section 5.3, each triplet corresponds to a link between a $\bot$-node (a peer) and a $\top$-node (a file) at time t_i. The $\bot$-projection $B_\bot$ of B is the graph in which two peers are linked if they provide one or more files in common. For two given timestamps x and y we consider:

– the reference period $[0, x[$ and the corresponding reference graph $B = (\bot, \top, L)$ induced by links observed from the beginning of the measurement (time 0) to time x, and
– the prediction period $[x, y[$ and the corresponding set of links $L' \subseteq (\bot \times \top) \setminus L$ added to B between x and y.

Basic features of the reference graph B are presented in Table 5.1, for different reference period durations x. Notice that the number of $\top$-nodes (files) is much larger than the number of $\bot$-nodes (peers), which is mostly due to the fact that we consider only peers which provide at least one file (most peers only download files).

	duration x of the reference period $[0, x[$				
	$x = 6$ hours	$x = 12$ hours	$x = 1$ day	$x = 3$ days	$x = 7$ days
number of $\bot$-nodes (peers)	$82,372$	$122,817$	$160,159$	$356,197$	$705,634$
number of $\top$-nodes (files)	$1,474,048$	$2,060,530$	$2,456,205$	$3,938,639$	$5,703,258$
number of links in L	$2,764,424$	$4,259,764$	$5,634,865$	$11,851,292$	$22,334,912$

Table 5.1 – Number of $\bot$-nodes (peers), $\top$-nodes (files), and links in the bipartite graph B, for five different reference period durations x.

5.5.2 Parameters for prediction methods

The performances of link prediction methods depend on various parameters. We explore in depth in the next section (Section 5.6) the impact of the reference and prediction period durations, as well as the impact of the weight functions. Even when these parameters are given, though, other parameters play a role: the weight threshold τ for *internal link prediction* and the values of N and k for collaborative filtering prediction. Exploring all possible values for all these parameters and their combinations is intractable in practice, and would actually have limited interest here as we are mostly concerned with *qualitative* results. We explain in this section how we choose values for these parameters for our experiments while avoiding extensive exploration of all possible values.

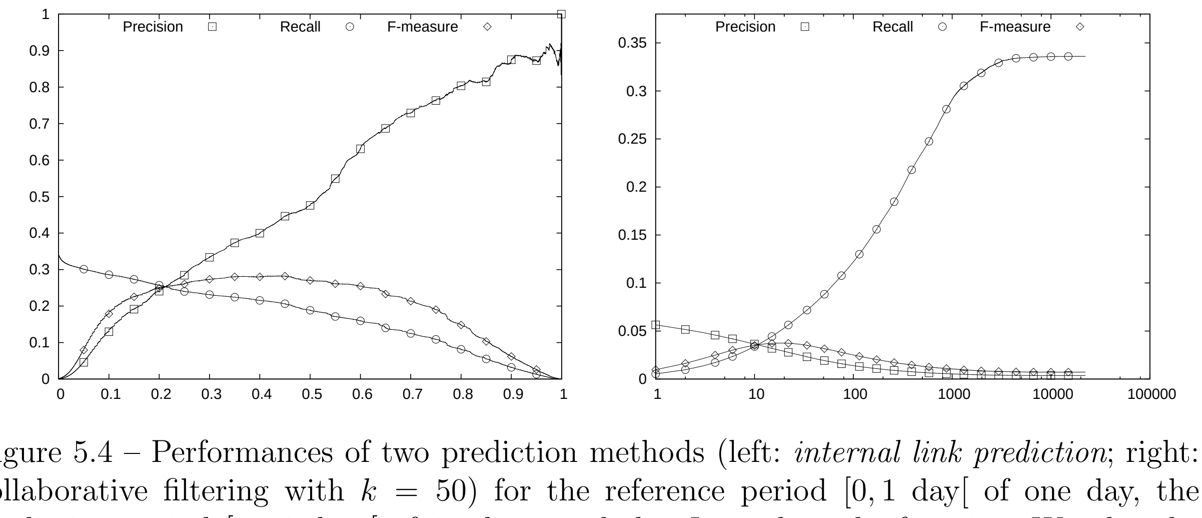


Figure 5.4 – Performances of two prediction methods (left: *internal link prediction*; right: collaborative filtering with $k = 50$) for the reference period $[0, 1$ day$[$ of one day, the prediction period $[1, 16$ days$[$ of 15 days, and the *Jaccard* weight function. We plot the precision, recall, and F-measure (vertical axis), defined in Section 5.3, as functions of the threshold τ (left), and the number N of predicted links per node (right).

Internal link prediction

As illustrated in Figure 5.4 (left), the performances of *internal link prediction* for given reference and prediction periods and a given weight function depend on the weight threshold τ. If $\tau = 0$ then all possible internal pairs are predicted, which corresponds in this example to 33% of all appearing links. However, many of these pairs do not actually appear, and so the corresponding precision is almost zero. Instead, if a very high threshold is used then only few internal pairs are predicted, and so the obtained recall is almost zero. However, most of these few pairs do appear, which corresponds to a precision of almost 100%. More generally, the precision increases with the threshold value, and the recall decreases. The F-measure which captures a tradeoff between the two reaches its maximal value of 0.28 for $\tau = 0.4$ in this example.

To avoid taking into account the impact of the threshold τ on the *internal link prediction* method, we will select in the experiments of Section 5.6 the value of τ which maximizes the F-measure. For instance, when studying the impact of the prediction period duration x for a given reference period (Section 5.6.1) we will plot the maximal value of the F-measure as a function of x.

Collaborative filtering

As explained in Section 5.4.2, the collaborative filtering method depends on a parameter k which is the number of similar neighbors considered for each node. We have experimented the performances of the collaborative filtering algorithm for three values of k: 10, 50 and 100. Results indicate that the precision is slightly better for small values of k, but recall is best for large values of k. The maximal F-measure was obtained for $k = 50$, therefore we will use this value in all our experiments in the following.

The other parameter, N, is the number of links predicted for each node. It also has a strong impact on the performance of the collaborative filtering method. Figure 5.4 (right) shows that the precision decreases and the recall increases when N increases.

Again, to avoid taking into account the impact of N on the performances of the collaborative filtering method, we will select in the following the value of N which maximizes the F-measure.

5.6 Experimental results

In this section, we study the performances of our approach for link prediction and compare it to the collaborative filtering approach in the experimental framework described above. We first explore the impact of the prediction period duration, which allows us to choose a relevant value for this parameter for the rest of our comparisons. We then study the impact of the reference period duration, and again select a relevant value for this parameter. Finally, we compare the performances of the different weight functions for these parameters.

5.6.1 Impact of the prediction period duration

In order to study the impact of the prediction period duration we consider several reference periods $[0, x[$ (from $x = 1$ hour to $x = 7$ days), and several prediction periods $[x, y[$ ($y = x + 1$ day, $y = x + 2$ days, ..., $y = x + 49$ days). We then compute, for each considered reference period duration, the maximal value of the F-measure observed over all values of the threshold τ (for *internal link prediction*) and all values of N (for collaborative filtering), and plot it as a function of the prediction period duration, as explained in Section 5.5.2.

Results are presented in Figure 5.5. The following key facts appear clearly:

- all plots have the same global shape (a fast increase followed by a slow decrease or steady regime), although their amplitude decreases when the reference period duration increases (we will deepen this in the next section);
- different weight functions give different results, which we deepen in Section 5.6.3;
- in most cases, and for all weight functions which perform well, *internal link prediction* surpasses significantly collaborative filtering (notice the different scales for the vertical axes for the two methods).

Finally, a prediction period of 15 days gives good results, and is representative of a wide range of prediction period durations for all reference period durations and weight functions. We will therefore use this prediction period duration in all the following.

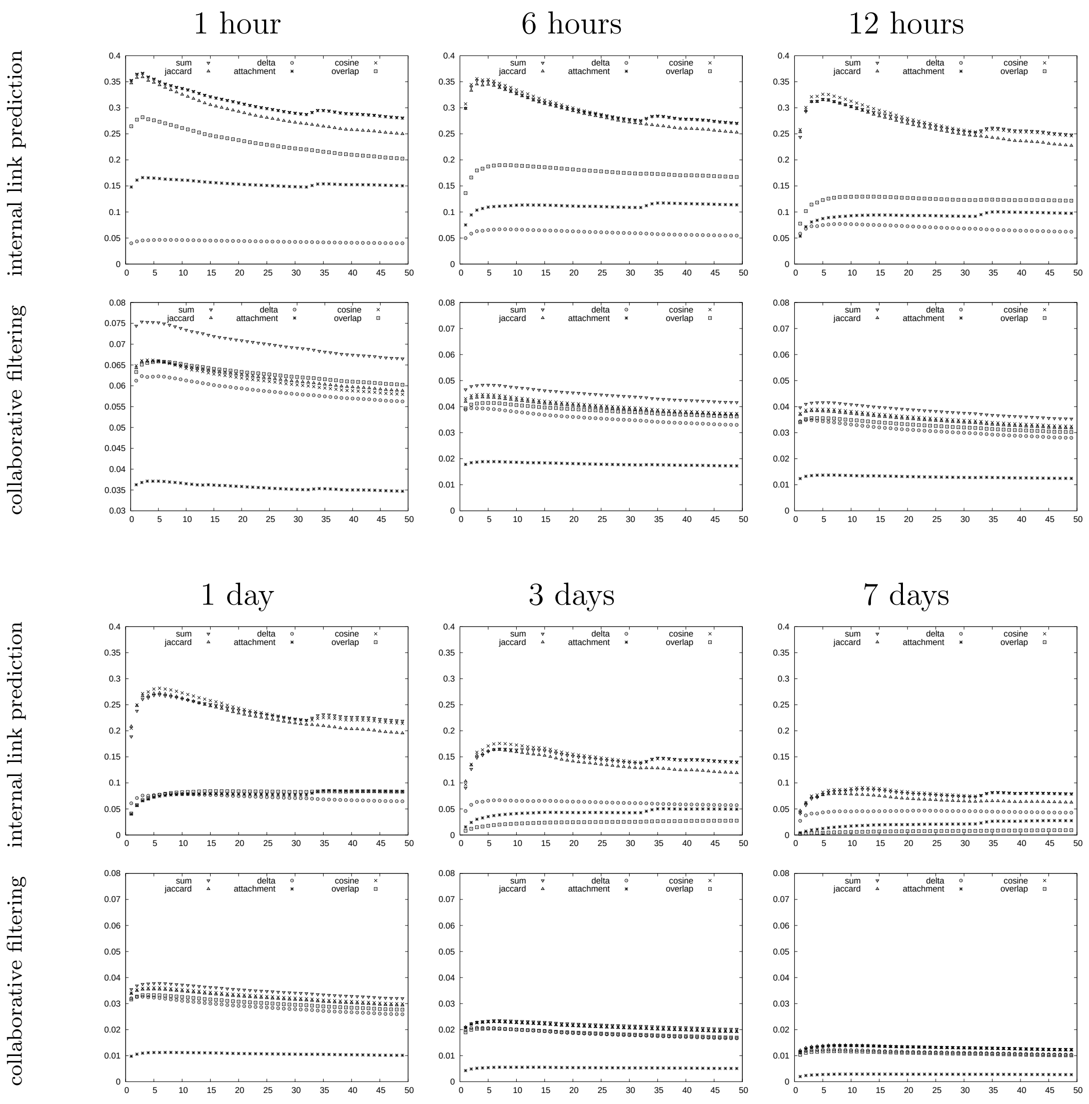


Figure 5.5 – Maximal value of the F-measure (vertical axis) as a function of the prediction period duration (horizontal axis, in days) for both link prediction methods with different weight functions. In order to help comparison between different reference period durations, we used the same scale for the vertical axes. Notice however that the scales are not the same for the two methods because otherwise the collaborative filtering plots would hardly be readable.

5.6.2 Impact of the reference period duration

In order to investigate the impact of the reference period duration $[0, x[$, we vary its duration x for $x = 1, 2, ..., 48$ hours, and we use the prediction period $[0, x + 15 \text{ days}[$ of 15 days, as explained in the previous section. We do not consider reference periods longer than 48 hours, because, as we can see in Figure 5.5, longer reference periods lead to poorer performances. We compute for all cases the maximal value of the F-measure observed over all values of the threshold τ (for *internal link prediction*) and all values of N (for collaborative filtering) and plot it as a function of the reference period duration, as explained in Section 5.5.2.

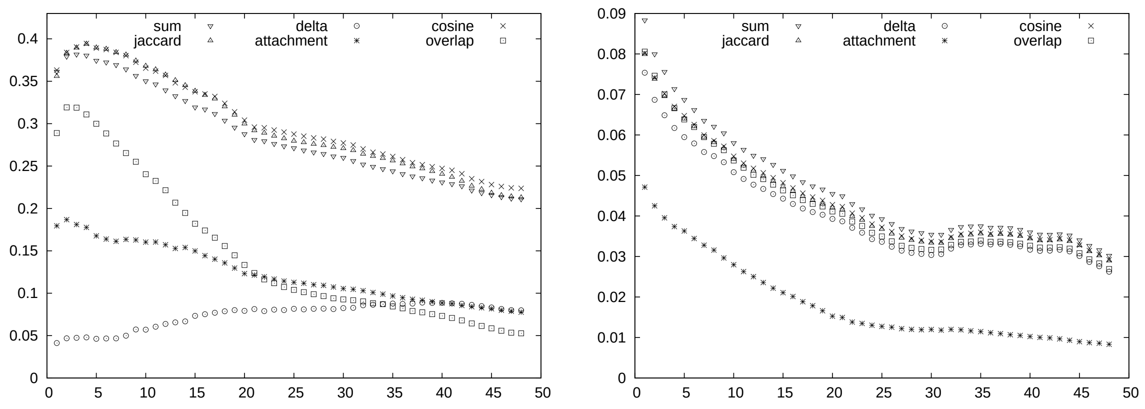


Figure 5.6 – Evolution of the maximal F-measure (vertical axis) as a function the reference period duration (horizontal axis, in hours) for the different weight functions. Left: *internal link prediction*. Right: collaborative filtering. The prediction period duration is 15 days in all cases.

Results are presented in Figure 5.6. The following key facts appear clearly:

- overall, the maximal F-measure decreases with the size of the reference period (except for very short reference periods with *internal link prediction*);
- different weight functions give different results, which we deepen in next section;
- in most cases, and for all weight functions which perform well, *internal link prediction* surpasses significantly collaborative filtering.

Finally, a reference period of 1 day gives good results, and is representative of a wide range of reference period durations. We will therefore use this reference period duration in all the following.

5.6.3 Impact of the weight function

In this section, we observe the impact of the weight function on both considered prediction methods. As explained in previous sections, we use reference period $[0, 1 \text{ day}[$ and prediction period $[1, 16 \text{ days}[$, which are representative of wide ranges of values for these parameters. We then compute the precision and recall for all possible values of the threshold τ for *internal link prediction* and all possible values of N for collaborative filtering; we plot the obtained precision as a function of the obtained recall in Figure 5.7.

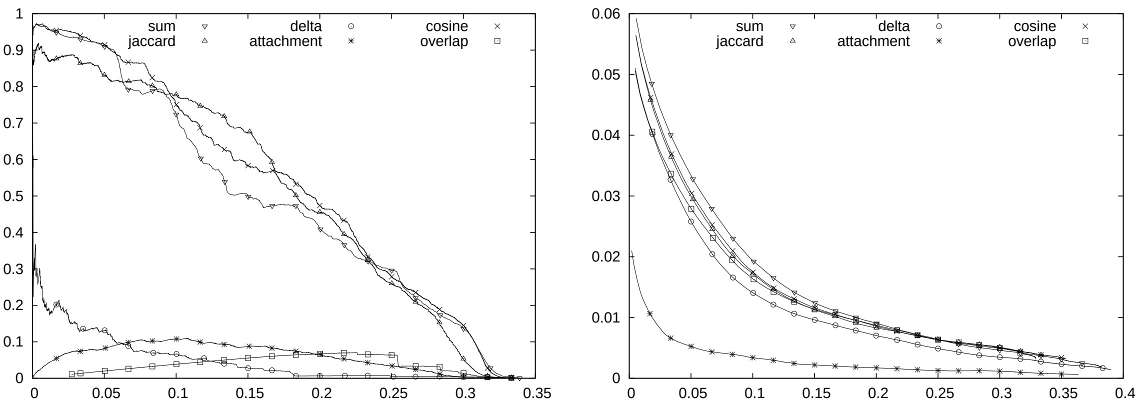


Figure 5.7 – Precision (vertical axis) as a function of recall (horizontal axis), for a 1 day reference period $[0, 1[$ and a 15 days prediction period $[1, 16 \text{ days}[$, for all weight functions. Left: *internal link prediction*; right: collaborative filtering. Each point corresponds to the precision and recall obtained for a given value of τ or N.

A first important observation is that the weight functions considered clearly split into two classes regarding the performances of *internal link prediction* (Figure 5.7, left): sum, Jaccard and cosine reach very high values of precision, and are also able to reach very good compromises between precision and recall (like a precision of 50% and a recall of 20%); instead, delta, overlap and attachment lead to poor performances of *internal link prediction*. No such behavior is observable for collaborative filtering (Figure 5.7, right): all weight functions lead to very similar results except attachment which performs worse than the others.

5.7 *Delicious-tags* and *Flickr-comments* datasets

We present here the results obtained for the *Delicious-tags* and *Flickr-comments* datasets. For each one, we choose reference and prediction periods which are representative of wide ranges of values for these parameters. Basic features of the reference graph B and the fraction of internal pairs appearing during the prediction period are presented in Table 5.2, for the two datasets.

	Delicious-tags	*Flickr-comments*
$n_\perp$	$13,851$	$16,281$
$n_\top$	$21,398$	$18,578$
$\|L\|$	$435,830$	$93,016$
reference period	12 *months*	3 *months*
prediction period	24 *months*	12 *months*
fraction of internal pairs in L'	21%	67%

Table 5.2 – Number of $\perp$-nodes, $\top$-nodes, links in the bipartite graph B, durations of the reference and prediction periods, and fraction of internal pairs among the new links in the prediction period, for the *Delicious-tags* and *Flickr-comments* graphs.

Results are presented in Figures 5.8 and 5.9. The following key facts appear clearly:

- different weight functions give different results, depending on the datasets;
- overlap leads to poor performances of *internal link prediction* for the two datasets;
- all weight functions lead to very similar results for collaborative filtering;
- different values of recall are observed in collaborative filtering. This is due to the fact that each weight function considers a different set of k similar neighbors of the node, and this leads to a different set of predicted links;
- in the two datasets, and for all weight functions which perform well, *internal link prediction* surpasses significantly collaborative filtering (notice the different scales for the vertical axes for the two methods).

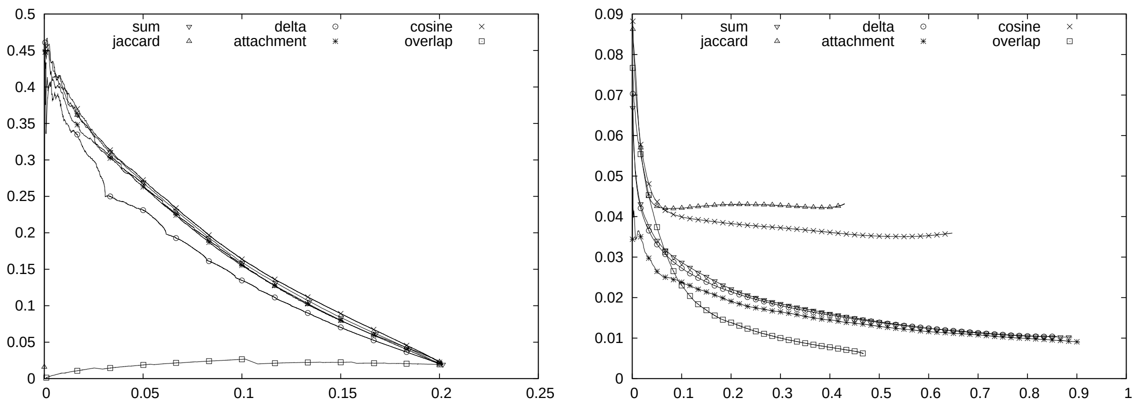


Figure 5.8 – Precision (vertical axis) as a function of recall (horizontal axis), for a 12 months reference period $[0, 12[$ and a 24 months prediction period $[12, 36$ months$[$, for all weight functions. Left: *internal link prediction*; right: collaborative filtering. Each point corresponds to the precision and recall obtained for a given value of τ or N.

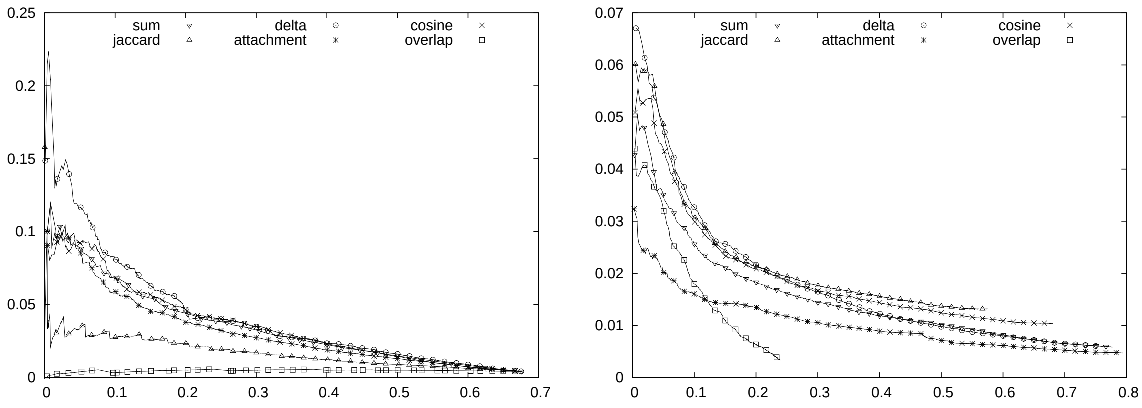


Figure 5.9 – Precision (vertical axis) as a function of recall (horizontal axis), for a 3 months reference period $[0, 3[$ and a 12 months prediction period $[3, 15$ months$[$, for all weight functions. Left: *internal link prediction*; right: collaborative filtering. Each point corresponds to the precision and recall obtained for a given value of τ or N.

5.8 Conclusion

In this chapter, we use a specific notion related to bipartite graphs introduced in the previous chapter, *internal pairs*, to propose a method addressing the link prediction problem. We evaluate the relevance of this method by comparing it to a classical collaborative filtering approach and perform experiments on various datasets, which show that our method performs very well. We present in details the results obtained for a P2P-file exchange graph in order to illustrate how results depend on various parameters, including which weight function is used for projection. We present more quickly results for two other datasets.

Our link prediction method has the following advantages. First, it performs very well, much better than a collaborative filtering approach, where no other method was previously available. Moreover, our method is purely structural: it relies on the identification of a specific kind of links which will probably appear in the future; this gives much insight on the properties of the underlying dynamics. Finally, the use of weight thresholds allows to tune the method in order to reach target tradeoffs in the quality of the prediction: one may use small thresholds to have excellent precision at the cost of a poorer recall, and conversely.

Our work may be extended in several ways. In particular, other (maybe more specific) weight functions may be introduced and tested. One may also predict internal pairs that induce *only* links with weight above the threshold (inducing *one* such link is sufficient in our current algorithm), or use both $\top$- and $\bot$-projections (our current algorithm only uses the $\bot$- one). There is therefore room for improving the method and its results.

Likewise, it would be interesting to conduct more experimentations and compare results on different datasets. Comparing our method with others, in particular machine learning approaches like the one presented in [29] is also appealing. Last but not least, our work calls for the development of link prediction methods for *external* pairs (those links which are not internal), which may be done with such methods or by modifying ours.

Another interesting direction would be to modify our approach in order to perform recommendation. As already explained, link prediction and recommendation are quite different problems, but they are strongly related. Just like we adapted collaborative filtering for link prediction in bipartite graphs, one may adapt our method and evaluate its relevance for recommendation.

Finally, notice that any graph $G = (V, E)$ may be seen as a bipartite graph $B = (\bot, \top, E)$ such that $\top$ and $\bot$ are the nodes in classical graph, and each node in $\top$ is linked to its neighbors in G. Using this, one may use our bipartite link prediction method for classical graphs, thus making it more general. Evaluating of this approach seems an interesting perspective.

Chapter 6

Conclusion

We have studied in this thesis real-world bipartite networks: their nodes may be separated into two classes, with links between nodes of different classes only. The approach classically used for studying these objects consists in transforming them into classical (non-bipartite) graphs where two nodes in the classical graph are linked if they have at least one neighbor in common in the bipartite graph. This process is called *projection*. However, it induces an important loss of information and leads to very large graphs. Studying bipartite graphs directly therefore is very appealing, but there is a lack of ground methods for doing so.

In addition, most of these networks are dynamic: they evolve during time, with node and link additions and removals. Studying such dynamics is extremely important for our understanding of these objects, but here again very limited knowledge and ground methodology is available, even for classical (non-bipartite) dynamic graphs. One of the main approaches developed for studying such dynamics is *link prediction*, which consists in identifying pairs of nodes which will be linked in the future.

We have addressed in this thesis the problem of analysis and link prediction in bipartite graphs. To do so, we have introduced a special kind of links that we have call *internal links*. These links have the specificity that their deletion does not change the projection of the bipartite graph.

Internal links may be used for measuring redundancy in bipartite graphs, and for measuring the information lost between a bipartite graph and its projections. Using a wide set of real-world examples, we observed that internal links are very frequent in practice, and that associated statistics are fruitful measures to point out similarities and differences among real-world networks. In addition, our notion is designed specifically for bipartite graphs. Moreover, removing internal links may be used to obtain compact bipartite encodings of graphs and to improve their modeling. This makes it as a relevant tool for analysis of bipartite graphs, which is an important research topic.

We have shown that internal links appear with high probability in the future evolution of the graph, and therefore that they play an important role regarding the dynamics.

We have proposed a method based on internal links for addressing the link prediction problem in bipartite networks. We evaluated the relevance of this method by comparing it to a classical collaborative filtering approach and performed experiments on various datasets, which show that our method performs very well. In addition, our method is purely structural: it relies on the identification of a specific kind of links which will probably appear in the future; this gives much insight on the properties of the underlying dynamics.

The work conducted in this thesis opens several perspectives. We have already presented the direct perspectives of our work, at the end of each chapters. We present more general ones below.

We can complete our classification by studying the *external links* and *pairs* (those links and pairs which are not internal) and define other specific classes of links. Typically, we can study their role regarding the structure and the dynamics of bipartite networks. In addition, we can observe their impact on the projection since their addition in, or removal from, the bipartite graph changes its projections. We can also isolate the set of links (and pairs) which are simultaneously $\perp$- and $\top$-external. This distinction will allow a better understanding of bipartite graphs without considering $\top$ and $\perp$ separately.

Another relevant investigation consists in observing the impact of internal links on weight functions. In fact, in some weight functions (Jaccard for instance), adding an internal link may increase the weight for some links and decrease it for others. There are therefore nontrivial relations, which call for further understanding.

We have studied specific graph dynamics in which nodes can be added, but never removed. The next step is to study the generic case where links can also be removed. In particular, we can study whether these links are with high probability internal or external links. We can also observe the time required for a link to become internal or external. Going further, taking into account the dynamics of nodes (addition and removal of nodes) is an important but challenging perspective.

Internal links can also be considered as useless information regarding the projection; deleting them from a bipartite graph gives a smaller dataset able to store the information contained in the projected graph. However, as we underlined, the deletion of an internal link can change the nature of other bipartite links. The challenge here consists in designing removal strategies which give a bipartite graph with no internal link but still the same projection (and so a *minimal* graph to this regard). It is not possible in general to delete all initial internal links without altering the structure of the projection. We suggested a possible way to do this by implementing a greedy algorithm that selects a link which minimizes the decrease of number of remaining internal links, but most remains to be done.

In some cases a bipartite graph is available without any information on its dynamics. Reconstructing the order in which links appeared is important for giving insight on the object. In addition, if the analysis of the whole graph is not feasible, because of its size and various other constraints, then cutting the graph into several snapshots is a relevant solution, but this is possible only when we know the order of appearance of links.
In a preliminary study, we have considered a set of links of which we know the timestamps. We then added randomly these links one by one, and at each time we computed the number of internal links in the bipartite graph. We have observed that this gives different results from what is observed in reality. This shows that the internal links play an important role regarding the order in which links are added.
To rebuild the order of appearance of links, one way consists in designing a function based on the number of internal links and pairs that is able to decide if a given link may be added to the current bipartite graph or if it should be added later.

Finally, a key perspective is to design models for bipartite graphs. When modeling complex networks using random bipartite graphs, the presence of internal links may be a problem as they are poorly captured by models. To this regard, removing internal links before generating a random bipartite graph may lead to better models.

Another interesting possibility consists in generating bipartite graphs with prescribed degree and internal degree distributions. The challenge would be to connect nodes in such a way that the final graph contains the desired number of internal links. This is difficult because creating a new link may cause previous internal links to become external, and vice-versa.

Another approach consists in developing a model of bipartite graph which allows by construction to capture those properties of internal links that make real graphs different from random ones. This process would take as input the number of nodes and desired fraction of internal links, and then generate bipartite graphs with the given fraction of internal links.

Such models would produce artificial graphs similar to real ones, which would allow to conduct more realistic simulations, and in particular to investigate the role of internal links in various situations.

Appendix A

A *peer-to-peer* measurement

A.1 Introduction

Real-world complex networks are in general not directly available: collecting data about them requires the use of *measurement* procedures. Much work has been done in this direction for the case of peer-to-peer system [3, 84, 122]. The most widely used approach is to watch the activity in a network of interest for a period of time. This raises important difficulties due mainly to the distributed and anonymous nature of peer-to-peer systems, their dynamics, and their sheer size.

In this appendix, we present such a contribution conducted during this thesis. We explore a new method for collecting information on what occurs in one of the main peer-to-peer systems currently in use, *eDonkey*[1]. It consists in introducing *honeypots* in the network, *i.e* peers pretending to offer files and logging all the queries they receive for these files from other peers.

We begin by presenting the current state of the art of peer-to-peer measurement (Section A.2), and then we describe the new solution we propose (Section A.3). Finally we will illustrate its relevance with some practical measurements (Section A.4), before presenting our conclusions (Section A.5).

A.2 Related work

Measurement of peer-to-peer systems is a very active area of research. We focus here on the measurement of peer activity, thus ignoring measurements of peer-to-peer overlays and protocols. Several approaches have been used to collect information on peer-to-peer

1. *eDonkey* is a semi-distributed peer-to-peer file exchange system based on directory servers. An unofficial documentation of the protocol is available [75].

activity, each with its own advantages and drawbacks. We rapidly describe them in this section.

A.2.1 Measurement at server level

In centralized or semi-centralized systems (like *eDonkey*), queries sent by peers are managed by servers; it is then possible to passively collect data directly on these servers. The measurement setting then consists in either modifying the server so it may log the queries it manages [64, 81], or capturing traffic at IP level and then decoding it [8].

This method has the advantage of collecting *all* the information managed by a given server. However, as actual file exchanges occur between peers (out of the sight of servers), this information is not captured. Moreover, it requires cooperation with server administrators.

A.2.2 Measurement at peer level

Passive measurements are also possible in fully distributed systems at client level: a modified client may observe the traffic going through it including in some cases keyword queries, file searches, etc.

In [3, 6, 71, 84] authors set up such measurements. The main issue of this approach is the need for users that agree to cooperate, which limits the amount of data obtained. To increase it, the author of [139] designed a large distributed architecture called GnuDC (Gnutella Distributed Crawler); it monitors the network by being attached to a large number of peers.

A.2.3 Measurement by client sending queries

An active measurement method from clients is also possible. It consists in designing a client that sends queries in the system and records the obtained answers (lists of files and providers, typically). This has been done in *Napster* [119] and *eDonkey* [66, 83] with success. The main drawback of this approach is that it is active: it may interfere with the observations, and the rate at which queries may be sent is limited.

A.2.4 Measurement at ISP level

Finally, one may capture peer-to-peer traffic directly on ISP infrastructures, in a passive way. In [72, 116, 122] for instance, data is collected from several routers, and different peer-to-peer applications (Gnutella, FastTrack, DirectConnect, *eDonkey*) are observed.

This approach provides network-level data, with limited information on users and exchanged files. This makes it quite different from other approaches discussed here. Moreover, it relies on cooperating with ISPs, which have limited rights to observe user traffic.

A.3 Measurement of eDonkey Activity with Distributed *Honeypots*

Our measurement infrastructure consists in a set of fake peers (the honeypots) connected to different servers, and a manager controlling these honeypots. We first present our manager and honeypots, and then we discuss privacy concerns.

A.3.1 Manager.

The manager's role is to set up the honeypots, and then coordinate them and centralize the data they collect.

The first function of the manager is to launch the honeypots. It specifies to each of them a server to connect to. Each honeypot then attempts to connect to the server, and reports its status (connected or not), as well as its *clientID*[2], if relevant, to the manager. This makes it possible to re-launch dead honeypots or to redirect them toward other servers. The manager regularly checks the status of each honeypot for the same reason.

Several strategies make sense for assigning honeypots to servers. One may typically choose a different sever for each honeypot, in order to obtain a more global view. The choice of servers may also be guided by their resources and number of users, so that the honeypots may reach the largest possible number of peers.

The second function of the manager is to tell honeypots to advertise fake files. It specifies the name, size and *fileID*[3] of each file. Again, many strategies are possible to choose the files to advertise, and the manager is in charge of implementing the chosen strategy. For instance, it is possible to study the activity on a specific topic by choosing the files accordingly. One may then ask to the discovered peers their list of shared files and add these files to the the honeypot's list. One may also advertise random files.

Finally, the manager periodically gathers the data collected by honeypots and does some basic data management (for instance, it merges and unifies the collected log files, see below).

A.3.2 Honeypots.

After receiving an order from the manager to advertise a file F , the honeypot adds this file to its shared file list. The *eDonkey* server then adds the honeypot to its list of providers for file F. A honeypot may therefore afterward be contacted by other peers looking for F.

2. In the *eDonkey* network, peers are identified by a *clientID*, which is their IP address if they are directly reachable (high ID) or a 24 bits number otherwise (low ID).

3. In the *eDonkey* network, files are identified by their *fileID* or *hash*. It is generated from the file's content, which ensures that two files with the same content are considered as identical, regardless of their names.

Notice that, due to peer exchange [127] and other high-level *eDonkey* features, the honeypot may be contacted by peers which are not connected to the server.

Each honeypot constructs a log file of all the queries it receives. The log file can be written directly on a hard disk or sent via network to the monitor.

Before going any further, we specify that, for each query, the honeypot saves the information contained in the *eDonkey* protocol concerning the message type, as well as metadata such as the IP address, port, name, *userID*[4], version of client and ID status (high or low) of the peers sending the queries; moreover, it collects information concerning the server (name, IP, port), as well as data concerning the network environment (such as the timestamps marking the reception of the packets by the system).

Once a peer is connected to a honeypot, the list and description of all shared files in its cache content are retrieved. Note that this feature is not available on all peers, as it can be disabled by the user.

A honeypot must pass for a normal peer on the network. For this purpose, we have modified the open-source *eDonkey* client Amule [1] so that it meets our needs. We detail the main modifications below.

File display. In the normal course of events, if a client has files to offer (the client application considers that all files belonging to a given directory are files to be shared with other users), an OFFER-FILES message describing these files is sent immediately after the connection to the server is established, or whenever the client's shared file list changes. This message can also be used as a *keep-alive* message, sent periodically to the server.

As our honeypots do not have any file in their shared file folder, we have added a module to change the shared file list, so that the honeypot automatically sends the desired OFFER-FILES messages to the server.

Communication with other peers. When a honeypot is connected to a server and advertises some files, it may be contacted by other peers wanting to download one of these files. The normal message exchange process between a peer wanting to download a file and a peer having the file is presented in Figure A.1. The downloading peer tries to establish a connection with a HELLO message, to which the provider peer answers with a HELLO-ANSWER message. The downloading peer sends a START-UPLOAD query, declaring which file it is interested in; the provider replies with an ACCEPT-UPLOAD message. The downloading peer then starts to query file parts and the provider sends the queried parts.

Our honeypots behave like provider peers up to the last part of the exchange: they do not send parts of the desired file to the downloading peer. In this aspect, honeypots do therefore not act as normal peers in the system, and run the risk of being noticed and then

4. The user ID (also called user hash) is unique and is used to identify a client across sessions,however,the client ID is valid only through a client's session with a specific server [75].

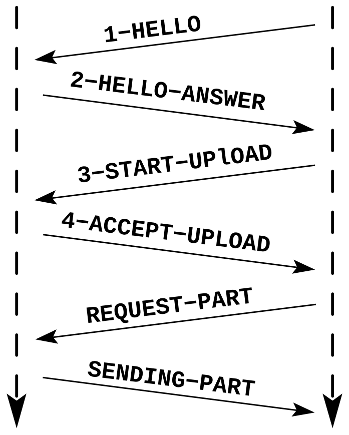


Figure A.1 – Series of messages exchanged between honeypots and peers.

blacklisted. To avoid this, we have implemented two different strategies for answering file parts queries. The first one consists in sending nothing: the corresponding honeypots do not reply to file parts queries. The other strategy consists in sending random content when queried for file parts.

Log file construction. Our goal is to record users' activity concerning the files a honeypot advertises. We have therefore modified the application to record the following message types received from other peers: HELLO, START-UPLOAD, and REQUEST-PART.

A.3.3 Privacy concerns

For ethical and legal reasons, we cannot record any sensitive data concerning users' privacy. In our context, the main such data are IP addresses and filenames. We want however to be able to know if a same user requested several files or not.

In order to attain a high level of anonymisation, while keeping the anonymised data coherent, we follow a two-steps procedure. First, each honeypot encodes IP addresses in its log using a one way hash function, cryptographically sure. This anonymisation takes place before any data is written to disk or sent to the manager. This is however not sufficient to achieve a secure anonymisation of IP addresses: somebody could apply the hash function to all 2^{32} possible IP addresses, to construct a reverse dictionary of the anonymisation. After having collected the data from the honeypots, the manager therefore conducts a second anonymisation step: it replaces each hash value, in a coherent way between honeypots' logs, by an integer: the first hash is replaced by 0, the second one by 1, and so on. This ensures that the final anonymised data is secure, and that it is not possible to obtain the users' IP addresses from it.

In addition, we also anonymise file names which may contain personal information [5, 15]. We replace each word that appears less often than a given threshold by an integer.

A.4 Experiments

Many parameters may have a strong impact on the measurements conducted with our tool: which files the honeypot claims to have; number of honeypots in a distributed measurement; number of files advertised by the honeypot; duration of the measurement; etc.

In this section, we study several measurements which we have conducted to illustrate what is feasible with our approach and tool, and to investigate the impact of some key parameters. This relies on two different measurements:

- The *distributed* measurement used 24 honeypots ran by different PlanetLab [2] machines during one month (October 2008). Among the 24 honeypots, half did not answer at all to queries received from other peers; the others sent files with random content to the peers contacting them (see Section A.4.2 below). All honeypots advertised the same four files (a movie, a song, a linux distribution and a text). They were all connected to the same large server and had a *HighID*.
- The *greedy* measurement used only one honeypot but aimed at advertising as many files as possible. To do so, the honeypot ran a two-steps procedure: during the first day of measurement, it asked their list of shared files to all peers contacting it, and added all these files to its own list of shared files; after the first day, it did not increase its list of shared files anymore, and just recorded the queries it received and the lists of files shared by peers contacting it. It did not send any content to other peers. We ran this measurement during the two first weeks of November 2008.

The key properties of the obtained data are summarized in Table A.1. These statistics already show that our measurement method succeeds in collecting large amounts of data, with hundreds of thousands distinct peers and files observed.

	distributed	**greedy**
Number of honeypots	24	1
Duration in days	32	15
Number of shared files	4	3,175
Number of distinct peers	110,049	871,445
Number of distinct files	28,007	267,047
Space used by distinct files	9 TB	90 TB

Table A.1 – Basic statistics regarding the data collected with our measurements.

In the following, we investigate the impact of four key parameters of the measurement: its duration; the fact that honeypots send random content or no content at all; the number of honeypots involved; and the number of advertised files.

A.4.1 Impact of measurement duration

Let us first observe how the number of distinct observed peers evolves as the duration of the measurement grows, displayed in Figures A.2 and A.3 for the distributed and greedy measurements respectively. It appears clearly that the number of observed peers grows rapidly, and linearily, during all the measurement. Even after 30 (resp. 15) days, the number of distinct peers observed by the distributed (resp. greedy) measurement still grows significantly: more than 2, 500 (resp. 50, 000) new peers per day.

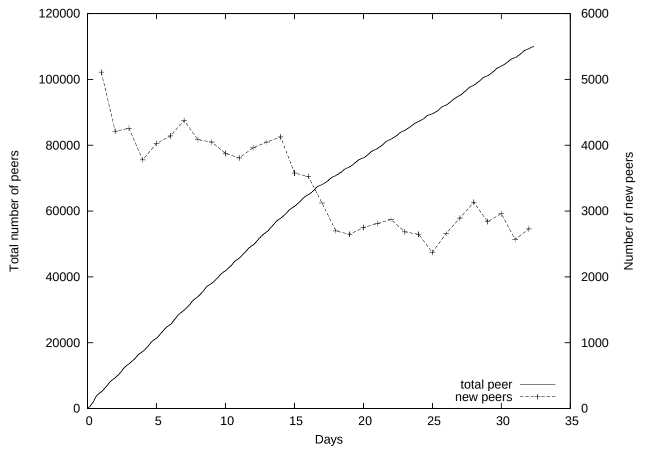


Figure A.2 – Evolution of the number of distinct peers observed during our distributed measurement (left vertical axis) and number of new peers observed each day (right vertical axis) as a function of time elapsed since the beginning of this measurement, in days (horizontal axis).

These observations are of prime importance for conducting measurements: they show that conducting very long measurements is relevant, as one continuously discovers a significant amount of new peers. They also show that blacklisting, even if it is present, does not prevent us from observing many peers despite the fact that our honeypots never provide any useful content. We deepen this in the next section.

The number of new peers discovered each day is also displayed in Figures A.2 and A.3. It decreases during time in the distributed measurement, which is probably due to the fact that the popularity of shared files decreases. This may also be due to the fact that we reach a situation where most peers interested in the files proposed by the honeypot already have contacted it. In any case, the important point here is that this happens only after a very long measurement duration (one month), and that even then the number of new observed peers remains large (more than 2, 500 per day). This shows that continuing the measurement for long periods of time makes sense, even with 24 distributed honeypots advertising only 4 files. In such a scenario, one may have guessed that all the peers potentially interested in the proposed files, or most of them, would have been observed before 30 days. Of course,

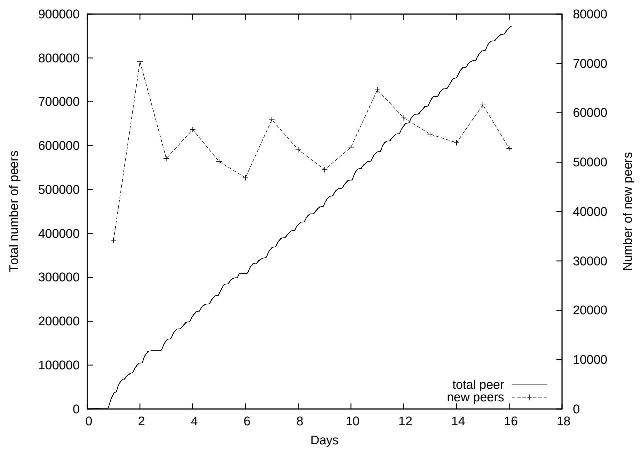


Figure A.3 – Evolution of the number of distinct peers observed during our greedy measurement (left vertical axis) and number of new peers observed each day (right vertical axis) as a function of time elapsed since the beginning of this measurement, in days (horizontal axis).

this depends on the popularity of the proposed files.

One may also notice that the number of peers observed during the first day of our greedy measurement (Figure A.3) is very low; this is due to the initialisation phase of this measurement strategy, as described above: during the first day, the honeypot mainly constructs its large list of shared files, starting with only a few. The number of observed peers during this period of time is non-zero, but it is much smaller than after the initialisation phase and thus it is not visible on the plot. After this initial period, the honeypot observes an average of $54,000$ new peers each day, which is stable during the 15 days of measurement.

A.4.2 Random content vs no content.

When contacted by peers wanting an advertised file, honeypots may apply two different strategies: they may simply ignore these queries and not answer them; or they may send random content. These strategies may play an important role in avoiding blacklisting at server and/or peer levels: if the system detects that honeypots do not provide relevant content, then other peers may stop contacting them. We do not consider the strategy consisting in providing the true files, which would raise bandwidth and storage problems, as well as legal and ethical issues in many cases.

In order to investigate this, half the honeypots in our distributed measurement applied the first strategy, and half applied the second one. This leads to two groups of 12 honeypots, which we call *no-content* and *random-content*, respectively. Figures A.4 and A.5 show the number of distinct peers sending HELLO and START-UPLOAD messages (see

Figure A.1) observed by each group during our measurement. Similarily, Figure A.6 displays the number of REQUEST-PART messages they received. Similar plots are obtained for each file advertised by our honeypots, independently (recall that the honeypots of this measurement setup advertised 4 different files).

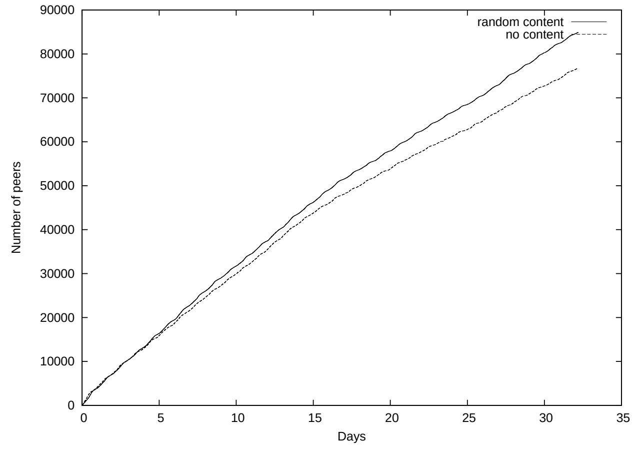


Figure A.4 – Number of distinct peers sending HELLO messages to our two groups of honeypots in our distributed measurements (vertical axis) as a function of time elapsed since the beginning of measurement, in days (horizontal axis).

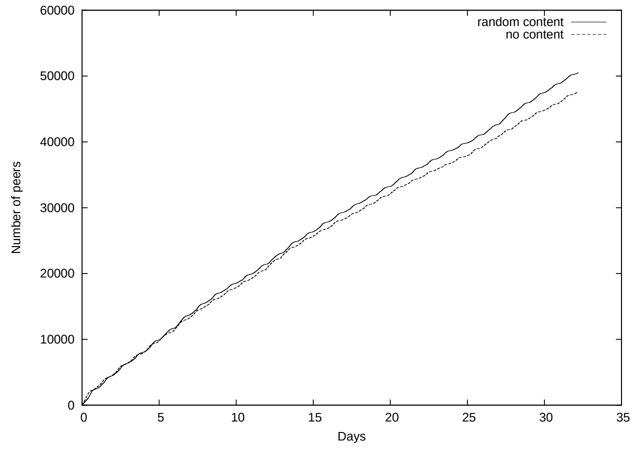


Figure A.5 – Number of distinct peers sending START-UPLOAD messages to our two groups of honeypots in our distributed measurements (vertical axis) as a function of time elapsed since the beginning of measurement, in days (horizontal axis).

All these plots clearly show that, although the difference is not huge, the *random-content* strategy leads to better results than the *no-content* one. This is particularly striking regarding the HELLO and START-UPLOAD messages, as the two strategies behave exactly

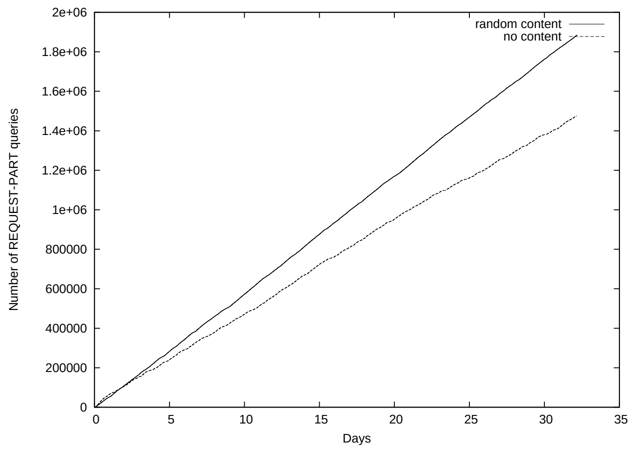


Figure A.6 – Number of REQUEST-PART messages received by our two groups of honeypots in our distributed measurements (vertical axis) as a function of time elapsed since the beginning of measurement, in days (horizontal axis).

in the the same up till this point. This demonstrates that, even though we still discover many new peers, there is some kind of blacklisting of honeypots, and that this blacklisting is more efficient when the honeypot sends no content. This is probably due to the fact that detecting honeypots which send invalid content takes more time than detecting honeypots which send nothing.

The larger difference between strategies regarding the number of REQUEST-PART messages, as observed in Figure A.6 (we finally observe $1,9$ million queries with the random-content strategy, and only $1,5$ million with the no-content strategy), may be explained as follows. First, when a peer receives irrelevant data or no data from our honeypots, it may stop using it. Detecting the fact that the honeypot sends random content is also longer than detecting that it does not answer, and thus a peer may send queries for more file parts in this case before deciding not to consider the honeypot anymore. This may be seen as an implicit blacklisting at client level, which stops using a honeypot after it observed that it sends no useful content.

Finally, these plots show that the *random-content* strategy is significantly more efficient than the *no-content* one. Other blacklisting techniques may operate, but we cannot observe them with our data.

In order to investigate further the difference between the *random-content* and *no-content* strategies, we plot in Figure A.7 (resp. Figure A.8) the number of START-UPLOAD (resp. REQUEST-PART) messages received by our two groups of honeypots from a *single* peer. The plot for HELLO messages is very similar to Figure A.7 so we do not reproduce it here.

The peer we have chosen is the one which sent the largest number of queries to our honeypots. For some periods of time, it does not send any queries (which induces a plateau on the plots), but in general this peer sends queries as fast as it can, provided that the

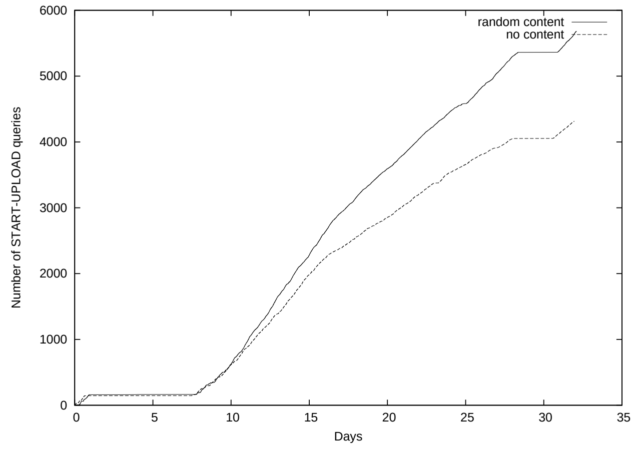


Figure A.7 – Number of START-UPLOAD messages received *from a single peer* by our two groups of honeypots in our distributed measurements (vertical axis) as a function of time elapsed since the beginning of measurement, in days (horizontal axis).

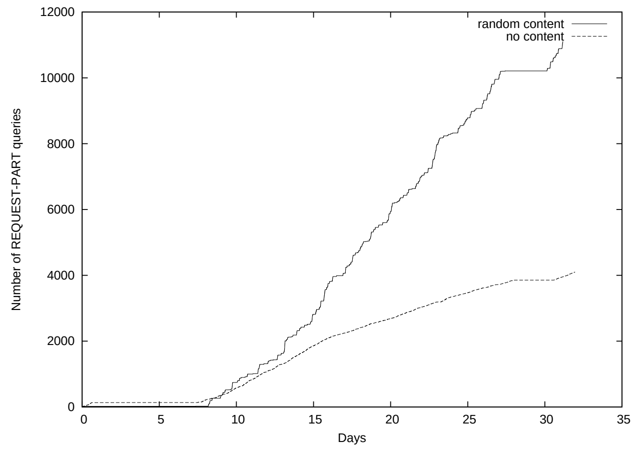


Figure A.8 – Number of REQUEST-PART messages received *from a single peer* by our two groups of honeypots in our distributed measurements (vertical axis) as a function of time elapsed since the beginning of measurement, in days (horizontal axis).

previous query was finished before it sends the next one.

The fact that it sends significantly less queries to our *no-content* honeypots than to our *random-content* ones confirms that the queries which receive no answer are sent at a lower rate than the ones which receive random content, as claimed above. This also explains the fact that the plot for *no-content* is smoother than the one for *random-content*, see Figure A.8: the time between two queries to a *no-content* honeypot is constant (it is the timeout of the peer waiting for an answer), while for *random-content* this time may vary.

Notice however that this is not sufficient to explain the difference between the two strategies observed in Figures A.4 and A.5, as they display the number of distinct peers (not messages) observed. As explained above, this is certainly due to a combination of some kind of client-level blacklisting combined to the difference of speed at which a query is processed when sent to *no-content* and *random-content* honeypots.

A.4.3 Impact of the number of honeypots.

Our tools makes it possible to conduct honeypot measurements from a large number of machines at the same time, in a distributed manner. In this section, we explore the benefit of this feature: one may imagine that increasing the number of honeypots does not improve the measurement, at least as soon as a quite small number of honeypots are in place. The key question we want to address here therefore is: given a number n of honeypots, what is the benefit of adding one more honeypot to the infrastructure.

To investigate this, we use our distributed measurement involving 24 honeypots, and explore what we would have seen if only a part of them were used. To do so, we select n random honeypots among the 24 ones, for n between 0 and 24, and we compute the number of distinct peers observed by these n honeypots alone. As the choice of the n honeypots may have a significant influence on the result, we repeat this computation 100 times (we would ideally consider *all* 2^{24} possible subsets of our honeypots, but this is not feasible), and we plot the average, maximal, and minimal obtained values, see Figure A.9. The average makes it possible to observe what one may expect. The maximal and minimal values give an indication of the dispersion of obtained results. In this case, they are quite close to the average, except at the very beginning of the plot (a honeypot leads to the observation of as many as $37,000$ peers, while another only sees $13,000$).

This plot clearly shows that there is an important benefit in using a few honeypots rather than only one. It also shows that, even when 24 honeypots are used, there is a significant benefit in adding more honeypots, thus calling for the use of large-scale distributed infrastructures. However, the benefit obtained by using more honeypots progressively decreases. This indicates that, even though many more honeypots may be used, at some reasonable point the benefit will become very low.

A.4.4 Impact of the number of files.

A natural way to increase the number of peers observed by our honeypot measurements is to increase the number of files we advertise, thus reducing the focus of the measurement. This is the aim of our *greedy* measurement, which starts by collecting for one day the list of files shared by the peers contacting a honeypot, and then adds all these files to the list of files shared by the honeypot, as described above.

Similarly to the previous section, we consider a measurement with a given number of files, and then, for any n lower than or equal to this number, we study the number of peers we would have observed if we used only n of these files in our measurement. Here, we could

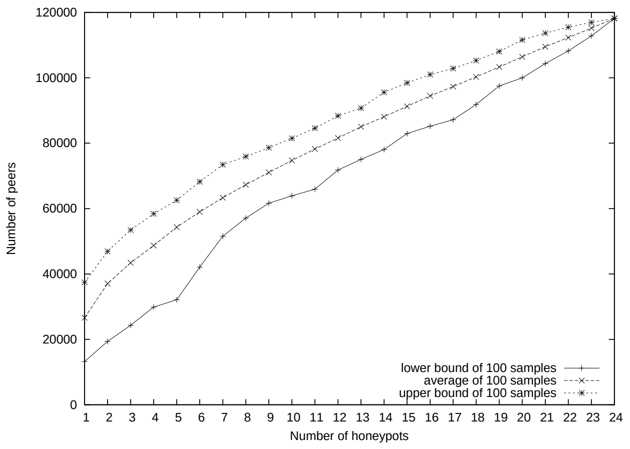


Figure A.9 – Number of distinct peers observed at the end of the measurement (vertical axis) as a function of the number n of involved honeypots (horizontal axis). For each n, we sample 100 random sets of n honeypots and plot the average, minimal and maximal obtained values.

in principle consider the 3, 175 files advertised in our measurement. However, this is not feasible in practice; we therefore consider two subsets of files: the *random-files* are a set of 100 randomly chosen files; and the *popular-files* are the 100 files for which we received queries from the largest number of peers.

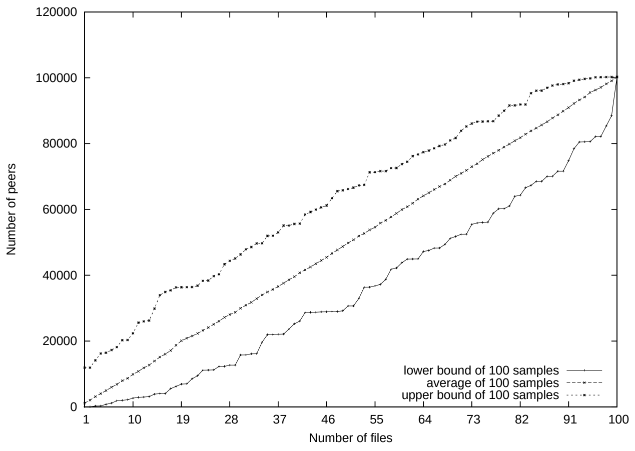


Figure A.10 – Number of distinct peers observed at the end of the greedy measurement (vertical axis) as a function of the number n of advertised files in the *random-files* set (horizontal axis). For each n, we sample 100 random sets of n files and plot the average, minimal and maximal obtained values. In average, each new file leads to the discovery of approximately 1, 000 new peers.

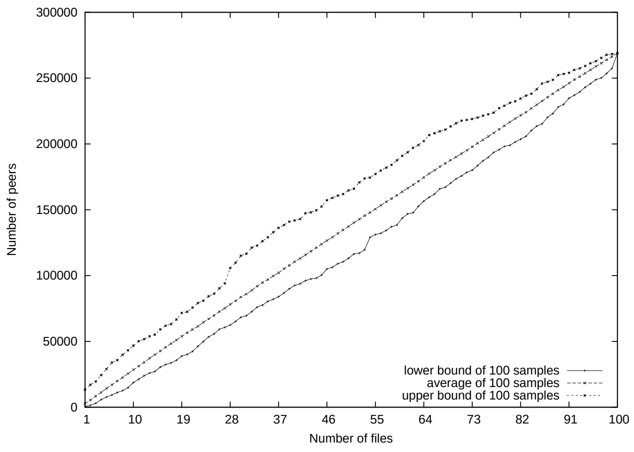


Figure A.11 – Number of distinct peers observed at the end of the greedy measurement (vertical axis) as a function of the number n of advertised files in the *popular-files* set (horizontal axis). For each n, we sample 100 random sets of n files and plot the average, minimal and maximal obtained values. In average, each new file leads to the discovery of approximately $2,700$ new peers.

For these two sets of files, we select n random ones among the 100 in the set, for n between 0 and 100, and we compute the number of distinct peers sending queries for at least one of these files. As the choice of the n files may have a significant influence on the result, we repeat this computation 100 times (we would ideally consider *all* 2^{100} possible subsets of files, or even the $2^{3\,175}$ ones, but this is not tractable), and we plot the average, maximal, and minimal obtained values, see Figures A.10 and A.11.

In both cases, the average, minimal and maximal values behave similarly, but the difference in the number of peers observed at the end of the measurement ($100,000$ for *random-files* and $270,000$ for *popular-files*) clearly shows that the advertised files have a significant impact on the obtained data. This is confirmed for any number of files; in particular, the most popular file leads to the discovery of $13,373$ peers, while some files lead to the observation of 2 peers only.

Importantly, these plots also show that the number of peers increases significantly and linearly as we add more files to our shared files list. This shows that the greedy approach certainly is relevant if one aims at observing huge number of peers, and that advertising more than 100 files definitively is relevant.

A.5 Conclusion.

We have described a method for peer-to-peer measurement via honeypots and implemented it on the *eDonkey* network. This method makes it possible to collect large amounts of data on peers searching for files. It complements other approaches and has its own advantages and drawbacks: it makes it possible to focus on a topic of interest, or even a specific file; it does not require cooperation with a server administrator and/or an ISP; but it gives a very partial view; and it may interfere with the systems as it is active.

We have illustrated the use of our approach by conducting some measurements which show that it is relevant: it captures information on many peers and files (hundreds of thousands) during long periods of time (several weeks), and using a relatively large number of distributed honeypots (24).

Using these measurements, we have evaluated several strategies and parameters. First, we studied the impact of measurement duration, and showed that it is useful to make very long measurements, as we continuously observe new peers and files. Then we compared the results obtained by sending random content to other peers *vs* not answering to their queries; there is no huge difference between the two, but sending random content is more efficient. We then studied the impact of the number of honeypots involved in a measurement, and the impact of the number of files they advertise; we showed that increasing both makes measurements much more efficient in terms of the number of observed peers and files.

Annexe B

Résumé en français

B.1 Introduction

On peut modéliser de nombreux objets issus du monde réel par des graphes. On peut citer de nombreux cas de réseaux informatiques, sociaux, biologiques ou issus du langage, par exemple la topologie de l'internet, des réseaux de collaboration, des réactions entre protéines au sein d'une cellule, des co-occurrences de mots ou des relations de synonymie. Bien que ces réseaux soient issus de contextes différents, ils se ressemblent au sens de certaines propriétés statistiques [22, 77, 136]. Ceci a montré qu'il est pertinent de considérer ces objets comme un ensemble cohérent. C'est pourquoi on les désigne sous le terme général de *graphes de terrain* (*complex networks* en anglais).

L'*analyse* des graphes de terrain est probablement la plus ancienne des activités du domaine. La problématique centrale de l'analyse vise à décrire la structure du graphe. Ceci se fait par l'introduction de notions statistiques et/ou structurelles qui résument l'information et en soulignent les principales caractéristiques. La définition de telles notions est toutefois loin d'être triviale, ainsi que l'évaluation de leur pertinence. De même l'interprétation des descriptions obtenues peut s'avérer délicate.

La plupart des graphes de terrain sont de plus dynamiques, c'est-à-dire que leur structure évolue au fil du temps par l'ajout et/ou le retrait de nœuds et/ou de liens. L'étude de la dynamique des graphes de terrain peut s'aborder par le problème de *la prédiction de nouveaux liens* dans ces graphes.

Plusieurs graphes de terrain ont par ailleurs une nature bipartie : les nœuds appartienent à deux classes distinctes, les liens étant uniquement entre nœuds de classes différentes. Dans l'exemple du réseau auteur-film, les auteurs et les films constituent deux classes disjointes et les acteurs sont reliés aux films dans lesquels ils ont joué.

Certaines notions utilisées dans l'analyse des graphes classiques (non bipartis) s'étendent directement au cas biparti, comme par exemple la taille, la densité, ou la distribution des degrés. Pour d'autres, l'extension est moins directe ; c'est le cas de la densité locale capturé par le *coefficient de clustering*, puisque dans un graphe biparti il ne peut y avoir le liens entre les voisins d'un nœud. Bien qu'il existe déjà des méthodes pour analyser les graphes de terrain bipartis [26, 34, 41, 54, 55, 113, 115, 123], il reste néanmoins beaucoup de travail dans cette direction.

De même, plusieurs travaux ont étudié le problème de la prédiction de liens dans les graphes classique (non-bipartis) [19, 67, 87, 103]. Toutefois, leurs méthodes ne sont pas directement applicables à, ou appropriées pour, les graphes bipartis. Un autre problème de recherche est étroitement lié à la prédiction de lien dans les graphes bipartis : le problème de recommandation [112]. Les systèmes de recommandation sont utilisés pour proposer des articles à des utilisateurs, tels que des produits à des clients. Notons cependant que les deux problèmes sont très différents : la recommandation vise généralement à trouver quelques produits d'intérêt pour chaque client ; la prédiction vise à trouver autant de liens qui apparaîtront dans l'avenir que possible.

Contribution. La nature bipartie des graphes de terrain fait appel au développement à de notions nouvelles ayant directement trait à la nature bipartie du graphe considéré, c'est-à-dire des notion qui n'auraient pas de sens ou d'intérêt dans le cas classique mais seraient naturelles dans le cas biparti. Dans cette perspective, nous identifions un type particulier de liens que nous appelons *liens internes*, et nous les proposons comme une notion importante pour l'analyse des réseaux bipartis. Ces liens sont très fréquents dans la pratique, et les statistiques associées permettent de souligner les ressemblances et les différences entre les graphes de terrain bipartis et les graphes bipartis aléatoires, ainsi que d'étudier leurs caractéristiques.

Les liens internes ont également un rôle important concernant la dynamique. Nous les étudions et proposons une approche basée sur ces liens pour la prédiction dans les graphes bipartis. Notre méthode fonctionne très bien, beaucoup mieux que l'approche de recommandation classique. En outre, notre méthode est purement structurelle : elle repose sur l'identification d'un type spécifique de liens qui apparaîtront probablement à l'avenir, ce qui donne un aperçu sur les propriétés de la dynamique sous-jacente.

B.2 graphes de terrain bipartis

De nombreux graphes de terrain ont une nature bipartie : leurs nœuds sont séparés en deux classes et les liens existent seulement entre des nœuds de classes différentes.

On peut citer comme exemples : la signature de publications où les auteurs sont reliés aux documents qu'ils ont signé, les systèmes pair-à-pair où les pairs sont reliés aux fichiers qu'ils ont fournis et/ou recherchés, et les réseaux d'achats en ligne, où les clients sont reliés aux produits qu'ils ont achetés.

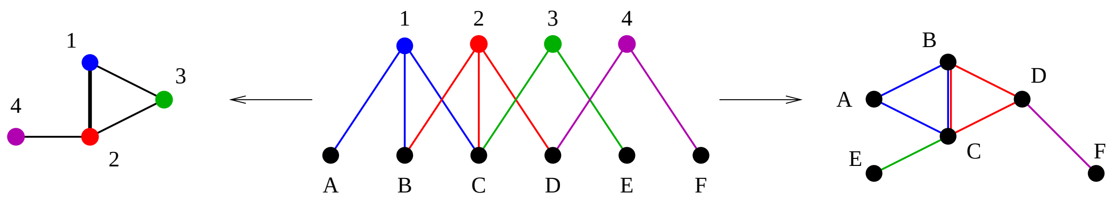


FIGURE B.1 – Un exemple de graphe biparti (au centre), avec ses deux projections : $\top$-projection $B_\top$(à gauche) et $\bot$-projection $B_\bot$(à droite). Ce processus réduit l'information disponible (par exemple le fait que B et C soient reliés à deux nœuds en commun dans le biparti, ou encore que les liens entre A, B et C dans la projection sont tous issus du même nœud, 1, dans le biparti).

Formellement, un graphe biparti est un triplet $B = (\top, \bot, L)$ où $\top$ et $\bot$ sont deux ensembles disjoints de nœuds, et $L \subseteq \top \times \bot$ est l'ensemble des liens. Nous notons $N(u) = \{v \in (\top \cup \bot),\ (u, v) \in L\}$ les voisins du nœud u ; $n_\bot = |\bot|$ et $n_\top = |\top|$ sont le nombre de nœuds dans $\top$ et $\bot$, respectivement, et $m = |L|$ est le nombre de liens. Le degré moyen dans $\bot$ est $k_\bot = \frac{1}{n_\bot} \sum_{u \in \bot} d(u) = \frac{m}{n_\bot}$ et le degré moyen dans $\top$ est $k_\top = \frac{1}{n_\top} \sum_{u \in \top} d(u) = \frac{m}{n_\top}$. La densité est $\delta = \frac{m}{n_\bot n_\top}$, c'est-à-dire le nombre de liens divisé par le nombre de liens possible entre toutes les paires de nœuds.

Bien qu'il existe aujourd'hui un nombre significatif de notions et d'outils pour analyser les graphes classiques (non bipartis), il y a encore un manque de notions pour analyser les graphes bipartis.

B.2.1 Projection.

Devant ce manque de notions adaptées au cas biparti, une approche classique consiste à transformer un graphe biparti en sa projection, qui est un graphe non-biparti. Considérons le graphe biparti $B = (\top, \bot, L)$: la $\bot$-projection de B est le graphe $B_\bot = (\bot, L_\bot)$ dans lequel $(u, v) \in L_\bot$ si u et v ont au moins un voisin en commun dans B : $N(u) \cap N(v) \neq \emptyset$. La $\top$-projection $B_\top$ est définie inversement.

Pour les exemples présentés ci-dessus, les projections sont par exemple : réseaux de co-vedette où deux acteurs sont reliés s'ils ont joué ensemble dans un film, réseaux de collaboration où deux auteurs sont reliés s'ils ont signé un document ensemble, réseaux d'intérêts où deux pairs sont reliés entre eux s'ils ont fourni/cherché une même donnée.

Cette approche permet d'étudier les graphes bipartis en utilisant les outils et les notions conçus pour les graphes classiques non-bipartis. Cependant elle entraîne une perte importante de données [78]. Il y a en effet beaucoup d'informations dans la structure bipartie qui disparaissent après projection. Par exemple, si deux acteurs ont joué dans de nombreux films ensemble, le nombre de ces films apporte beaucoup d'information. Celui-ci est disponible dans le graphe biparti, mais pas dans la projection où les deux acteurs sont simplement reliés entre eux.

De plus, chaque $\top$-nœud de degré d dans le biparti induit $\frac{d(d-1)}{2}$ liens dans la $\bot$-projection, et inversement. Ceci provoque une augmentation très forte du nombre de liens quand on passe d'un graphe biparti à sa projection.

B.2.2 Projection valueé

Une approche pour pallier à ce problème consiste à considérer une projection valuée, c'est-à-dire dans laquelle des poids sont attribués aux liens. Plusieurs approches peuvent être utilisées pour le calcul du poids de chaque lien.

Le poids du lien (u, v) peut-être défini comme étant le nombre de voisins que u et v ont en commun dans le graphe biparti : $\sigma(u, v) = |N(u) \cap N(v)|$.

Si u et v ont beaucoup de voisins, alors $\sigma(u, v)$ aura naturellement tendance à être élevé. Inversement, si u et v ont seulement peu de voisins mais que ces voisins sont les mêmes, alors $\sigma(u, v)$ sera faible, ce qui ne reflète pas le fait que u et v sont très similaires. Pour capturer ceci, on peut utiliser *le coefficient de Jaccard* : $\gamma(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$.

Remarquons que les nœuds jouent un rôle déséquilibré : un $\top$-nœud avec n voisins dans le graphe biparti a une influence sur les poids de $n * (n-1)/2$ liens. Lorsqu'un nœud n'a que deux voisins, alors cela indique probablement une forte similarité entre eux. Au contraire, si son nombre de voisins est grand alors son importance sur leurs similarité est faible. Pour capturer ceci on peut considérer que chaque nœud vote sur la similarité de ses voisins, ce qui conduit à calculer $\delta(u, v) = \sum_{x \in N(u) \cap N(v)} \frac{2}{|N(x)|(|N(x)|-1)}$.

Toutes les fonctions de pondération présentées ci-dessus sont naturelles et capturent des informations pertinentes sur le graphe biparti. Chacune a ses avantages et ses inconvénients.

B.3 Jeu de données et statistiques de bases

Nous présentons dans cette section les jeux de données que nous utilisons et leurs caractéristiques générales. Dans tous nos graphes de terrain bipartis, les utilisateurs sont des nœuds $\bot$ et leurs intétêts sont des nœuds $\top$. Nous avons utilisé les graphes de terrain bipartis suivant :

- *Imdb-movies* [22] où les acteurs sont reliés aux films dans lesquels ils ont joué.
- *Delicious-tags* [61] où les utilisateurs de *Delicious* sont reliés aux *tags* qu'ils utilisent pour indexer leurs favoris.
- *Flickr-tags* [109] où les utilisateurs de *Flickr* sont reliés aux *tags* qu'ils utilisent pour indexer leurs photos.
- *Flickr-comments* [109] où les utilisateurs de *Flickr* sont reliés aux photos qu'ils commentent.
- *Flickr-favorites* [109] où les utilisateurs de *Flickr* sont reliés aux photos qu' ils marquent comme favorites.

- *Flickr-groups* [109] où les utilisateurs de *Flickr* sont reliés aux groupes auxquels ils appartiennent.
- *P2P-files* [8] où les pairs sont liés aux fichiers qu'ils fournissent.

	$n_\perp$	$n_\top$	m	$k_\perp$	$k_\top$	δ
Flickr-comments	$122,561$	$1,489,485$	$4,190,415$	34.1	2.8	0.000023
Flickr-favorites	$321,312$	$6,450,934$	$17,871,828$	55.6	2.7	0.0000086
P2P-files	$122,599$	$1,920,353$	$4,502,704$	36.7	2.3	0.00002
PRL-papers	$15,414$	$41,633$	$249,474$	16.2	6.0	0.0003

TABLE B.1 – Statistiques de base de nos huit exemples de graphes bipartis de terrains.

Les propriétés de base de nos huit exemples de graphes de terrain bipartis sont données dans le tableau B.1. Il apparaît clairement que ce sont tous des grands graphes avec un degré moyen petit par rapport à leur taille. Cependant, il ya une différence d'ordre de grandeur entre les densités. Par exemple *Flickr-groups* est dix fois plus dense que *P2P-files*. Nos graphes de terrain bipartis couvrent une variété de cas rencontrés en pratique. C'est un point important qui sera utile dans la section suivante, car ils sont représentatifs de beaucoup de comportements différents.

B.4 Liens et paires internes

Dans cette section, nous introduisons les notions de liens et paires internes. Nous utilisons ces notions pour décrire les graphes de terrain bipartis présentés dans la section précédente.

Considérons un graphe biparti $B = (\perp, \top, L)$. Pour chaque paire de nœud $(u, v) \notin L$, nous notons par $B + (u, v)$ le graphe $B' = (\perp, \top, L \cup \{(u, v)\})$ obtenu par l'ajout du nouveau lien (u, v) à B. Pour chaque lien $(u, v) \in L$, nous notons par $B - (u, v)$ le graphe $B' = (\perp, \top, L \setminus \{(u, v)\})$ obtenu par la suppression du lien (u, v) de B.

Définition 1 (paire interne) *Une paire de nœuds $(u, v) \in \perp \times \top$ avec $(u, v) \notin L$ est une paire* $\perp$-interne *de B si la $\perp$-projection de $B' = B + (u, v)$ est identique à la projection de B. Nous définissons les paires $\top$-interne de façon similaires.*

Définition 2 (lien interne) *Le lien $(u, v) \in L$ est un lien* $\perp$-interne *de B si la $\perp$-projection de $B' = B - (u, v)$ est identique à la projection de B. Nous définissons les liens $\top$-internes de façon similaires.*

En d'autres termes, (u, v) est une paire $\perp$-interne de B si l'ajout du nouveau lien (u, v) à B ne change pas sa $\perp$-projection ; (u, v) est un lien $\perp$-interne si la suppression du lien (u, v) de B ne change pas sa $\perp$-projection. Voir les exemples des figures B.2 et B.3.

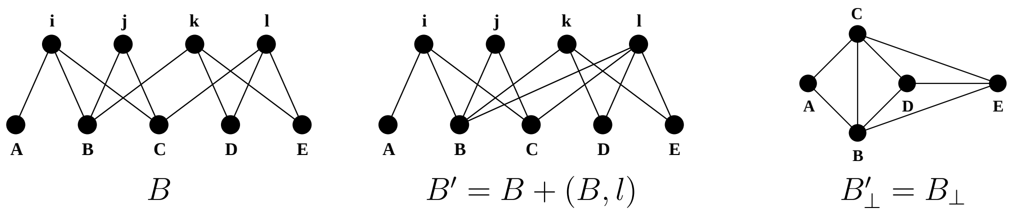


FIGURE B.2 – **Exemple d'une paire ⊥-interne**. De gauche à droite : un graphe biparti B, le graphe biparti B' obtenu en ajoutant le lien (B, L) à B et la ⊥-projection de ces deux graphes. Comme $B'_\perp = B_\perp$, (B, l) est une paire ⊥-interne de B.

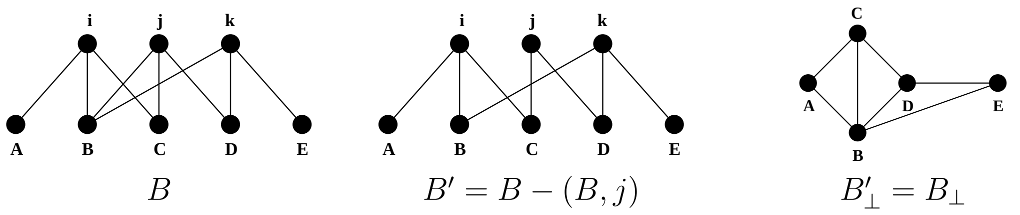


FIGURE B.3 – **Exemple d'un lien ⊥-interne**. De gauche à droite : un graphe biparti B, le graphe biparti B' obtenu en supprimant le lien (B, j) de B et la ⊥-projection de ces deux graphes. Comme $B'_\perp = B_\perp$, (B, j) est lien ⊥-interne de B.

Nous donnons une caractérisation des liens internes qui n'est pas fondée explicitement sur la projection et fournit un autre point de vue sur cette notion.

Lemme 1 *Un lien (u, v) de B est ⊥-interne si et seulement si $N(v)\backslash\{u\} \subseteq N(N(u)\backslash\{v\})$.*

B.4.1 Quantité de liens et de paires internes

Afin de capturer la redondance dans la structure bipartie, nous calculons le nombre de paires et de liens ⊤- et ⊥-internes. La fraction de liens internes, notée f_{E_I} et présentée dans la Table B.2, semble en général non négligeable. Une analyse quantitative de ces valeurs nécessite cependant la définition d'un référentiel. C'est pourquoi nous comparons les résultats des bipartis réels avec ceux obtenus avec des graphes bipartis aléatoires ayant les mêmes tailles et distributions des degrés, qui est un modèle aléatoire typique pour évaluer l'écart par rapport à un comportement attendu – voir par exemple [100, 101]. Les mesures liées à ce modèle seront désignées par le symbole *.

Nous notons $\mathcal{P}_I(\perp)$ (resp. $\mathcal{P}_I(\top)$) l'ensemble des paires ⊥-internes (resp. paires ⊤-internes) et $E_I(\perp)$ (resp. $E_I(\top)$) l'ensemble des liens ⊥-internes (resp. liens ⊤-internes). Nous normalisons le nombre de paires et de liens internes mesurés sur les graphes réels par les valeurs obtenues avec le modèle décrit ci-dessus. Les résultats correspondants sont également présentés dans la Table B.2.

	$f_{E_I}(\bot)$	$\frac{\mathcal{P}_I(\bot)}{\mathcal{P}_I^*(\bot)}$	$\frac{E_I(\bot)}{E_I^*(\bot)}$	$f_{E_I}(\top)$	$\frac{\mathcal{P}_I(\top)}{\mathcal{P}_I^*(\top)}$	$\frac{E_I(\top)}{E_I^*(\top)}$
Imdb-movies	0.031	0.441	47.0	0.026	0.491	147
Delicious-tags	0.112	0.972	1.47	0.104	1.823	5.31
Flickr-tags	0.117	0.920	1.51	0.048	1.040	2.50
Flickr-comments	0.398	0.258	4.22	0.002	0.151	22.0
Flickr-groups	0.228	0.491	2.21	0.015	0.249	2.86
Flickr-favorites	0.172	0.574	2.02	0.002	0.704	12.4
P2P-files	0.337	0.082	8.53	0.136	0.092	1430
PRL-papers	0.718	0.033	7.17	0.487	0.001	11.2

TABLE B.2 – Fraction de liens internes (f_{E_I}), nombre de paires internes ($\mathcal{P}_I$) et de liens internes (E_I) des graphes réel normalisés par les valeurs obtenues pour des graphes bipartis aléatoires ayant la même taille et les mêmes distributions des degrés.

On observe que les comportements concernant la quantité de liens internes sont très hétérogènes. Cependant quelques tendances générales peuvent être soulignées : dans le cas aléatoire, les liens $\bot$- et $\top$-internes sont sous-estimés. Ainsi, la probabilité d'avoir des nœuds qui partagent le même voisinage est plus élevée dans les graphes réels que dans les graphes aléatoires. On peut en effet s'attendre, par exemple, à ce que les personnes participant au même papier aient une probabilité plus élevée d'être co-auteurs d'un autre papier qu'une paire d'auteurs aléatoires.

Cependant le nombre de paires internes est généralement surestimé dans les graphes aléatoires. Pour comprendre cet effet, prenons le cas extrême où deux $\bot$-nœuds dans un graphe ont soit exactement le même voisinage, soit pas de voisins communs. Alors tous les liens sont $\bot$-internes, et le graphe ne contient aucune paire interne. Cet exemple indique que le nombre de paires internes est probablement anti-corrélé au nombre de liens internes.

En général, il existe une corrélation entre le fait que le nombre de liens internes est sous-estimé dans les graphes aléatoires et le fait que le nombre de paires interne est surestimé, mais cette corrélation n'est pas valable dans tous les cas. En outre, il n'y a pas de lien direct entre ces observations et la taille ou le degré moyen des graphes considérés.

Enfin, nous observons un comportement spécifique pour les deux graphes qui correspondent aux jeux de données *tags*, c'est-à-dire *Delicious-tags* et *Flickr-tags*. Pour ces graphes on observe le plus faible écart entre les cas réel et aléatoire pour les liens et paires $\bot$-internes. Inversement, ce sont les seuls graphes pour lesquels la quantité de paires $\top$-internes est sous-estimée dans les graphes aléatoires.

B.4.2 Corrélation entre le nombre de liens internes et le degré des nœuds

Nous appelons le nombre de liens internes d'un nœud son degré interne, le nombre total de liens étant son degré. Nous étudions dans cette section le rapport entre ces deux quantités, et présentons dans la Figure B.4 le degré moyen d'un nœud en fonction de son degré ($\perp$-)interne pour les données réelles et les graphes aléatoires.

Pour des raisons de brièveté, nous limitons notre analyse aux liens $\perp$-internes.

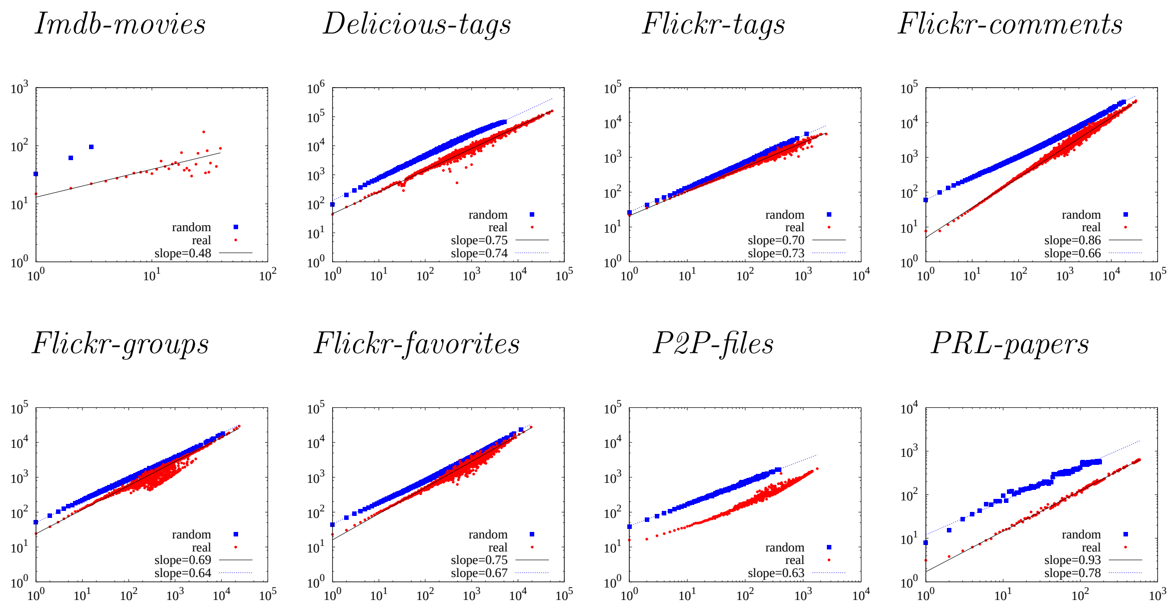


FIGURE B.4 – Degré moyen en fonction du degré $\perp$-interne.

Nous observons que les deux courbes réelles et aléatoires peuvent être approximées par une loi sous-linéaire dans plusieurs cas. Cependant, ce modèle est insatisfaisant pour *P2P-files*, et discutable pour les cas où les valeurs sont trop rares ou trop dispersées : en particulier *Imdb-movies* et *Flickr-groups*. La dispersion observée pour les grands degrés est une conséquence de la distribution des degrés qui est hétérogène (nous ne la présentons pas ici), le nombre de nœuds avec un degré élevé étant faible.

Si le fait qu'un lien donné est interne ou non était indépendant du degré du nœud, ces courbes seraient linéaires. Comme les graphes aléatoires ont un comportement sous-linéaire, cela signifie que les nœuds ayant des degrés importants ont, en moyenne, une proportion plus élevée de liens internes. Cet effet peut être expliqué qualitativement : l'augmentation du degré d'un nœud u implique une augmentation de la probabilité que l'un de ses voisins v soit tel que $N(v) \setminus \{u\} \subseteq N(N(u) \setminus \{v\})$.

D'autre part, la pente pour les graphes réel est dans la plupart des cas plus grande que la pente pour les graphes aléatoires – à nouveau les jeux de données de *tags* présentent un comportement différent.

Donc, le cas réel fournit plus de liens internes et moins de nœuds ayant une fraction faible (mais non nulle) de ces liens, qui doivent être des nœuds de fort degré. Cela provient du fait que si les nœuds u et v sont des voisins, la probabilité que $N(v)\backslash\{u\} \subseteq N(N(u)\backslash\{v\})$ est d'autant plus importante que le degré de v est faible et de u est grand.

B.5 Prédiction de liens dans les graphes bipartis

La prédiction de liens est une problématique clé dans l'étude des graphes de train dynamiques. Certaines travaux ont étudié ce problème dans les graphes classiques (non-bipartis), et ne sont pas directement applicables ou appropriés pour les graphes bipartis [19, 29, 45, 67, 69, 87, 103, 129, 134]. Par exemple, les méthodes basées sur le voisinage commun de deux nœuds s'appuient sur la présence de triangles dans le graphe. Cependant, il n'y a pas de triangles dans les graphes bipartis. Les méthodes basées sur les chemins entre les nœuds ou les marche aléatoire doivent être adaptées pour ne prendre en compte que les chemins de longueur impaire, car un lien ne peut apparaître entre deux nœuds que s'ils sont déjà reliés par des chemins de longueurs impaires dans le graphe biparti.

Comme expliqué dans la section B.1, la prédiction de lien et la recommandation sont deux problème étroitement liés. Diverses approches ont été développées pour la recommandation [20, 25, 74], et le filtrage collaboratif est l'approche la plus couramment utilisée [74]. Nous allons utiliser cette approche dans cette section pour des fins de comparaison avec notre méthode.

B.5.1 La prédiction de liens dans les graphes bipartis

Considérons un graphe biparti dynamique défini par un ensemble de n liens horodatés $D = \{(t_i, u_i, v_i), i = 1...n\}$. Soit $B = (\bot, \top, L)$ le graphe observé à partir d'un instant donné a à un autre instant $b > a$: $\bot = \{u, \exists(t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$, $\top = \{v, \exists(t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$ et $L = \{(u, v), \exists(t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$. Nous appelons B le *graphe de référence* et $[a, b[$ la *période de référence*.

Maintenant, considérons un instant $c > b$. Ceci induit un ensemble de liens L' ajoutés à B durant la période $[b, c[$, que nous appelons la *période de prédiction* : $L' = \{(u, v), \exists(t, u, v) \in D \text{ s.t. } b \leqslant t < c\} \cap (\bot \times \top \setminus L)$. Notez que nous considérons seulement les liens entre les nœuds de G (nous ignorons les nouveaux nœuds apparaissant dans la période $[b, c[$) qui ne sont pas présents dans B (nous considérons seulement les liens dans $\bot \times \top \setminus L$).

L'objectif d'une méthode de prédiction de liens est de trouver un ensemble P de *liens prédits* qui contient de nombreux liens de L', mais seulement peu qui ne sont pas dans L'.

Remarquons que dans le cas extrême où l'on prédit *tous* les liens possibles, c'est-à-dire $P = \bot \times \top \setminus L$, alors on arrive à prédire tous les liens de L', mais on prédit également de nombreux liens qui ne sont pas dans L', et inversement.

L'évaluation des performances d'une méthode de prédiction consiste donc à évaluer sa capacité à atteindre un compromis entre ces deux objectifs, ce qui n'est pas trivial. Nous présentons ci-dessous une méthode classique pour le faire [43,118], que nous utilisons dans cette section.

Notons $\overline{P}$ l'ensemble de liens dont la méthode prédit qu'ils n'apparaîtront pas $\overline{P} = (\perp \times \top \setminus L) \setminus P$. La figure B.5 illustre les quatre cas possibles qui peuvent se produire lors de la prédiction de liens : l'ensemble $P \cap L'$ des *vrais positifs* est l'ensemble des liens qui apparaissent et que la méthode prédit avec succès ; l'ensemble $\overline{P} \setminus L'$ des *vrais négatifs* est l'ensemble de liens non prédit qui n'appairassent pas. À l'inverse, les *faux positifs* sont les liens de $P \setminus L'$, c'est-à-dire les liens prédit, mais n'appairassant pas, et les *faux négatifs* sont les liens de $\overline{P} \cap L'$.

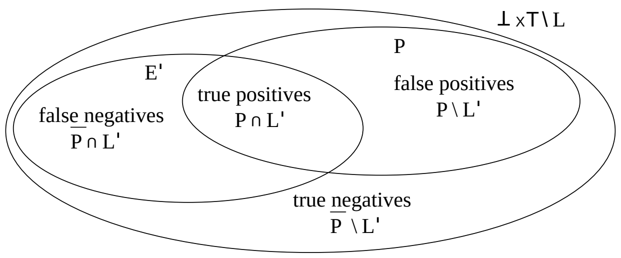


FIGURE B.5 – Une méthode de prédiction divise l'ensemble des liens possibles $\perp \times \top \setminus L$ en quatre catégories : les vrais positifs, $P \cap L'$; les vrais négatifs, $\overline{P} \setminus L'$; les faux positifs, $P \setminus L'$; et les faux négarifs, $\overline{P} \cap L'$.

Le but d'une méthode de prédiction de lien est de maximiser le nombre de vrais positifs et négatifs, tout en minimisant le nombre de faux positifs et négatifs. Ceci est capturé par deux quantités, appelé *précision* et *rappel*.

La *précision* est la fraction de vrais positifs parmi les liens prédit, c'est-à-dire $\frac{|P \cap L'|}{|P|}$. En d'autres termes, c'est la probabilité que la méthode est bonne quand elle prédit qu'un lien donné apparaîtra, et est donc une mesure de la correction.

Le *rappel* est la fraction de vrais positifs parmi les liens qui apparaissent, c'est-à-dire $\frac{|P \cap L'|}{|L'|}$. En d'autres termes, c'est la probabilité qu'un lien qui apparait sera en effet prédit par la méthode, et est donc une mesure de l'exhaustivité.

Comme expliqué plus haut, il existe un compromis entre la précision et le rappel, comme, en général, l'amélioration de l'une dégrade l'autre et inversement. Afin de saisir cela en une seule valeur, qui est souvent plus pratique, on considère généralement la *F-mesure*, $\frac{2 \times |P \cap L'|}{|P| + |L'|}$, qui est la moyenne harmonique de la précision et le rappel [130]. L'objectif d'une méthode de prédiction est alors de maximiser la F-mesure.

B.5.2 La prédiction de liens internes

La principale caractéristique de notre méthode de prédiction est qu'elle se concentre sur les paires internes : elle ne prédit que des liens qui sont des paires internes dans le graphe de référence. L'intuition derrière cela est que deux nœuds de $\perp$ qui ont déjà un voisin commun dans B (c'est-à-dire qui sont liés dans $B_\perp$) vont probablement en acquérir plus à l'avenir. Par contre, si deux nœuds n'ont pas de voisin commun dans B, alors ils vont probablement encore ne pas en avoir à l'avenir. Les liens qui peuvent être ajoutés à B qui correspondent à ces deux critères sont précisément les paires internes.

Pour aller plus loin, deux nœuds de $\perp$ ayant de nombreux voisins communs dans B sont plus susceptibles d'avoir plus de nouveaux voisins à l'avenir que des nœuds qui ont un seul voisin en commun. Plus généralement, toutes les fonctions de poids présentés dans la Section B.2.2 sont des mesures (de différents points de vue) de notre attente à ce que deux nœuds ayant au moins un voisin en commun en auront probablement plus à l'avenir. Par conséquent, nous nous attendons à ce que les liens qui apparaissent soient les paires internes induisant les $\perp$-liens ayant un poids élevé.

Cela conduit à la méthode de prédiction suivante, que nous appelons *la prédiction de liens internes*. Prenons une fonction poids ω comme celles décrites dans la Section B.2.2, et un seuil de poids τ donné. On note $L_{\perp\tau} = \{(u,w) \in L_\perp,\ \omega(u,w) \geq \tau\}$ l'ensemble des liens dans la projection qui ont un poids supérieur ou égal à τ. Nous alors prédire toutes les paires internes qui induisent au moins un lien de $L_{\perp\tau}$.

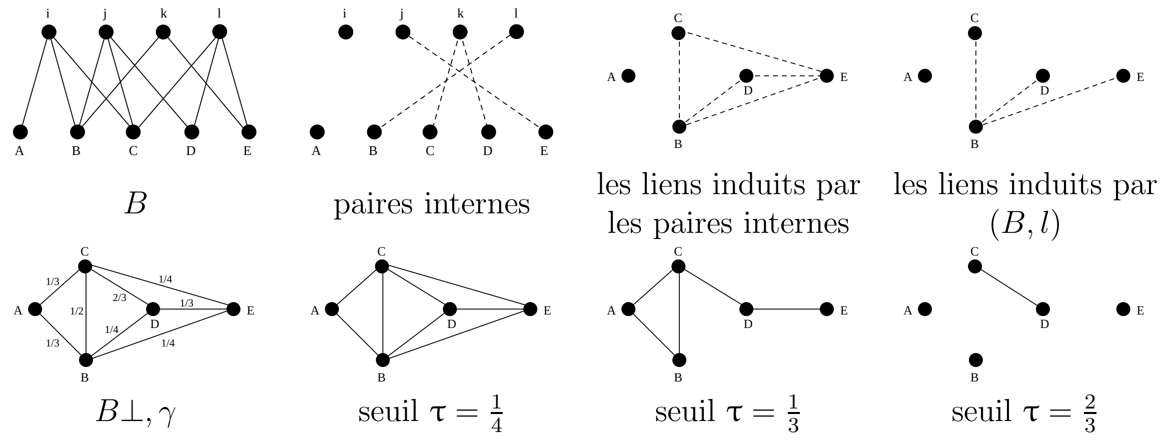

B
paires internes
les liens induits par les paires internes
les liens induits par (B, l)
$B\perp, \gamma$
seuil $\tau = \frac{1}{4}$
seuil $\tau = \frac{1}{3}$
seuil $\tau = \frac{2}{3}$

FIGURE B.6 – **Exemple de prédiction de liens internes**. Première ligne (de gauche à droite) : un graphe biparti B, les paires internes de B, les $\perp$-liens qu'elles induisent, et les liens de $B_\perp$ induits par la paire (B, l). Deuxième ligne (de gauche à droite) : $\perp$-projection de B valuée par *Jaccard* $(B_\perp, \gamma)$, et les liens $L_{\perp\frac{1}{4}}$, $L_{\perp\frac{1}{3}}$, $L_{\perp\frac{2}{3}}$ obtenus en utilisant les seuils τ égaux respectivement à $\frac{1}{4}$, à $\frac{1}{3}$ et à $\frac{2}{3}$.

La Figure B.6 présente un exemple de prédiction de liens internes en utilisant la fonction de poids *Jaccard*. L'ensemble des liens internes de B est $\{(B,l),(C,k),(D,k),(E,j)\}$;

concentrons-nous sur la paire interne (B, l). Elle induit les liens (B, C), (B, D), et (B, E). Étant donné un seuil τ nous prévoyons (B, l) si un de ces liens a un poids d'au moins τ. Par exemple :

- si $\tau = \frac{1}{4}$, tous les liens dans la projection ont un poids supérieur ou égal à τ, et ainsi nous prédisons toutes les paires internes possibles dans le graphe biparti, y compris (B, l) ;
- si $\tau = \frac{1}{3}$, seulement 5 liens dans la projection ont un poids supérieur ou égal à τ , incluant (B, C), qui est induit par (B, l) ; donc nous prédisons (B, L) ;
- si $\tau = \frac{2}{3}$, un seul lien a le poids supérieur ou égal à τ, et qui n'est pas un lien induit par (B, L) ; donc nous ne prédisons pas (B, L).

B.5.3 Résultats expérimentaux

Les performances des méthodes de prédiction de lien dépendent de différents paramètres, en particulier la durée des périodes de référence et de prédiction, et la fonction de poids. Dans cette section nous nous concentrons sur l'impact des fonctions de poids. Nous nous focalisons sur le jeu de données *P2P-files* qui est représentatif de tous les résultats obtenus pour d'autres jeux de données.

Nous utilisons la période de référence $[0, 1$ jour$[$ et la période de prédiction $[1, 16$ jours$[$, qui sont représentatives d'une large plage de valeurs de ces paramètres. Nous calculons ensuite la précision et le rappel pour toutes les valeurs possibles du seuil τ pour notre méthode et toutes les valeurs possibles de N (le nombre de liens recommandés pour chaque nœud) pour le filtrage collaboratif ; nous traçons la précision obtenue en fonction du rappel obtenu dans la Figure B.7.

Une première observation importante est que les fonctions de poids peuvent être divisées en deux classes concernant les performances de notre méthode (Figure B.7, à gauche) : *sum*, *Jaccard* et *cosine* atteignent des valeurs très élevées de précision, et sont également en mesure de parvenir à de très bons compromis entre précision et rappel (une précision de 50% et un rappel de 20%) ; par contre, *delta*, *overlap* et *attachment* donnent de mauvaises performances. Un tel comportement n'est pas observé pour le filtrage collaboratif (Figure B.7, à droite) : toutes les fonctions de poids donnent des résultats très similaires à l'exception de l'attachment qui obtient des résultats pires que les autres.

B.6 Conclusion

Nous avons abordé dans cette thèse le problème de l'analyse et de la prédiction de lien dans les graphes de terrain bipartis. Pour ce faire, nous avons introduit un type particulier de liens que nous avons appelé *liens internes*. Ces liens ont la particularité que leur suppression ne change pas la projection du graphe biparti.

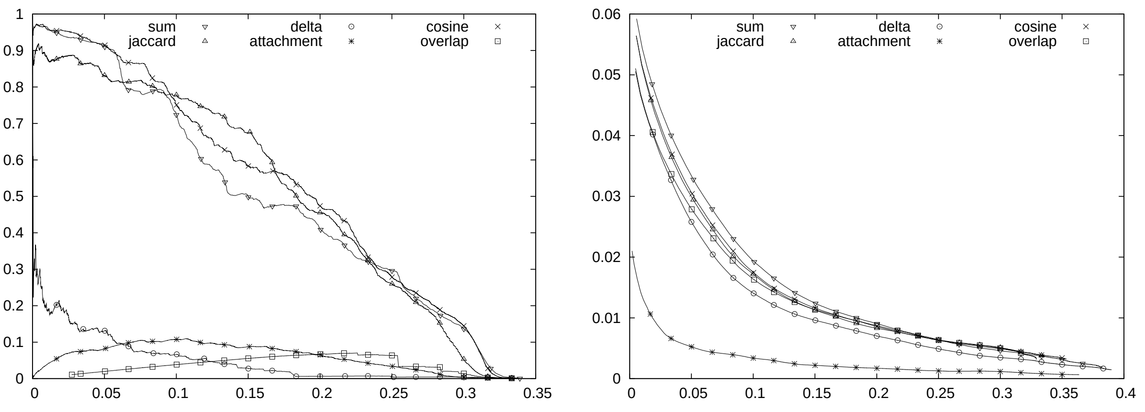


FIGURE B.7 – Précision (axe vertical) en fonction du rappel (axe horizontal), pour une période de référence de 1 jour [0, 1 jour[et une période de prédiction de 15 jours [1, 16 jours[, pour toutes les fonctions de poids. Gauche : *prédiction de liens internes*; droite : filtrage collaboratif. Chaque point correspond à la précision et au rappel obtenu pour une valeur donnée de τ ou N.

Les liens internes peuvent être utilisés pour mesurer la redondance dans les graphes bipartis, et pour mesurer la perte d'information entre un graphe biparti et ses projections. En utilisant un vaste ensemble d'exemples du monde réel, nous avons montré que les liens internes sont très fréquents en pratique, et que les statistiques associées permettent de souligner des ressemblances et des différences entre les graphes de terrain bipartis. En outre, notre notion est spécifiquement conçue pour les graphes bipartis. De plus, la suppression des liens internes peut être utilisée pour obtenir des codages bipartis compacts et améliorer la modélisation des graphes de terrain bipartis. Cela en fait un outil pertinent pour l'analyse des graphes bipartis, qui est un sujet de recherche important.

Nous avons aussi proposé une méthode basée sur les liens internes pour aborder le problème la prédiction de liens dans les graphes de terrain bipartis dynamiques. Nous avons évalué la pertinence de cette méthode en la comparant au filtrage collaboration avec des expériences réalisées sur différents jeux de données, et avons montré que notre méthode fonctionne très bien. En outre, notre méthode est purement structurelle : elle repose sur l'identification d'un type spécifique de liens qui apparaîtront probablement à l'avenir, ce qui donne une intuition sur les propriétés de la dynamique sous-jacente.

Les travaux menés dans cette thèse ouvrent plusieurs perspectives.

Nous pouvons tout d'abord compléter notre classification par l'étude des *liens* et *paires externes* (les liens et paires qui ne sont pas internes) et définir d'autres classes spécifiques de liens.

Les liens internes peuvent également être considérés comme des informations inutiles pour la projection ; leur suppression donne un graphe biparti *minimal* capable de stocker les informations contenues dans le graphe projeté. Cependant, la suppression d'un lien interne peut changer la nature d'autre liens dans le biparti. Le défi consiste ici à concevoir des stratégies d'élimination qui donnent un graphe biparti sans liens internes ayant toujours la même projection.

Notons que tout graphe $G = (V, E)$ peut être vu comme un graphe biparti $B = (\bot, \top, E)$ tel que $\top$ et $\bot$ sont les nœuds du graphe classique, et chaque nœud dans $\top$ est lié à ses voisins dans G. Avec cela, on peut utiliser notre méthode de prédiction de lien pour les graphes classiques, la rendant ainsi plus générale.

Finalement, une perspective essentielle est de concevoir des modèles pour graphes bipartis. Une façon intéressante pour ce faire consiste à générer des graphes bipartis avec des distributions des degrés et des degrés internes données. Le défi serait de connecter les nœuds de telle sorte que le graphe final contienne le nombre souhaité de liens internes. Ceci est difficile parce que la création d'un nouveau lien peut faire que des liens précédemment internes deviennent externes, et vice-versa.

B.7 Annexe–Mesure de l'activité pair-à-pair.

Il existe plusieurs façons de collecter des traces d'usages pair-à-pair, chacune ayant ses avantages et ses inconvénients en fonction du type d'informations recherché et/ou du système concerné. On peut par exemple concevoir un client pair-à-pair effectuant des requêtes afin de savoir quelles données sont proposées par les autres pairs [66, 83, 119]. Il est également possible de capturer le trafic directement sur les routeurs et chez les FAI [72,116,122]. On peut également installer un pair de grande capacité traitant de nombreuse requêtes, et qui peut les enregistrer [3].

Nous avons conçu un client de type *honeypot* pour effectuer des mesures dans le système *eDonkey*. Dans ce système, des serveurs indexent les fichiers proposés par les pairs. Lorsqu'un pair recherche un fichier, il fait une requête auprès du serveur qui lui fournit une liste de fournisseurs. Le pair contacte alors directement les fournisseurs pour télécharger le fichier. Notre client contacte le serveur et prétend fournir certains fichiers, puis enregistre les requêtes faites par d'autres pairs pour ces fichiers.

Nous illustrons notre approche en menant une mesure distribuée avec 24 *honeypots* implantés sur différentes machines de PlanetLab pendant un mois (octobre 2008). Tous les *honeypots* ont annoncé les quatre mêmes fichiers (un film, une chanson, une distribution Linux et un texte).

Impact de la durée de mesure. Nous observons d'abord la façon dont le nombre de pairs distincts observés évolue lorsque la durée de mesure croît. La figure B.8 montre clairement que le nombre de pairs observés croît rapidement, et quasiment linéairement, pendant toute la mesure. Même après 30 jours, le nombre de pairs distincts observés augmente encore de façon significative : plus de 2 500 nouveaux pairs par jour.

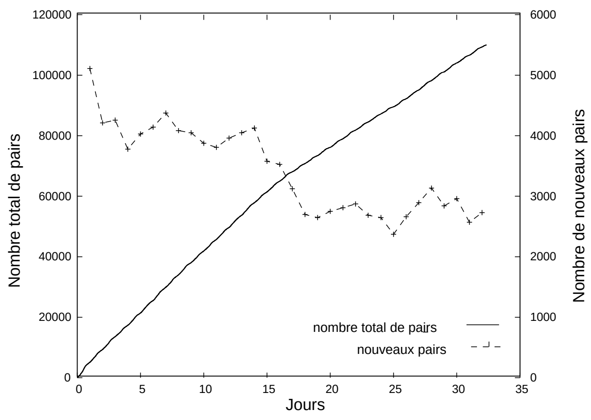


FIGURE B.8 – Évolution du nombre de pairs distincts observés depuis le début de notre mesure (axe vertical de gauche) et du nombre de nouveaux pairs observés chaque jour (axe vertical de droite) en fonction du temps écoulé depuis le début de cette mesure, en jours (axe horizontal).

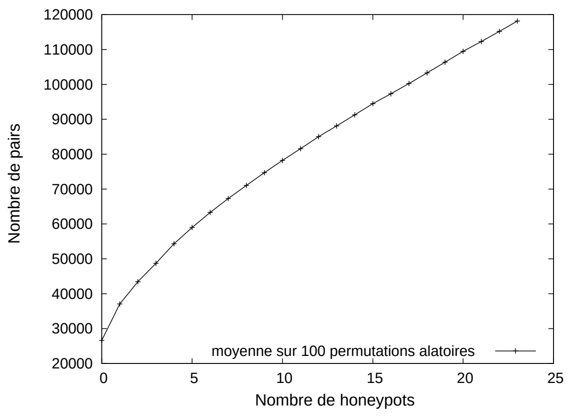


FIGURE B.9 – Nombre de pairs distincts observés à la fin de la mesure (axe vertical) en fonction du nombre n de *honeypots* impliqués (axe horizontal). Pour chaque n, on échantillonne 100 ensembles aléatoires de n *honeypots* et on trace la valeur moyenne obtenue.

Le nombre de nouveaux pairs découverts chaque jour diminue au cours du temps, ce qui est probablement dû au fait que la popularité des fichiers partagés a diminué. Ceci peut également être dû au fait que nous parvenons à une situation où la plupart des pairs qui s'intéressent aux fichiers proposés ont déjà contacté un de nos *honeypots*. Le point important est que cela se produit seulement après une longue durée de mesure (un mois). Ceci signifie que la poursuite de la mesure pendant de longues périodes de temps a un sens, même avec 24 *honeypots* distribués partageant 4 fichiers seulement.

Impact du nombre de *honeypots* utilisés. Nous étudions maintenant l'avantage d'utiliser plusieurs *honeypots*, en termes de nombre de pairs observés. Pour cela, nous étudions ce que nous aurions obtenu si une partie seulement de nos *honeypots* avaient été utilisés. Pour ce faire, nous sélectionnons n *honeypots* parmi les 24, et nous calculons le nombre de pairs distincts observés par ces n *honeypots*. Comme le choix des n honeypots peut avoir une influence significative sur le résultat, nous répétons ce calcul avec 100 échantillons aléatoires, et nous traçons les valeurs moyennes obtenues dans la Figure B.9.

Cette figure montre clairement que, même lorsque 24 *honeypots* sont utilisés, il y a un avantage significatif à en ajouter encore. Toutefois, le bénéfice obtenu en utilisant plus de *honeypots* diminue progressivement, et on peut imaginer qu'au bout d'un moment en ajouter n'apporte qu'un bénéfice négligeable.

References

[1] aMule: `http://www.amule.org/`.

[2] planet-lab: `http://www.planet-lab.org/`.

[3] William Acosta and Surendar Chandra. Trace driven analysis of the long term evolution of gnutella peer-to-peer traffic. In *PAM*, pages 42–51, 2007.

[4] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2003.

[5] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Workshop on Query Log Analysis at the 16th World Wide Web Conference*, 2007.

[6] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5, 2000.

[7] Ian Foster Adriana Iamnitchi, Matei Ripeanu. Small-world file-sharing communities. *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies In INFOCOM*, 2, 2004.

[8] F. Aidouni, M. Latapy, and C. Magnien. Ten weeks in the life of an eDonkey server. In *Proceedings of the Sixth International Workshop on Hot Topics in Peer-to-Peer Systems*, 2009.

[9] R Albert, H Jeong, and A-L Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–82, 2000.

[10] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, Jan 2002.

[11] Oussama Allali, Matthieu Latapy, and Clémence Magnien. Measurement of eDonkey activity with distributed honeypots. In *Proceedings of Sixth International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P'09)*, 2009. In conjunction with IEEE IPDPS'09.

[12] Oussama Allali, Clémence Magnien, and Matthieu Latapy. Internal link prediction: a new approach for predicting links in bipartite graphs. Submitted.

[13] Oussama Allali, Clémence Magnien, and Matthieu Latapy. Link prediction in bipartite graphs using internal links and weighted projection. In *Proceedings of the third International Workshop on Network Science for Communication Networks (Netscicom'11)*, 2011. In Conjunction with IEEE Infocom'11.

[14] Oussama Allali, Lionel Tabourier, Clémence Magnien, and Matthieu Latapy. Internal links and pairs as a new tool for the analysis of bipartite complex networks. Submitted.

[15] Mark Alllman and Vern Paxson. Issues and etiquette concerning use of shared measurement data. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 135–140, New York, NY, USA, 2007. ACM.

[16] Juan A. Almendral and Albert Díaz-Guilera. Dynamical and spectral properties of complex networks. *New Journal of Physics*, 9, 2007.

[17] M.M. Babu, N.M. Luscombe, L. Aravind, M. Gerstein, and S.A. Teichmann. Structure and evolution of transcriptional regulatory networks. *Curr Opin Struct Biol*, 14(3):283–291, 2004.

[18] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks : Membership, growth, and evolution. In *Proc. 12th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2006.

[19] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'11)*, 2011.

[20] Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.

[21] A. L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311(3-4):590 – 614, 2001.

[22] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509, 1999.

[23] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11), March 2004.

[24] Marc Barthélem, Alain Barrat, Romualdo Pastor-Satorras, and Alessandro Vespignani. Characterization and modeling of weighted networks. *PHYSICA A*, 346, 2005.

[25] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[26] Vladimir Batagelj, Patrick Doreian, and Anuska Ferligoj. Generalized blockmodeling of two-mode network data. *Social Networks*, 26(1):29–53, 2004.

[27] S. Battiston and M. Catanzaro. Statistical properties of corporate board and director networks. *European Physical Journal B*, 38, 2004.

[28] Nesserine Benchettara, Rushed Kanawati, and Céline Rouveirol. Supervised machine learning applied to link prediction in bipartite social networks. In *Advances in Social Networks Analysis and Mining (ASONAM'10)*, 2010.

[29] Nesserine Benchettara, Rushed Kanawati, and Céline Rouveirol. A supervised machine learning link prediction approach for academic collaboration recommendation. In *Proceedings of the fourth ACM conference on Recommender systems*, 2010.

[30] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statstical Mechanics : Theory and Experiment*, page P10008, 2008.

[31] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Combin*, 1:311–316, 1980.

[32] B. Bollobás. Random graphs. *Academic Press*, 1985.

[33] Stephen P. Borgatti and Martin G. Everett. Regular blockmodels of multiway, multimode matrices. *Social Networks*, 14, 1992.

[34] Stephen P. Borgatti and Martin G. Everett. Network analysis of 2-mode data. *Social Networks*, 19(3):243–269, 1997.

[35] John Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

[36] B. Bui-Xuan, A. Ferreira, and A. Jarry. Evolving graphs and least cost journeys in dynamic networks. In *Proceedings of WiOpt'03 Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks*, pages 141–150, 2003.

[37] Guido Caldarelli, Stefano Battiston, Diego Garlaschelli, and Michele Catanzaro. Emergence of complexity in financial networks. *Lecture Notes in Physics*, 650, 2004.

[38] R. Calegari, M. Musolesi, F. Raimondi, and C. Mascolo. CTG : A connectivity trace generator for testing the performance of opportunistic mobile systems. In *European Software Engineering Conference and the International ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE07)*, 2007.

[39] D.S. Callaway, M.E.J. Newman, S.H. Strogatz, and D.J. Watts. Network robustness and fragility : Percolation on random graphs. *Physical Review Letters*, 85:5468–5471, 2000.

[40] John Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002.

[41] Peter J. Carrington, John Scott, and Stanley Wasserman. *Models and methods in social network analysis.* Structural analysis in the social sciences. Cambridge Univ. Press Cambridge [u.a.], 2005.

[42] A. Chaintreau, J. Crowcroft, C. Diot, R. Gass, P. Hui, and J. Scott. Pocket switched networks and the consequences of human mobility in conference environments. In *WDTN*, pages 244–251, 2005.

[43] Prabhakar Raghavan Christopher D. Manning and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[44] A. Clauset and N. Eagle. Persistence and periodicity in a dynamic proximity network. In *DIMACS Workshop*, 2007.

[45] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[46] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin. Resilience of the internet to random breakdown. *Physical Review Letters*, 85, 2000.

[47] Jean-Philippe Cointet and Camille Roth. Socio-semantic dynamics in a blog network. *In IEEE SocialCom 09 Intl Conf Social Computing*, pages 114–121, 2009.

[48] U. Vazirani D. Aldous. "go with the winners" algorithms. In *proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994)*, pages 492–501, 1994.

[49] Mukund Deshpande and George Karypis. Item based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, 2004.

[50] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks. *Advances in Physics*, 51:1079 – 1187, 2002.

[51] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. Structure of growing networks with preferential linking. *Physical Review Letters*, 85(21):4633–4636, Nov 2000.

[52] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[53] Guler Ergun. Human sexual contact network as a bipartite graph. *Physica A*, 308, 2002.

[54] Katherine Faust. Centrality in affiliation networks. *Social Networks*, 19, 1997.

[55] Katherine Faust, Karin E. Willen, D. Rowlee David, and Skvoretz John. Scaling and statistical models for affiliation networks: patterns of participation among soviet politicians during the Brezhnev era. *Social networks*, 24(3):231–259, 2002.

[56] Ramon Ferrer-i-Cancho and Ricard V. Sole. The small world of human language. In *Proceedings of The Royal Society of London. Series B, Biological Sciences*, November 2001.

[57] F. Le Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in Peer-to-Peer file sharing workloads. In *In 3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2004.

[58] Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical Review E*, 70(5), 2004.

[59] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[60] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4:133–151, 2001.

[61] O. Görlitz, S. Sizov, and S. Staab. Pints: Peer-to-peer infrastructure for tagging systems. In *Proceedings of the Seventh International Workshop on Peer-to-Peer Systems*, 2008.

[62] Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. In *proceedings of the 1-st international workshop on Combinatorial and Algorithmic Aspects of Networking CAAN'04*, 2004.

[63] Jean-Loup Guillaume and Matthieu Latapy. Bipartite structure of all complex networks. *Information Processing Letters (IPL)*, 90(5):215–221, 2004.

[64] Jean-Loup Guillaume, Matthieu Latapy, and Stevens Le-Blond. Statistical analysis of a P2P query graph based on degrees and their time-evolution. In *Proceedings of the 6-th International Workshop on Distributed Computing (IWDC'04)*, 2004.

[65] Assia Hamzaoui, Matthieu Latapy, and Clémence Magnien. Detecting Events in the Dynamics of Ego-centered Measurements of the Internet Topology. In *WiOpt'10: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 491–498, 2010.

[66] S. B. Handurukande, A.-M. Kermarrec, F. Le Fessant, L. Massoulié, and S. Patarin. Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. *SIGOPS Oper. Syst. Rev.*, 40(4):359–371, 2006.

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

[68] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the Conference on Research and Development in Information Retrieval*, 1999.

[69] Zan Huang. Link Prediction Based on Graph Topology: The Predictive Value of Generalized Clustering Coefficient. *Workshop on Link Analysis Dynamics and Static of Large Networks (LinkKDD'06)*, 2006.

[70] Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the Joint Conference on Digital Libraries (JCDL05). ACM*, 2005.

[71] Daniel Hughes, James Walkerdine, Geoff Coulson, and Stephen Gibson. Peer-to-peer: Is deviant behavior the norm on p2p file-sharing networks? *IEEE Distributed Systems Online*, 7(2), 2006.

[72] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of p2p traffic. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134, New York, NY, USA, 2004. ACM.

[73] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proc. 32nd ACM Symposium on Theory of Computing*, 2000.

[74] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40:77–87, 1997.

[75] Yoram Kulbak and Danny Bickson. The emule protocol specification. Technical report, School of Computer Science and Engineering The Hebrew University of Jerusalem, January 2005.

[76] M. Lad, D. Massey, and L. Zhang. Visualizing internet routing changes. *Transactions on Visualization and Computer Graphics, special issue on Visual Analytics*, 2006.

[77] Matthieu Latapy. Grands graphes de terrain – mesure et métrologie, analyse, modélisation, algorithmique. *Mémoire d'habilitation à diriger les recherches, UPMC*, 2007. `http://www-rp.lip6.fr/~latapy/HDR/`.

[78] Matthieu Latapy and Clémence Magnien. Complex network measurements : Estimating the relevance of observed properties. In *Proceedings of IEEE Infocom*, 2008.

[79] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1), 2008.

[80] Stevens Le Blond, Fabrice Fessant, and Erwan Merrer. Finding good partners in availability-aware P2P networks. In *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'09)*, pages 472–484, 2009.

[81] Stevens Le-Blond, Jean-Loup Guillaume, and Matthieu Latapy. Clustering in P2P exchanges and consequences on performances. In *IPTPS*, pages 193–204, 2005.

[82] Stevens Le-Blond, Jean loup Guillaume, and Matthieu Latapy. Clustering in P2P exchanges and consequences on performances. In *Proceedings of the 4-th International workshop on Peer-to-Peer Systems IPTPS'05*, pages 193–204. Springer, 2005.

[83] F. Le Fessant, S. Handurukande, A. M. Kermarrec, and L. Massoulié. Clustering in peer-to-peer file sharing workloads. In *3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, San Diego, CA, February 2004.

[84] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the kazaa network. In *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, page 112, Washington, DC, USA, 2003. IEEE Computer Society.

[85] E. A. Leicht, G. Clarkson, K. Shedden, and M. E. J. Newman. Large-scale structure of time evolving citation networks. *Eur. Phys J. B*, 59, 2007.

[86] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution : Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (ACM TKDD)*, 1(1), 2007.

[87] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management(CIKM '03)*, 2003.

[88] Pedro G. Lind, Marta C. González, and Hans J. Herrmann. Cycles and clustering in bipartite networks. *Physical Review E*, 72(5), Nov 2005.

[89] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1), 2003.

[90] S. H. Strogatz M. E. J. Newman, D. J. Watts. Random graph models of social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(1):2566–2572, February 2002.

[91] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6:161–179, 1995.

[92] Michael Molloy and Bruce Reed. The size of the giant component of a random graph with a given degree sequence. *Comb. Probab. Comput.*, 7:295–305, September 1998.

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

[94] M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Physical Review E*, 64(1), Jun 2001.

[95] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64(1), Jun 2001.

[96] M. E. J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20), Oct 2002.

[97] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45, 2003.

[98] M. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70(5), Nov 2004.

[99] Mark Newman, Albert-László Barabási, and Duncan J. Watts. The structure and dynamics of networks. *Princeton University Press, Princeton, USA*, 2006.

[100] M.E.J. Newman and J. Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3):36122, 2003.

[101] M.E.J. Newman, S.H. Strogatz, and D.J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):26118, 2001.

[102] R. Oliveira, B. Zhang, and L. Zhang. Observing the evolution of internet AS topology. In *ACM SIGCOMM*, 2007.

[103] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

[104] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 2007.

[105] Jean-Jacques Pansiot. Local and dynamic analysis of internet multicast router topology. *Annales des télécommunications*, 62:408–425, 2007.

[106] S.-T. Park, D. M. Pennock, and C. L. Giles. Comparing static and dynamic measurements and models of the Internet's AS topology. In *IEEE Infocom*, 2004.

[107] Saverio Perugini, Marcos André Gonçalves, and Edward A. Fox. A connection-centric survey of recommender systems research. *CoRR*, cs.IR/0205059, 2003.

[108] Bonacich Phillip. Technique for analyzing overlapping memberships. *Sociological Methodology*, 4, 1972.

[109] C. Prieur, D. Cardon, J.S. Beuscart, N. Pissard, and P. Pons. The stength of weak cooperation: A case study on flickr. *Computing Research Repository*, 2008.

[110] Robert J J. Prill, Pablo A A. Iglesias, and Andre Levchenko. Dynamic properties of network motifs contribute to biological network organization. *PLoS Biol*, 3(11):1881–1892, 2005.

[111] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Collaborative Work Conference*, 1994.

[112] Paul Resnick and Hal Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[113] J. M. Roberts. Correspondence analysis of two-mode network data. *Social Networks*, 22:65–72, 2000.

[114] G. Robins and M. Alexander. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory*, 10, May 2004.

[115] Camille Roth and Paul Bourgine. Epistemic communities: Description and hierarchic categorization. *Mathematical Population Studies*, 12:107–130, 2005.

[116] Walid Saddi and Fabrice Guillemin. Measurement based modeling of edonkey peer-to-peer file sharing system. *Managing Traffic Performance in Converged Networks*, pages 974–985, 2007.

[117] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18, 1975.

[118] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, 1986.

[119] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Storage and Retrieval for Image and Video Databases*, 2002.

[120] A. Scherrer, P. Borgnat, E. Fleury, J.-L. Guillaume, and C. Robardet. Description and simulation of dynamic mobility networks. *Computer Network*, 52:2842–2858, 2008.

[121] Fabian Schneider, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. Understanding online social network usage from a network perspective. In *Internet Measurement Conference*, pages 35–48, 2009.

[122] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Trans. Netw.*, 12(2):219–232, 2004.

[123] John Skvoretz and Katherine Faust. Logit models for affiliation networks. *Sociological Methodology*, 29:253–280, 1999.

[124] Ralf Steuer, Adriano Nunes Nesi, Alisdair R. Fernie, Thilo Gross, Bernd Blasius, and Joachim Selbig. From structure to dynamics of metabolic pathways : application to the plant mitochondrial TCA cycle. *Bioinformatics*, 23(11), 2007.

[125] Alina Stoica and Christophe Prieur. Structure of neighborhoods in a large social-network. In *IEEE International Conference on Social Computing (SocialCom'09)*, 2009.

[126] Steven H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001 2001.

[127] D. Stutzbach and R. Rejaie. Improving lookup performance over a widely-deployed dht. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[128] Pierre Ugo Tournoux, Jérémie Leguay, Marcelo Dias de Amorim, Farid Benbadis, Vania Conan, and John Whitbeck. The accordion phenomenon : Analysis, characterization, and impact on dtn routing. In *Proceedings of the 28rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1116–1124, 2009.

[129] Tomasz Tylenda, Ralitsa Angelova, and Srikanta Bedathur. Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis (SNA-KDD '09)*, 2009.

[130] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[131] A. Vázquez, R. Dobrin, D. Sergi, J. P. Eckmann, Z. N. Oltvai, and A. L. Barabási. The topological relationship between the large-scale attributes and local interaction. *patterns of complex networks*, 101:17940–17945, 2004.

[132] Fabien Viger and Matthieu Latapy. Random generation of large connected simple graphs with prescribed degree distribution. In *proceedings of the 11-th international conference on Computing and Combinatorics (COCOON'05)*, 2005.

[133] S. Voulgaris, A.M. Kermarrec, L. Massoulié, and M. Van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *In 10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004), Suzhu*, 2004.

[134] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. *Data Mining, IEEE International Conference on*, 0:322–331, 2007.

[135] S. Wasserman and K. Faust. *Social network analysis.* Cambridge University Press, 1994.

[136] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684), jun 1998.

[137] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, April 1998.

[138] Sholom M. Weiss, Sholom M. Weiss, Nitin Indurkhya, and Nitin Indurkhya. Lightweight collaborative filtering method for binary encoded data. In *Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2001.

[139] Demetris Zeinalipour-yazti and Theodoros Folias. Quantitative analysis of the gnutella network traffic. Technical report, Department of Computer Science University of California, 2002.

[140] Katharina Zweig and Michael Kaufmann. A systematic approach to the one-mode projection of bipartite graphs. *Social Network Analysis and Mining*, pages 1–32, 2011. 10.1007/s13278-011-0021-0.

Structure et dynamique des graphes de terrain bipartis : liens internes et prédiction de liens

---

Résumé.

Beaucoup de graphes de terrain comme les relations acteur-film ou fichier-fournisseur sont modélisables par des graphes bipartis, dont les nœuds sont divisés en deux ensembles avec des liens entre les nœuds de différents ensembles seulement. Cependant, des méthodes manquent actuellement pour analyser correctement ces graphes, la plupart des méthodes existantes étant conçues pour des graphes classiques. Une approche courante, mais limitée, consiste à transformer les graphes bipartis en graphes classiques, par un procédé appelé projection. Cependant ceci entraîne une perte importante d'informations. Nous introduisons dans cette thèse les *liens internes*, et les proposons comme une nouvelle notion importante pour analyser les graphes de terrain bipartis : elle permet de mesurer la redondance dans ces graphes, et de mesurer la perte d'information entre un graphe biparti et ses projections. Nous montrons en étudiant différents jeux de données que les liens internes sont très fréquents, et que les statistiques associées permettent de souligner leurs ressemblances et leurs différences avec les graphes bipartis aléatoires. Ensuite, nous montrons que nous pouvons tirer profit de cette notion pour modéliser les graphes de terrain bipartis et les stocker dans un format compact.

La plupart des graphes de terrain sont de plus dynamiques, c'est-à-dire que leur structure évolue au fil du temps par l'ajout et/ou le retrait de nœuds et/ou de liens. L'étude de la dynamique des graphes de terrain peut s'aborder par le problème de *la prédiction de nouveaux liens* dans ces graphes. Plusieurs travaux ont étudié le problème de la prédiction de liens dans les graphes classique (non-bipartis). Toutefois, leurs méthodes ne sont pas directement applicables aux graphes bipartis ou sont inappropriées. Nous proposons une approche basée sur les liens internes pour la prédiction dans les graphes bipartis. Nous montrons que notre méthode fonctionne très bien, beaucoup mieux que l'approche de recommandation classique.

Mots clés. graphes de terrain, graphes bipartis, projection, liens internes, stockage de graphe, analyse de graphe, dynamique de graphe, prédiction de liens.
