



## MASTER DEGREE INTERNSHIP REPORT

Computer Science Department  
ESIEE Paris

---

### A STUDY OF A RANDOM MODEL PRESERVING MAXIMAL BICLIQUES IN BIPARTITE GRAPHS

---

By

**Cécile POV**

At

**Complex Networks team, LIP6**  
Sorbonne University and French National Center for Scientific Research  
(UMR 7606 Sorbonne University - CNRS)

Submitted in October 2020

Defense Committee composed of:

Mr Fabien TARISSAN	Complex Networks - LIP6, CNRS	Internship supervisor
Mr Lionel TABOURIER	Complex Networks - LIP6	Internship supervisor
Mr Nabil Hassan MUSTAFA	ESIEE Paris, LIGM	University supervisor

# Contents

<b>Acknowledgements</b>	4
<b>Abstract</b>	5
<b>1 Introduction</b>	<b>6</b>
Complex networks in nature and society.	6
Complex networks as mathematical objects.	7
The complexity and emergency of studying real-world networks.	7
Purpose of this work.	7
<b>2 Useful notions</b>	<b>9</b>
2.1 Family of graphs	9
2.1.1 Theory	9
Graph	9
Directed graphs	9
Weighted graphs	9
Multipartite and bipartite Graphs	10
Notion of projection	11
2.2 Network properties	12
2.2.1 Feature for both unipartite and bipartite graphs	12
Adjacency and neighbourhood of a vertex	12
Density	12
Path	13
Connected component	13
Degree and degree distribution	13
2.2.2 Unipartite features only	14
Path length	14
Assortativity	14
Clustering coefficient	15
2.2.3 Bipartite features only	16
Bipartite clustering coefficient	16
Redundancy	16
Biclique and maximal biclique	17
<b>3 Problem</b>	<b>18</b>
3.0.1 Highlighting the problem	18
Bicliques and maximal bicliques in nature and society.	18
Bicliques and bipartite Configuration Model.	18
3.0.2 A tripartite model	19
Step 2: Tripartite encoding	20
Step 2: Randomization process	21
Step 3: Bipartite projection	23
Multiple edges	24
3.0.3 Methodology and purpose of the work	25

<b>4</b>	<b>Implementation</b>	<b>29</b>
4.1	Tools used and organization of the program . . . . .	29
4.2	Some implementation elements . . . . .	30
4.2.1	Graph modelisation . . . . .	30
4.2.2	Finding maximal bicliques on bipartite graphs . . . . .	30
	Consensus set . . . . .	31
	Theorem 1 . . . . .	31
	Theorem 2 . . . . .	31
4.2.3	Returning a set of non-overlapping maximal bicliques . . . . .	35
4.2.4	Tripartite encoding . . . . .	35
4.2.5	Paths calculation . . . . .	35
	Breadth First Search . . . . .	35
	Depth First Search . . . . .	35
4.2.6	How to use the code . . . . .	35
	Damaschke . . . . .	35
<b>5</b>	<b>Experiments, analysis and results</b>	<b>36</b>
5.1	Datasets used . . . . .	36
5.2	Results . . . . .	36
	Appendix - how to read plot of this section . . . . .	36
5.2.1	Results for bipartite graphs . . . . .	39
	Degrees . . . . .	39
<b>6</b>	<b>Conclusion and perspective</b>	<b>43</b>

# Acknowledgments

Foremost, my sincere gratitude goes first to my internship advisors, Prof. Lionel Tabourier and Fabien Tarissan for giving me the golden opportunity to do research and for their guidance, patience and caring. Thank you for trusting me and letting me build my own research experience, by exploring the aspects of the project I enjoyed and was curious about, without losing sight of my initial objective. I greatly appreciate the time they have taken to share their knowledge with me, but also by giving me technical and personal advice despite their busy schedules. It was a pleasure to work with both of them.

I would like to acknowledge Prof. Nabil Hassan Mustafa for accepting being my university supervisor for this internship, but also for all the precious advice he gave me. His algorithms classes, along with Prof. Jean Cousty and Michel Couprie ones, have surely grown my interest in graph algorithms. Thanks a lot for those great knowledge I learned during my studies.

I would also like to thank my whole research team, ComplexNetworks, for welcoming me. To Maximilien Danisch, Matthieu Latapy, Clémence Magnien, Hong-Lân, Alexis, my two supervisors and other members I occasionally met - despite the particular circumstances linked to the COVID-19 pandemic, I really enjoyed each and every one of the interactions I had with them. The small talks I had with some of them reflect their positiveness, kindness, dedication and passion for their work. I hope that we will cross paths again, so I can know everybody better. Many thanks for sharing their experience, research activities and anecdotes with me, and also for those lunches at "Arènes de Lutèce". Their mail exchanges in the "flood" mailing list during the lockdown were (and are still!) a pleasure to read, and were a flavour of the good atmosphere to expect in the corridor 25-26.

I must mention my intern-workmates, Matthieu Pont and Hugo Manet, with whom I had the pleasure to share the same 26-00/301 office with. Thanks for creating such a studious and friendly atmosphere in our room. I would also like to thank the Ph.D. students from team APR: Jules Vidal, Martin Pépin, Yi-Ting Chen and Raphaël Monat, and other members that I occasionally met. Thanks for those lunches, break times, video games we had together. I really appreciate their company, support and advice.

At last, I would like to convey my heartiest thanks to my friends and family for giving me the constant support I needed during all this time.

About graph theory, the research world but also myself - I came to know more about so many things during this experience. But I still have so much more to learn. I am really thankful to all the inspiring people I have met as an intern at LIP6 and as a student at ESIEE Paris.

## Abstract

Real-world networks share non-trivial properties, such as a skewed degree-distribution, a low global density, a high local density, etc. However, existing random graph models do not capture all the properties of real world networks at the same time. In particular, the bipartite configuration model achieves in preserving a degree distribution close to the original network one, but fails in generating graphs that keeps the overlapping structures.

F. Tarissan & L. Tabourier proposed a random model that relies on maximal bicliques to preserve overlaps in bipartite networks, by exploiting the tripartite version of the configuration model [17]. The purpose of this study is to tackle the realistic random graph model problem, by evaluating the relevance of the tripartite model as a possible generalized answer to this issue. We will both validate and further current knowledge of this model, by examining other characteristics of real-world network.

**Keywords:** bipartite graphs, configuration model, tripartite encoding, maximal bicliques, random graph models

# Chapter 1

## Introduction

### **Complex networks in nature and society.**

The World Wide Web, biological networks and human social interactions are only a few famous example of complex networks that surround us. A complex network is a system composed by a large number of interconnected dynamical units. From natural systems – such as proteins interaction, food web or social networks – to infrastructure networks – power grids, transports or the World Wide Web – complex networks form the backbone of modern society.

In recent years, we have collectively gathered more diverse data than ever before. The growing demand of data specialists and interest in data enhancement are only the tip of the iceberg proving that there is an emergency of understanding all this data. More and more frequently, what we are interested in the relationship, i.e. how the data is related: it is precisely this type of information that is captured by complex networks.

Understanding the underlying structure of such networks would allow us to have a better understanding of various real-world systems and interactions that emerge from diverse disciplines – or, even better, predict them and control them.

For instance, network science, which is the study of complex networks, has many applications in biology. In biological neuronal networks, where interactions between neurons are represented, small-world property has been demonstrated in connections between cortical areas of the primate brain, suggesting that cortical areas of the brain are not directly interacting with each other, but most areas can be reached from all others through only a few interactions [1]. For protein-protein interaction networks, it has been discovered that proteins with high degrees of connectedness are more likely to be essential for survival than proteins with lesser degrees [12]. In food web, complex network analysis is used to measure the global impact of a species removal in the web [9]. This work is particularly critical considering the potential extinction of species due to global climate change.

Besides biology, the study of complex networks has infinite applications. In sociology, for example, social network analysis plays a central role in understanding social mechanisms. Thus, it has been used to identify influential actors in criminal gangs or understand disagreements among scientific concepts [4]. Also, if temporality in complex networks is considered, i.e. if we know at which time  $t$  two elements of the network had interacted together, we may be able to detect and predict cyber attack in network traffic, or suspicious activities in financial transactions [15].

More particularly, the study of diffusion in complex networks makes an important contribution not only in the understanding of behavior adoption, but also in infectious diseases spread [5]. Knowing the nature of diffusion (information, behavior, disease, etc.) and the underlying diffusion pattern of the network allows us to understand why things that we would like to spread, such as barrier gestures and social distancing, has often more difficulty to diffuse than things that we would want to prevent from spreading: the COVID-19 pandemic is a sad example of this phenomena. This knowledge provides precious information for both prediction and control of most type of diffusion.

**Complex networks as mathematical objects.**

From a mathematical point of view, complex networks are viewed as graphs, where network entities are represented as vertices and edges are interactions between them. This mathematical tool has two main advantages: the first one is, we benefit from all the powerful measurement tools and algorithms developed in the field of graph theory, which are a good fit for network science. The second reason is that we provide a universal tool to analyze data coming from various disciplines, as long as we have entities interacting together.

At the end of the last century, with major progress in network science and computer performance, it has been found out that, despite the apparent randomness of the World Wide Web network, the Internet topology exhibits some surprisingly simple power-law distribution for degree [11] and other non-trivial distributions for other measured metrics. These new results confirm that complex networks can be modeled as either regular graphs (same number of edges for every vertex and predictable neighbors) or completely random graphs (edges appear totally randomly). Further works revealed that most real-world networks – in contrast with graphs generated theoretically or with a model – share non-trivial properties, such as a skewed degree distribution, a low global density, a high local density, etc. This discovery immediately reveals a new approach to analyze complex networks.

**The complexity and emergency of studying real-world networks.**

Since the beginning of 2000s, many studies have been conducted in order to capture those properties in theoretical models. These models are used for both network simulation and complex network analysis. In fact, real-world networks are often compared to random graphs generated with a model: the aim is to determine whether their underlying structure or statistical measures could be the result of a purely random process or not.

The well-known Erdős–Rényi model already attempts to reproduce the density of real-world networks. In regards to the degree distribution, among those new theoretical models that have been built, the Barabási–Albert model succeeds in generating a heterogeneous (scale-free) degree distribution as observed in most real-world networks [3]. Moreover, the Watts–Strogatz model can reproduce high local density graphs [18], while the Configuration Model generates random graphs with a given degree sequence [16].

However, none of those models should be viewed as a fully realistic real-world network. In fact, both the Erdős–Rényi and Barabási–Albert models fail at producing graphs with a high local density, as well as the Configuration Model. With Watts–Strogatz model, this property is preserved but we end up with an unrealistic degree distribution. In the end, despite all the different attempts and models that have been developed, we are not able to generate one realistic graph that preserves, at the same time, *all* the non-trivial properties observed in most real-world networks. This problem is still an open issue.

Considering various applications of network science in critical tasks and phenomena described above, understanding complex networks is part of the great scientific challenges of our time. Inability of contemporary science to understand real-world networks limits advance in many disciplines, ranging from criminology to biology.

**Purpose of this work.**

This inability to simulate a realistic network is not restricted to 1-mode complex networks, i.e. networks that carry interactions between entities that have the same nature. In fact, this problem also occurs with 2-mode networks. These networks represent interactions between two types of entities. Many real-world networks actually have a 2-mode nature. For example, if we consider the actors collaboration dataset (1-mode), it is in fact more natural to consider the actors-movies dataset (2-mode) which associates each actor to the movies he played in. Similarly, for the co-authorship network (1-mode), it is more realistic to consider the author-paper network (2-mode), where authors are linked to the papers they have signed [14].

In particular, F. Tarissan [17] proposed a random model that relies on maximal bicliques to preserve overlaps in bipartite networks, by exploiting the tripartite version of the configuration model [17][6].

The purpose of this work is to tackle this realistic model for complex networks problem, by analyzing the purposed model. The aim of our work is to both validate and further current knowledge of this model, by examining other characteristics of real-world network such as redundancy, or clustering coefficient.



# Chapter 2

## Useful notions

### Contents

<b>2.1</b>	<b>Family of graphs</b>	<b>9</b>
2.1.1	Theory	9
<b>2.2</b>	<b>Network properties</b>	<b>12</b>
2.2.1	Feature for both unipartite and bipartite graphs	12
2.2.2	Unipartite features only	14
2.2.3	Bipartite features only	16

## 2.1 Family of graphs

### 2.1.1 Theory

#### Graph

A graph  $G = (V, E)$  is a data structure defined by 2 sets:

- $V$  is called the set of nodes/vertices ;
- $E \subseteq V \times V$  are the edges. Each edge  $e = (v_1, v_2)$  links two vertices  $v_1 \in V$  and  $v_2 \in V$ . We say that  $v_1$  and  $v_2$  are endpoints of  $e$ , and  $e$  is incident to the vertices.

In this paper, we denote  $V = \{v_1, v_2, \dots, v_n\}$  where  $n = |V|$  is the number of vertices, and  $E = \{e_1, e_2, \dots, e_m\}$  with  $m = |E|$  the number of edges.  $n$  is also called the size of the graph.

Graphs are often used to model pairwise relations or interactions between entities. For instance, nodes can represent people and edges the friendship between them.

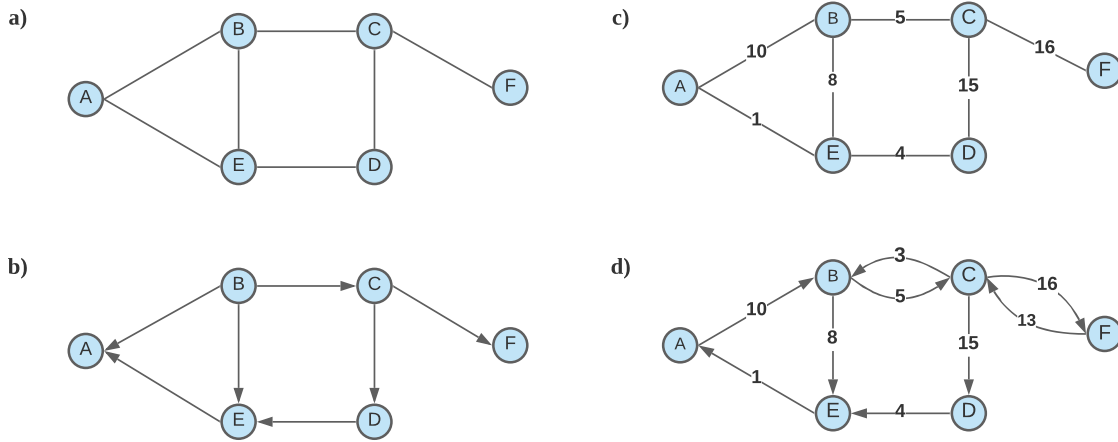
#### Directed graphs

Graphs can be directed or undirected. A graph  $G$  is undirected when for every edge  $e = (v_1, v_2)$  of  $G$ , there is a symmetric relation between  $v_1$  and  $v_2$ , i.e. there is a relation from  $v_1$  to  $v_2$  and also from  $v_2$  to  $v_1$ . The pair  $(v_1, v_2)$  is an unordered pair.

On the contrary, a graph  $G$  is directed when edges are ordered pairs. For every edge  $e = (v_1, v_2)$ , there is a relation from  $v_1$  to  $v_2$  but not from  $v_2$  to  $v_1$ . Edges are represented as arrows in directed graphs.

#### Weighted graphs

An edge may also have a number associated with it, called a weight or a cost. In Fig. 3.3.c., the cost of edge  $(A, B)$  is denoted by a  $W(A, B) = 10$ . Such a graph  $G = (V, E, W)$  is called a weighted graph, with  $W : V \times V \rightarrow \mathbb{R}$  the function that associate a weight to each edge in  $G$ .

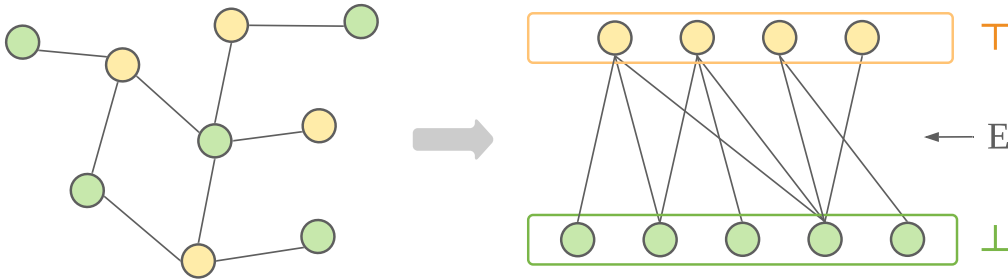


**Figure 2.1:** Examples of directed and weighted graphs. a) An undirected and unweighted graph. b) A directed and unweighted graph. c) An undirected and weighted graph. d) A directed and weighted graph.

Weighted graphs are often used to model real-life information. Thus, the graph of Fig. 3.3.c. can be considered as a map: each vertex is a city and each edge  $(v_1, v_2)$  represents a road between  $v_1$  and  $v_2$ . The weight of each edge is the distance between two cities.

### Multipartite and bipartite Graphs

A  $k$ -partite graph is a graph whose vertices can be divided into  $k$  distinct sets. Edges are only allowed between nodes from 2 distinct sets. In other words, there cannot be any edge between two nodes from the same set. Equivalently, nodes of a  $k$ -partite graph  $G$  can be colored with  $k$  colors, such that every edge  $e$  must have its endpoints colored with different colors.



**Figure 2.2**

A multipartite graph is a  $k$ -partite graph with  $k > 1$ . A  $k$ -partite graph with :

- $k = 1$  is called a unipartite or 1-mode graph.
- $k = 2$  is called a bipartite or 2-mode graph. vertices.
- $k = 3$  is called a tripartite or 3-mode graph.

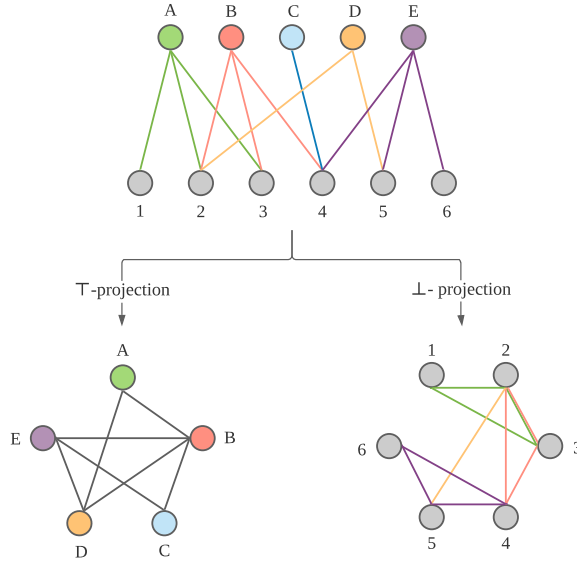
Formally, a bipartite graph is a triplet  $G = (\top, \perp, E)$  where  $\top$  is the set of top nodes,  $\perp$  the set of bottom nodes and  $E \subseteq \top \times \perp$  the set of edges.

### Notion of projection

Let  $G = (\mathbb{T}, \perp, E)$  be a bipartite graph. The  $\perp$ -projection is the graph  $G_{\perp} = (\perp, E_{\perp})$ , in which two vertices of  $\perp$  are linked together in  $G_{\perp}$  if they have at least one mutual neighbor  $n \in \mathbb{T}$  in  $G$ .

The  $\mathbb{T}$ -projection is defined dually.

The projection process allows us to pass from a  $n$ -partite set to a  $(n - 1)$ -partite set. Thus, the bipartite graph  $G$  is compressed into a unipartite graph  $G_{\perp}$  (or  $G_{\mathbb{T}}$ ). Usually, one may consider the  $\perp$ -projection rather than the  $\mathbb{T}$ -projection.

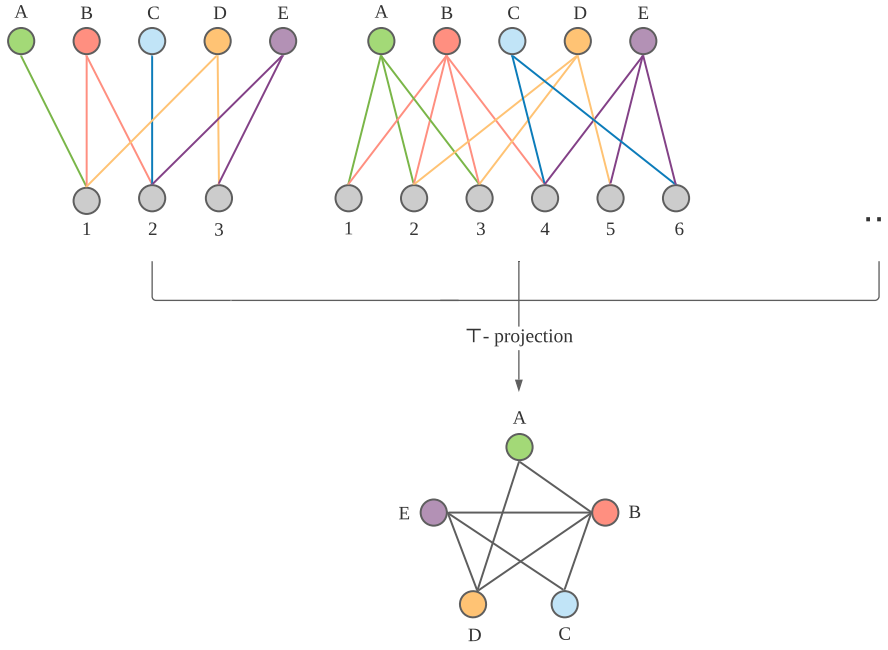


**Figure 2.3:** Example of a  $\mathbb{T}$ -projection and  $\perp$ -projection of a bipartite graph  $G$ . In the  $\mathbb{T}$ -projection, edge  $(A, B)$  exists because  $A$  and  $B$  have *at least* one neighbor in common in  $G$ , which are vertices 2 and 3. On the contrary,  $A$  and  $E$  doesn't share any neighbors in  $G$ , so they are not connected in  $G_{\mathbb{T}}$ . Dually, in the  $\perp$ -projection, 2 and 3 are linked because they share  $A$  and  $B$  as mutual neighbors in  $G$ . Since nodes 1 and 6 are not connected in  $G$ , there isn't any edge  $(A, E)$  in  $G_{\perp}$ .

Projecting a bipartite graph into its  $\mathbb{T}$ -projection allows us to analyze it using the well-studied and powerful tools provided for unipartite graphs. However, the projection process induces a loss of information : projecting a bipartite graph only produce exactly one  $\mathbb{T}$ -projection and exactly one  $\perp$ -projection, but those projections can also be obtained by projecting other bipartite graphs. This means that much information hold by the bipartite structure disappears after the projection. Moreover, some properties observed in the unipartite projection may be due to the projection process rather than the underlying data itself : for instance, the projection may produced a very dense graph, even if the original bipartite graph is not dense.

To avoid some of those side effect, one approach is to use a weighted projection process. For example, the weight of an edge between 2 vertices in the weighted  $\perp$ -projection can be defined as the number of common neighbors in the bipartite graph. Despite important progresses in the recent years, actual weighted projections still lead to information loss.

Thus, it is still necessary to consider the original bipartite graph, because it encodes more information than its projection.



**Figure 2.4:** Examples of bipartite graphs that have the same  $T$ -projection as Fig. 2.3.

## 2.2 Network properties

Several tools and metrics are used to analyze the properties and characteristics of a network. Most of these metrics directly come from graph theory.

### 2.2.1 Feature for both unipartite and bipartite graphs

#### Adjacency and neighbourhood of a vertex

Let  $G = (V, E)$  be a graph and  $v \in V$ . The neighbourhood of a vertex refers to all vertices that are connected to it by an edge. Formally, the neighbourhood of  $v$  in  $G$ , denoted as  $N(v)$ , is the set:

$$N(v) = \{u \in V | (u, v) \in E\}$$

Vertices in  $N(v)$  are called *neighbors* of  $v$ , and we say that they are *adjacent* to  $v$ .

For directed graph, we consider 2 families of neighbourhood:

- in-neighbourhood:  $N^{in}(v) = \{u \in V | (u, v) \in E\}$ .
- out-neighbourhood:  $N^{out}(v) = \{u \in V | (v, u) \in E\}$ .

The neighbourhood of a vertex  $v$  can be seen as a subgraph of  $G$ , composed of vertex neighbors of  $v$  and edges between them. We say that this graph is *induced* by the neighbourhood of  $v$ .

Studying the neighbourhood of a particular vertex / actor is interesting in network analysis, since it may give information about the nature of the connection: in a friendship network, where people are represented as vertices and friendship as edges, some nodes may tend to connect with vertices with a high degree ("people person") or people that have similar hobbies.

#### Density

The density  $\delta(G)$  of an undirected graph  $G$  is defined as:

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

The density is the number of existing edges divided by the number of possible edges.

For a directed graph, we have  $\delta(G) = \frac{m}{n(n-1)}$ .

### Path

A path in a graph  $G = (V, E)$  is a sequence of vertices  $(v_1, v_2, \dots, v_p)$  such that 2 consecutive vertices  $v_i$  and  $v_{i+1}$  of this sequence are connected by an edge  $(v_i, v_{i+1})$ . Dually, a path can also be described by its sequence of edges  $(e_1, e_2, \dots, e_{p-1})$ .

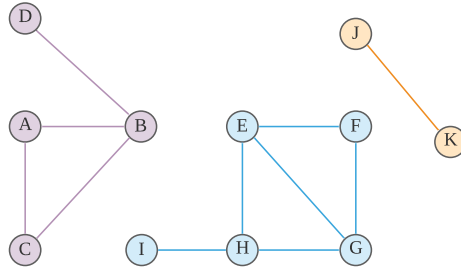
A path is called a *simple graph* if no edges appears more than once.

A path is called an *elementary graph* if no vertices appears more than once.

Note: the literature introduces several definitions for a simple path. Here we are using the one followed by Berge, Liu, Rosen and others. The other definition is similar to the elementary path one.

### Connected component

A connected component of a graph  $G$  is a subgraph in which any pair of vertices are connected together by a path.



**Figure 2.5:** This graph contains 3 connected components. The first one, drawn in purple, is the subgraph induced by  $\{A, B, C, D\}$ . The second one drawn in blue is the subgraph induced by  $\{E, F, G, H, I\}$ . The last connected component is drawn in orange and is induced by vertices  $\{J, K\}$ .

### Degree and degree distribution

The degree of a vertex refers to the number of edges connected to it. Formally, the degree of a vertex  $v$  is defined as:  $d(v) = |N(v)|$ .

In a directed graph, each vertex has an in-degree and an out-degree:

- in-degree: number of edges that are coming into  $v$ , noted as  $d^{in}(v) = |N^{in}(v)|$ .
- out-degree: number of edges that are going out from  $v$ , noted as  $d^{out}(v) = |N^{out}(v)|$ .

The total degree can then be calculated:  $d(v) = d^{in}(v) + d^{out}(v)$ , i.e. the total degree of  $v$  is the sum of edges coming into and going out from  $v$ .

Since degree is a local metric (we can compute it for each vertex of the graph), we can extract statistical measures from it, such as the degree

the average degree of  $G$ :

$$d(G) = \frac{1}{n} \sum_{v \in V} d(v) = \frac{\text{sum of all degrees}}{\text{number of vertices}} = \frac{2m}{n}$$

We note that the sum of all degrees of  $G$  is equal to twice the number of edges ( $2|E| = 2m$ ): intuitively, whenever an edge is introduced in a graph, it will link 2 vertices, so degree of both vertices will increase by

1. As a consequence, every time an edge is added, the sum of all degrees increase by 2, with corresponds to  $2m$ .

### 2.2.2 Unipartite features only

#### Path length

The path length  $L$  is the number of edges in the shortest path between two vertices, averaged over all pairs of vertices.

#### Assortativity

Assortativity measures the preference for vertices with same degree to connect together. In assortative networks, hubs are connected together ("big ones with big ones and small ones with small ones"); on the contrary, high degree nodes tends to connect to low degree nodes in disassortative networks.

In the context of graphs, the assortativity coefficient is the Pearson correlation coefficient of degree between pairs of linked nodes. Pearson correlation coefficient measures the linear correlation between two variables  $X$  and  $Y$ .

We consider:

- $x$  as the initial endpoint of the node;
- $y$  as the terminal endpoint.

Since we are working with undirected graph, we must consider each edge twice (edges  $(i, j)$  and  $(j, i)$ ), so in our case we have  $2 \times m$  samples.

$r$  has a value between  $+1$  and  $-1$  :

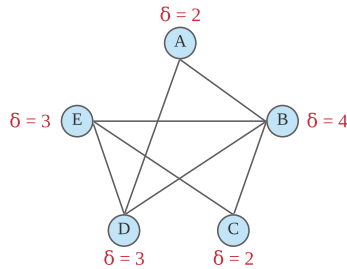
- $r = 1$  : total positive linear correlation between  $x$  and  $y$ ;
- $r = 0$  : no linear correlation between  $x$  and  $y$ ;
- $r = -1$  : total negative linear correlation between  $x$  and  $y$ .

The Pearson correlation is computed as follows:

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

With:

- $n$  the number of samples;
- $\bar{x}$  the mean of all  $x_i$ ;
- $\bar{y}$  the mean of all  $y_i$ .



**Figure 2.6:** Degree  $\delta$  of each vertex of the T-projection in Fig.2.3.

edge	$x_i$	$y_i$	edge	$x_i$	$y_i$
(A,B)	2	4	(B,A)	4	2
(A,D)	2	3	(D,A)	3	2
(B,E)	4	3	(E,B)	3	4
(B,D)	4	3	(D,B)	3	4
(B,C)	4	2	(C,B)	2	4
(C,E)	2	3	(E,C)	3	2

**Table 2.1:** Edges to consider for calculating the assortativity coefficient of the  $\mathbb{T}$ -projection graph of Fig.2.3. Since it is a undirected graph, both edges  $(x, y)$  and  $(y, x)$  must be considered, so there is  $s = 12$  samples. Since  $\bar{y} = \bar{x} = \frac{1}{s} \sum x_i = \frac{1}{12} \times 36 = 3$ , we have  $\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) = -4$  and  $\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 (y_i - \bar{y})^2} = \sqrt{8 \times 8} = 8$ . Thus,  $r_{xy} = -0.5$ : there is a negative linear correlation between edges with same degree.

### Clustering coefficient

The global clustering coefficient  $CC(G)$  measures the cliquishness of a graph  $G$ , i.e. how well the vertices are connected together. It gives a score between 0 and 1: the closer to one the score is, the better the vertices are connected together.

The global clustering coefficient is computed as follows (in general we consider only vertices with degree  $> 1$ ):

$$CC(G) = \text{mean}(CC(v)) \quad v \in G$$

The local clustering coefficient for a vertex  $v$  is equal to:

$$CC(v) = \frac{2E_\delta}{\delta(\delta - 1)}$$

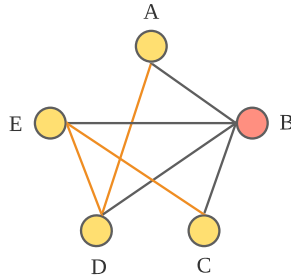
With :

- $E_\delta$  the number of edges between neighbors of  $v$ ;

$CC(v)$  is a "ratio":

- $2E_\delta$  is the actual number of existing interconnections (multiplied by 2 because an edge links 2 nodes);
- $\delta(\delta - 1)$  is the maximum number of possible interconnections

There is at most  $\delta(\delta - 1)/2$  edges between neighbors.



**Figure 2.7:** Clustering coefficient calculations for  $\mathbb{T}$ -projection in Fig. 2.3. In the figure, we explicit the calculation for vertex B: its neighborhood  $\delta$  is composed of vertices A,C,D,E (in yellow) and 3 edges (in orange) exist between nodes of  $\delta$ , so  $E_\delta = 3$ . So we have  $CC(B) = \frac{2E_\delta}{\delta(\delta-1)} = \frac{2 \times 3}{4 \times 3} = \frac{6}{12} = \frac{1}{2}$ . And since  $CC(A) = 1$ ,  $CC(C) = 1$ ,  $CC(D) = \frac{2}{3}$ ,  $CC(E) = \frac{2}{3}$ , the global clustering coefficient of the graph is  $CC(G) = \frac{(1 + \frac{1}{2} + 1 + \frac{2}{3} + \frac{2}{3})}{5} \approx 0,76$

### 2.2.3 Bipartite features only

#### Bipartite clustering coefficient

cc\_bullet, cc\_bullet\_max, cc\_bullet\_min

$$cc_{\bullet}(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} = \frac{\text{common}}{\text{all}}$$

$$cc_{\bullet}(u, v) = \frac{|N(u) \cap N(v)|}{\min(|N(u)|, |N(v)|)}$$

$$cc_{\bullet}(u, v) = \frac{|N(u) \cap N(v)|}{\max(|N(u)|, |N(v)|)}$$

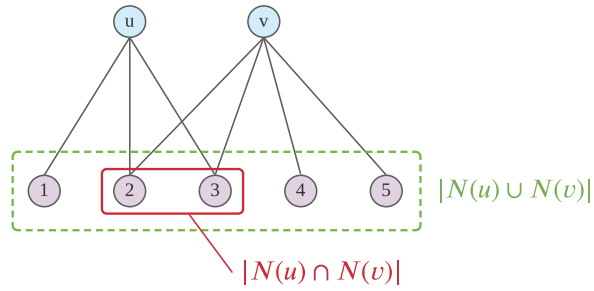


Figure 2.8: test

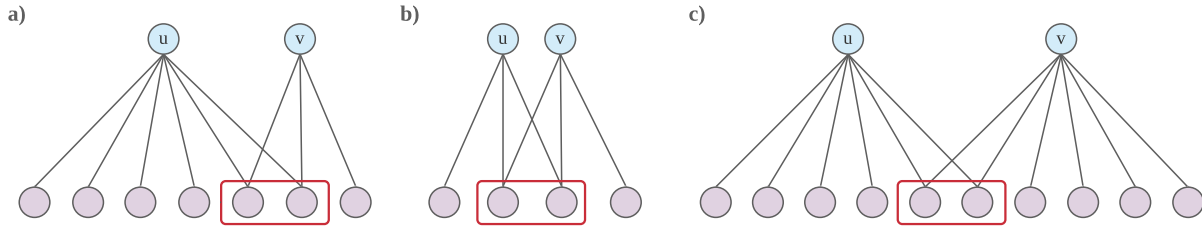


Figure 2.9: test

#### Redundancy

The redundancy coefficient of a vertex  $v$  is the fraction of pairs of neighbors of  $v$  that are both linked to other vertices. In a one-mode projection, these vertices would be linked together even if  $v$  were not there.

More formally, for any vertex  $v$ , the redundancy coefficient of  $v$  is defined by

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

Where  $N(v)$  is the set of neighbors of  $v$  in  $G$ :



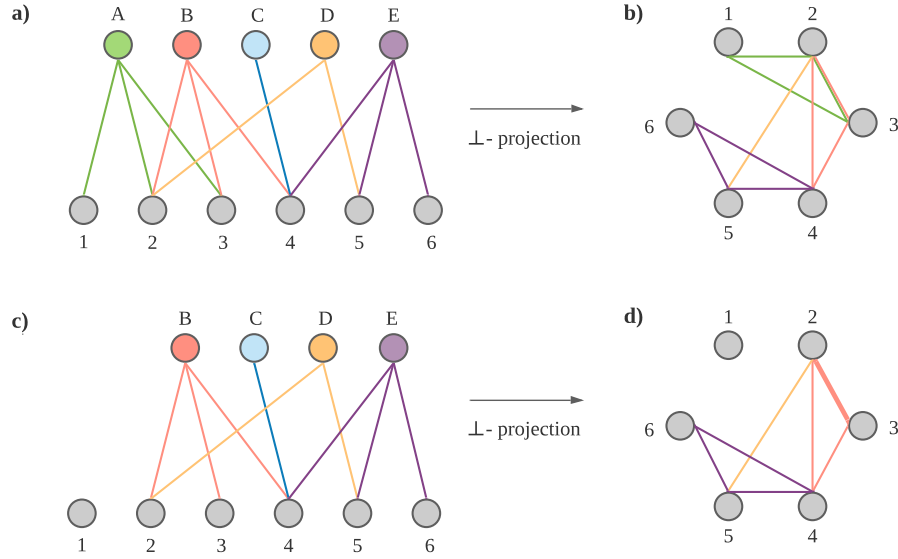


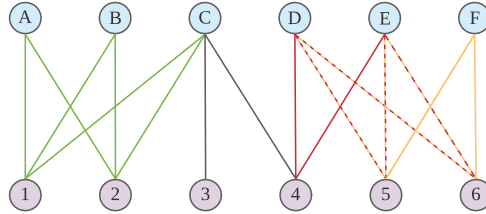
Figure 2.10: test

### Biclique and maximal biclique

A biclique  $(S_{\top}, S_{\perp})$  of a bipartite graph  $G = (\top, \perp, E)$  is subset of  $G$  where every vertex of  $S_{\top} \in \top$  is connected to every vertex of  $S_{\perp} \in \perp$ . A biclique is maximal if no proper superset of  $(S_{\top}, S_{\perp})$  is a biclique, i.e. there is no biclique  $(S'_{\top}, S'_{\perp})$  such that  $S_{\top} \subseteq S'_{\top}$  and  $S_{\perp} \subseteq S'_{\perp}$ .

Informally, a biclique  $(S_{\top}, S_{\perp})$  is maximal if, for any vertex  $v$  that is added to  $S_{\top}$  (or  $S_{\perp}$ ),  $(S_{\top} \cap \{v\}, S_{\perp})$  (or  $(S_{\top}, S_{\perp} \cap \{v\})$ ) is not a biclique. It is a complete bipartite subgraph of  $G$ .

In our case of study, if  $S_{\top}$  or  $S_{\perp}$  are singletons,  $(S_{\top}, S_{\perp})$  will not be considered as a biclique.



**Figure 2.11:** A graph containing 3 bicliques:  $b_1 = (\{A, B, C\}, \{1, 2\})$  with edges in green,  $b_2 = (\{D, E\}, \{4, 5, 6\})$  in red and  $b_3 = (\{D, E, F\}, \{5, 6\})$  in yellow.  $(\{A, B\}, \{1, 2\})$  is a biclique but not a maximal biclique since  $(\{A, B, C\}, \{1, 2\})$  is also a biclique.  $(\{C\}, \{3, 4\})$  is not considered as biclique in our case of study. We note that  $b_1$  and  $b_2$  are overlapping bicliques because they share the 4 dotted edges.

test

# Chapter 3

## Problem

### Contents

3.0.1	Highlighting the problem . . . . .	18
3.0.2	A tripartite model . . . . .	19
3.0.3	Metholodogy and purpose of the work . . . . .	25

### 3.0.1 Highlighting the problem

#### **Bicliques and maximal bicliques in nature and society.**

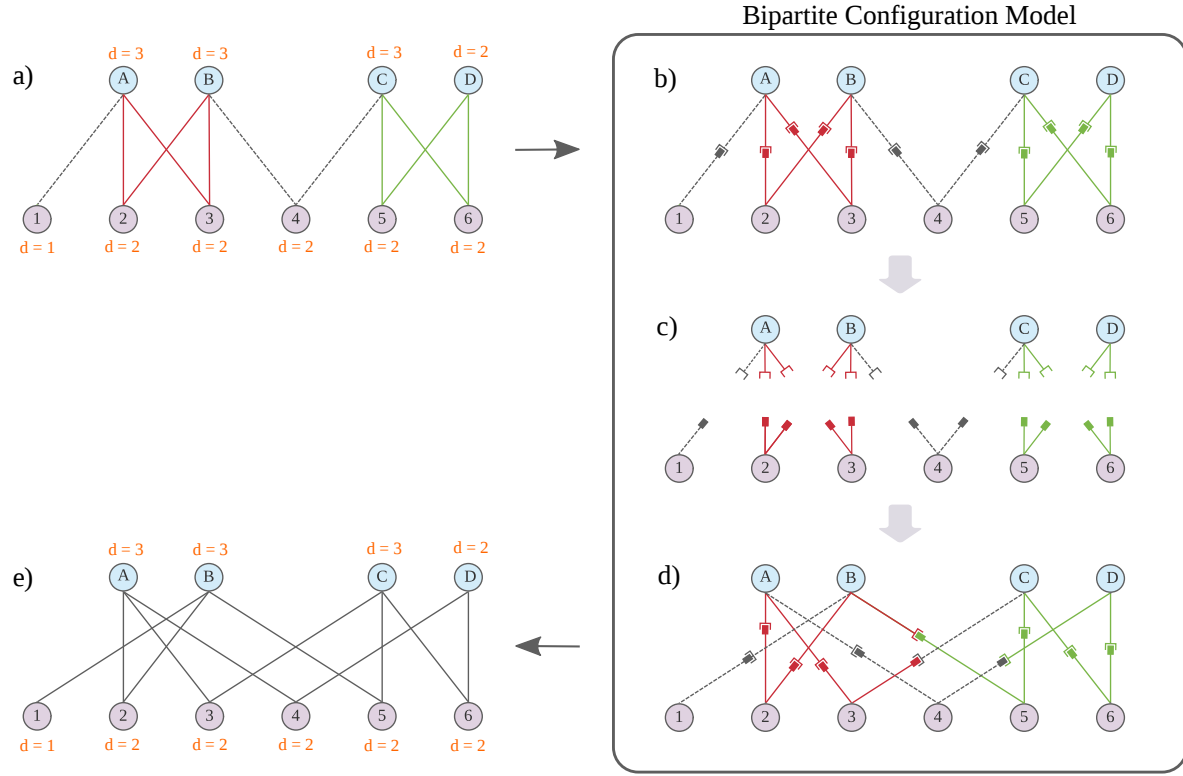
Bicliques have been studied extensively in many different contexts, as they are well-known and fundamental patterns that form naturally in most real-world networks. Enumerating and detecting maximal bicliques is a relevant task in a wide range of domains, ranging from language theory, artificial intelligence to biology.

For instance, in chemistry, enumeration of bicliques can be useful for the analysis of the structure of organic compounds. Also, in gene expression data analysis, different genes will respond in different conditions. The group of genes that have many common responses over multiple conditions is considered as a significant gene group. In protein-protein interaction study, hidden topological structures such as cliques and bicliques, consists of biologically relevant functional groups.

Furthermore, detection of maximal bicliques also finds applications in social and traffic networks. In fact, some interesting interactions can be captured by maximal bicliques. In E-commerce websites, a large group of users purchasing or rating a product together is considered as suspicious, as there is a high probability that they are making fraudulent transactions to increase the rankings of their businesses by selling their own product. In social network analysis, bicliques play a key role in detecting communities.

#### **Bicliques and bipartite Configuration Model.**

Considering various applications of maximal biclique enumeration, it is essential to preserve those patterns in our random models. In particular, the natural extension of Configuration Model to bipartite graphs attempts to preserve the degree distribution of every node while shuffling the edges. However, the bicliques completely vanished. (Fig. 3.1)



**Figure 3.1:** Bipartite version of the configuration model. **a)** A graph with 2 maximal non-overlapping bicliques, in red and green. Each vertex is labeled with its degree  $d$ . **b)** Same graph with visible stubs. As for the unipartite version of the configuration model, we will consider half-edges. The endpoint of a stub that is not linked to a node is represented as either a bracket (top stubs) or a rectangle (bottom stubs). A bracket and a rectangle connected together means that the two stubs they come from form an edge. **c)** Edges of the graph are "cut". **d)** Stubs rematching process. A top (respectively bottom) stub is randomly rematched to any bottom (respectively top) stub. **e)** The initial graph after the randomization process. The initial degrees has been preserved, but the two bicliques in **a)** completely vanished.

### 3.0.2 A tripartite model

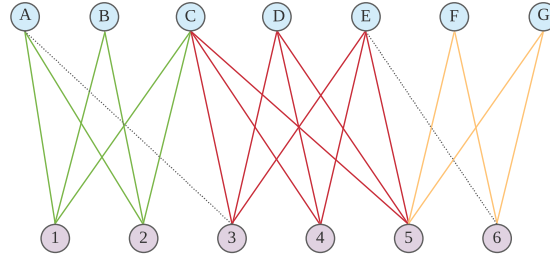
To overcome this issue, F. Tarissan and L. Tabourier purpose a random graph model able to preserve both degree distribution and overlapping structures (maximal bicliques) of real-world networks. This model relies on a third level, defining thus a *tripartite graph*, that will encode the bicliques observed in the bipartite graph.

More precisely, the four main steps of this model are the following:

1. Enumerate all maximal bicliques in the bipartite graph;
2. Encode them in a third level of a tripartite graph;
3. Perform the randomization process on the tripartite graph;
4. Project the tripartite graph into its corresponding bipartite structure.

We will now explain, through an example, how the tripartite encoding (step 2) and randomization process (step 3) are performed.

Let's consider a bipartite graph  $G_{bip} = (X, Y, E)$  (Fig 3.2), where  $X$  is the set of top vertices,  $Y$  the set of bottom vertices and  $E$  the edges.  $M$  refers to a set of non-overlapping maximal bicliques and  $E_{\overline{M}} \in E$  the set of edges involved in none of those bicliques. In the following, a maximal biclique of  $G_{bip}$  is described as an ordered pair  $m_n = (X_n, Y_n)$ , where  $X_n \subseteq X$  and  $Y_n \subseteq Y$ .

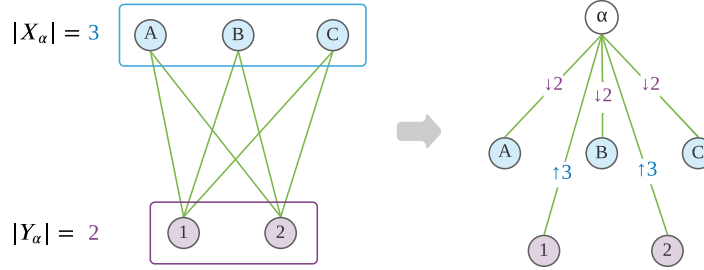


**Figure 3.2:** A bipartite graph  $G_{bip} = (X, Y, E)$  with  $X = \{A, B, C, D, E, F, G\}$  and  $Y = \{1, 2, 3, 4, 5, 6\}$ .  $G_{bip}$  contains 3 non-overlapping maximal bicliques:  $m_\alpha = (\{A, B, C\}, \{1, 2\})$  in green,  $m_\beta = (\{C, D, E\}, \{3, 4, 5\})$  in red, and  $m_\gamma = (\{F, G\}, \{5, 6\})$  in yellow. These 3 bicliques form the set  $M$ . The set of edges  $E_{\overline{M}} = \{(A, 3), (E, 6)\}$ , drawn with dotted lines, define edges that are not involved in any biclique in  $M$ .

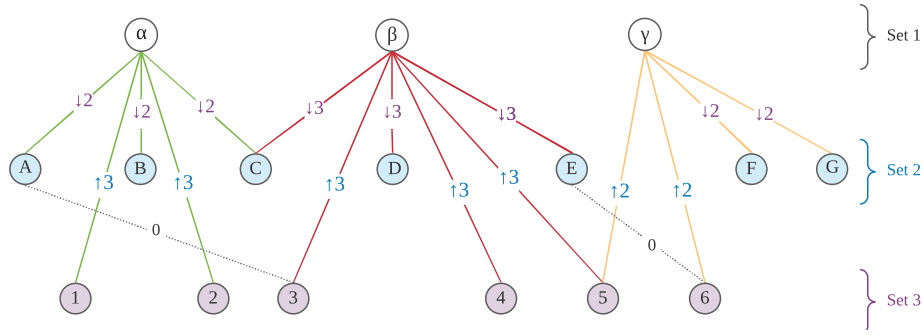
### Step 2: Tripartite encoding

We construct a tripartite graph  $G_{tri} = (W, X, Y, E_{WX}, E_{WY}, E_{XY})$  such that:

- Each biclique  $m_n = (X_n, Y_n)$  of  $M$  is *captured* by a vertex  $n \in W$ . Thus, we link  $n$  to all vertices in  $X_n$  and  $Y_n$ . Formally, the neighbourhood of vertex  $n$  is equal to  $N(n) = X_n \cup Y_n$ .
- $E_{XY} = E_{\overline{M}}$
- Every edge  $\{(n, x_n) \mid x_n \in X_n \wedge n \in W\} \in E_{WX}$  is labeled with  $|Y_n|$ .
- Every edge  $\{(n, y_n) \mid y_n \in Y_n \wedge n \in W\} \in E_{WY}$  is labeled with  $|X_n|$ .
- All edges  $\{(x, y) \mid x \in X \wedge y \in Y\} \in E_{XY}$  are labeled with 0.



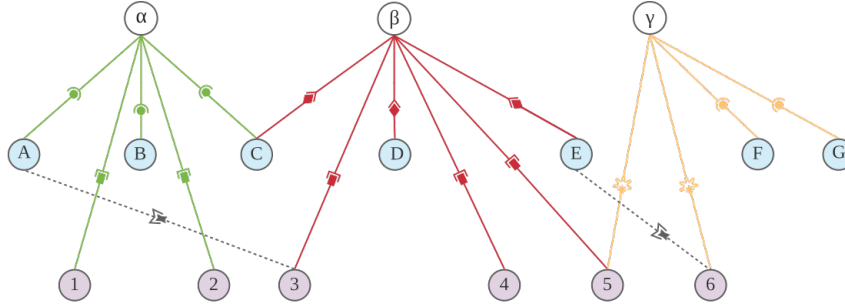
**Figure 3.3:** Maximal biclique encoding in a third level. The figure on the left is the subgraph induced by the maximal biclique  $m_\alpha = (X_\alpha, Y_\alpha) \in M$  of  $G_{bip}$ . In the tripartite graph on the right, a vertex  $\alpha$  is created in an upper level  $W$  to encode the biclique. The vertex  $\alpha$  gathers all the information about  $m_\alpha$ , because it is connected to all elements of  $X_\alpha$  and  $Y_\alpha$ . Since  $|Y_\alpha| = 2$ , the edges  $(\alpha, A)$ ,  $(\alpha, B)$  and  $(\alpha, C)$  are labeled " $\downarrow 2$ ". Therefore, the label of edge  $(\alpha, A)$  means "Vertex A is involved in a maximal biclique  $(X_n, Y_n)$  such as, in the bottom level of  $G_{bip}$ ,  $|Y_n| = 2$ ." Dually, since  $|X_\alpha| = 3$ , the edges  $(\alpha, 1)$  and  $(\alpha, 2)$  are labeled " $\uparrow 3$ ". Therefore, the label of edge  $(\alpha, 1)$  means "Vertex 1 is involved in a maximal biclique  $(X_n, Y_n)$  such as, in the upper level of  $G_{bip}$ ,  $|X_n| = 3$ ." We say that the vertex  $\alpha$  *captures* the maximal biclique  $m_\alpha$ .



**Figure 3.4:** Tripartite graph  $G_{tri}$  encoding the maximal bicliques of  $G_{bip}$ . Each vertex of set  $W$  encodes a maximal biclique of  $G_{bip}$ . Edges in  $E_{XY}$  are the dotted edges in  $G_{bip}$ . Their label "0" is a arbitrary value meaning that the edges aren't encoding any biclique.

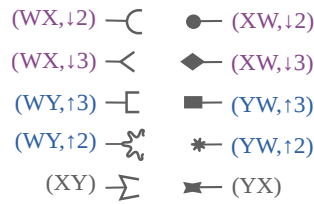
### Step 2: Randomization process

As for the bipartite version of the configuration model, we will consider half-edges. However, in this case, two stubs from different set can be randomly connected only if they stem from edges that have the same label in  $G_{tri}$ . The graph obtained after this rematching process is called  $G_{tri}'$ .



**Figure 3.5:** The same tripartite graph  $G_{tri}$  but with visible stubs. Each label is represented with a unique pair of interlockable symbols.

An edge  $(u, v)$  labelled  $l$  (with  $u \in S_u$  and  $v \in S_v$ , such that  $S_u$  and  $S_v$  are vertices sets) can be cut into two type of stubs;  $(S_u S_v, l)$  and  $(S_v S_u, l)$ . The naming convention used to defined a stub is a pair (set1-set2, label):  $(S_u S_v, l)$  denotes a stub that stems from an edge  $(u, v)$  and have one endpoint linked to a vertex of  $S_u$  while the other endpoint is left floating, unassigned. Dually,  $(S_v S_u, l)$  denotes a stub stemming from an edge  $(u, v)$ , such that one endpoint is linked to a vertex of  $S_v$  while the other endpoint is left floating.

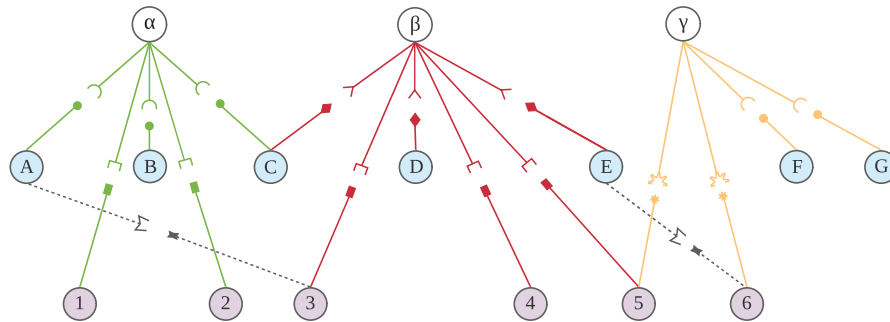


**Figure 3.6:** Types of stubs and their corresponding symbol. For instance, a half-edge of type " $(WX, \downarrow 2)$ " denotes a stub that stems from an edge of  $\{(w, x) | w \in W, x \in X\}$  and with one endpoint linked to a vertex of  $S_u$ , while the other one is left floating.

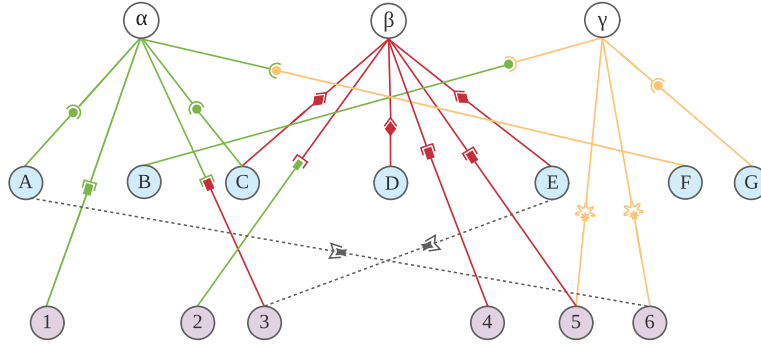
Joining two stubs produces an edge. Two stubs are interlockable only if one is of type  $(S_u S_v, l)$  and the other is of type  $(S_v S_u, l)$ .

interlockable pairs	not interlockable pairs
$\downarrow 2$	
$\downarrow 3$	
$\uparrow 3$	
$\uparrow 2$	
0	...

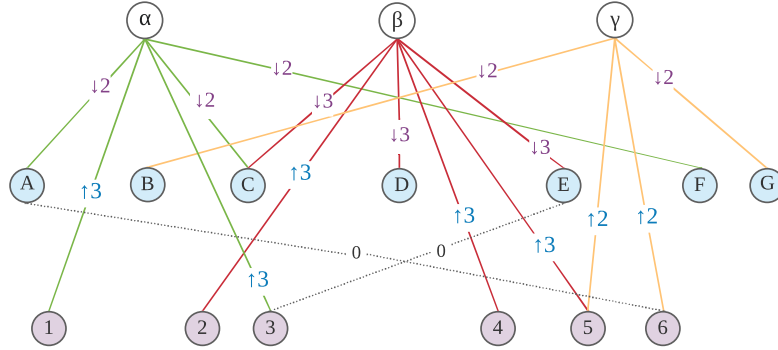
**Figure 3.7:** Interlockable and some non-interlockable symbols of Fig.3.5.



**Figure 3.8:** Edges of the tripartite graph are "cut" and ready to be rematched.



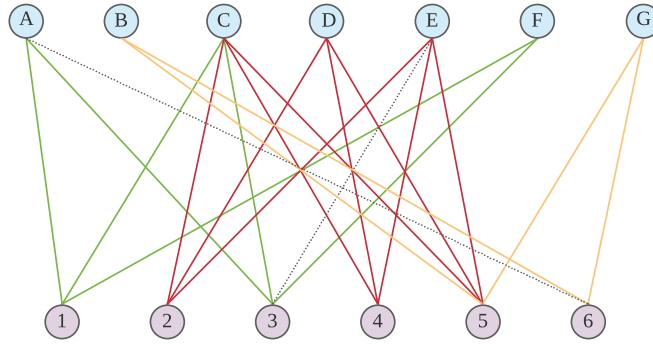
**Figure 3.9:** Stubs rematching process. For each stub, we pick another stub according to Fig.3.7. Stubs matched together may either: stem from the same biclique (e.g. edge  $(\alpha, A)$ ); stem from 2 different biclique (e.g. edge  $(\alpha, F)$ ); stems from 0 biclique for both of them (only possible case for  $\{(x, y) \mid x \in X \wedge y \in W \wedge (x, y) \in E_{XY}\}$  edges).



**Figure 3.10:** Tripartite graph  $G_{tri}' = (W, X, Y, E_{WX}', E_{WY}', E_{XY}')$  obtained after performing randomization process on  $G_{tri}$ . It is the same figure as Fig.3.9 but with the corresponding labels instead of symbols.

### Step 3: Bipartite projection

Then, the tripartite randomized graph  $G_{tri}'$  is projected into its bipartite projection  $G_{bip}' = (X, Y, E')$ : vertices in  $X$  and  $Y$  that are connected to the same vertex  $n$ , with  $n \in W$ , forms a biclique in  $G_{bip}'$ . Edges in  $E_{XY}'$  are simply added to  $E'$



**Figure 3.11:** Graph  $G_{bip}'$  obtained after bipartite projection of  $G_{tri}'$ . The degree distribution of  $G_{bip}$  and  $G_{bip}'$  are equal. We notice three non-overlapping bicliques: one preserved by  $\alpha$  in  $G_{tri}'$  (in green), one by  $\beta$  (in red) and one by  $\gamma$  (in yellow). Since these bicliques have exactly the same size as the ones detected in the initial graph  $G$ , the tripartite model produces a graph preserving both the degree distribution and the maximal bicliques.

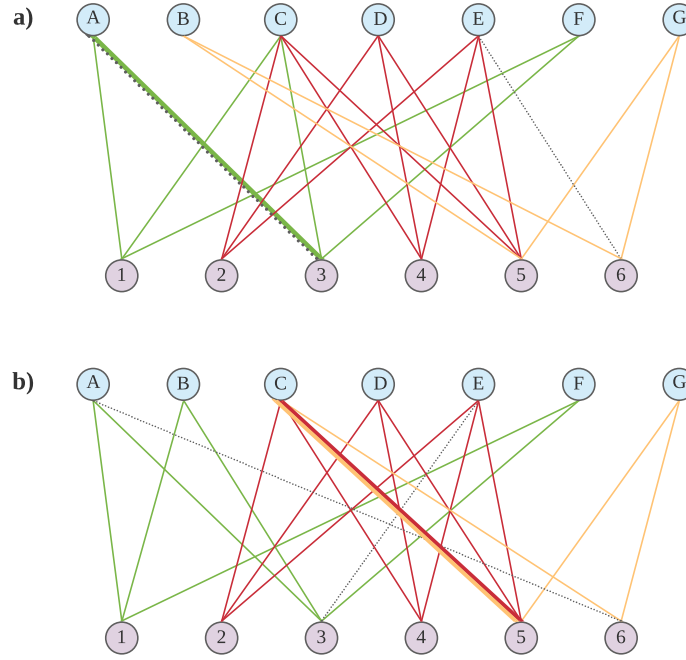
### Multiple edges

Multiple edges can appear during different steps of the tripartite model. In fact, during the randomization process, 2 stubs of a node  $i$  can be connected to 2 other stubs of a node  $j$ . This is the "classical" case that also appears in unipartite and bipartite version of the Configuration Model. It induces a biclique size reduction if stubs from set  $w$  is involved.

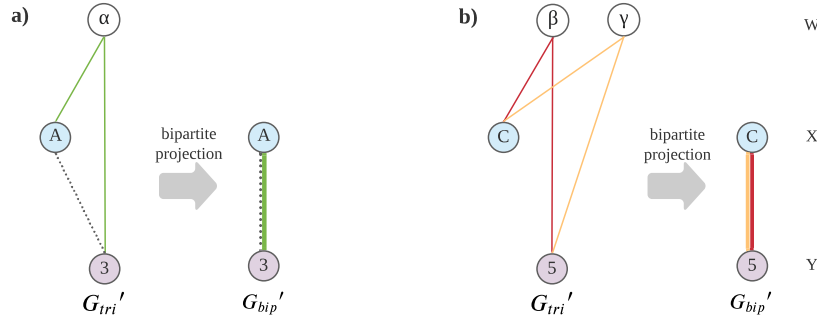
Furthermore, we can also have multiple edge while projecting  $G_{tri}'$  into its bipartite graph  $G_{bip}'$ , between :

- A biclique edge and a non-biclique edge : this case is detectable in the tripartite level, by identifying triangles.
- Two bicliques edges : this case occurs when two vertices in  $W$  both share a vertex in  $X$  and a vertex in  $Y$ . This case is also detectable in the tripartite level, by identifying cycles of length 4, such that  $\{(n_1, x, n_2, y, n_1) \mid \{n_1, n_2\} \in W \wedge x \in X \wedge y \in Y\}$ , where  $(n_1, x, n_2, y, n_1)$  refers to a hamiltonian path of the cycle.
- The maximal biclique  $(\{F, G\}, \{5, 6\})$  (in yellow).





**Figure 3.12:** Two possible multigraphs obtained after projection of alternative randomized tripartite graphs. **a)** Multiple edges between vertices A and 3. One edge were involved in biclique encoding, the other was not. **b)** Multiple edges between C and 5. Both of these edges were involved in biclique encoding



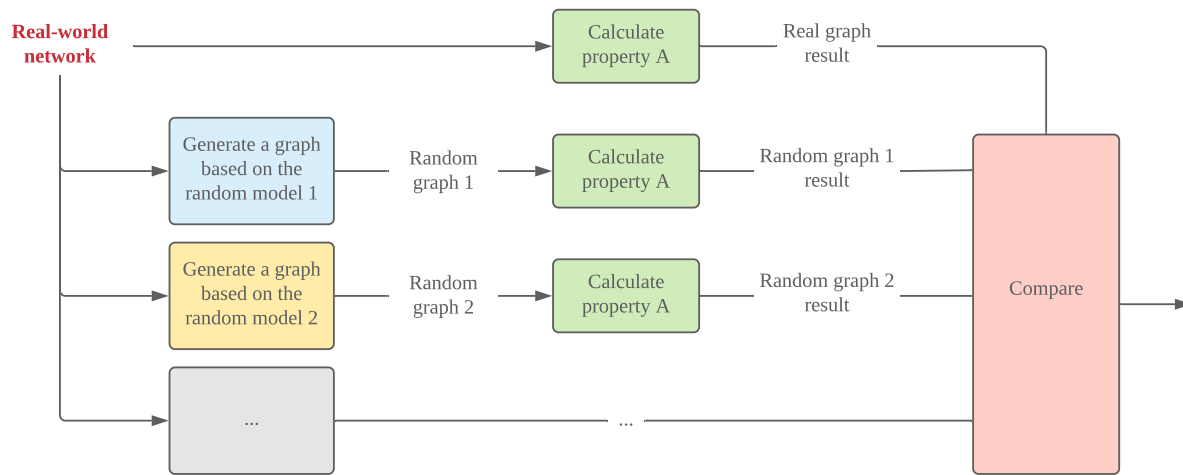
**Figure 3.13:** Patterns in tripartite graphs that form multiple edges in bipartite projection. **a)** The triangle  $\{\alpha, A, 3\}$  is projected into 2 multiples edges between vertices A and 3. Informally,  $A, 3 \subset N(\alpha)$  means that there is an edge  $(A, 3)$  in the bipartite projection, but there is already an edge  $(A, 3)$  in  $G_{tri}'$ . **a)** The cycle  $(\beta, C, \gamma, 5, \beta)$  is projected into 2 multiples edges between vertices C and 5. Informally,  $C, 5 \subset N(\beta)$  means that there is an edge  $(C, 5)$  in the bipartite projection, and similarly  $C, 5 \subset N(\gamma)$  also means that there is an edge  $(C, 5)$ .

### 3.0.3 Methodology and purpose of the work

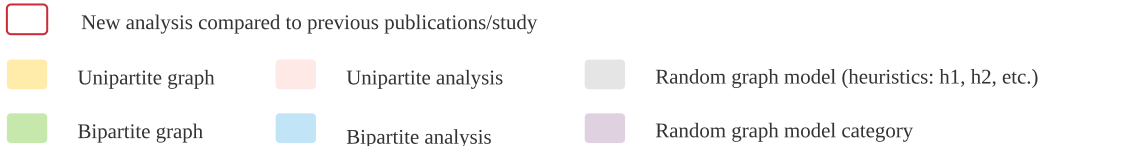
The purpose of this study is to tackle the realistic random graph model problem, by evaluating the relevance of the tripartite model as a possible generalized answer to this problem.

The main questions to be addressed in our work are the following: for which features the tripartite model performs better or worse results than the classical bipartite configuration model, i.e. results closer to the ones calculated with the real-world network? To what extent does this model preserve bicliques of the original bipartite graph? Does it preserve other characteristics such as bipartite clustering coefficient, redundancy, density? Eventually, in the unipartite projection level, are properties - such as density or assortativity - preserved? If not, what this model lacks to preserve those characteristics? How

can we improve it ? Finally, can the tripartite model be (or be extended to) a unified random bipartite graph model that can produce a realistic graph ?



**Figure 3.14:** Models comparison.



**Figure 3.15:** Models comparaison.

-

# Chapter 4

## Implementation

### Contents

<b>4.1</b>	<b>Tools used and organization of the program</b>	<b>29</b>
<b>4.2</b>	<b>Some implementation elements</b>	<b>30</b>
4.2.1	Graph modelisation	30
4.2.2	Finding maximal bicliques on bipartite graphs	30
4.2.3	Returning a set of non-overlapping maximal bicliques	35
4.2.4	Tripartite encoding	35
4.2.5	Paths calculation	35
4.2.6	How to use the code	35

### 4.1 Tools used and organization of the program

We choose to use Python 3.4 to implement our program. If fact, since our main problem is to evaluate the *relevance* of the tripartite model ("prototyping"), performance is not a critical requirement at this stage of the model understanding. Using Python allows us to speed up the software development process and benefit from a robust standard library. Even if lower level languages such as C++ or C# offer higher performance, Python provides a decent performance for our case of study: especially, it brings packages for high-performance mathematical operations (Numpy) and various librairies for data analysis and data visualisation (pandas, seaborn, matplotlib, etc.) that we will useful for the models comparaisn part. For performance-critical code, it is still possible to call C/C++ code from a Python script.

Once the project will be on its way to become an analytical tool or application, i.e. we would like to provide a tripartite model generator for external users or to do speed optimisation, it can be ported to more sophisticated performance-sensitive language such as C++.

In order to keep a good understanding of measured properties and handling of our data structures, we didn't use any Python package for creation, manipulation and study of graphs and complex networks. All the functions have been made from scratch using the basics data structures provided by Python and the following list of packages:

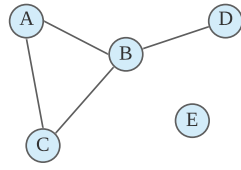
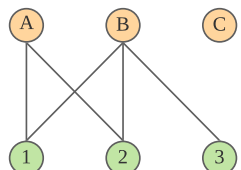
- Numpy;
- Basics data science librairies: pandas, matplotlib, seaborn.

We use Jupyter Notebook for the data analysis part and Git for code version control.

## 4.2 Some implementation elements

### 4.2.1 Graph modelisation

There is 2 ways to modelise a graph: by using an adjacency matrix or adjacency lists (Fig.4.1).

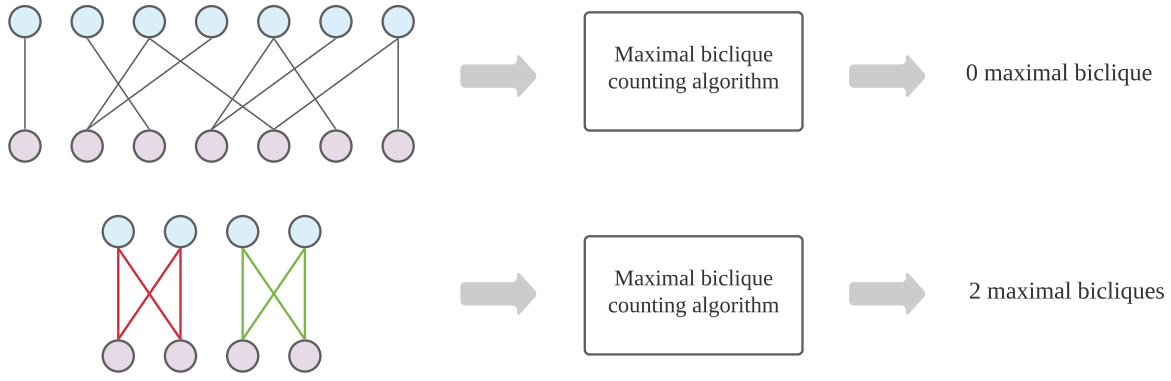
	Example	Adjacency matrix	Adjacency list																																				
Unipartite graph		<table><tr><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td></tr><tr><td>A</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>B</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>C</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>D</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>E</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>		A	B	C	D	E	A	0	1	1	0	0	B	1	0	1	1	0	C	1	1	0	0	0	D	0	1	0	0	0	E	0	0	0	0	0	A: {C,B} B: {A,C,D} C: {A,B} D:{B} E: {}
	A	B	C	D	E																																		
A	0	1	1	0	0																																		
B	1	0	1	1	0																																		
C	1	1	0	0	0																																		
D	0	1	0	0	0																																		
E	0	0	0	0	0																																		
Bipartite graph		<table><tr><td></td><td>A</td><td>B</td><td>C</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>2</td><td>1</td><td>1</td><td>0</td></tr><tr><td>3</td><td>0</td><td>1</td><td>0</td></tr></table>		A	B	C	1	1	1	0	2	1	1	0	3	0	1	0	<u>top</u> A: {1,2} B: {1,2,3} C: {} <u>bottom</u> 1: {A,B} 2: {A,B} 3: {B}																				
	A	B	C																																				
1	1	1	0																																				
2	1	1	0																																				
3	0	1	0																																				

**Figure 4.1:** Models comparaison.

We choose to represent our graph using adjacency list.

### 4.2.2 Finding maximal bicliques on bipartite graphs

A preliminary step before implementing the tripartite model is finding all maximal bicliques of bipartite graphs. This problem is a variation of the MGBP (Maximal Biclique Generation Problem) NP-hard problem, which consists in generating all the maximal bicliques of a given simple graph. This problem is particularly challenging in terms of execution time because it cannot be solved in polynomial time with respect to the graph size only [10]. Informally, a graph can be small but contains more bicliques than a bigger graph containing 0 biclique. Thus, today, it only exists output sensitive algorithms so solve this problem: the running time depends on the size of the output, instead of / in addition to the size of the output.



**Figure 4.2:** Output sensitive algorithm . The first graph contains more vertices and edges than the second one, but maximal biclique enumeration may take more time for the second graph. Indeed, it contains 2 maximal bicliques while the first one doesn't contain any biclique.

In our case of study, we will focus on a practical aspect of this problem. For a first implementation, we will study the FIND-ALL-MAXIMAL algorithm purposed by Enver Kayaaslan, which aims to enumerate all maximal bicliques in bipartite graphs. This algorithm is a specified version for bipartite graph of the work done by Alexe et. al. [2], where the authors tackle the MGBP problem. E. Kayaaslan provides a practical algorithm that runs in polynomial time in respect to both input (i.e. graph) and output (maximal bicliques) size with a naive implementation.

In the following, we will go through main notions and theorems defined in the article. For each of them, we give an informal explanation and example.

- **Consensus set:**

#### Consensus set

Given a bipartite graph  $G = (X, Y, E)$ , for a subset  $S_x \subseteq X$  ( $S_y \subseteq Y$ ), the consensus set  $P_y(S_x)$  ( $P_x(S_y)$ ) is defined as the intersection of neighbor set of each vertex  $x_i \in S_x$  ( $y_j \in S_y$ ). [13] Formally:

$$P_y(S_x) = \bigcap_{x_i \in S_x} N_y(x_i)$$

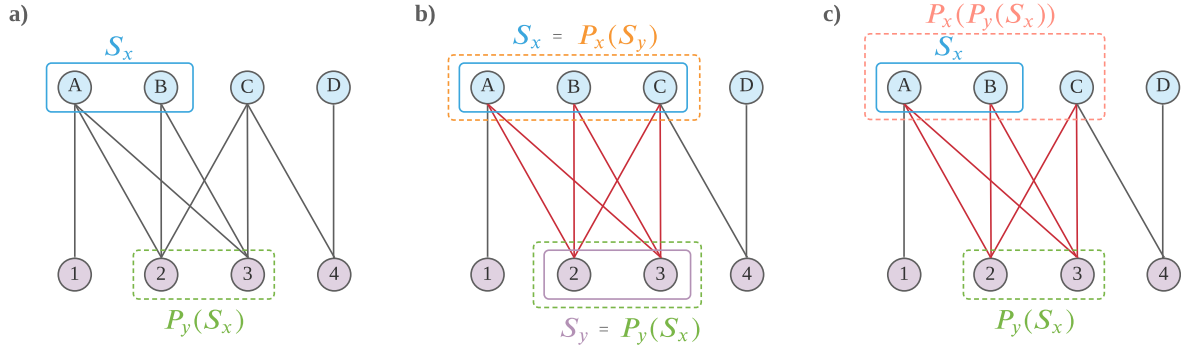
One can also note that  $(S_x, P_y(S_x))$  is a biclique. Similarly,  $(S_y, P_x(S_y))$  is also a biclique.

#### Theorem 1

$$(S_x, S_y) \neq \emptyset \text{ is a maximal biclique} \Leftrightarrow S_y = P_y(S_x) \text{ and } S_x = P_x(S_y)$$

#### Theorem 2

For any  $S_y \subseteq Y$  such that  $P_x(S_y) \neq \emptyset$ ,  $(P_y(P_x(S_y)), P_x(S_y))$  is a maximal biclique. Similarly, for any  $S_x \subseteq X$  such that  $P_y(S_x) \neq \emptyset$ ,  $(P_x(P_y(S_x)), P_y(S_x))$  is a maximal biclique.



**Figure 4.3:** Examples to illustrate notions described in E. Kayaaslan article [13]. **a)** The consensus set  $P_y(S_x)$  of  $S_x = A, B$  is the intersection of neighbor set of each vertex in  $S_x$ .  $N(A) = 1, 2, 3$  and  $S_x$ .  $N(B) = 2, 3$ , hence  $P_y(S_x) = N(A) \cap N(B) = 2, 3$ . **b)** The subgraph  $(S_x, S_y)$  with  $S_x = A, B, C$  and  $S_y = 2, 3$  is a maximal biclique because the consensus set  $P_y(S_x)$  of  $S_x$  is equal to  $S_y$ , and the consensus set  $P_x(S_y)$  of  $S_y$  is equal to  $S_x$ . **c)** Given the set  $S_x = A, B$ , we find a maximal biclique such that  $S_x$  is a subset of its top vertices. This maximal biclique is defined by  $(P_x(P_y(S_x)), P_y(S_x))$ , where  $P_x(P_y(S_x))$  is the consensus set of  $P_y(S_x)$ .

Theoretical results suggests than it is sufficient to enumerate on the consensus sets in order to find all maximal bicliques [13]. In our study, maximal bicliques  $(S_x, S_y)$  with singletons  $S_x$  or  $S_y$  are ignored, therefore the FIND-ALL-MAXIMAL algorithm must be slightly re-adapted to fit our application.



---

**Algorithm 1:** FIND-ALL-MAXIMAL-2: Re-adapted version of the algorithm with exclusion of 1-top and 1-bottom bicliques.

---

```

1 Input: a bipartite graph  $G = (X, Y, E)$ 
2 Initialisation: ;
   // Feed a list  $S$  and a queue  $Q$  with neighbor sets of each vertex in
   //  $Y$  and exclude all 1-top bicliques and empty neighborhood
3  $S \leftarrow \{N(y_i) : y_i \in Y : |N(y_i)| > 1\}$  ;
4  $Q \leftarrow S$ ;
5 while  $Q \neq \emptyset$  do
6    $S_x \leftarrow \text{DEQUEUE}(Q)$  ;
7   foreach  $y_j \in Y/P_y(S_x)$  do
8      $S_{new} \leftarrow S_x \cap N(y_j)$  ;
     // If  $S_{new}$  is not in  $S$  and 1-top bicliques are excluded
9     if  $S_{new} \notin S$  and  $|S_{new}| > 1$  then
10       $S \leftarrow S \cup \{S_{new}\}$  ;
11       $\text{ENQUEUE}(Q, S_{new})$ 
12    end
13  end
14 end

   // Return a list of maximal bicliques excluding 1-bottom bicliques
15  $C_{max} = \{(S_x, P_y(S_x)) : S_x \in S : |P_y(S_x)| > 1\}$  ;
16 return  $C_{max}$ 

```

---

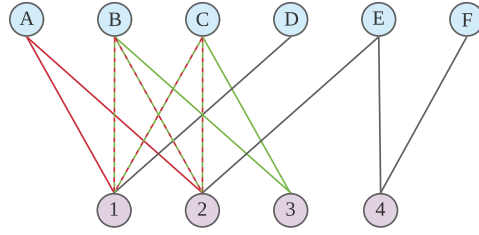
The FIND-ALL-MAXIMAL-2 procedure excludes all 1-top and 1-bottom maximal bicliques. The algorithm takes a bipartite graph  $G = (X, Y, E)$  as an input. First, a list  $S$  and a FIFO queue  $Q$  are initialized with neighbor sets of every vertex  $y_j \in Y$  that have more than one neighbor, i.e.  $|N(y_j)| > 1$ . With this condition, we ensure that  $S$  and  $Q$  are not initialized with 1-top bicliques. A neighbor set  $N(y_j)$  is in fact the *consensus set* of the singleton  $S_y = y_j \subseteq Y$ : we have  $P_x(Y_j) = N(y_j)$ .

In the main loop,  $Q$  holds all the consensus sets that haven't been checked yet. while  $S$  gradually memorizes all the consensus sets (checked and unchecked ones).  $S$  expands as far as new consensus sets are found while iterating in the main loop.

At each iteration, we consider a  $S_x \in Q$  that hasn't been checked yet by simply dequeuing  $Q$ . For each vertex  $y_j \in Y$  not in the consensus set of  $S_x$ , a new  $S_{new}$  is built as the intersection between  $S_x$  and the neighbourhood of  $y_j$ . We don't consider vertices in  $S_x$  because its consensus set will result again as  $S_x$ , whereas we would like to find new ones. If  $S_{new}$  is not in  $S$  (it hasn't been identified previously) and is not a singleton (we ensure that 1-top bicliques are not added to  $S$ ),  $S_{new}$  is added to  $S$  and enqueued to  $Q$ .

The loop ended while the queue is empty: there is no more consensus sets to generate new ones. Finally, in order to exclude all 1-bottom bicliques, we return all the maximal bicliques of  $G$  by taking pairs  $(S_x, P_y(S_x))$  for each subset  $S_x \in S$  that has a non-singleton consensus set, i.e.  $|P_y(S_x)| > 1$ .

An implementation in Python can be found in the appendix.



**Figure 4.4:** A bipartite graph with 2 overlapping maximal bicliques:  $b_1 = (\{A, B, C\}, \{1, 2\})$  in red and  $b_2 = (\{B, C\}, \{1, 2, 3\})$  (in green).

	Iteration 0	Iteration1	Iteration 2	Iteration 3	Iteration 4
$S_x$	{A,C,D,B}	{A,C,B,E}	{C,B}	{F,E}	{A,C,B}
$P_y(S_x)$	{1}	{2}	{1,2,3}	{4}	{1,2}
$y_j \in Y/P_y(S_x)$ :	{2,3,4}	{1,3,4}	{4}	{1,2,3}	{3,4}
Loop over $y_j \in Y/P_y(S_x)$	$y_j$ : 2 $N(y_j)$ : {A,C,B,E} $S_{new}$ : {A,C,B} ->add to S and Q	$y_j$ : 1 $N(y_j)$ : {A,C,D,B} $S_{new}$ : {A,C,B} ->pass		$y_j$ : 1 $N(y_j)$ : {A,C,D,B} $S_{new}$ : $\emptyset$ ->pass	
	$y_j$ : 3 $N(y_j)$ : {C,B} $S_{new}$ : {C,B} ->pass	$y_j$ : 3 $N(y_j)$ : {C,B} $S_{new}$ : {C,B} ->pass	$y_j$ : 4 $N(y_j)$ : {F,E} $S_{new}$ : $\emptyset$ ->pass	$y_j$ : 2 $N(y_j)$ : {A,C,B,E} $S_{new}$ : {E} ->pass	$y_j$ : 3 $N(y_j)$ : {C,B} $S_{new}$ : {C,B} ->pass
	$y_j$ : 4 $N(y_j)$ : {F,E} $S_{new}$ : $\emptyset$ ->pass	$y_j$ : 4 $N(y_j)$ : {F,E} $S_{new}$ : {E} ->pass		$y_j$ : 3 $N(y_j)$ : {C,B} $S_{new}$ : $\emptyset$ ->pass	$y_j$ : 4 $N(y_j)$ : {F,E} $S_{new}$ : $\emptyset$ ->pass

#### Values of $Q$ and $S$ :

- Initialisation:  $Q = (\{A, C, D, B\}, \{A, C, B, E\}, \{C, B\}, \{F, E\})$   
 $S = \{\{A, C, D, B\}, \{A, C, B, E\}, \{C, B\}, \{F, E\}\}$
- At the end of iteration 0:  $Q = (\{A, C, D, B\}, \{A, C, B, E\}, \{C, B\}, \{F, E\}, \{A, C, B\})$   
 $S = \{A, C, D, B\}, \{A, C, B, E\}, \{C, B\}, \{F, E\}, \{A, C, B\}$
- At the end of iteration 1:  $Q = (\{C, B\}, \{F, E\}, \{A, C, B\})$
- At the end of iteration 2:  $Q = (\{F, E\}, \{A, C, B\})$
- At the end of iteration 3:  $Q = (\{A, C, B\})$
- At the end of iteration 4:  $Q = \emptyset$   
 $S = \{A, C, D, B\}, \{A, C, B, E\}, \{C, B\}, \{F, E\}, \{A, C, B\}$

**Returns:**  $\{(\{A, B, C\}, \{1, 2\}), (\{B, C\}, \{1, 2, 3\})\}$

**Table 4.1:** Execution trace of FIND-ALL-MAXIMAL-2 for the graph given in Fig.4.4. The algorithm returns only the 2 bicliques identified in Fig.4.4. 1-top and 1-bottom bicliques have been well ignored.

### 4.2.3 Returning a set of non-overlapping maximal bicliques

#### 4.2.4 Tripartite encoding

Let's say we have enumerated the bicliques of the graph  $G$  of Fig. 3.2 using the FIND-ALL-MAXIMAL-2 algorithm. After a procedure returning a list of non-overlapping maximal bicliques, we will consider those following bicliques:

- $m_\alpha = (\{A, B, C\}, \{1, 2\})$  in green,
- $m_\beta = (\{C, D, E\}, \{3, 4, 5\})$  in red,
- $m_\gamma = (\{F, G\}, \{5, 6\})$  in yellow.

#### 4.2.5 Paths calculation

Useful for: things related to connected components, distance, path length

**Breadth First Search**

**Depth First Search**

#### 4.2.6 How to use the code

**Some improvements ideas**

**Damaschke**

$$\sigma(A) = \cap_{v \in A} N(v)$$

$$\phi(A) = \sigma(\sigma(A))$$

$$\phi(\phi(A)) = \phi(A)$$

# Chapter 5

## Experiments, analysis and results

### Contents

<b>5.1</b>	<b>Datasets used . . . . .</b>	<b>36</b>
<b>5.2</b>	<b>Results . . . . .</b>	<b>36</b>
5.2.1	Results for bipartite graphs . . . . .	39

### 5.1 Datasets used

dataset	nb top vertices	nb bottom vertices	nb edges	nature
hepB	24663	49214	121363	scientific collaborations on hepatitis B
arxiv	38741	16726	58595	co-authorship
bpse	1	6	3	proteins of Burkholderia pseudomallei bacteria involve in metabolic pathway

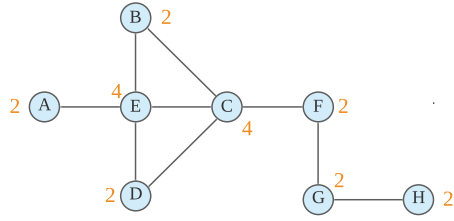
**Table 5.1:** Some datasets used for experimentation.

### 5.2 Results

#### Appendix - how to read plot of this section

We will start this section with a reminder of the main statistical notions used in the following. The aim is not to give a deep mathematical description of all the concepts used there, but rather hints and sufficient intuition to interpret and understand their relevance in our work.

Let's consider the following graph.



**Figure 5.1:** An unipartite graph with degree for every vertex (orange)

degree	distribution	CDF	ICDF
1	1	1	8
2	4	5	7
3	1	6	3
4	2	8	2

**Table 5.2:** Degree distribution, Cumulative degree distribution function (CDF) and Inverse cumulative degree distribution (ICDF) of the graph in Fig. 5.1

We define 3 main notions:

- **Distribution:** a distribution function  $f$  is a function that shows the possible values for a variable  $X$  and how often they occur. In our example, we are looking at the degree of each vertex of the graph ( $X$ ). For each degree  $x$ , we count how many vertices have this degree. For example, the table tells us that 4 vertices are degree 4.
- **Cumulative distribution function (CDF):** the cumulative distribution function  $F_X$  is the probability that a variable  $X$ , evaluated at  $x$ , will take a value less than or equal to  $x$ . Formally:

$$F_X(x) = P(X \leq x)$$

where the right-hand side represents the probability that the random variable  $X$  takes on a value less than or equal to  $x$ . In 5.3, CDF of a degree  $x$  is calculated by summing all the degrees  $\leq x$ . For instance, for  $x = 3$ ,  $F_X(3) = f(1) + f(2) + f(3) = 1 + 4 + 1 = 6$ .

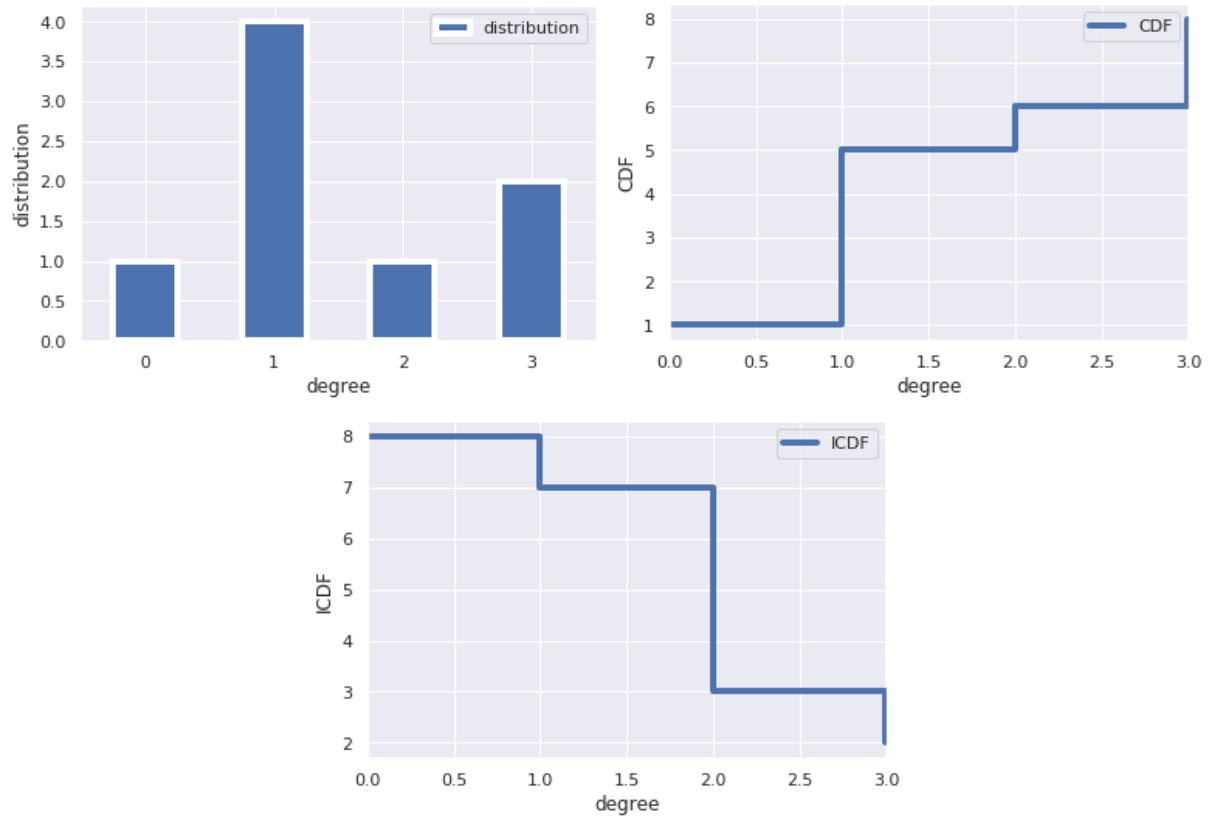
- **Inverse Cumulative distribution function (ICDF):** the inverse cumulative distribution function  $F_X^{-1}$  is the probability that a variable  $X$ , evaluated at  $x$ , will take a value greater than or equal to  $x$ . Formally:

$$F_X^{-1}(x) = P(X \geq x)$$

where the right-hand side represents the probability that the random variable  $X$  takes on a value greater than or equal to  $x$ . In 5.3, ICDF of a degree  $x$  is calculated by summing all the degrees  $\geq x$ . For instance, for  $x = 3$ ,  $F_X(3) = f(3) + f(4) = 1 + 2 = 3$ .

In some cases, it is more relevant to consider cumulative distributions (CDF and/or ICDF) rather than classical distribution. In fact, instead of plotting the fraction of samples that have *exactly*  $x$  as the measured value, cumulative distributions plot for each  $x$  the *fraction* of samples having a measured value lower than or equal to  $x$  for CDF, and greater than or equal to  $x$  for ICDF.

This approach is particularly interesting when  $x$  can take real values and not only integers. In this case, it is more relevant to estimate the number of samples that have an  $X \geq x$  or  $X \leq x$  than knowing how many samples have its  $X$  exactly equal to a specific real value. Moreover, it is then easier to estimate the number of samples with a low or high measured value.[14]



**Figure 5.2:** Distribution function (top left), CDF (top right) and ICDF (bottom) for degrees of the graph in Fig.5.1. For  $x = 1$ , the figures can be interpreted as: - Distribution: the graph has 4 vertices with degree  $x = 1$ ; - CDF: 5 vertices have degree less or equal to 1; - ICDF: 7 vertices have degree greater or equal to 1.

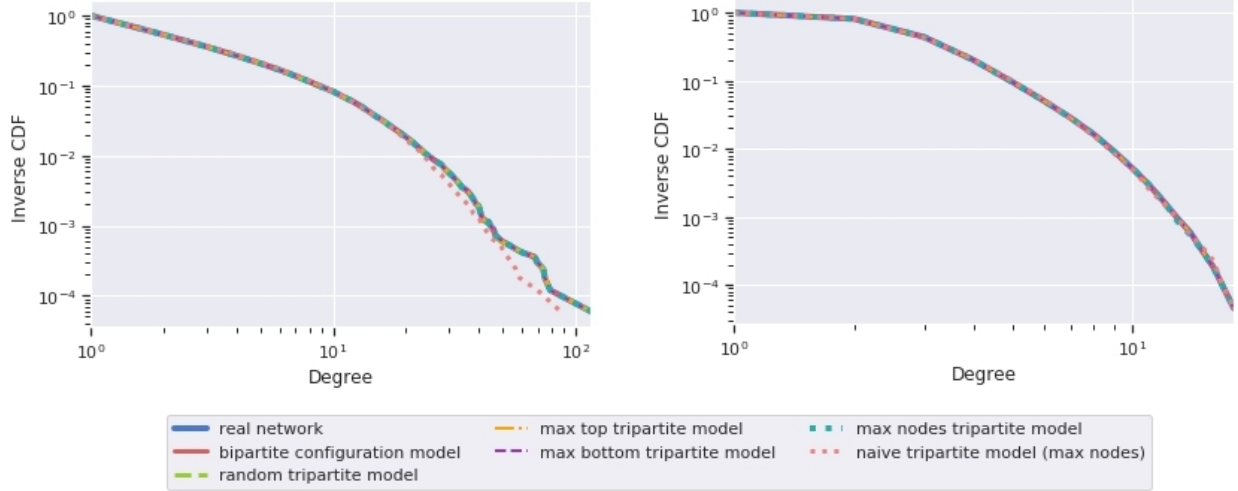
### 5.2.1 Results for bipartite graphs

Since we are not doing network analysis but random graph model analysis and comparison, in our plots we are more interested in understanding how the curves behave than having *exact* values is y-axis given an  $x$ . Therefore, we have normalized CDF and ICDF values to 0-1 range by simply dividing all the values by the max value. Normalizing the data to range 0-1 will simplify the percentage distribution study and comparison between results obtained with different datasets.

Note: When a log-log scale is used, since  $\log(0)$  is not defined, the minimum x-axis has been set to a positive number above 0 ( $10^5$ ).

#### Degrees

Dataset "arxiv"



**Figure 5.3:** Inverse cumulative distribution of degrees. Left: top nodes. Right: bottom nodes.

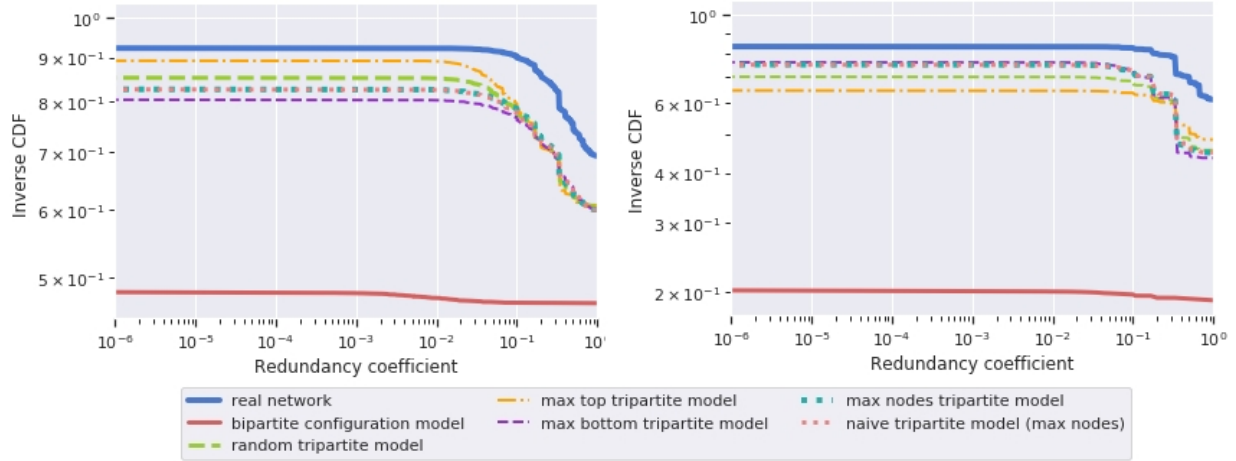
First, let's plot the inverse cumulative function of degree distribution (Fig. ??). In particular, one may want to detect *small* variation in high degree distribution. In fact, since real-networks usually have a skewed degree distribution, we want to ensure that after any randomization process, we still have few vertices with high degree distribution at the tail: those nodes are very important for network analysis. That is the reason why we choose to plot ICDF in a log-log instead of CDF: since ICDF decreases, the more we move to the right of the plot, the more small variations on y-axis are noticeable as curve slope variation is sensitive to small variation due to logarithmic scale. Thus, plotting ICDF in a log-log scale does exactly what we want in this case: "compress"/minimize variations of small degrees and detect small distribution variation of higher degrees. On the contrary, if one is interested in studying small variations of low degrees, he should plot the information with CDF on a log-log scale.

#### Observation and interpretation:

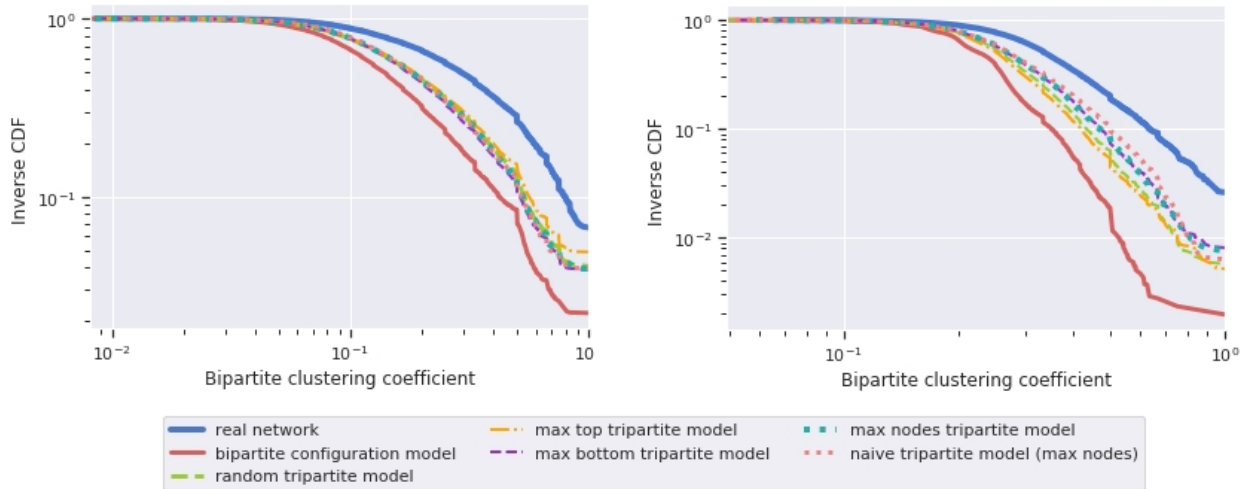
- As expected from a real-world network, the degree distribution is very skewed for both top and bottom nodes. In fact, as  $F^{-1}(10) \approx 10^{-1} \approx 0.1 \approx 10\%$  for top nodes of the real network, it means that 10% of the vertices have degree  $\geq 10$ . Hence, it means that approximately 90% ( $1 - 0.1 = 0.9$ ) of vertices have degree lying in the open interval  $[1, 10)$ .

We have also computed a random graph using a naive approach of the tripartite model. In this approach, randomization process is differs since bicliques edges are not labeled; in fact, we simple apply 3 bipartite configuration model: one for the graph induced by edges  $E_{WX}$ , one for the one induced by  $E_{XY}$  and one induced by  $E_{WY}$ . A more detail explanation of this approach is provided in the appendix. In Fig. 5.3 clearly appears that this naive model results in a worse degree estimation than the tripartite

model with any heuristic. This observation suggests that the "selective" edge matching required by the tripartite model (two stubs from different sets can be randomly connected only if they stem from edges that have the same label in  $G_{tri}$ ) is necessary to preserve the degree distribution of the original network.



**Figure 5.4:** Inverse cumulative distribution of redundancy. Left: top nodes. Right: bottom nodes.



**Figure 5.5:** Inverse cumulative distribution of redundancy. Left: top nodes. Right: bottom nodes.



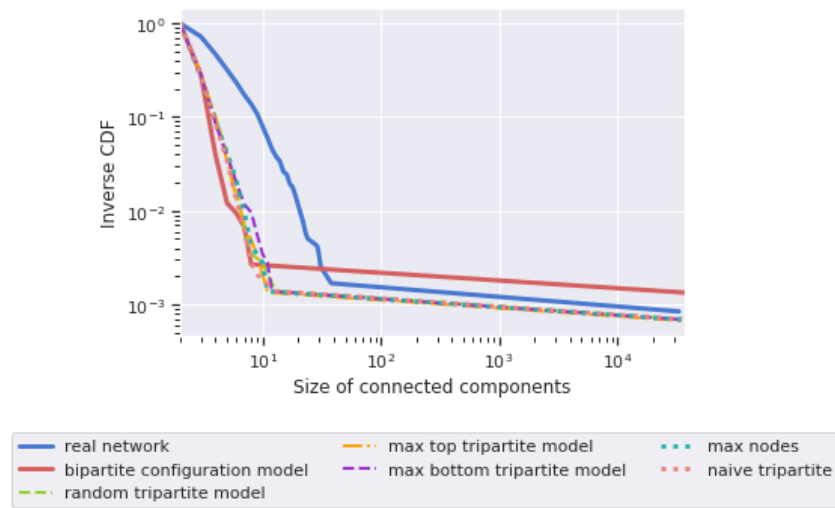


Figure 5.6

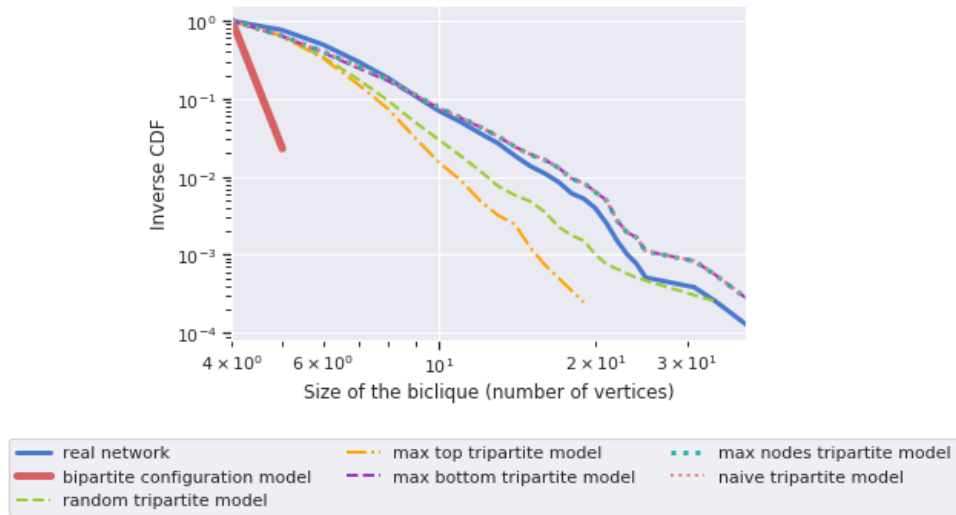


Figure 5.7

	real	configuration model	random	maxtop	maxbottom	maxnodes	naive_tri
nb vertices	22015	22015	22015	22015	22015	22015	22015
nb edges	226873	277863	256460	262314	244802	244597	221955
density	0.000936	0.001147	0.001058	0.001083	0.001010	0.001009	0.000916
clustering coeff	0.804555	0.634117	0.708705	0.695835	0.722123	0.721821	0.720963
nb connected components	1188	747	1437	1490	1454	1450	1441
diameter	17	9	10	11	10	10	11
assortativity	0.706990	0.395438	0.450726	0.416876	0.483616	0.492045	0.380354
path length	5.959748	3.724516	4.002997	3.995446	4.066474	4.099031	4.181454
degree avg	20.610765	25.243062	23.298660	23.830480	22.239564	22.220940	20.163980
degree sum	453746	555726	512920	524628	489604	489194	443910
degree min	0	0	0	0	0	0	0
degree max	176	286	224	239	216	263	176

**Table 5.3**

## Chapter 6

# Conclusion and perspective

# Bibliography

- [1] Computational analysis of functional connectivity between areas of primate cerebral cortex. *Philosophical Transactions: Biological Sciences*, 355(1393):111–126, 2000.
- [2] Gabriela Alexe, Sorin Alexe, Yves Crama, Stephan Foldes, Peter Hammer, and Bruno Simeone. Consensus algorithms for the generation of all maximal bicliques. 09 2003.
- [3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
- [4] Martin Bouchard and Aili Malm. Social network analysis and its contribution to research on crime and criminal justice. 11 2016.
- [5] Damon Centola. *How Behavior Spreads: The Science of Complex Contagions*. Princeton University Press, 2018.
- [6] Émilie Coupechoux and Fabien Tarissan. Un modèle pour les graphes bipartis aléatoires avec redondance. In *4ème Journées Modèles et l'Analyse des Réseaux : Approches Mathématiques et Informatique (MARAMI'13)*, Saint-Etienne, France, October 2013.
- [7] Luca Dall'Asta. *Dynamic Phenomena on Complex Networks*. Theses, Université Paris Sud - Paris XI, September 2006.
- [8] Peter Damaschke. Enumerating maximal bicliques in bipartite graphs with favorable degree sequences. *Information Processing Letters*, 114:317–321, June 2014.
- [9] Jennifer Dunne, Richard Williams, and Neo Martinez. Network structure and biodiversity loss in food webs: Robustness increases with connectance. *Ecology Letters*, 5:558 – 567, 07 2002.
- [10] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Inf. Process. Lett.*, 51:207–211, 1994.
- [11] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. SIGCOMM '99, page 251–262, New York, NY, USA, 1999. Association for Computing Machinery.
- [12] Hawoong Jeong, S.P. Mason, Albert-Laszlo Barabasi, and Zoltan Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–2, 06 2001.
- [13] Enver Kayaaslan. On enumerating all maximal bicliques of bipartite graphs. pages 105–108, 01 2010.
- [14] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic Notions for the Analysis of Large Two-mode Networks. volume 30, pages 31–48. Elsevier, January 2008.
- [15] Matthieu Latapy and Tiphaine Viard. Complex networks and link streams for the empirical analysis of large software. In Gianfranco Ciardo and Ekkart Kindler, editors, *Application and Theory of Petri Nets and Concurrency*, pages 40–50, Cham, 2014. Springer International Publishing.
- [16] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64:026118, Jul 2001.

- [17] Fabien Tarissan and Lionel Tabourier. A random model that relies on maximal bicliques to preserve the overlaps in bipartite networks. In *8th International Conference on Complex Networks and their Applications*, Lisbonne, Portugal, December 2019.
- [18] Duncan J. Watts and Steven H. Strogatz. *Collective dynamics of 'small-world' networks*, volume 393. Nature Publishing Group, June 1998.

# List of Figures

2.1	Examples of directed and weighted graphs. <b>a)</b> An undirected and unweighted graph. <b>b)</b> A directed and unweighted graph. <b>c)</b> An undirected and weighted graph. <b>d)</b> A directed and weighted graph. . . . .	10
2.2	. . . . .	10
2.3	Example of a $\top$ -projection and $\perp$ -projection of a bipartite graph $G$ . In the $\top$ -projection, edge $(A, B)$ exists because $A$ and $B$ have <i>at least</i> one neighbor in common in $G$ , which are vertices 2 and 3. On the contrary, $A$ and $E$ doesn't share any neighbors in $G$ , so they are not connected in $G_{\top}$ . Dually, in the $\perp$ -projection, 2 and 3 are linked because they share $A$ and $B$ as mutual neighbors in $G$ . Since nodes 1 and 6 are not connected in $G$ , there isn't any edge $(A, E)$ in $G_{\perp}$ . . . . .	11
2.4	Examples of bipartite graphs that have the same $\top$ -projection as Fig. 2.3. . . . .	12
2.5	This graph contains 3 connected components. The first one, drawn in purple, is the subgraph induced by $\{A, B, C, D\}$ . The second one drawn in blue is the subgraph induced by $\{E, F, G, H, I\}$ . The last connected component is drawn in orange and is induced by vertices $\{J, K\}$ . . . . .	13
2.6	Degree $\delta$ of each vertex of the $\top$ -projection in Fig.2.3. . . . .	14
2.7	Clustering coefficient calculations for $\top$ -projection in Fig. 2.3. In the figure, we explicit the calculation for vertex B: its neighborhood $\delta$ is composed of vertices A,C,D,E (in yellow) and 3 edges (in orange) exist between nodes of $\delta$ , so $E_{\delta} = 3$ . So we have $CC(B) = \frac{2E_{\delta}}{\delta(\delta-1)} = \frac{2 \times 3}{4 \times 3} = \frac{6}{12} = \frac{1}{2}$ . And since $CC(A) = 1$ , $CC(C) = 1$ $CC(D) = \frac{2}{3}$ $CC(E) = \frac{2}{3}$ , the global clustering coefficient of the graph is $CC(G) = \frac{(1+\frac{1}{2}+1+\frac{2}{3}+\frac{2}{3})}{5} \approx 0,76$ . . . . .	15
2.8	test . . . . .	16
2.9	test . . . . .	16
2.10	test . . . . .	17
2.11	A graph containing 3 bicliques: $b_1 = (\{A, B, C\}, \{1, 2\})$ with edges in green, $b_2 = (\{D, E\}, \{4, 5, 6\})$ in red and $b_3 = (\{D, E, F\}, \{5, 6\})$ in yellow. $(\{A, B\}, \{1, 2\})$ is a biclique but not a maximal biclique since $(\{A, B, C\}, \{1, 2\})$ is also a biclique. $(\{C\}, \{3, 4\})$ is not considered as biclique in our case of study. We note that $b_1$ and $b_2$ are overlapping bicliques because they share the 4 dotted edges. . . . .	17
3.1	Bipartite version of the configuration model. <b>a)</b> A graph with 2 maximal non-overlapping bicliques, in red and green. Each vertex is labeled with its degree $d$ . <b>b)</b> Same graph with visible stubs. As for the unipartite version of the configuration model, we will consider half-edges. The endpoint of a stub that is not linked to a node is represented as either a bracket (top stubs) or a rectangle (bottom stubs). A bracket and a rectangle connected together means that the two stubs they come from form an edge. <b>c)</b> Edges of the graph are "cut". <b>d)</b> Stubs rematching process. A top (respectively bottom) stub is randomly rematched to any bottom (respectively top) stub. <b>e)</b> The initial graph after the randomization process. The initial degrees has been preserved, but the two bicliques in <i>a)</i> completely vanished. . . . .	19

3.2	A bipartite graph $G_{bip} = (X, Y, E)$ with $X = \{A, B, C, D, E, F, G\}$ and $Y = \{1, 2, 3, 4, 5, 6\}$ . $G_{bip}$ contains 3 non-overlapping maximal bicliques: $m_\alpha = (\{A, B, C\}, \{1, 2\})$ in green, $m_\beta = (\{C, D, E\}, \{3, 4, 5\})$ in red, and $m_\gamma = (\{F, G\}, \{5, 6\})$ in yellow. These 3 bicliques form the set $M$ . The set of edges $E_{\overline{M}} = \{(A, 3), (E, 6)\}$ , drawn with dotted lines, define edges that are not involved in any biclique in $M$ . . . . .	20
3.3	Maximal biclique encoding in a third level. The figure on the left is the subgraph induced by the maximal biclique $m_\alpha = (X_\alpha, Y_\alpha) \in M$ of $G_{bip}$ . In the tripartite graph on the right, a vertex $\alpha$ is created in an upper level $W$ to encode the biclique. The vertex $\alpha$ gathers all the information about $m_\alpha$ , because it is connected to all elements of $X_\alpha$ and $Y_\alpha$ . Since $ Y_\alpha  = 2$ , the edges $(\alpha, A)$ , $(\alpha, B)$ and $(\alpha, C)$ are labeled " $\downarrow 2$ ". Therefore, the label of edge $(\alpha, A)$ means "Vertex A is involved in a maximal biclique $(X_n, Y_n)$ such as, in the bottom level of $G_{bip}$ , $ Y_n  = 2$ ." Dually, since $ X_\alpha  = 3$ , the edges $(\alpha, 1)$ and $(\alpha, 2)$ are labeled " $\uparrow 3$ ". Therefore, the label of edge $(\alpha, 1)$ means "Vertex 1 is involved in a maximal biclique $(X_n, Y_n)$ such as, in the upper level of $G_{bip}$ , $ X_n  = 3$ ." We say that the vertex $\alpha$ captures the maximal biclique $m_\alpha$ . . . . .	20
3.4	Tripartite graph $G_{tri}$ encoding the maximal bicliques of $G_{bip}$ . Each vertex of set $W$ encodes a maximal biclique of $G_{bip}$ . Edges in $E_{XY}$ are the dotted edges in $G_{bip}$ . Their label "0" is a arbitrary value meaning that the edges aren't encoding any biclique. . . . .	21
3.5	The same tripartite graph $G_{tri}$ but with visible stubs. Each label is represented with a unique pair of interlockable symbols. . . . .	21
3.6	Types of stubs and their corresponding symbol. For instance, a half-edge of type " $(WX, \downarrow 2)$ " denotes a stub that stems from an edge of $\{(w, x)   w \in W, x \in X\}$ and with one endpoint linked to a vertex of $S_u$ , while the other one is left floating. . . . .	21
3.7	Interlockable and some non-interlockable symbols of Fig.3.5. . . . .	22
3.8	Edges of the tripartite graph are "cut" and ready to be rematched. . . . .	22
3.9	Stubs rematching process. For each stub, we pick another stub according to Fig.3.7. Stubs matched together may either: stem from the same biclique (e.g. edge $(\alpha, A)$ ); stem from 2 different biclique (e.g. edge $(\alpha, F)$ ); stems from 0 biclique for both of them (only possible case for $\{(x, y)   x \in X \wedge y \in W \wedge (x, y) \in E_{XY}\}$ edges). . . . .	23
3.10	Tripartite graph $G_{tri}' = (W, X, Y, E_{WX}', E_{WY}', E_{XY}')$ obtained after performing randomization process on $G_{tri}$ . It is the same figure as Fig.3.9 but with the corresponding labels instead of symbols. . . . .	23
3.11	Graph $G_{bip}'$ obtained after bipartite projection of $G_{tri}'$ . The degree distribution of $G_{bip}$ and $G_{bip}'$ are equal. We notice three non-overlapping bicliques: one preserved by $\alpha$ in $G_{tri}'$ (in green), one by $\beta$ (in red) and one by $\gamma$ (in yellow). Since these bicliques have exactly the same size as the ones detected in the initial graph $G$ , the tripartite model produces a graph preserving both the degree distribution and the maximal bicliques. . . . .	24
3.12	Two possible multigraphs obtained after projection of alternative randomized tripartite graphs. <b>a)</b> Multiple edges between vertices A and 3. One edge were involved in biclique encoding, the other was not. <b>b)</b> Multiple edges between C and 5. Both of these edges were involved in biclique encoding . . . . .	25
3.13	Patterns in tripartite graphs that form multiple edges in bipartite projection. <b>a)</b> The triangle $\{\alpha, A, 3\}$ is projected into 2 multiples edges between vertices A and 3. Informally, $A, 3 \subset N(\alpha)$ means that there is an edge $(A, 3)$ in the bipartite projection, but there is already an edge $(A, 3)$ in $G_{tri}'$ . <b>a)</b> The cycle $(\beta, C, \gamma, 5, \beta)$ is projected into 2 multiples edges between vertices C and 5. Informally, $C, 5 \subset (\beta)$ means that there is an edge $(C, 5)$ in the bipartite projection, and similarly $C, 5 \subset N(\gamma)$ also means that there is an edge $(C, 5)$ . . . . .	25
3.14	Models comparaison. . . . .	26
3.15	Models comparaison. . . . .	27
4.1	Models comparaison. . . . .	30

4.2	Output sensitive algorithm . The first graph contains more vertices and edges than the second one, but maximal biclique enumeration may take more time for the second graph. Indeed, it contains 2 maximal bicliques while the first one doesn't contain any biclique. .	31
4.3	Examples to illustrate notions described in E. Kayaaslan article [13]. <b>a)</b> The consensus set $P_y(S_x)$ of $S_x = A, B$ is the intersection of neighbor set of each vertex in $S_x$ . $N(A) = 1, 2, 3$ and $S_x$ . $N(B) = 2, 3$ , hence $P_y(S_x) = N(A) \cap N(B) = 2, 3$ . <b>b)</b> The subgraph $(S_x, S_y)$ with $S_x = A, B, C$ and $S_y = 2, 3$ is a maximal biclique because the consensus set $P_y(S_x)$ of $S_x$ is equal to $S_y$ , and the consensus set $P_x(S_y)$ of $S_y$ is equal to $S_x$ . <b>c)</b> Given the set $S_x = A, B$ , we find a maximal biclique such that $S_x$ is a subset of its top vertices. This maximal biclique is defined by $(P_x(P_y(S_x)), P_y(S_x))$ , where $P_x(P_y(S_x))$ is the consensus set of $P_y(S_x)$ . . . . .	32
4.4	A bipartite graph with 2 overlapping maximal bicliques: $b_1 = (\{A, B, C\}, \{1, 2\})$ in red and $b_2 = (\{B, C\}, \{1, 2, 3\})$ (in green). . . . .	34
5.1	An unipartite graph with degree for every vertex (orange) . . . . .	37
5.2	Distribution function (top left), CDF (top right) and ICDF (bottom) for degrees of the graph in Fig.5.1. For $x = 1$ , the figures can be interpreted as: - Distribution: the graph has 4 vertices with degree $x = 1$ ; - CDF: 5 vertices have degree less or equal to 1; - ICDF: 7 vertices have degree greater or equal to 1. . . . .	38
5.3	Inverse cumulative distribution of degrees. Left: top nodes. Right: bottom nodes. . . . .	39
5.4	Inverse cumulative distribution of redundancy. Left: top nodes. Right: bottom nodes. . .	40
5.5	Inverse cumulative distribution of redundancy. Left: top nodes. Right: bottom nodes. . .	40
5.6	. . . . .	41
5.7	. . . . .	41