Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea in Fisica

Implementation of network entropy algorithms on HPC machines, with application to high-dimensional experimental data

Relatore:
Prof. Daniel Remondini

Presentata da:
Marco Molari

Correlatore:
Prof. Giuseppe Levi

Sessione II
Anno Accademico 2012/2013

To Martina, who shares everything with me.
To my parents, to whom I owe more than they know, and even more than I know.
To Ilaria, whose green eyes make me smile like nothing else in the world.
To Elia, Matteo, Simone, Lisa.
To the ones I love. To the ones that have a place in my heart.
What follows is a work of the mind. But my mind couldn't work right without them.

Contents

Abstract

Network Theory is a prolific and lively field, especially when it approaches Biology. New concepts from this theory find application in areas where extensive datasets are already available for analysis, without the need to invest money to collect them. The only tools that are necessary to accomplish an analysis are easily accessible: a computing machine and a good algorithm. As these two tools progress, thanks to technology advancement and human efforts, wider and wider datasets can be analysed.
The aim of this paper is twofold. Firstly, to provide an overview of one of these concepts, which originates at the meeting point between Network Theory and Statistical Mechanics: the **entropy of a network ensemble**. This quantity has been described from different angles in the literature.[1] Our approach tries to be a synthesis of the different points of view. The second part of the work is devoted to presenting a **parallel algorithm** that can evaluate this quantity over an extensive dataset. Eventually, the algorithm will also be used to analyse high-throughput data coming from biology.

The work is divided into many chapters. Here is a list of their titles, along with a brief description of their content.

Introduction In this chapter we shall examine the context in which the work is set, providing the reader with a short introduction to Systems Biology. We will start by defining what Systems Biology is, and how Physics is related to this subject. Then, in order to illustrate how the data we will analyse were collected, a very basic description of Cell Biology has been provided, focusing on cellular structure, main molecular components and the process of gene expression. Finally the two types of data we will make use of are presented, along with a description of how they were collected: gene expression profiling, obtained making use of DNA micro-array analysis, and protein-protein interaction networks, retrieved from on-line protein interactions databases. We will also briefly discuss the reasons why a physicist could prove a helpful figure in researches conducted in such a field.

[1] Cf. [3], [12].

Statistical Mechanics of Networks The aim of this chapter is to present to the reader the concepts of network and entropy, and combine them in the calculation of the entropy of a network ensemble. We start by defining what a network is, and giving an overview of Graph Theory, introducing the main properties we can define on a graph and describing the important role played by the adjacency matrix. Then we proceed by illustrating Erdős-Rényi model of a random graph, and how we can extrapolate interesting ideas from it, like the idea of evolution of a network and of network ensemble. The chapter continues with an exposition of the main concept of Statistical Mechanics that will prove useful in our analysis: microstates and macrostates of a system, the different kinds of constraints we can impose on our system, and the definition of Boltzmann Entropy and Gibbs Entropy. Also Shannon Entropy, a concept belonging to Information Theory we will make use of, is presented. Finally, we apply Statistical Mechanics on networks. We define the micro-canonical and the canonical network ensemble, and the different measures of entropy we can implement on these sets.

The Algorithm This chapter is devoted to the description of an algorithm for the calculation of the network entropy of a canonical network ensemble, subject to constraint extracted from a single real network. We will first describe the meaning of this value, and how we can obtain a formula that allows us to actually calculate this quantity. In the second part we will describe the translation of these operation into code in *C++* language, explaining the considerable advantages of such an implementation. We will also illustrate its features, focusing on four main aspects: input-output, memory allocation, parallelization and customization.

Performance and Data Analysis In this chapter, after the algorithm has been described, we analyse its performance. The code is integrated with a random network generator, capable of providing the algorithm with user-customized data on which to test the precision and computation time of the program. After examining the properties of these computer-generated data, we finally analyse the performance of the algorithm, comparing a non-parallel version with a parallel version running on a 8 cores machine and a 32 cores hpc machine. In the second part of the chapter we illustrate the results of the analysis conducted on high-throughput data extracted from biological samples. The meaning of these data is explained through some references to concepts included in the first chapter. A possible interpretation of the results is provided, along with some suggestions on possible future applications of the algorithm in totally different contexts.

Conclusions Finally, we draw the conclusions of our work. We try to sum up the main passages of our paper in an organic description, and illustrate the main results we achieved.

Sommario

La Teoria dei Network rappresenta oggi un campo vivo ed in via di sviluppo, specialmente nelle sue applicazioni vicine alla Biologia. Nuovi concetti provenienti da questa teoria trovano applicazione in aree dove grandi moli di dati sono già disponibili, senza che sia necessario investire denaro per raccoglierle. Gli unici strumenti necessari per l'analisi sono ampiamente accessibili alla maggior parte dei laboratori: un calcolatore ed un buon algoritmo. Con il progredire di questi strumenti, grazie all'avanzamento tecnologico e all'intuizione umana, è possibile processare insiemi di dati sempre più consistenti.
Lo scopo di questa tesi è duplice: innanzitutto quello di illustrare al lettore il concetto di **entropia di un ensemble di network**, concetto che nasce dall'incontro fra la Teoria dei Network e la Meccanica Statistica. Esso è già presente in letteratura,[2] affrontato da punti di vista leggermente diversi. Nell'elaborato si è cercato di operare una sintesi fra questi approcci. Il secondo scopo è quello di presentare al lettore un **algoritmo parallelo** in grado di valutare questa quantità partendo da un consistente insieme di dati. L'algoritmo viene inoltre utilizzato per l'analisi di un set di dati *high-throughput* provenienti da analisi biologiche.

Il lavoro è diviso in capitoli. Presentiamo di seguito una loro breve descrizione.

Introduction In questo capitolo si esamina il contesto in cui è inserito il lavoro, presentando al lettore una breve introduzione alla *Systems Biology*. Iniziamo definendo cosa sia la Biologia Sistemica e indicando le sue relazioni con la Fisica. Quindi, per arrivare a descrivere il significato dei dati che analizzeremo e come essi siano stati prelevati, abbiamo inserito una breve introduzione alla Biologia Cellulare, soffermandoci sulla struttura della cellula, sui principali componenti molecolari e sui processi coinvolti nell'espressione genica. È quindi presentato il significato delle due misure utilizzate nella nostra analisi: l'analisi di espressione genica (*gene-expression profiling*), ottenuta tramite l'utilizzo di *microarray di DNA*, e i network di interazione proteina-proteina (*protein-protein interaction networks*), estratti da database virtuali e disponibili in rete. Nello stesso capitolo discuteremo anche brevemente le ragioni per cui la figura del fisico rappresenta una risorsa importante in questo campo.

[2] Cfr. [3], [12]

Statistical Mechanics of Networks Lo scopo di questo capitolo è quello di presentare al lettore i concetti di *network* ed *entropia*, e infine di combinarli definendo l'*entropia di un ensemble di network*. Per fare ciò iniziamo con la definizione di network e presentiamo le basi della Teoria dei Grafi. Sono esposte le principali proprietà definite su un grafo, e la descrizione tramite matrice di adiacenza. Si procede quindi illustrando il modello di grafo random introdotto da Erdős e Rényi, e come da esso sia possibile estrarre idee interessanti, quali quella di *evoluzione di un network* e di *ensemble di network*. Il capitolo continua con un'esposizione dei principali concetti della Meccanica Statistica, concetti che si dimostreranno utili nella nostra successiva analisi. In particolare tratteremo la differenza fra microstati e macrostati di un sistema, i diversi tipi di vincoli a cui un sistema termodinamico è soggetto e le tre differenti ma compatibili definizioni di entropia fornite da Boltzmann, Gibbs e Shannon. Infine la meccanica statistica è applicata ai network. Si definiscono gli ensemble microcanonico e canonico di network, e le diverse misure di entropia che possiamo applicare su di essi.

The Algorithm Il capitolo è dedicato alla descrizione di un algoritmo per il calcolo dell'entropia di un ensemble canonico di network, soggetto a vincoli estratti da un network reale. In una prima parte si descrive il significato di questo valore, e come possiamo ottenere una formula che ci permette effettivamente di calcolarlo. Nella seconda parte è presentata una vantaggiosa implementazione di questo algoritmo in linguaggio *C++*, facendo uso di programmazione parallela. L'implementazione è analizzata sotto quattro aspetti: input-output di dati, allocazione di memoria, parallelizzazione e adattabilità alle diverse esigenze degli utenti.

Performance and Data Analysis In questo capitolo, dopo aver descritto le caratteristiche dell'algoritmo, ne analizziamo le prestazioni. Il codice è integrato con un generatore di random network, in grado di fornire dati provenienti da un *toy-model* secondo parametri scelti dall'utente. Tramite questi dati è possibile testare la precisione e il tempo di computazione dell'algoritmo. Presenteremo prima le proprietà dei dati generati dal programma e quindi le prestazioni sull'analisi di questi dati, confrontando una versione non parallela dell'algoritmo con una versione parallela eseguita su macchine ad 8 e 32 *cores*. Nella seconda parte del capitolo illustriamo i risultati di un analisi condotta su dati *high-throughput* provenienti dall'analisi di campioni biologici. Il significato di questi dati è spiegato facendo riferimento a concetti introdotti nel primo capitolo. È anche accennata una possibile interpretazione dei risultati ottenuti. Infine sono suggerite alcune possibili applicazioni dell'algoritmo in contesti totalmente differenti dai network biologici.

Conclusions Nell'ultimo capitolo forniamo un riassunto organico dei passaggi più importanti e dei maggiori risultati ottenuti nella trattazione.

List of Figures

Chapter 1

Introduction

> Identifying all the genes and proteins in an organism is like listing all the parts of an airplane... By itself is not sufficient to understand the complexity underlying the engineered object.
>
> *Hiroaki Kitano*[1]

The study of *Complex System* is receiving more and more attention from the scientific community. In Complex System studies Physics and Mathematics meet other important fields, such as Biology, Sociology and Economics. From their intertwining, new interesting concepts and new approaches to the study of modern problems are born. In this paper we will concentrate on a particular field, where a complex system approach can yield interesting results: *Systems Biology*. But what is Systems Biology? And what is the role of a physicist in such a field?

The two questions are obviously linked, and in order for us to answer the latter, we first have to face the former. We ask the reader for a little faith while we do so. It may seem we are going astray, but it will be worth the while. This field is full of interesting problems which cannot be answered but making use of some Physics.

1.1 Systems Biology

How could we define Systems Biology? We shall try to do so quoting a work by the biologist Ludwig von Bertalanffy,[2] who is considered a precursor theorist for Systems Biology.

[1] Cf. [7]

[2] The work in question is the introduction to his book "*General System Theory, Foundations, Development, Applications.*" published in 1969.

In the biologist's opinion, old-fashioned science "tried to explain observable phenomena by reducing them to an interplay of elementary units investigatable independently of each other." On the other hand, contemporary science recognized the importance of "wholeness", defined as "problems of organization, phenomena not resolvable into local events, dynamic interactions manifest in the difference of behavior of parts when isolated or in higher configuration, etc.; in short, 'systems' of various orders not understandable by investigation of their respective parts in isolation."
This kind of approach is what identifies Systems Biology: the attention given to studying biological systems in their "*wholeness*". In fact, we often tend to analyse phenomena by breaking up a system in its components. In spite of making things easier, this kind of approach is not capable of explaining some behaviours, the ones in which a system acts as more than just the sum of its parts. And this often happens in Biology, when we try to investigate how an organism works. Complexity is inevitable, when we study life.

Some time passed between these early pronouncements and following results. This time was necessary to accumulate sufficient knowledge on biological systems and ways to analyse and interpret this knowledge. As a matter of fact, in order to understand a complex behaviour, we first need to collect data on the state of the system and its evolution over time; and in the case of biological systems, data are extensive. In particular our analysis could not have been possible, if not for a technique called *Micro-array DNA analysis*, which will be illustrated later in this chapter. Secondly, given the big amount of data to be analysed, computational tools are needed. Finally, and most importantly, data have to be interpreted. There has to be a shift of paradigm, from the comprehension of the behaviour of single small components, to the wider comprehension of the way the whole systems works. Biologist are really good at the components analysis, but sometimes to understand the way a whole-system works we have to make use of concepts and tools coming from Physics. Concepts such as *entropy*, which we will be using in this paper.
We are approaching the answer to the second question: "what is the role of a physicist in such a field?". He is not an expert on Biology, so he would not be able to conduct a research without the help of someone who is more familiar with this subject. But he possesses skills that enable him to analyse great amounts of data, and create models of the complex behaviour of the system. We will discuss this in more detail in the last section of this chapter: *"A place for Physics"*.

As we have said before, a physicist is not supposed to be an expert in Biology, and he can not be either. That is not his strong point. But it may be useful for him, matter-of-factly, to have some knowledge of the field he is working on. This can help him in comprehending the meaning of the data he is analysing, what is the best approach he can have and what results he expects to find. We shall therefore give a short overview on the subject of *Cellular Biology* and some of its main concepts, and explain the origin

and the meaning of the data we will later analyse. This brief overview has no claim of completeness, it will just be enough for the reader to get an idea of the concepts we will make use of later in the work. So any reader who is already familiar with these concepts may as well skip what follows, without compromising the comprehension of what comes next.

1.2 Cell Biology in short

Cell Biology is quite a wide subject, and it would not be possible to cover it all in this paper. We shall therefore illustrate only the concepts strictly related to our work. We will first see what a cell is, and how important is the role of proteins in the cell. Secondly, we will analyse the process of protein synthesis, that consists in the translation of DNA strands into amino-acids chains.

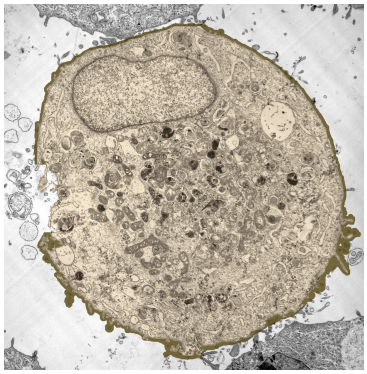


Figure 1.1: Image of a human cell.

1.2.1 Cell structure

A cell is the simplest subunit of life. But even so, it is an extremely complicated system. First of all, let us analyse its structure. Every cell is surrounded by a membrane, which separates it from the surrounding environment. It is called the *cell membrane* or *plasma membrane*. It is semi-permeable, and the traffic of substances going in and out of it is restricted to some specific molecules, or regulated by proteins incorporated in the membrane. There is another main compartment inside the cell, called *nucleus*. It is separated from the rest of the cell by a *nuclear membrane*, and contains the *DNA*, a macro-molecule fundamental for life, which will be analysed later. The remainder of the cell is called the *cytoplasm*. It contains small structures, called *organelles*, that fulfill specific cellular functions. Among these we will just mention *ribosomes*, the *endoplasmic reticulum* and the *Golgi complex*, which are the ones where protein formation takes place. What we have just described is the general structure of an animal cell. If we were to consider also

plant cells, or even bacteria, we would have had to make some distinctions. But again, there is no need for it in this work.
Many physiological processes take place inside a cell every instant: molecules are disassembled to extract energy from their chemical bonds, or to make their subunits available for the construction of other molecules. Some other molecules are assembled, or undergo changes. The molecules which make up a cell belongs to four main categories. We have already named some of them. They are *carbohydrates*, *lipids*, *proteins* and *nucleic acids*. Carbohydrates serve mainly as energy storage molecules. Energy can be extracted from them through chemical reactions, and used in other physiological processes. Lipids have many functions. They are made up mainly of non-polar groups; for this reason they are not soluble in polar solvents, and used to separate different compartments inside the cell. They make up the cell membrane. As carbohydrates, energy can be extracted from them. Finally some lipids, such as steroids, have the function of hormones.
We will now describe nucleic acids and proteins more extensively.

1.2.2 Nucleic Acids

There are two main nucleic acids: *DNA* (deoxyribonucleic acid) and *RNA* (ribonucleic acid). The former is the storage molecule for hereditary information, while the latter is fundamental for protein synthesis and the formation of ribosomes. All nucleic acids are polymers built up of single structural units, called *nucleotides*. Each nucleotide consists of three parts: a *nitrogen-containing base*, a *pentose sugar* and one or more *phosphate groups*. In DNA's nucleotides only one phosphate group is present. The typical structure of a nucleotide is represented in figure 1.2.

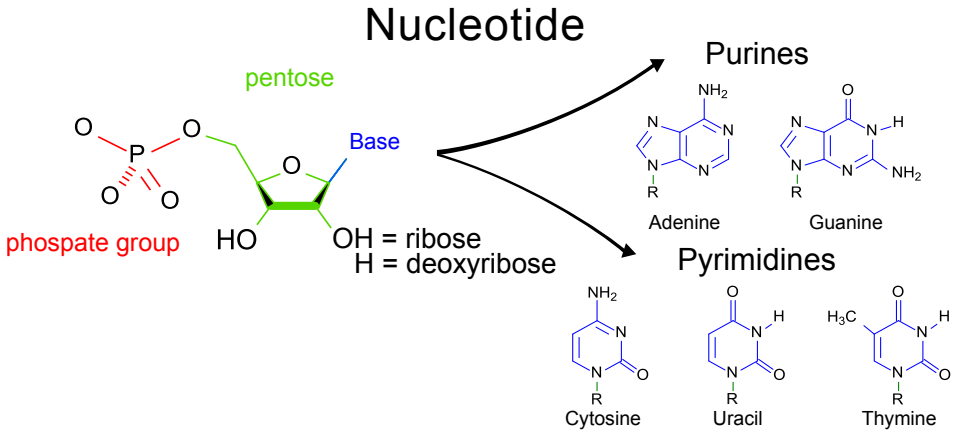


Figure 1.2: The structure of a nucleotide.

There are five bases, divided in two groups: Purines and Pyrimidines. *Adenine* (A) and *Guanine* (G) are the two Purines, while *Cytosine* (C), *Thymine* (T) and *Uracil*

(U) are Pyrimidines. Each nucleotide in the polymer is linked to the next one through covalent bonds: a carbon of the pentose sugar forms the bond with an oxygen of the phosphate group belonging to the next nucleotide. This creates a long chain of nucleotides. In DNA single chains couple together, thanks to hydrogen bonds that form between the nitrogen-containing bases. These bonds are allowed only between specific bases though: Guanine with Cytosine (G-C), and Adenine with Thymine or Uracil (A-T or A-U). The result of these bonds is a coupling of complementary chains of DNA in a double-helix structure, firstly discovered in 1953, by famous scientists J. Watson and F. Creek. The twin chains have to be compatible in order to link, so at each base in one chain corresponds the complementary base in the other chain. This structure is represented in figure 1.3, altogether with an example of the bonds between the subunits of DNA.

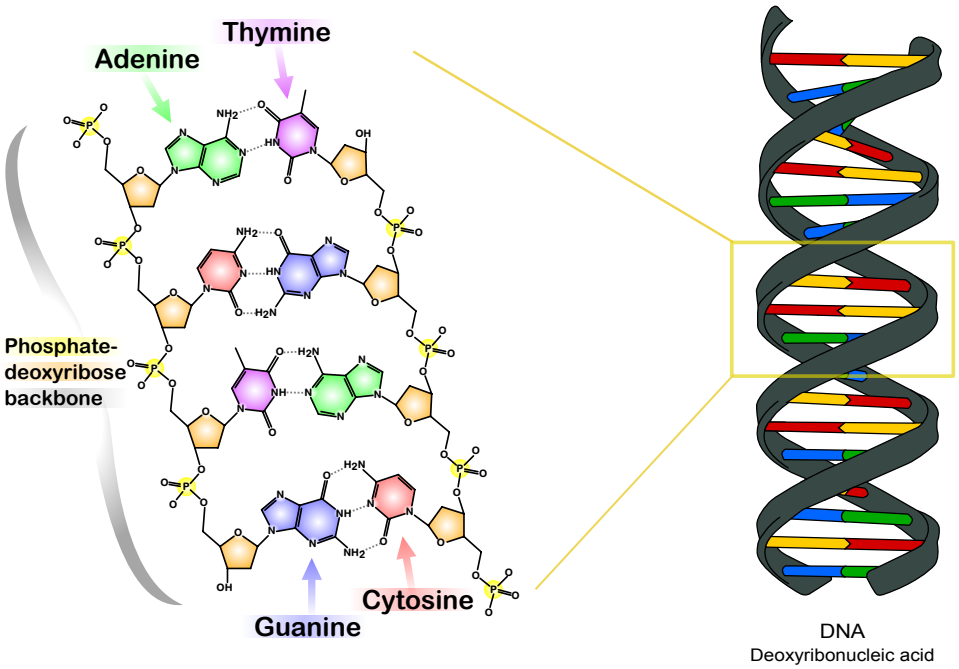


Figure 1.3: DNA chemical structure.

In the nucleotides of DNA and RNA only four of the five possible nitrogen-containing bases are present. In the case of DNA, the absent one is Uracil, while in RNA it is Thymine. This is the first main structural difference between DNA and RNA. Secondly, DNA's pentose sugar is deoxyribose, while RNA's pentose sugar is ribose. Finally, DNA usually assumes the double-helix conformation, while RNA is usually present in a single filament.

1.2.3 Proteins

The last class of substances we shall consider are *proteins*.
They fulfill many highly important functions inside and outside the cell: amongst their many functions, they act as catalytic enzymes for biochemical reactions, control the metabolism and are responsible for the degradation of other proteins. They even control the transcription and translation of genes into other proteins, a process which we shall soon illustrate.
A protein in an assembly of one or more *polypeptides*. Each polypeptide consists of a polymer chain of *amino acids*. The common features of all amino acids are amine (NH_2) and carboxylic acid ($COOH$) functional groups. They are linked to a central carbon atom, which also carry a *residual group*, different for each amino acid and thus displaying different physiochemical properties. Two amino acids can link to each other through a covalent bond, called peptide bond, as illustrated in figure 1.4. In this bond the amine group of the first amino acid links to the carboxylic group of the second amino acid. Proceeding this way, a chain of these molecules can be formed.

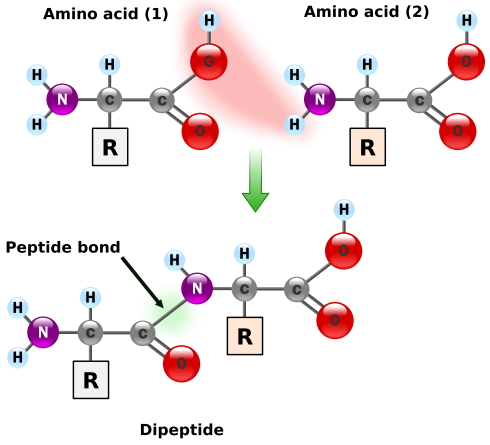


Figure 1.4: Formation of a peptide bond

There is a total of 20 different amino acids in human cells. They differ from each other in the residual group, and therefore also in chemical and physical properties. By forming chains of these molecules, with amino acids in the right order, one can obtain a macromolecule with specific properties, which could, for example, act as catalyst for reactions, or combine with other molecules to form bigger structures. This is why proteins are so important in the physiological processes of a cell. But what are the right sequences of amino acids that form human proteins? How is this information stored inside the cell? It is stored into the nucleus of the cell, coded into the DNA.

1.2.4 Gene Expression

We say that the genetic information is "*coded*" in the DNA because there is actually a code through which the DNA can be interpreted and translated into proteins. What matters is the order of the nitrogen-containing bases in the DNA strand. Each triplet of bases corresponds to a specific amino acid. These strands are read in sequence, and the corresponding amino acids are subsequently added in the chain, slowly forming a protein. Not every section of the human DNA codes for proteins though. Some fragments are translated in *ribosomal RNA* (*rRNA*), which makes up ribosomes, or *transport RNA* (tRNA) which has an important role in the formation of proteins. However, most of the fragments[3] are transcribed into mRNA, but can not be translated into proteins. The collection of these fragments is called *non-coding DNA*.
Each fragment of DNA that is actually translated into a protein is called "*gene*". And the process of translation is called *gene expression*. This process is performed in many steps, illustrated in the following list:

Transcription Firstly, a fragment of DNA encoding a gene is transcribed in a complementary strand of *messenger RNA* (mRNA). This step takes place in the nucleus. It is performed by a special enzyme called *RNA polymerase*, and always starts from DNA sequences called *promoters*, to whom the enzyme binds with a special affinity. As the enzyme proceeds along the DNA string, a complementary mRNA strand is created, until a specific sequence, called the *terminator*, is encountered. At this sequence the transcription process ends. The resulting strand of mRNA is called *pre-mRNA*.

Processing of mRNA Before getting out of the nucleus, the strand of pre-mRNA undergoes a series of structural changes: some specific sequences in the string, called *introns*, are removed. The ones that remain are named *exons* instead. This process is called *splicing*. The removed fragments will not take part in the formation of the protein, and now the resulting mRNA strand is no more complementary to the initial DNA strand.

Translation The mRNA strand gets out of the nucleus, and it reaches a ribosome, in which the next step takes place. The ribosome is a macro-molecule composed of rRNA and proteins. It has a site where the mRNA can bind, and the process of *translation* can take place. Translation occurs according to the *genetic code*, a correspondence between triplets of mRNA bases (called *codons*) and amino acids. During the translation process each codon is scanned in the ribosome, and the corresponding amino acid chained to the forming protein. Since there are four possible bases, the number of all possible codons is bigger than the total number of amino acids ($4^3 = 64 > 20$), so more than one codon often codes for the same amino

[3]Up to 98.5% of human genome is non-coding DNA. (Lander et al. 2001, Venter et al. 2001)

First Base	Second Base U		C		A		G		Third Base
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys	C
	UUA	Leu	UCA	Ser	UAA	Stop	UGA	Stop	A
	UUG	Leu	UCG	Ser	UAG	Stop	UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC	Leu	CCC	Pro	CAC	His	CGC	Arg	C
	CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg	A
	CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg	G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser	C
	AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg	A
	AUG	Met	ACG	Thr	AAG	Lys	AGG	Arg	G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly	C
	GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly	A
	GUG	Val	GCG	Ala	GAG	Glu	GGG	Gly	G

Table 1.1: The genetic code

acid. The genetic code is represented in table 1.1, in the form of a list of all possible codons, each one accompanied by its corresponding amino acid. It is interesting to notice that the codon AUG (corresponding to the amino acid *Methionine*) is the one that always starts the translating sequence. The codons UAA, UAG and UGA are the ones that always stop the translation instead.

An important role in translation is played by *RNA transfer* (tRNA), which is the link between the "nucleic acid language" and the "protein language". Each tRNA molecule carries an amino acid on one side, and an *anticodon* (a codon complementary to the one corresponding to the amino acid it carries) on the other side. When the anticodon meets the corresponding codon on the mRNA, the amino acid is released and chained to the forming protein. Adding one amino acid after the other, the protein slowly takes form, until the stop sequence is reached and the protein is released from the ribosome.

Sorting and Post-translational modifications The cell has also a complicated sorting and distribution system, thanks to whom each protein is directed to the place where it is needed. Most of the proteins are not yet ready to perform their task though: the last step required for the formation of a protein is the *folding* process. The specific function of a protein can often be fulfilled only if the protein takes a specific form. The long chain of amino acid has to fold on itself, assuming a

determined structure. Once this is done, the protein is ready to be released, and to take part in the biological processes of the cell.

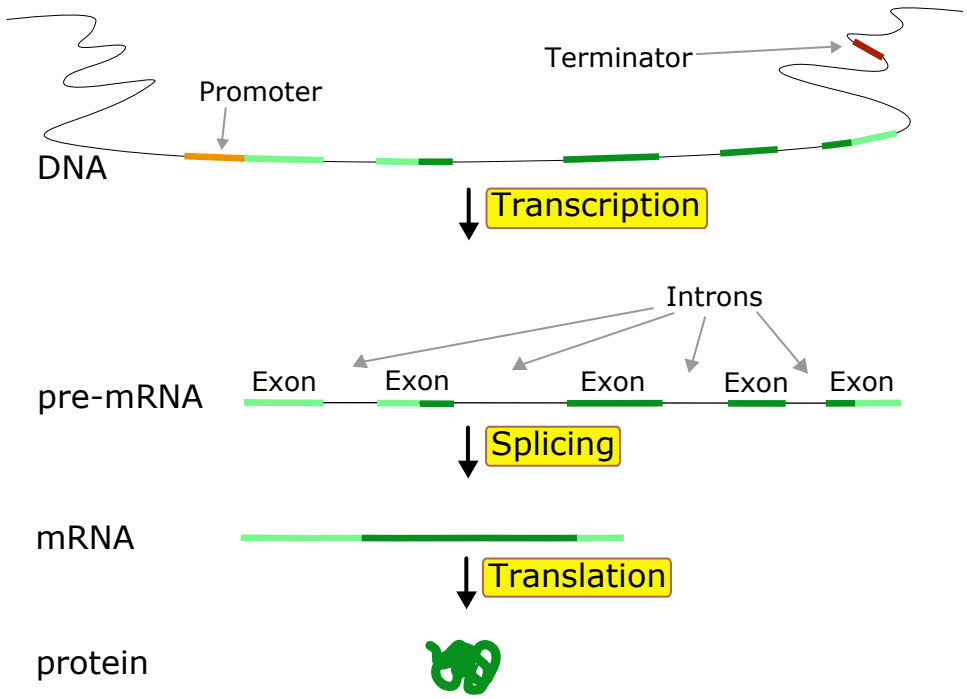


Figure 1.5: Schematic representation of the gene expression process.

The whole process of gene expression is represented schematically in figure 1.5. There is one last thing worth noticing: the amount of protein produced from a gene is directly proportional to the amount of mRNA transcribed from the gene. This proves useful when we face the problem of measuring how much a gene is expressed, because it spares us the trouble of having to actually measure how much of each protein is present in the cell: it is sufficient to measure the amount of the corresponding mRNA sequence.

1.2.5 Gene Expression Regulation

Human genome has a total of 20'000 - 25'000 protein-coding genes, each on average having a length of 30'000 bases[4]. Not all these proteins are produced in every cell though. The difference between cells belonging to one tissue or the other consists mainly on the difference in the expression level of specific genes. For example, detoxification enzymes produced by liver cells are not present in epidermal cells; or secretions produced by stomach cells are absent in the muscular tissue.

[4]The sum of all the coding sequences corresponds to just a 1.5% of human genome.

These different levels of expression are controlled by a very complex regulation system in eukaryotic cells. Expression can be regulated in many ways: through transcription control, by inhibition of the promoter in the DNA sequence, blocking the export of mRNA from the nucleus, controlling the translation ratio or the decay rate of mRNA and specific proteins.
Expression levels can change from one person to the other, and also in the same person with age. Errors in the regulation of genes expression levels are the cause of serious illness. Because of its importance and complexity, the regulatory network is nowadays the object of many studies and researches.

1.3 Dataset: Gene-Expression, Protein-Protein Interaction

If we want to understand the way a whole system works, we need to extract comprehensive data, enough to provide us with a complete description of the state of the system. Nowadays, thanks to progresses in molecular analysis, we can do so. But even describing the state of "simple" systems like cells requires a great amount of information. This information is collected with automated methods, through what are called *high-throughput measurements*. Data are often stored in virtual databases, and made available to anyone who may need them. As we have said, the big amount of information makes a calculator necessary in order to analyse the data. We will now illustrate the meaning of the data we will make use of, and what kind of measurements have been used to collect them.

1.3.1 Gene-expression Profiling

The first information we will need for our analysis is how much a gene is expressed into a cell, or equivalently how much of the corresponding protein is present in the cell. This has to be measured simultaneously for all the genes, starting from a small sample. How to perform such a complicated measure? The best technique is through what is called a *DNA Micro-Array*.
This technique was developed at the end of last century (DeRisi et al. 1997) and represents a high-throughput method for the analysis of gene expression. It allows us to monitor the expression of several thousand genes in a single experiment, and have a global picture of the cellular activity. This is as close as we can get to knowing the state of our system. This technique is based on the correlation we previously pointed out, between the amount of mRNA related to a particular gene, and the produced amount of corresponding protein. It measures the first in order to know the second. It is performed in the following steps, represented in figure 1.6:

1. The *DNA chip* is fabricated. It is made by spotting samples of DNA from a DNA

library on a glass slide or a nylon membrane, following a matrix pattern. Each site of the matrix has a different DNA sample, and they are all mapped in a digital software. This passage is performed by automated machines. The samples from the DNA library are obtained through reverse transcription from mRNA strands[5].

2. Total mRNA is extracted from the samples we want to analyse. It is the ensemble of all the mRNA strings present in the cell, coding for all the expressed genes.

3. Through reverse transcriptase *complementary DNA* (cDNA) is created from the mRNA strands, and labelled with a fluorescent dye (usually red or green).

4. The solution containing the labelled cDNA is now put in contact with the surface of the DNA chip. They hybridize to the spot where the complementary strand of DNA has been spotted. The cDNA strands that do not undergo hybridization are washed away from the chip.

5. After washing, the DNA chip is put into a scanner and the intensity of the fluorescence is measured for each spot. If the intensity of a spot is high, it means that a considerable amount of the relative labelled mRNA was present in the sample, so probably also the corresponding protein was there in good quantity. If the intensity is low instead, there was little mRNA in the sample, and little protein too.

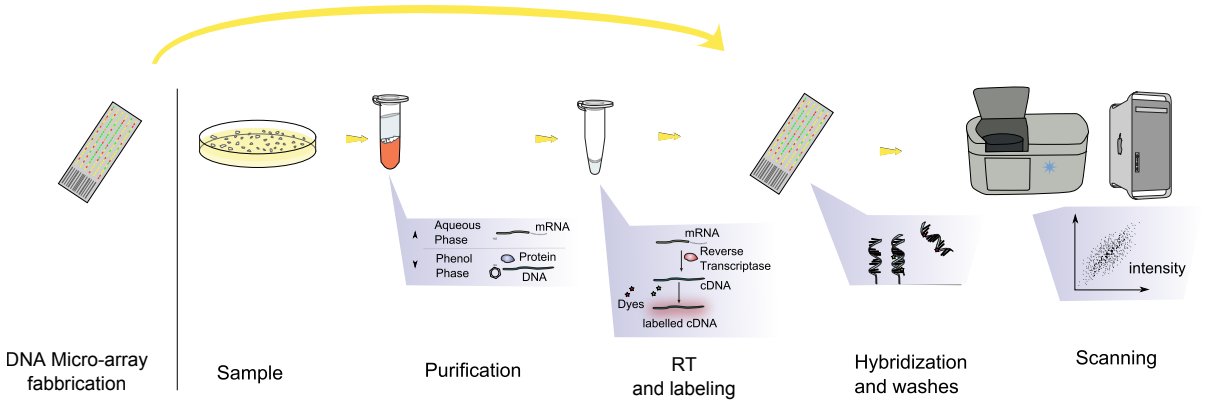


Figure 1.6: Micro-Array Analysis

[5]Reverse transcription is performed easily thanks to a particular enzyme, *reverse transcriptase*, which performs the opposite of transcription: given an mRNA strand, it builds the complementary DNA strand. Notice that because of the splicing, the resulting DNA fragments are not present in the original DNA, because all the introns have been eliminated.

A single DNA chip can have up to 2.1 million probes spotted on it, allowing the experimenter to monitor the expression of several thousand genes simultaneously, in a single experiment. Later in the work we will make use of data obtained with this technique.

1.3.2 PPI Network

We already showed how important is the role of proteins inside the cell. They perform many tasks and are involved in many physiological processes, they often combine with other proteins or physically interact with them to perform their function. In order for us to understand the way a biological system works, we need a further important information. Every single protein is not capable of interacting with all the other proteins in the system, it physically couples with only a specific fraction of them. If we had this information, and knew precisely which proteins can interact with each other, we could create a *network*[6], a sort of net where the nodes represent the proteins of our system, and the lines connecting the nodes represent interactions between two proteins. This is what is called a *Protein-Protein Interaction Network* (PPI network in short), and from the analysis of this model we can extract many important information on our system.

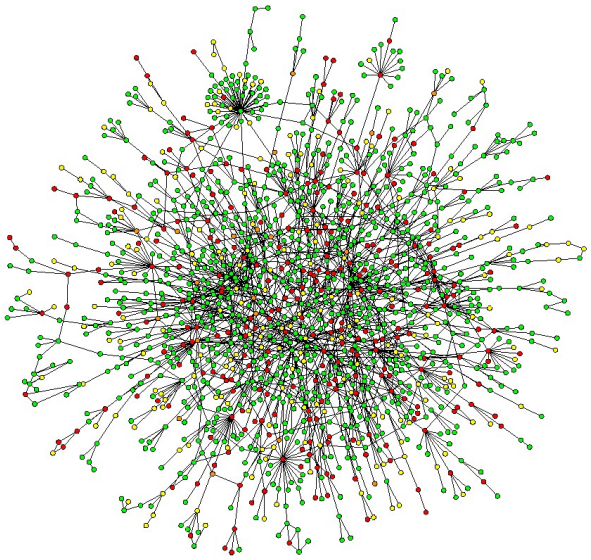


Figure 1.7: PPI network of yeast.

There are several experimental techniques that allows researchers to understand which proteins can interact with each other. For example using *protein chips*: they

[6]The concept of network will be discussed in detail in the next chapter.

are similar to DNA chips, and work in the same way. However, they are subject to more experimental problems: protein interaction can often take place only under certain condition (e.g. the presence of an enzyme, or a particular temperature), and are thus not so easy to detect.
Results of these experiments are put together and stored in on-line databases, easily accessible from all over the world. Because of the complexity and heterogeneity of the data, particular tools are available, that allow the user to perform simultaneous queries to many databases at once. The data are then collected, integrated and presented to the user, ready to be analysed.
In particular, the protein-protein interaction data we will be using were extracted using "*Pathway Commons*"[7], an on-line data mining tool. It collects the data from many public databases and integrates them, offering also additional information to the user, such as an indicator of the quality and reliability of each data.

1.4 A place for Physics

Now that the reader has an idea of what Systems Biology is, and what concepts and tools it makes use of, we can finally face the question: *what is the role of a physicist in SB?*
We already pointed out that he is not an expert in Biology. He has an important role in this field nonetheless, thanks to two useful qualities.

The first is his analysis ability. He has the competence and statistical tools to interpret great amounts of data and extract relevant information from them. Given the considerable number of dimensions of the variable space in all living systems, it is a very helpful quality in this field.
The second ability is the interpretation he can give to the data. A physicist possesses a wide set of concepts coming from many areas of the physical knowledge, that allows him to create models that account for complex behaviours, and give the data a significant interpretation.

Our work is set on both these abilities: we will have to handle a consistent dataset[8] and also interpret it making use of the concept of *Entropy*, which comes from Statistical Mechanics and Information Theory.

[7] It can be accessed from the web address `http://www.pathwaycommons.org`

[8] The great number of data in our dataset, obtained by *high-throughput* measurements, is what justifies a statistical mechanics approach.

Chapter 2

Statistical Mechanics of Networks

One of the many tools used in System Biology are *Networks*. The concept of network is very general and has a wide applicability: it is used in Sociology as well as Communication Theory, Mathematics, Biology and Physics. But the rigorous language for the description of networks comes form *Graph Theory*, a mathematical theory whose origin traces back to the work of Euler, solving the Königsberg bridges problem (1736). In particular, we will make use of the concept of *Random Network*, introduced by the work of Erdős and Rényi.

Entropy, on the other hand, was firstly introduced by Rudolf Clausius in Classical Thermodynamics as a state function. This quantity reflected the reversibility of thermodynamic processes. Later Ludwig Boltzmann gave a new description of the same concept, in the context of Statistical Mechanics, in terms of the relationship between macroscopic and microscopic states of the system. In his description, Entropy is a measure of the number of microstates a system can be into, given its measured macrostate. Also J.W. Gibbs gave a similar interpretation, based on the concept of statistical ensemble. Finally, C. E. Shannon defined Entropy in Information Theory, as a measure of the information coded in a message.
In recent studies[1] the same concept was applied to networks, in order to quantify the variabiliy of a network ensemble, and the information coded in the constraints.

In this chapter we will briefly provide a mathematical description of graphs and the properties they have. We will also introduce Random Graphs, and try to describe some of their main features as well as their importance in network science. It is a basic introduction, so any reader who is familiar with these concepts may simply skip this chapter. Then we will introduce the concept of *Entropy*, in the context of Statistical Mechanics and Information Theory. Finally, we shall apply this concept to networks.

[1] In particular cf. [3], [4], [12]

2.1 What is a network?

In general terms, a network is any system that can be mathematically represented as a graph. Thus, it can be reduced to a set of elements, called *nodes*, and a series of relations amongst them, called *edges*, visualized as links going from one node to the other.
This description is very general, so this model suits a wide variety of systems. This broad applicability of graph theory is the one feature that allows us to compare systems really different from each other (e.g. network of links between web pages, network of scientific collaborations, communication networks...), and try to find common features.
Graph are present in a wide variety, and a wide variety of properties over them has been introduced. We will provide a very basic description of these objects, just to the extent useful for our aim. We shall not consider *multigraphs*, in which more than one edge between the same pair of nodes is allowed, nor *pseudo-graphs*, which can present both multiple edges and "loop" edges, that is edges going to and from the same node. These require some more mathematical tools, and are not necessary for our aim. We will therefore just describe *simple graphs*, in which no more than one edge can connect two different nodes, and a node can not be connected to itself.

2.2 Basics of Graph Theory

An *undirected simple graph* G is mathematically defined as a pair of sets $G = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is a countable set whose elements are called "*vertices*" or "*nodes*"; $\mathcal{E}$ is a set whose elements, called *edges*, are *unordered* pairs of elements of nodes. Each pair in $\mathcal{E}$ represents a link between two nodes. If we consider *ordered* pairs of nodes as elements of $\mathcal{E}$ instead, we can describe *directed* simple graphs, where each link has also a direction. Given two elements of the set $\mathcal{V}$, namely i and j, if the pair (i, j) belongs to the set $\mathcal{E}$: $(i, j) \in \mathcal{E}$, then the two nodes are said to be *adjacent* or *connected*.

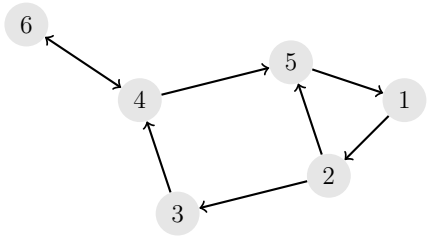
The *order of the graph*, denoted with N, is defined as the cardinality of the set $\mathcal{V}$, i.e. the number of nodes in the graph. On the other hand, the cardinality of $\mathcal{E}$ is referred to as the *size of the graph*, and denoted with E. With simple combinatorial calculations[2] we can show that the maximum number of edges for a graph of order N is equal to $E_{max} = \binom{N}{2} = \frac{N!}{2!(N-2)!}$. Such a graph, where all the N edges are connected, is said *complete N-graph*.

[2] In fact, a graph can have as many edges as the number of different unordered pairs of vertices we can form, which is the number of all the possible combinations of 2 elements out of a set of N

2.2.1 Representation of a Graph

Undirected graphs are often graphically depicted as a set of dots connected by lines. The dots corresponds to the nodes, and the lines to the edges. A directed graph is represented in a similar fashion, the only difference being in the lines, replaced with arrows to convey the directionality of the link. As we have seen, the presence of an edge between nodes i and j in an undirected graph grants a connection in both directions. On the contrary, in directed graphs the fact that node i is linked with node j does not imply that node j is linked with node i.

In figure 2.1 we show an example of a directed graph, followed by the description of its own nodes set, and edges set. The couples of nodes that represent an edge are ordered, and this conveys the directionality of the link. In this case, the set of edges is a subset of the Cartesian product of the set of nodes with itself: $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$.

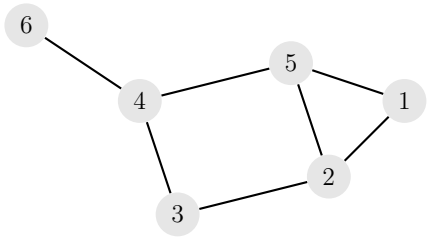


$$\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$$

$$\mathcal{E} = \Big\{(6,4), (4,6), (4,5), (5,1) \\ (1,2), (2,5), (2,3), (3,4)\Big\}$$

Figure 2.1: Directed Graph

In figure 2.2 instead, we have an example of undirected graph. The arrows are replaced with lines, since there is no directionality in the link. Moreover, the ordered pair is now replaced with a simple unordered set.



$$\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$$

$$\mathcal{E} = \Big\{\{6,4\}, \{4,5\}, \{5,1\} \\ \{1,2\}, \{2,5\}, \{2,3\}, \{3,4\}\Big\}$$

Figure 2.2: Undirected Graph

2.2.2 Adjacency Matrix

If the order of the graph is known, all the information we need to reconstruct a graph is, given any couple of nodes, whether they are linked or not. This information is encoded in the *adjacency matrix*: $A_{N\times N} = \{a_{ij}\}_{i,j}$. It is a square matrix, whose number of lines and columns is equal to N, the order of the graph. The value of the element a_{ij} tells us if node i and node j are linked or not, according to the following rule:[3]

$$a_{ij} = \begin{cases} 0 \quad \text{if} \quad (i,j) \notin \mathcal{E} & \text{thus if node } i \text{ is not linked with node } j \\ 1 \quad \text{if} \quad (i,j) \in \mathcal{E} & \text{thus if node } i \text{ is linked with node } j \end{cases} \tag{2.1}$$

We can thereby write the two matrices, for the directed and undirected graph of figure 2.1 and 2.2. Notice that, as a result of the non-directionality of the link, the adjacency matrix of an undirected graph is always *symmetrical*: $A = A^T$. Moreover, since we excluded the possibility of a "loop" edge, in both directed and undirected graphs all the elements on the diagonal of the matrix have null value: $\forall\, i \in \{1, \ldots, N\} \quad a_{ii} = 0$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.2}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.3}$$

Matrix (2.2) is referred to the directed graph, while matrix (2.3) refers to the undirected graph.

2.2.3 Connectivity Degree of a Node

One of the most important properties we can define in a graph is the *connectivity degree* (or simply *degree*) of a vertex. We shall indicate the degree of vertex i with the symbol k_i. It is defined as the number of edges in the graph that share a link with the specified node. This measure gives us information on the relevance of the vertex in the graph. The connectivity of every node can be extracted easily from the adjacency matrix: connectivity of node i corresponds to the sum of all the elements in the i-th row of the matrix.

$$k_i = \sum_j a_{ij} \tag{2.4}$$

[3]In equation 2.1, the notation (i,j) presumes a notion of order, and is valid for directed graphs. In case of undirected graphs, it is sufficient to replace it with the notation: $\{i,j\}$.

Moreover, if we were to sum all the elements of the matrix, we would sum all the connectivities of the nodes. In this sum, each edge is considered twice: once for each node it connects. As a consequence, the sum of all elements of the adjacency matrix is equal to twice the size of the graph E:

$$\sum_{i,j} a_{ij} = \sum_{i} k_i = 2E \tag{2.5}$$

Actually, the definition of degree of a node presents some ambiguity when we start to consider directed graphs. To solve this problem we can define an *in-degree* k_i^{in} and an *out-degree* k_i^{out} as follows:

$$k_i^{in} = \sum_{j} a_{ij} \qquad k_i^{out} = \sum_{j} a_{ji} \tag{2.6}$$

If we consider again an undirected graph, we have $A = A^T$, so for all nodes $k_i^{in} = k_i^{out} = k_i$.
In addition to this, we can introduce one more quantity to evaluate how many links a node has on average: the *average connectivity* $\langle k \rangle$.

$$\langle k \rangle = \frac{1}{N} \sum_{i} k_i = \frac{2E}{N} \tag{2.7}$$

The last equation comes easily from (2.5), N being the order of the graph.

Finally, we can define the *degree sequence* of a graph as a vector $\vec{k} = (k_1, k_2, \ldots, k_N)$ containing a list of the connectivity degrees of all the nodes in the graph. This vector will have an important role in the further analysis we will perform. In fact it will represent one of the constraint we will extract from our real network, in a sense that will become clear to the reader later.

2.2.4 Metric on a Network

Often modelling a system making use of just a simple network would be too vast a simplification. It would mean neglecting part of the information the system has in itself; information which is important in order to understand the way the system works. Sometimes this information can be included in our model if we code it as a further property of our network. In our case this property will be the *distance* between nodes.
We define a *distance matrix* $D_{N\times N}$. It has the same dimensions of the adjacency matrix, and each element d_{ij} of the matrix represents the distance of the node i from the node j. We can derive some properties of the distance matrix starting from the mathematical definition of distance.

A distance function defined on a set χ is any binary function $d : \chi \times \chi \to \mathbb{R}$, satisfying the following four properties:

1. Non negativity: $\forall x, y \in \chi \quad d(x, y) \geq 0$
2. Coincidence axiom: $\forall x, y \in \chi \quad d(x, y) = 0 \Leftrightarrow x = y$
3. Symmetry: $\forall x, y \in \chi \quad d(x, y) = d(y, x)$
4. Triangle Inequality: $\forall x, y, z \in \chi \quad d(x, y) + d(y, z) \geq d(x, z)$

If these properties are satisfied, $d(x, y)$ represents the distance between the element x and the element y of the set χ. In our case, the set χ will obviously be the set of the nodes of our graph $\mathcal{V}$.

From the first property we notice that every element of our distance matrix is a positive real number. Then, from symmetry it follows that $\forall i, j \in \{1, 2 \dots N\} \quad d_{ij} = d_{ji}$, thus the distance matrix is symmetrical: $D^T = D$. Furthermore, because of the coincidence axiom all the elements on the diagonal of the matrix have null value: $\forall i \in \{1, 2 \dots N\}, \quad d_{ii} = 0$. Finally, the triangle inequality sets a number of constraints[4] on the distance matrix.
We called this property "distance". Needless to say, it does not have to be a physical distance. It can be any feature of our system that can be represented as a matrix of real numbers satisfying the properties we just listed.

2.3 Random Graph and Network Ensembles

Random Graphs are very important tools for the study of real networks. They are defined as graphs whose formation occurs according to some kind of random rule. In their article "On Evolution of Random Graphs"[5] Erdős and Rényi write: "The evolution of random graphs may be considered as a (rather simplified) model of the evolution of certain real communication-nets, e. g. the railway, road or electric network system of a country or some other unit, or of the growth of structures of inorganic or organic matter, or even of the development of social relations."

Random Graphs can indeed provide a model for network growth; a model capable of explaining data coming from many different contexts. They proved crucial for the comprehension of some properties of real networks, and they allowed scientists to get a deeper understanding of how a real network works.
We will briefly introduce the main example of random graph. This will eventually lead us to the concept of network ensemble.

[4] If we have N nodes in our graph, then there are $3 \cdot \binom{N}{3}$ inequalities that have to be satisfied.
[5] Cf. [6]

2.3.1 Erdős–Rényi model

In the same article, the two mathematicians proposed a simple model for a random graph: let us suppose we have a particular set of nodes, $\mathcal{V}$, having N elements. At time $t = 0$ there are no edges in our graph. At every unit of time, an edge is added to the graph, chosen at random amongst all the possible combinations of two nodes still not linked by an edge, which at time $t = 1$ are $\binom{N}{2}$. At time $t = 2$ the second edge will be chosen from a set of $\binom{N}{2} - 1$ possibilities, and so on. The process ends at time $t = M \leq N$, the final graph having a total of M edges.

Can we make any prediction on the outcome of this process? We know for sure that the result will be a graph having N nodes and M edges. Let us denote with $G_{N,M}$ the set of all the graphs that satisfy such conditions. If we are to evaluate the cardinality of $G_{N,M}$, we can consider that to build a graph belonging to this set we have to choose M edges from the set of all the possible edges we could draw between our N nodes, this last set counting $\binom{N}{2}$ elements, as we have already explained. As a result, the cardinality of $G_{N,M}$ is equal to the number of combinations of M elements from a set of $\binom{N}{2}$.

$$\mathbf{card}(G_{N,M}) = \binom{\binom{N}{2}}{M}$$

What is the probability that a given graph $\mathcal{G} \in G_{N,M}$ is the outcome of the random process we just described? We can notice that each graph in $G_{N,M}$ has the same probability of resulting from such a process. As a consequence, if we indicate the probability of the graph $\mathcal{G}$ being the result of the process as $P(\mathcal{G})$, we have that:

$$P(\mathcal{G}) = \frac{1}{\mathbf{card}(G_{N,M})} = \binom{\binom{N}{2}}{M}^{-1}$$

It is interesting to notice that the connectivity degree distribution of the nodes follows a well known pattern. In fact for each couple of nodes there is a $M \times \binom{N}{2}^{-1}$ probability of being connected by a link. For the sake of simplicity, let us call this probability π. As a result, considering a node in our network, the probability of it having a connectivity degree equal to a value k is:

$$p(k) = \binom{N-1}{k} (\pi)^k (1-\pi)^{N-1-k} \tag{2.8}$$

This is exactly the binomial distribution. For large values of N, it can be approximated with the Poisson ditribution:

$$p(k) = \frac{(N\pi)^k}{k!} e^{-(N\pi)} \tag{2.9}$$

2.3.2 Evolution and Network Ensembles

The work of Erdős and Rényi continues with the investigation of the arising of some properties in the graph as time passes and links are added. This does not concern our paper, but two interesting ideas have already been introduced, which will be really important for what will follow:

Evolution of a network. Networks are not stable and immutable entities: they evolve. Nodes can be added, new edges can form between existing nodes. Sometimes the arising of edges is totally random, other times it follows some kind of rule. But considering the fact that the specific real network we are investigating has an history can always help us understand many of its properties. Many of them originated from the way the network evolved. This leads us to the second important idea.

A real network is one particular instance of a wider set of possible networks. This set, which we will call *Network Ensemble* henceforth, reunites all the networks whose features are somehow similar to the ones of the real network we are considering. It provides us with an environment whose analysis can give us additional information on the real network. It will be of capital importance in this paper.

Ensembles are built according to a number of constraints we require the networks that belong to them to satisfy. In the previous example, we required a fixed number of edges M for every network. The result was the set $G_{N,M}$. But we could make other kinds of requests: for example, we could fix the connectivity of each node; or even the distance between nodes, if we have previously defined a distance function on our network. The result would be another set, necessarily a strict subset of the previous one, since we added some constraints.

This kind of language may have already triggered a comparison in the mind of the reader. We are using terms such as "ensemble" and "constraint", which, combined with a certain knowledge of Physics, recall a whole different context: *Statistical Mechanics*. If we persist in this direction, we will see that this parallel is consistent. It gives us the opportunity to analyse the subject from a new perspective, and it can yield interesting results.

2.4 Statistical Mechanics and Entropy

Statistical Mechanics, also known as Statistical Thermodynamics, is the branch of Physics that studies the behaviour of a thermodynamic system, but with a perspective really different from the one of Classical Thermodynamics. It originates from the necessity of

explaining the meaning of thermodynamic quantities, such as *Temperature* and *Entropy*, from a microscopic point of view, making use of two tools: *Classical Mechanics laws* and *Statistical considerations* (thus the name "Statistical Mechanics").

A Statistical Mechanics system is composed by a great number of sub-units (e.g. particles, oscillators...) and is analysed under two points of view: the *microscopic* one, which accounts for the state of every single sub-unit, and a *macroscopic* one, which describes the global state of the system, without specifying the state of every single sub-unit. By *microstate* we thus mean a microscopic description of the state of our system. On the contrary, by *macrostate* we indicate a macroscopic description, which carries much less information than the microscopic one.

2.4.1 Ergodicity, Microstates and Macrostates

Our system is required to have a fundamental property, if we want a Statistical Mechanics analysis to be effective over it.
This property is *Ergodicity*. It is generally caused by frequent and random[6] collisions between the sub-units of the system, which are responsible for sudden changes of the microstate of the system. If a system is ergodic, then there is no privileged microstate amongst all the ones accessible by the system.[7] If we were somehow able to measure the microstate of our system at regular intervals of time, we would find a uniform distribution over all the possible microstates.

Before proceeding into further considerations we have to state the nature of what we called "macrostates", and of Classical Thermodynamics measures. These measures are what allow us to define a macrostate. From the microscopic point of view, they are but average measure of microscopic quantities in our system. They involve a consistent loss of information on our system, compared to the measure of every microscopic quantity. In fact, given the set of all possible microstates Σ and the set of all possible macrostates Γ of our system, there is always a function $f : \Sigma \to \Gamma$. This function is not injective in general though. So, given a macrostate, we are not able to know in which microstate

[6]They are not actually "random", since they are often governed by exact laws (e.g. the laws of Mechanics), but their overall effect on the system is comparable to sudden random transfers of energy from one sub-unit to the other.

[7]This is actually an over-simplification. To be more precise, ergodicity states that if we map the state of our system in a phase space, after a long period of time the probability of finding our system in any subregion of the accessible phase space is proportional to the integral measure of the subregion. Combined with Liouville's theorem, this property allow us to transform average quantities evaluated on the trajectory of the system in the phase space, in average quantities evaluated on the accessible phase space itself.

our system is. But we can make an interesting consideration.

While every accessible microstate is equiprobable with respect to the others, this is no more true for macrostates: when we measure the macrostate of our system, it is more likely the measure results in a macrostate with thousands of microstates mapping on it, than resulting in a macrostate realized by just a dozen microstates. This is why the equilibrium state of the system is the one that maximizes Boltzmann's Entropy, in a sense that will become clear later. In fact, because of the great number of sub-units present in our system, there is in general a macrostate which is by far more probable than any other possible macrostate.

But what do we mean by "accessible states"? In Statistical Mechanics systems are often subject to constraints. These constraints limit the possible states of the system to a particular subset of the phase space. We shall give a short description of the different categories of constraints.

2.4.2 Categories of constraints

Three different kinds of constraints are used in Statistical Mechanics, distinguished according to the system's number of sub-units and total energy:

Microcanonical Constraint This kind of constraint is the one to whom isolated thermodynamic systems are subject. Both the total number of sub-units N and the total energy E are constant over time.

Canonical Constraint This constraint is milder than the previous one, and characterizes systems subject to a thermal bath. The total number of sub-units N is still fixed, but this is no more true for the total energy E. What is maintained constant is the temperature T.

Grand Canonical Constraint This is the kind of constraint we refer to when we consider small systems put in contact with bigger systems, without any kind of barrier. The total energy E is not constant, nor the total number of sub-units N. What is invariant over time is the temperature T and the *chemical potential* μ.[8]

We shall now analyse a simple model to introduce the concept of Boltzmann's Entropy, and later use the same concept on networks.

[8]The chemical potential is defined as the *free energy* $\Phi = E - TS$ which we have to add to the system if we want to add a single sub-unit and maintain the temperature and entropy constant.

2.4.3 Boltzmann's Entropy: a simple model

Let us consider a system composed by N small sub-units - we shall call them *particles* - labelled with indices going from 1 to N. Each particle has an energy (we call e_j the energy of the j-th particle) belonging to an infinite discrete spectrum $\{\epsilon_i\}_{i\in\mathbb{N}}$. We identify the microstate of the system by providing a configuration vector $\vec{e} = (e_1, e_2, \ldots, e_N)$ containing the energy of every single particle. The discreteness of the spectrum of confined systems comes from quantum theory, and it is fundamental in our model.[9] Let us define the population of the i-th energy level by the number $n_i = \sum_{j=0}^{N} \delta(\epsilon_i - e_j)$[10], which is the number of particles having an energy equal to ϵ_i. The macrostate of the system is described by the population vector $\vec{n} = (n_1, n_2, \ldots)$ having infinite components.

Let us consider the case in which our system is subject to a microcanonical constraint, the total energy being equal to E, and the total number of particles to N. Because of these constraints, the population vector $\vec{n}$ and the configuration vector $\vec{e}$ are forced to belong to two particular subsets:

$$\vec{e} \in \Sigma\ , \qquad \Sigma = \left\{ \vec{e}\ \ s.t.\ \left(e_i \in \{\epsilon_i\}_{i\in\mathbb{N}}\ \ \forall i \in \{1,\ldots,N\} \right) \wedge \left(\sum_{i=1}^{N} e_i = E \right) \right\}$$

$$\vec{n} \in \Gamma\ , \qquad \Gamma = \left\{ \vec{n}\ \ s.t.\ \left(n_i \in \mathbb{N} \cup \{0\}\ \ \forall i \in \mathbb{N} \right) \wedge \left(\sum_{i=1}^{\infty} n_i = N \right) \wedge \left(\sum_{i=1}^{\infty} n_i\, \epsilon_i = E \right) \right\}$$

Both of these sets are necessarily finite. We can also map each microstate $\vec{e}$ to a macrostate $\vec{n}$ by a function $f\ :\ \Sigma \to \Gamma$, defined as follows: $f(\vec{e})_i = \sum_{j=0}^{N} \Big(4\ \theta(e_j - \epsilon_i)\ \theta(\epsilon_i - e_j) \Big) = n_i$. This function is in general not injective, so we can ask ourselves again: "how many microstates map on a given macrostate"? This time we can answer precisely to that question making combinatorial considerations.

Given a macrostate $\vec{n} \in \Gamma$ the number of microstates that map into it (we shall call this number $W_{boltz}(\vec{n})$) can be calculated as the product of the number of different ways we can shuffle the particles in each energy level, given by the combination of n_i elements chosen from a set of $N - n_1 - \ldots - n_i$:

$$\begin{aligned} W_{boltz}\,(\vec{n}) &= \binom{N}{n_1} \cdot \binom{N-n_1}{n_2} \cdot \binom{N-n_1-n_2}{n_3} \cdot \ldots \cdot \binom{N-\ldots-n_{k-1}}{n_k} \\ &= \frac{N!}{\prod_{i=1}^{k} n_k!} \end{aligned}$$

[9] A second quantum concept, namely *indistinguishability*, is not taken into account instead. According to quantum theory in fact, it is not possible to assign a label to each particle, and a switching of two particles in the same state can not be considered a switch of microstate. In this case, microstates would be identified by the vector $\vec{n}$.

[10] In this case $\delta(x)$ stands for the discrete delta function, whose value is 1 when $x = 0$ and 0 elsewhere.

Where n_k is the last non-null value of the infinite vector $\vec{n}$, which has to exist because of the condition imposed on the set Γ.

We suppose that our system is ergodic, and thus has the same probability to be in any microstate $\vec{e} \in \Sigma$. The probability $P(\vec{n}^\star)$ of finding our system in the macrostate $\vec{n}^\star \in \Gamma$ is therefore equal to:

$$P(\vec{n}^\star) = \frac{W_{boltz}\left(\vec{n}^\star\right)}{\sum_{\vec{n}\in\Gamma} W_{boltz}\left(\vec{n}\right)}$$

As a result, the most probable macrostate in which our system can be found is the one in Γ that maximizes the probability function $P(\vec{n})$. Or equally, the one which maximizes the function:

$$S_B(\vec{n}) = k_b \ \ln\left(W_{boltz}(\vec{n})\right) \tag{2.10}$$

Where k_b is Boltzmann constant, equal to $1.3806488 \times 10^{-23} J \cdot K^{-1}$.
This function is Boltzmann's definition of Entropy.

2.4.4 Gibbs' Entropy

W. Gibbs introduced a different definition of Entropy; even though it can be proved that it is consistent with Boltzmann's definition.[11] Instead of considering a single system which can be found in many states, Gibbs refers to a *Statistical Ensemble*, defined in the following way: let us consider a system, and sample the position of its state in the phase space at regular intervals of time. Doing so, we obtain a collection of n points of the phase space: $(s_1, s_2, \ldots, s_n)$. In order for us to apply Statistics on it, we will work in the limit $n \to \infty$.
Now, instead of considering these points as different states of the same system measured at different times, we consider them as states of different systems, measured in a single instant of time. The collection of these systems is called *Statistical Ensemble*.

For every microstate $\omega \in \Omega$ (Ω being the set of all possible microstates) we can build a statistic, and have the probability to pick a system in microstate ω in our ensemble. We shall call this probability $P(\omega)$. Given a macrostate A, we call $\Omega_A \subseteq \Omega$ the set of all possible microstates compatible with A. The *Gibbs Entropy* of macrostate A is now defined as:

$$S_G(A) = -k \sum_{\omega\in\Omega_A} P(\omega) \log P(\omega) \tag{2.11}$$

In Gibbs Entropy we do not require the equiprobability of all microstates,[12] but by imposing it we obtain Boltzmann's formula. In fact, if we consider a system where all

[11] Cf. [8]
[12] Or equally, we do not require the ergodicity of the system.

microstates are equally alike, and also there is one major macrostate A to whom most of them map, for which is valid $W_A = \mathbf{card}(\Omega_A) \approx \mathbf{card}(\Omega) = W$,[13] then:

$$\left(\forall\ \omega \in \Omega \quad P(\omega) = \frac{1}{W}\right) \quad \Rightarrow$$

$$S_G(A) = -k \sum_{i=1}^{W_A} \frac{1}{W} \log \frac{1}{W} = -k \frac{W_A}{W} \log \frac{1}{W} \approx k \log(W_A) = S_B(A) \quad (2.12)$$

2.4.5 Entropy in information theory

The last definition of Entropy we will analyse is *Shannon Entropy*. It was introduced by C. E. Shannon (1948) in a whole different context, but it presents evident similarities with Gibbs' formulation.
Let us consider a communication channel, in which a source $\mathcal{S}$ transmits *information* to a receiver, in form of a sequence of symbols belonging to a finite alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$, N being the cardinality of the alphabet. Each symbol appears in the sequence with a particular frequency, thus we are able to associate it an occurrence probability, that is a probability that the symbol is the next one transmitted in the sequence by the source. We name p_i the probability of the i-th symbol. These probabilities obviously have to satisfy the equation $\sum_{a \in \mathcal{A}} p(a) = 1$.

Shannon's aim is to find a function $H(\mathcal{S})$ capable of measuring the amount of *information* the source $\mathcal{S}$ produces, or better, the rate at which the information is produced. This has to be done with the only knowledge of the occurrence probabilities of each symbol. Shannon individuates three properties that have to be satisfied by such a function:

1. $H(\mathcal{S})$ should be continuous in the p_i
2. If all the p_i are equal, and thus $p_i = \frac{1}{N}$, $H(\mathcal{S})$ should be a monotonic increasing function of N. In fact, if the alphabet has more symbols, more information can be transmitted with messages of the same length.
3. The third property concerns an increase of complexity of an alphabeth: let us consider two sources, $\mathcal{S}$ and $\mathcal{S}'$. They are identical, if not for the fact that the symbol a_i in $\mathcal{S}$, with associated probability p_i, is replaced with a pair of symbols a'_i, a'_{i+1} in $\mathcal{S}'$, with associated probabilities p'_i and p'_{i+1}. It is also valid that, for what we have just stated, that $p'_i + p'_{i+1} = p_i$.
   We require that the increase of complexity causes an increase of the H function, according to the following equation: $H(\mathcal{S}') = H(\mathcal{S}) + p_i\ H(\mathcal{S}' \setminus \mathcal{S})$, where $\mathcal{S}' \setminus \mathcal{S}$ is the source for whom $\mathcal{A} = \{a'_i, a'_{i+1}\}$ and the respective probabilities are $(\frac{p'_i}{p_i}, \frac{p'_{i+1}}{p_i})$.

[13] This second assumption can be showed to be valid also in the previous Boltzmann Entropy model in the limit $N \to \infty$.

It can be proved that the only function which can satisfy all these requests is the function:[14]

$$S_S = H(\mathcal{S}) = -k \sum_{a \in \mathcal{A}} p(a) \log p(a) \tag{2.13}$$

Where k is a positive constant.

This function is called *Shannon Entropy*, named this way after the similarity with the Statistical Mechanics concept.

2.5 Statistical Mechanics of networks

Finally, it is time to analyse the parallel we so far just outlined, between Network Ensembles and Statistical Ensembles. Here we introduce the fundamental idea that justifies every further analysis. In section 2.3.2 we introduced considerations on a real network being the result of a process of evolution, and being a particular instance of a wider set of possibilities. This idea and its consequences have been examined by G. Bianconi in the article "*the entropy of randomized network ensembles*".[15] In this article the author states that:

> "Every real network can be considered as a specific instance of a particular network evolution compatible to its functional constraints... We propose here to consider a real network as belonging to an *ensemble of networks* which would perform the same task equally well."

This is especially true in biology, where we can observe a certain variability of biological networks performing the same function across different species. Therefore, let us suppose that a real network is a particular instance belonging to an *ensemble of networks*. What are the common features of all the elements in our ensemble? Can we somehow infer them from the only information we have: the real network?

It is reasonable to think that every other possible network has to be able to perform the task the real network performs equally well, but we do not know precisely what features are essential in order for this to happen. We can proceed by successive approximations, adding every time a further constraint, to make all the graphs in our ensemble more and more similar to the real graph. The result of this process is an increasing complexity of the networks in our ensemble, and a decrease of the variability, that is the

[14] The choice of a base for the logarithm corresponds to a choice of a unit to measure information. There are two conventional choices: either base 2 is used, the unit of measure being called *bit*, meaning "binary digit"; or base e, ant the corresponding unit *natural unit*.

[15] Cf. [4]

number of network in the ensemble.

Now the parallel with Statistical Mechanics suggest that every network belonging to the ensemble represents a microstate, and the whole ensemble groups all the networks mapping in the same macrostate. With these considerations we can easily define a measure of Entropy of the network ensemble. We will use the words of G.Bianconi, from the same article:[16]

> The entropy of a given network ensemble is proportional to the logarithm of the number of networks belonging to the ensemble. We expect that a very complex network is belonging to an ensemble of functionally equivalent networks of small entropy. Since it is difficult to characterize the minimal entropy ensemble a real network belongs to, we take successive approximations of the real network.

By measuring the decrease of entropy caused by the addition of a constraint, we have a measure of the "strength" of the constraint, the information coded into it, and somehow its importance.
There are two main approaches we can follow, according to the strength of the constraints we require the network in our ensemble to satisfy: the *microcanonical approach* and the *canonical approach*.[17] In both of these approaches the number of nodes in the network corresponds to the number of sub-units in the thermodynamical system, and the other constraints, such as the number of links, are assimilated to the energetic constraint.

2.5.1 Microcanonical Ensemble of Networks

In this first kind of approach, we consider a *microcanonical ensemble of networks*, that is to say an ensemble of networks each of whom has a definite number of nodes N, and which precisely satisfies all the constraints we require it to satisfy. We shall refer to this ensemble as E_{mic}. From what we stated in the previous section, it follows that the definition of entropy Σ of the microcanonical ensemble is:

$$\Sigma(E_{mic}) = \log\left(\mathbf{card}(E_{mic})\right) \tag{2.14}$$

Which is similar to Boltzmann's definition of Entropy.
In Statistical Mechanics this kind of approach requires that the number of sub-units is constant. Our microcanonical ensemble must thus contain only networks with a fixed total number of nodes N. Also the total energy must be exactly the same for all microstates. We can translate this request in many ways. In the first example, which is

[16] As before, cf. [4]

[17] Also a *grand canonical approach* is possible, in which the number of nodes in the network can vary, but it will not be examined in this paper.

also the simpliest one, we shall assimilate the total energy to the total number of links L.

Under these conditions, our microcanonical ensemble E_{mic} is exactly the set $G(N, L)$ which we presented in section 2.3.1. We already showed the cardinality of this set is $\mathbf{card}(G(N, L)) = \binom{\binom{N}{2}}{L}$ and so the entropy of the ensemble is:

$$\Sigma = \log\left(\mathbf{card}(E_{mic})\right) = \log\left(\binom{\binom{N}{2}}{L}\right)$$

This is the simplest model we can present. The next constraint we can impose on our ensemble is the *degree sequence*: we can reduce our ensemble to only the networks with a given *degree sequence* $\vec{k}^{\star}$.[18] This constraint presents a considerable difference with Statistical Mechanics though, in fact it is *extensive*. The system has to satisfy N equations: $\forall\ i\ \in\ \{1, \ldots, N\}\ \ k_i = k_i^{\star}$.
The evaluation of the number of networks that exactly satisfy this constraint is a complicated matter. We can approximate it in the large limit network ($N \to \infty$), building the *partition function* of the ensemble; but we will not tackle this particular problem in our work.[19]

2.5.2 Canonical Approach

In this last kind of approach, the requests of the constraint are milder: the number of nodes N is fixed, as in microcanonical approach, but the other conditions only have to be satisfied *on average* in our canonical ensemble E_{can}. This means that, for example, if we request the total number of nodes to be $L^{\star}$ on average, it has to be true that:

$$\langle L \rangle = \sum_{\mathcal{G} \in E_{can}} \frac{L(\mathcal{G})}{\mathbf{card}(E_{can})} = L^{\star} \tag{2.15}$$

Where by $\mathcal{G}$ we indicate a particular graph in our ensemble, and $L(\mathcal{G})$ is the total number of links of that graph. The total number of links can be replaced by any measurable property on our graph $X(\mathcal{G})$ and the previous definition would still hold. Basically, even networks that do not satisfy the requested properties can be part of the ensemble, if the properties are still satisfied on average by all the networks.

Unfortunately, there are often multiple ways to realize such an ensemble. We shall limit this freedom of choice with two possible alternative descriptions. In the first, instead of defining the canonical ensemble as a set of equiprobable networks, we use an approach similar to the one used by Gibbs. We define our ensemble E_{can} as the set of all possible

[18] For the definition of *degree sequence*, see section 2.2.3
[19] For a solution to this problem, cf. [3].

networks with N nodes, and assign each of them a probability $P(\mathcal{G})$ of being the one real network we consider. This results in a probability distribution over our ensemble. Obviously it has to be true that $\sum_{\mathcal{G}\in E_{can}} P(\mathcal{G}) = 1$. Then the constraints are satisfied on average if, taking any measure $X(\mathcal{G})$ on a graph whose average value has to be $X^\star$:

$$\langle X\rangle = \sum_{\mathcal{G}\in E_{can}} P(\mathcal{G})\ X(\mathcal{G}) = X^\star \tag{2.16}$$

And Entropy is defined, in accordance with Gibbs' approach, as:

$$\Sigma = -\sum_{\mathcal{G}\in E_{can}} P(\mathcal{G}) \log P(\mathcal{G}) \tag{2.17}$$

This still leaves us with a considerable amount of freedom. In the second description we define our ensemble in a very different way, starting from a single network having N nodes. We assign to any possible edge a *link-probability*. We shall call p_{ij} the probability of a link between node i and node j. This results in a link-probability matrix P, which has to be symmetrical ($P^T = P$) since we consider undirected graphs; moreover all the elements on the diagonal must have null value ($p_{ii} = 0\ \ \forall\ i\ \in\ \{1,\dots,N\}$).
In this second description, we impose the constraints in the following way. We consider a measure on the network $X(A)$, acting on the adjacency matrix A. We ask the average value of this measure to be equal to $X^\star$ in our ensemble, by imposing the following equation:

$$\langle X\rangle = X(P) = X^\star \tag{2.18}$$

For example, if we requested the total number of links to be equal to $L^\star$ as in the previous example, we could write:

$$\langle L\rangle = \sum_{i<j} p_{ij} = L^\star \tag{2.19}$$

Entropy, in this last kind of approach, is defined as:

$$\Sigma = -\sum_{i<j} \left(p_{ij}\log p_{ij} + (1-p_{ij})\log(1-p_{ij})\right) \tag{2.20}$$

This definition presents evident similarity with Shannon Entropy. In fact we are dealing with a two-bits alphabet: the element of the adjacency matrix a_{ij} takes the value 1 with probability p_{ij}, and the value 0 with probability $1 - p_{ij}$. We have to consider both of these values when we calculate entropy.
We can turn this in the previous description, if we consider that for any graph $\mathcal{G}$ having an adjacency matrix A:

$$P(G) = \prod_{i<j} p_{ij}^{a_{ij}} (1-p_{ij})^{|1-a_{ij}|} \tag{2.21}$$

This second kind of approach, involving the probability matrix P, will be the one used in the next chapter. We will describe and implement an algorithm capable of evaluating the entropy of a canonical network ensemble that satisfies particular constraints extracted from a real network.

Chapter 3

The Algorithm

This chapter is devoted to the description of the algorithm used to analyse the biological data. Starting from a real network, the algorithm can evaluate the entropy of the corresponding maximum-entropy canonical ensemble. This quantity can be an indicator of the strictness of the constraints that act on our network. By comparing entropies of networks coming from different samples, for example old and young samples, or healthy and ill samples, we can try to understand how a given illness, or simply old age, act on the constraints of our biological network.
But why, amongst all possible network ensembles that satisfy the same constraints our real network satisfies, are we looking for the one with maximum entropy? First of all, because this is the *biggest* ensemble we can consider, and thus the most probable. Secondly, it is the ensemble created with the minimum number of hypotheses, just the ones contained in our constraints.
We shall begin by describing the mathematical procedure we will use, and then comment on its implementation in *C++* language. A *Matlab* version of the algorithm was already available, developed by G.Bianconi et al.[1] and subsequently modified by G. Menichetti in her Master Degree graduation paper[2]. *Matlab* is an high-level language, and in spite of being easier to write instructions into code, the compiled result is not always optimized, and this slows the computation when many iterative cycles are present. We considerably enhanced the algorithm performance by implementing the algorithm in *C++* language, and especially by employing parallel computation. This allows us to process more extensive dataset than it was previously possible in a reasonable amount of time.

[1] G. Bianconi, P. Pin and M. Marsili: *Assessing the relevance of node features for network structure*, Proceedings of the National Academy of Science 106, 11433 (2009).

[2] Cf. [11]

3.1 Mathematical framework

The first step we shall do describe our algorithm is illustrating to the reader the mathematical procedure we make use of in the code. Starting from the data we have at our disposal concerning a particular real network, we try to define precisely how to extract topological and spatial constraints from it. Then we associate to these constraints a set of canonical network ensembles. Finally, we shall find a way to select amongst all the possible ensembles the one with the maximum entropy. Once this will have been done, the reader will have a rather precise idea of how the algorithm works.

3.1.1 Input data

The data at our disposal, as we have already stated, have biological origin. They are:

1. The *protein-protein interaction network* of human cells, common to all the samples.[3]
2. The gene-expression profile of a sample.[4]

The first kind of data provides us with a network containing N nodes, each one representing a protein. The edges in the network are associated with an interaction between the two proteins. The second kind of data is a vector containing N positive real numbers, each one representing the level of *expression* of the gene coding for that protein. Each number is proportional to the logarithm of the number of molecules of the corresponding protein, within some inevitable experimental error.
The data will thus be provided to the algorithm in the form of:

1. An adjacency matrix $A = \{a_{ij}\}_{i,j\in\{1,\dots,N\}}$.
2. A vector $\vec{g} = (g_1, \dots, g_N)$ containing the level of expression of each gene.

This second kind of data is immediately processed by the algorithm, and a distance matrix D is extrapolated from it in the following way: we define the distance d_{ij} between node i and node j as

$$d_{ij} = |g_i - g_j|$$

This is by all means a distance function, and satisfies all the properties listed in section 2.2.4.[5]

[3] In particular, cf. section 1.3.2. This data was retrieved from `www.pathwaycommons.org`.

[4] These data come from the analysis of liver cells, extracted from organs destined to transplant. The measure was carried out with the DNA micro-array technique, as described in section 1.3.1

[5] We will use a distance matrix, but this is not strictly necessary. Even a matrix whose members does not satisfy the *triangular inequality* is fit for the same kind of analysis, if it represents some property of the network.

3.1.2 Requested constraints

We use this data to impose two different kinds of constraints on our network:

- The connectivity degree of each node.
- The number of nodes in a certain interval distance.

Let us translate this constraints in the language of canonical network ensembles. The connectivity of our real network is extracted from the adjacency matrix, as illustrated in section 2.2.3. The result is a vector $\vec{k} = (k_1, \ldots, k_N)$. If our ensemble is described by the probability matrix P, we impose this constraint on our network, as described in section 2.2.3, by requesting the following N equations to be satisfied:

$$\forall i \in \{1, \ldots, N\} \quad , \quad \sum_{j=1}^{N} p_{ij} = k_i \tag{3.1}$$

As for the constraint on the distance matrix, we proceed as follows. We find the maximum element d_{max} and the minimum non-null element d_{min} of the distance matrix D. Then we divide the interval $[d_{min}; d_{max}]$ in a number B of bins,[6] defining a vector $\vec{\lambda} = (\lambda_1 = d_{min}, \lambda_2, \ldots, \lambda_{B+1} = d_{max})$ containing the edges of the bins.[7] We call $\chi_l(x)$ the characteristic function of the l-th bin, defined as follows:

$$\chi_l(x) = \begin{cases} 1 & \text{if } x \in [\lambda_l; \lambda_{l+1}] \\ 0 & \text{if } x \notin [\lambda_l; \lambda_{l+1}] \end{cases} \tag{3.2}$$

Once the binning operation is completed, we count the number of links present in each bin, obtaining a bin-population vector $\vec{b} = (b_1, \ldots, b_B)$. The l-th element of the vector is evaluated as follows:

$$b_l = \sum_{i<j} (\chi_l(d_{ij}) \cdot a_{ij}) \tag{3.3}$$

Finally, we are able to impose the distance constraint, requesting that the average population of each distance bin is equal to the bin-population of the real network. This results in the following set of B equations.

$$\forall l \in \{1, \ldots, B\} \quad , \quad \sum_{i<j} (\chi_l(d_{ij}) \cdot p_{ij}) = b_l \tag{3.4}$$

[6] Usually a linear binning is used, but for particular distance distributions a logarithmic binning could be more suitable. In our analysis we will test them both, to compare the effect of such a choice on the total entropy value.

[7] Obviously, $\lambda_1 < \lambda_2 < \ldots < \lambda_{B+1}$

3.1.3 Maximization of Entropy

Entropy can be easily evaluated knowing the matrix P. But still, the matrix has $\frac{N(N-1)}{2}$ degrees of freedom, and we imposed a total of just $N+B$ equations. How can we evaluate the p_{ij}?
We are seeking the maximum-entropy ensemble, that is to say, the one ensemble of network amongst all the ones satisfying the constraints, that has the maximum entropy. The easiest way to find the parameters that maximize the values of a function, and are subject to constraints, is using *Lagrange multipliers*.

$$\begin{cases} \Sigma(P) = -\sum_{i<j} \left(p_{ij} \log p_{ij} + (1-p_{ij}) \log(1-p_{ij})\right) & \text{Entropy function, to be maximized} \\ \forall i \in \{1,\dots,N\} \quad , \quad \sum_{j=1}^{N} p_{ij} = k_i & \text{Connectivity constraints} \\ \forall l \in \{1,\dots,B\} \quad , \quad \sum_{i<j} \left(\chi_l(d_{ij}) \cdot p_{ij}\right) = b_l & \text{Spatial constraints} \end{cases}$$

We thus write the Lagrangian function:

$$\Lambda(P) = \Sigma(P) + \sum_{i=0}^{N} \left[\alpha_i \left(\sum_{j=1}^{N} p_{ij} - k_i\right)\right] + \sum_{l=0}^{B} \left[\beta_l \left(\sum_{i<j} \left(\chi_l(d_{ij}) \cdot p_{ij}\right) - b_l\right)\right] \tag{3.5}$$

The variables $\alpha_i,\ i \in \{1,\dots,N\}$ being the connectivity Lagrange multipliers, and the $\beta_l,\ l \in \{1,\dots,B\}$ being the spatial Lagrange multipliers.

We can estimate the value of the elements of P and of the Lagrange multipliers by imposing the constraints, plus the set of $\frac{N(N-1)}{2}$ equations:

$$\forall\ i,j \in \{1,\dots,N\}\ ,\ i<j \qquad \frac{\partial \Lambda}{\partial p_{ij}} = 0 \tag{3.6}$$

Which translates into:

$$\forall\ i,j \in \{1,\dots,N\}\ ,\ i<j \qquad \log\frac{p_{ij}}{1-p_{ij}} + \alpha_i + \alpha_j + \sum_{l=1}^{B} \chi(d_{ij})\beta_l = 0 \tag{3.7}$$

We isolate the p_{ij} from each equation:

$$p_{ij} = \frac{e^{-\left(\alpha_i+\alpha_j+\sum_{l=1}^{B}\chi_l(d_{ij})\beta_l\right)}}{1+e^{-\left(\alpha_i+\alpha_j+\sum_{l=1}^{B}\chi_l(d_{ij})\beta_l\right)}} = \sum_{l=0}^{B}\chi_l(d_{ij})\frac{e^{-(\alpha_i+\alpha_j+\beta_l)}}{1+e^{-(\alpha_i+\alpha_j+\beta_l)}} = \sum_{l=0}^{B}\chi_l(d_{ij})\frac{z_i z_j w_l}{1+z_i z_j w_l} \tag{3.8}$$

In the last member of the equation we considered $z_i = e^{-\alpha_i}$ and $w_l = e^{-\beta_l}$. We shall call these values the *generalized Lagrange multipliers*.

3.1.4 Iterative process for the calculation of Lagrange multipliers

The fastest way to solve such a complex system of equations is the iterative algorithm proposed by G.Menichetti,[8] to calculate iteratively the values of the generalized Lagrange multipliers.
We start from the sets of equations 3.1 and 3.4, and substitute the expression of the p_{ij} obtained in equation 3.8:

$$k_i = \sum_{j=1}^{N} \left[\sum_{l=0}^{B} \chi_l(d_{ij}) \frac{z_i z_j w_l}{1 + z_i z_j w_l} \right] = \sum_{j \neq i} \frac{z_i z_j e^{\sum_l \chi_l(d_{ij}) g_l}}{1 + z_i z_j e^{\sum_l \chi_l(d_{ij}) g_l}} \tag{3.9}$$

$$b_l = \sum_{i<j} \left[\chi_l(d_{ij}) \cdot \sum_{t=0}^{B} \chi_t(d_{ij}) \frac{z_i z_j w_t}{1 + z_i z_j w_t} \right] = \sum_{i<j} \chi_l(d_{ij}) \frac{z_i z_j w_l}{1 + z_i z_j w_l} \tag{3.10}$$

Now we can easily isolate z_i in the first equation, and w_l in the second, obtaining the expressions:

$$z_i = \frac{k_i}{\sum_{j \neq i} \frac{z_j e^{\sum_l \chi_l(d_{ij}) g_l}}{1 + z_i z_j e^{\sum_l \chi_l(d_{ij}) g_l}}} \tag{3.11}$$

$$w_l = \frac{b_l}{\sum_{i<j} \chi_l(d_{ij}) \frac{z_i z_j}{1 + z_i z_j w_l}} \tag{3.12}$$

These last two equations are the fundamental equations for the iterative cycle. Let us suppose to have an approximation of the correct Lagrange multipliers in the vectors $\vec{z}_{old}$, $\vec{w}_{old}$. By making use of the previous equations we write:

$$z_i^{new} = \frac{k_i}{\sum_{j \neq i} \frac{z_j^{old} e^{\sum_l \chi_l(d_{ij}) g_l^{old}}}{1 + z_i^{old} z_j^{old} e^{\sum_l \chi_l(d_{ij}) g_l^{old}}}} \tag{3.13}$$

$$w_l^{new} = \frac{b_l}{\sum_{i<j} \chi_l(d_{ij}) \frac{z_i^{old} z_j^{old}}{1 + z_i^{old} z_j^{old} w_l^{old}}} \tag{3.14}$$

The two resulting vectors $\vec{z}_{new}$, $\vec{w}_{new}$ are a better approximation to the real values of the Lagrange multipliers $\vec{z}$, $\vec{w}$. Our algorithm iterates these successive approximations, improving the precision of the solution. The iteration ends when both the *Chebyshev distances*[9] $d_{Cheb}(\vec{z}_{new}, \vec{z}_{old})$ and $d_{Cheb}(\vec{z}_{new}, \vec{z}_{old})$ are smaller than a *threshold precision*

[8] Cf. [11].

[9] The *Chebyshev distances* between two vectors is equal to the maximum distance between every single component: $d_{Cheb}(\vec{a}, \vec{b}) = \max_i\{|a_i - b_i|\}$

p_{tresh}. The smaller this value, the bigger the number of iterations required to reach the desired precision.
This number can be diminished if we adopt a good *starting guess* for our Lagrange multipliers. Our choice is obtained from the following considerations. From equation 3.8 we approximate:

$$p_{ij} \approx \sum_t \chi_t(d_{ij}) z_i z_j w_t = z_i z_j w_l \tag{3.15}$$

And also, from 2.5 and 2.7 we have:

$$\sum_{i,j} p_{ij} = N\langle k \rangle \tag{3.16}$$

Then a good choice for our starting guess is:

$$z_i \approx \frac{k_i}{\sqrt{N\langle k \rangle}} \tag{3.17}$$

$$w_l \approx \frac{b_l}{N\langle k \rangle} \tag{3.18}$$

After the computation of the Lagrange multipliers has been accomplished, the program calculates the elements of the matrix P using equation 3.8. Finally, the Entropy of the ensemble is evaluated through equation 2.20.

3.2 Implementation of the Algorithm

The reader should now have a precise idea of what the algorithm does: it Now that we have described the theory that lies under the algorithm and the operations we need to perform, we can concentrate on more practical matters. In particular on three main aspects of the implementation of our algorithm in *C++* language: input-output, memory allocation, parallelization and customization.

3.2.1 Input-output

The first aspect we shall analyse is the input and output of data. The program acquires input data from data files, and also the output is written on a file, in order for it to be available for further data elaboration. Only the main information about the network and the computation status is printed on the terminal. The format we choose for the input and output is text format (i.e. files whose extension is `.txt`). This has both advantages and disadvantages. For example, text format files occupy more memory than binary format files, and they sometimes generate compatibility issues. However, it is the most suitable format for our aim, since the data we analysed were extracted from databases

and pre-processed with Matlab, and the result could easily be exported in text format. Moreover, since the results of the calculation were to be read by the user, or processed with high-level languages such as Python to plot graphics, the text format was suitable also for output files.

The source code is customizable, and the user can chose which of the following information are evaluated and eventually displayed in the output file:

- General information on the analysed network, such as: total number of nodes in the network, total number of links, average connectivity, minimum and maximum value of distance between nodes, total number of bins and population per bin.
- Precision reached in the approximation of the Lagrange multipliers at every iteration of the cycle.
- Total number of iterations required to reach the desired precision.
- Final values of the Lagrange multipliers.
- Values of the elements of the probability matrix P.
- Amount of time employed to accomplish the computation.
- Entropy of the network ensemble.
- *Single Node Entropy* of every node of the ensemble.[10]

3.2.2 Memory Allocation

Memory management represented a delicate issue in the implementation of the algorithm for two main reasons. The first was the flexibility in the input data. We want the program to evaluate the size of the network we are providing in the input file, and allocate the necessary memory consequently. In *C++* language this issue can be easily solved by a wise use of pointers.

The second issue concerns memory usage. In fact we want to enhance the performance of our algorithm and enable it to handle big datasets. This is necessary when we handle high-throughput data. In fact the size of the networks we want to analyse is about 10^4 nodes. This translates into 10^8 distance elements into the distance matrix, coded into *double* variables, each of whom occupies 8 bytes. The total size of such a matrix would

[10]The *Single Node Entropy* measure is a local measure of entropy, that concern a single node of the network. It can be easily evaluated from the probability matrix P and the connectivity vector $\vec{k}$. We will not describe this measure further.

be about 0.75Gb, which is of the same order of magnitude of an average RAM memory. Since we have to analyse more than one sample in the same run of the program, this could lead to a depletion of the available memory. In order for us to avoid this condition, we need to dis-allocate the memory occupied by data that are no longer necessary. Pointers can again be of help, and avoid the re-allocation of memory when specific functions are called.

3.2.3 Parallelization

The main improvement from the previous version of the code is the use of *parallel computation*. Through this technique, a list of operations can be split in more sub-lists - called *threads* - than can be performed independently. Each sub-list is executed from a different core. Thanks to the sharing of the computation load on more processors, the computation time can be reduced up to $\frac{1}{N.\text{ of cores}}$ the original time. It is not usually possible to reach this level of optimization because of the *threading time*, which is the time needed to split the main list in its sub-units. If this time is more than the time needed to perform the operations in a *thread*, the parallelization is not a suitable solution to diminish the computation time, in fact it has the opposite effect. For this reason we created a highly customizable source code, as we will later explain.

The parallelization has been performed making use of *OpenMP*,[11] an open-source library created to implement parallel calculation in *Fortran* and *C/C++* languages. OpenMP directives are really easy to use, and the parallelization of a program can be easily accomplished adding just a few lines in the right places. In particular, parallelization is usually really effective if performed on a cycle, on condition that the next iteration of the cycle is independent from the previous. In our algorithm, this is the case of the computation of the generalized Lagrange multipliers vectors $\vec{z}_{new}$ and $\vec{w}_{new}$ starting from $\vec{z}_{old}$ and $\vec{w}_{old}$. In fact the computation of each component can be carried out independently.[12] In the next chapter we will see that this parallelization causes a considerable improvement of the computation time on a multi-core machine.

3.2.4 Customization

Finally, one of the most important features of a program is its customization and versatility, its ability to satisfy different requests that the user may have. For example, this is achieved by the program being able to analyse networks of different size. But on a wider

[11] The library is available at `http://openmp.org/wp`

[12] On average, for the networks we will analyse the number of components of these vectors are respectively $N \approx 10^4$ and $B \approx 10^2$. In this case the computation load for each thread is enough to grant a beneficial parallelization.

level, this is realized in the source code. In fact, by just changing some numeric values in the initial pre-processor directives different versions of the programs can be created. The difference lies in three main points:

Output customization The user can select which of the values listed in section 3.2.1 are to be printed in output. This avoids unnecessary confusion in the output file, and waste of calculation efforts and time to evaluate quantities which are not requested by the user.

Data from input file / data generated from a toy model The second customization is the opportunity for the user to select, instead of the acquisition of data from a file, the creation of data according to a toy model. In this second case the user can select the size, number of nodes and maximum distance at which the nodes are distributed. The program then generates a random adjacency matrix and a random distance matrix, and proceeds then with the analysis. This can be useful to test the efficacy of the algorithm or of the parallelization in particular conditions (e.g. for network with a given connectivity).

Parallel / non-parallel Sometimes (e.g. the analysis of many particularly small networks, or the case in which the machine used has just one core) the parallelization is not a beneficial solution. If this is the case, the user can easily obtain a non-parallel version of the algorithm from the same source code. In this version minor improvements which were not compatible with the parallelization have been made on the code.

Chapter 4

Performance and Data Analysis

In this chapter we shall analyse the performance of our parallel algorithm running on a hpc machine, and compare it with the performance of a non-parallel version of the algorithm. We will see that the parallelization enables us to carry out the computations considerably faster, and on networks of bigger size than it was previously possible. Moreover, we will illustrate the results of an analysis of real biological data.

4.1 Performance on toy models

As we stated in the previous chapter, from the same source code we can easily obtain a version of the program which generates a random network and then carries out its analysis. Thanks to this program, we can test the performance of our algorithm on networks of different size.

4.1.1 Generation of random networks

In the random network generator, the user can select the number of nodes N and links L he wants the random network to have. He can also select the maximum distance d_{max} at which the nodes are distributed, and optionally, the number of bins B used to create the spatial constraint. If he does not specify this value, it is automatically selected equal to the square root of the number of links: $B = floor[\sqrt{L}]$.

The adjacency matrix A is generated starting from a null-values matrix, selecting randomly L different elements of the upper triangular matrix (the elements on the diagonal are excluded) and setting them to "1". Then the values of the lower triangular matrix are copied symmetrically from the upper one. As a result we have the same probability of obtaining any of the graphs belonging to the microcanonical ensemble $G(N, L)$.[1]. The

[1]This ensemble was described in section 2.3.1

distance matrix D is generated from a uniform distribution of the distance elements d_{ij} in the interval $[0; d_{max}]$. Since we do not require the elements to satisfy the triangular inequality, this does not necessarily result in a "distance matrix" strictly speaking. The analysis is effective nonetheless, in fact we already pointed out that its validity is extended to any matrix which somehow represents a property of the network, even if this property is not akin to distance.

At the same time, since the networks generated are random, we need to analyse more than one sample for each size, in order for us to get an average value of entropy. In figure 4.1.1 we plot the distribution of entropy values for 15'000 random networks having $N = 100$, $L = 200$.

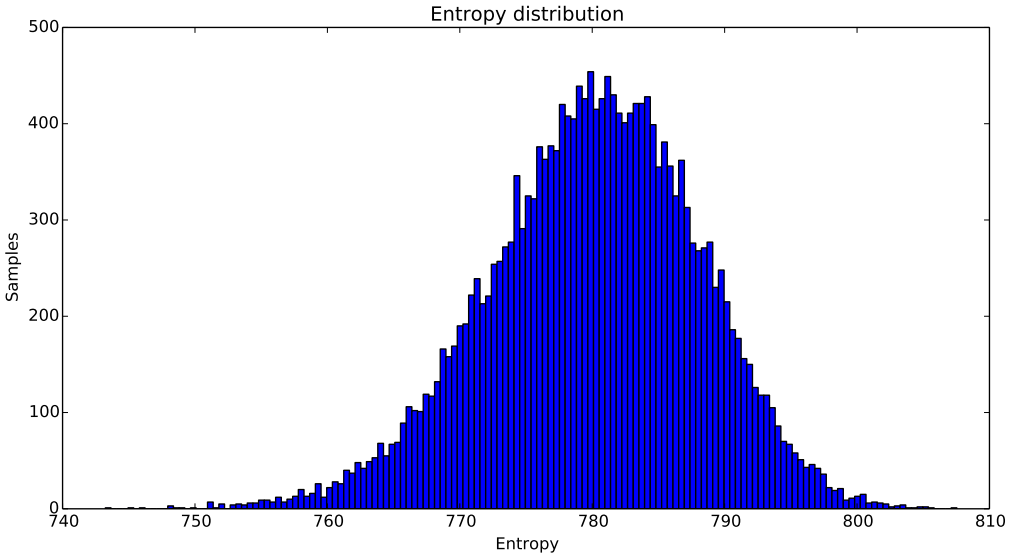


Figure 4.1: Entropy distribution

The standard deviation of this distribution is relatively bigger for networks of smaller size. This is due to the fact that constraints on smaller networks play a stronger role, they can be satisfied in a fewer number of ways, but with a relatively bigger difference of entropy. On the contrary, in bigger networks the effect of constraints is statistically lessened by the large number of nodes.

4.1.2 Precision and computation time

In order to test the performance of our program, we carried out the analysis of some sets of random networks in three different ways: with the non-parallel version of the algorithm (plotted in red), with the parallel version on a 8 cores machine (in green) and again with the parallel version on a 32 cores machine (in blue). We selected group of graphs with size varying from 10 to 20'000 nodes, but all of them having an average connectivity $\langle k \rangle = 2$.[2] In figure 4.2 we plotted the average entropy of different groups of networks. Each group is composed by elements of $G(N, 2N)$, as we just specified. We plot the size N of these groups of networks on the x-axis, and the relative entropy on the y-axis. The measures on both the axes are in logarithmic scale.

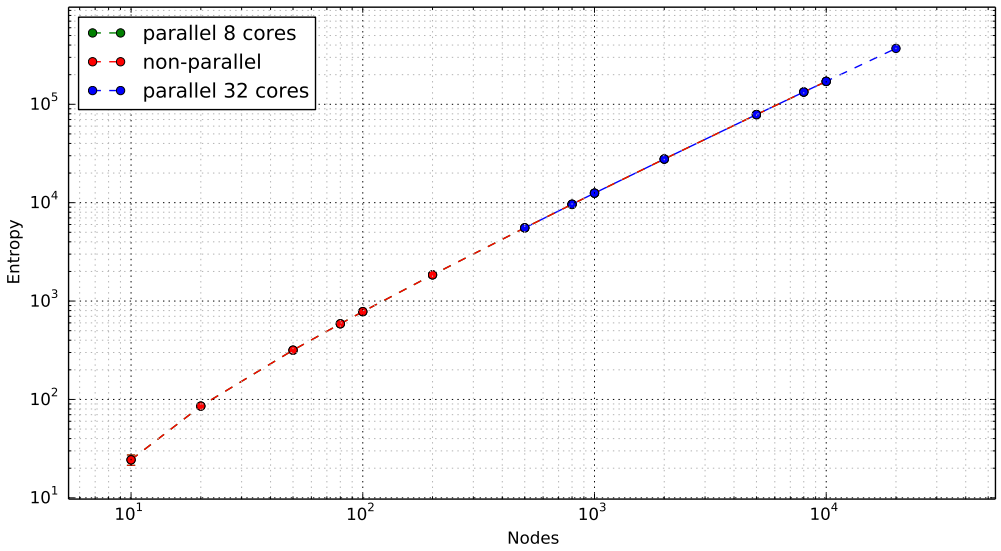


Figure 4.2: Entropy distribution of random networks generated by the program.

The results are the same for all the different versions of the program. This proves that they are equivalent, and there is not one more precise than the other. We also notice that entropy increases exponentially with the increase of the number of nodes of the network.

In figure 4.3 instead we plotted the value of the average convergence time for the different kinds of algorithms. The subject of the analysis were the same groups of random

[2]This means that for every graph the number of links is two times the number of nodes: $L = 2N$.

networks considered before. Again, both of the axes are in logarithmic scale.

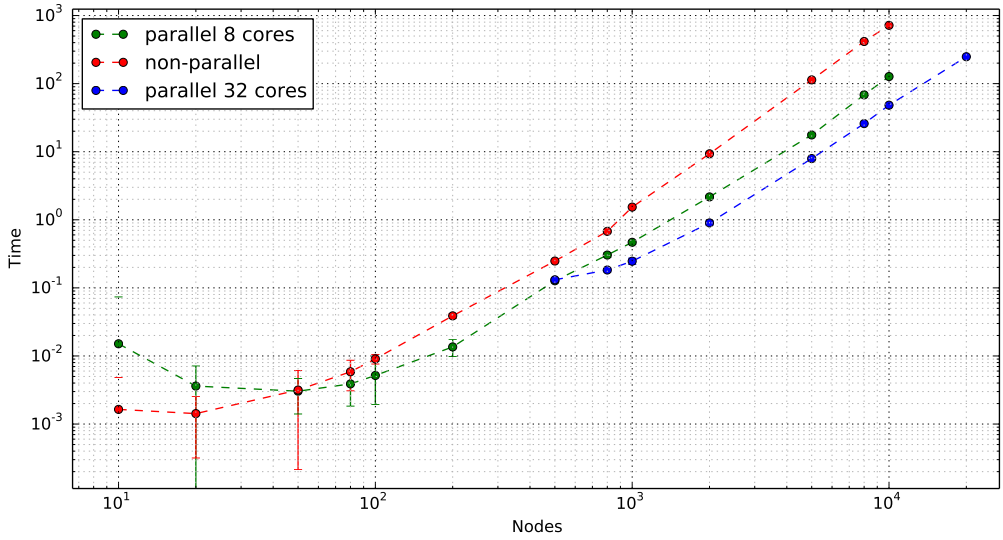


Figure 4.3: Average convergence time.

We notice there are two main regions in the plot, the first including the small networks, from 10 to 50 nodes, and the second including big networks, counting from 50 to 20'000 nodes. In the first region a non-parallel algorithm is the most effective solution for the analysis of networks in a short time.[3] In the second region the parallel algorithm is the fastest to compute the entropy of a network ensemble. All the more if it runs on a machine with a significant number of cores, as in the case of a 32-cores machine.
The bigger the network, the more substantial the difference between the computation time of the three versions. Since in the region going from networks with 2'000 nodes on, the three lines in the graph are parallel, we deduce that there is a constant ratio between the computation time of the different versions of the algorithm: the parallel version on a 8-cores machine is about 6 times faster than the non parallel version, while the parallel version on a 32-cores machine is more than one order of magnitude faster (about 14 times).
The result is that with a 32-cores machine and the parallel algorithm we can evaluate the entropy associated to a real network having 20'000 nodes (so we operate with matrices of

[3]The reader shall notice that by *a short time* we mean intervals of the order of 10^{-2} / 10^{-3} seconds. It is not really a big difference if we are to analyse a single network. It may be appreciable if we have to analyse many thousands of networks.

4×10^8 elements) faster than the time it takes for the non-parallel algorithm to compute the entropy of a network with just 8'000 nodes (matrices having 1.6×10^7 elements, 30 times smaller).
It is also important to notice that many different factors other than the number of nodes in the network can contribute to the increase or decrease of entropy and convergence time: factors such as average connectivity, required precision, number of spatial bins employed.

In conclusion, the performance improvement is relevant. It is what allowed us to perform the analysis of 32 real networks having about 10^4 nodes each in a little more than an hour, instead of it taking almost a whole day. The next section will be devoted to the description of the results of this analysis.

4.2 Data analysis

In this last section we will present the results of an analysis conducted on biological high-throughput data. The data, as we already stated, concern human liver cell, extracted from organs destined to transplant. The donors' age vary from a minimum of 13 to a maximum of 87 years. For each of the 32 samples we have the results of the gene expression profiling, describing the different expression levels of 9994 genes. The second kind of data is the protein-protein interaction network, which, since we refer to the same genes in all the samples, is common for all the networks.[4] This means that the topology of all the networks we are going to analyse is exactly the same. What differs from one sample to the other is the distance matrix D, obtained from the gene-expression profile of each sample as described is section 3.1.1. In particular, from this matrix we extract the spacial constraint that has to be satisfied by our canonical network ensemble, and we do so by dividing the distance interval in a number of bins, and counting the number of connections belonging to each bin. As a consequence, the resulting value of entropy should be somehow sensitive to the kind of binning we use.

In order to prove how stable this measure is, we used three kinds of binning: a canonical logarithmic binning, a logarithmic binning realized with a different exponent and a linear binning. In all the three cases we considered a total of 50 bins. In figure 4.4 we plot the number of connected nodes in each of the bins for the three different kinds of binning. This is referred to the first sample we will analyse, but the same trend is maintained also in the other samples.

We can see that the linear binning (plotted in green) has an elevated number of nodes in the first bins, while the last ones are empty. The opposite is valid for the classical log-

[4]For the meaning of these two kinds of data, see section 1.3.1 and 1.3.2.

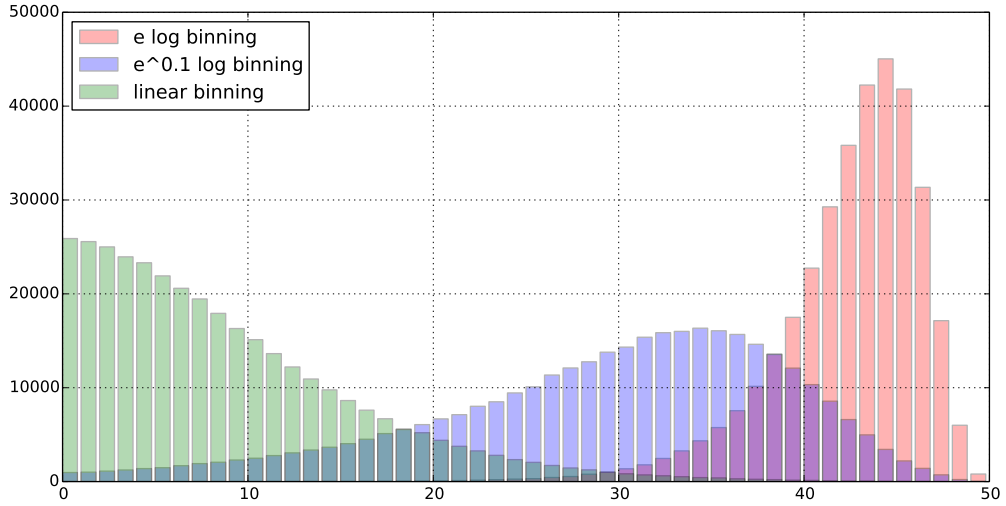


Figure 4.4: Connections per bin for the three different kinds of binning.

arithmic binning (in red). That is why we chose a third binning, which is a logarithmic binning (in blue), but realized with a different base than the natural one: we used the value $e^{0.1}$. In this last kind of binning the nodes are distributed more evenly.
An empty bin is a strong constraint on the probability matrix, in fact it forces all the elements of the matrix that are referred to couple of nodes whose distance would be in that bin, to be equal to zero.

We performed the analysis of all the 32 samples with the three kinds of binning. The average time interval required for the analysis of a network of this size is about 170 seconds. As a result, the total time necessary to carry out the three analysis is about 4 hours and an half. If the considerations we made in the previous section about the decrease of the computation time in the parallel version still holds, then we would have had to wait a total of three days in order for the computation to finish. That is a considerable gain. In figure 4.5 we plot the results of the computation for all the three different analysis. On the x-axis we plotted the age of the donor from which the sample comes, and on the y-axis the relative entropy.

We notice that the three analysis are consistent with each other. Differences in the binning choice cause a global increase or decrease of entropy, but the trend across the sample is constant in all three kinds of analysis. To quantify this similarity between the measures, we calculated the correlation matrix of the three vectors containing the

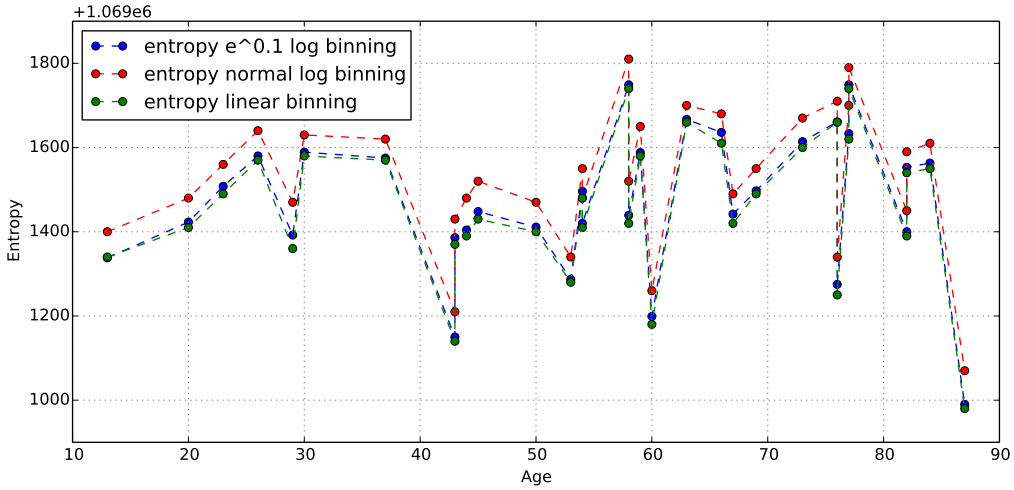


Figure 4.5: Entropy vs Age.

entropy evaluated with the three different binning choices:

$$C = \begin{pmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0.99788294 & 0.99644456 \\ 0.99788294 & 1 & 0.99917003 \\ 0.99644456 & 0.99917003 & 1 \end{pmatrix} \tag{4.1}$$

Where the binning choice are ordered as follows: first the logaritmic binning, then the logarithmic binning with a different exponent factor, and finally the linear binning. And $\rho_{ij} = \frac{\mathbf{cov}(\vec{x}_i,\vec{x}_j)}{\sigma(\vec{x}_i)\sigma(\vec{x}_j)}$ is the Pearson correlation coefficient. The correlation between the three measures is really strong. This means our measure is stable, and does not depend too much on the binning we use.
We can also plot the difference between the entropy values obtained considering the three different kinds of binning, to see if they actually behave similarly. In figure 4.6 we show these differences. Each series of values is correlated with a line representing the mean value, and an highlighted area. Points in this area are less than a standard deviation away from the mean value.

From the analysis of the figure we conclude that the three different entropy measures are essentially alike. The difference between them is represented by an offset.

We stress again the fact that the differences of entropy from one sample to the other are not caused by the network topology, since the adjacency matrix (extracted from the PPI matrix) is the same for all the samples, nor from the choice of the binning, since we showed that the three choices are consistent with each other. The only value that

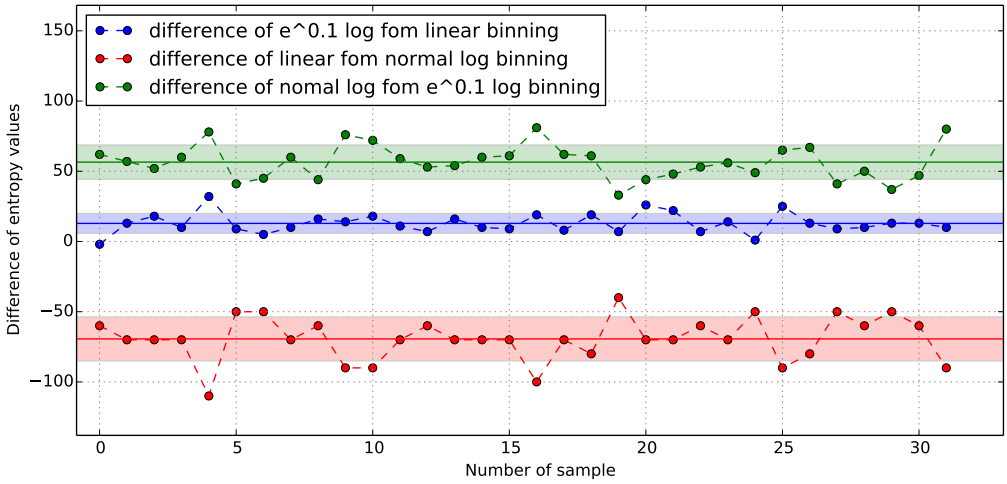


Figure 4.6: Difference between entropy values.

changes, and causes the variation of entropy from one sample to the other, is the *gene-expression profile* of each sample. This is what our algorithm highlights.

The last thing we can notice is the ratio between the magnitude of variations of entropy from one measure to the other, and the mean value of entropy. While the latter is about 1.1×10^6, the former gets to a maximum of 8×10^2. This is due to the relative relevance of the connectivity constraints and the spatial constraints. In fact we have a number of connectivity constraints equal to the number of nodes $N = 9996$, while we just have as many spatial constraints as the number of bins $B = 50$. That is why the variation we register are small compared to the average value of entropy.

4.2.1 Interpretation

Since this is not the aim of the paper, we will dedicate just a short section to the interpretation of the results of our analysis. Interpreting data coming from biological systems is never an easy matter, it requires a wide and deep knowledge of the field, which unfortunately we do not possess. Nonetheless, we may provide the reader with at least a viewpoint on the subject, giving some simple key information.

What is the meaning of the analysis we make? Why do we integrate the protein-protein interaction network, common to all the samples, with data of gene expression? The idea is that, while *genotype* is equal for all the cells, *phenotype* is different among

cell types (e.g. a neuron and a liver cell), and can also change in the same cell type during replication or as a function of different perturbations (e.g. a pathology, or ageing).[5] The phenomenology underlying network entropy (gained with other experimental observations) is that it should represent the extent of the phenotypic landscape available to the cell. In fact the phenotype is strictly related to the expression of genes: while all the genes are present in every cell, not all of them are expressed equally. We could find that a different phenotypic condition, such as the onset of an illness or just the effect of old age (in particular this is our case), is correlated to a change of entropy; and by examining this change we could get a better insight into complicated processes, in our case the process of ageing.

By looking at figure 4.5 though, we notice that entropy seems not to have a uniform monotonic trend with respect to age. This may be due to many factors, such as:

- The lack of a robust statistic. We have just 32 samples, and only one for each age. General trends may be veiled under individual variations.
- Protein-protein interaction networks are vast entities composed of smaller sub-networks, each one responsible for different functions of the cell. Individual and distinct behaviours of these sub-units may contribute to the measure of entropy, making a general point of view not effective to understand the phenomenon.
- The effective lack of relevant trends of entropy with respect to age in liver cells. This last point is supported by the fact that in livers and their cells the process of ageing is particularly slow. This is true to the extent that even eighty-year-old people can be liver donors, as in the case of our samples.

We can try to partially make up for the first point, namely the lack of data, and enhance our statistic by reuniting data in age groups. We shall call *young* the group of subjects whose age is less than 40 years, *middle-aged* the ones going from 40 to 70 years old, and *old* the remaining subjects. In figure 4.7 we plot the average entropy of each group in a box-plot.[6]

From the figure we notice a growing trend, but we also see it is not significant.

[5] The *genotype* is a term indicating the organism's full hereditary information, stored in the DNA, which is common to all the cells of an individual. The *phenotype* on the contrary describes the set of the particular physical properties of the cell, which can be different from one cell to the other. Anything that is part of the observable structure, function or behaviour of a living organism is considered phenotype.

[6] A box plot is a convenient way of graphically depicting a data distribution through its quartiles. A red line marks the mean value of the distribution. The box delimits the upper and lower quartile, and a pair of whiskers extend to the highest point less that 1.5 times the *inter-quartile range* (IQR) distant from the upper quartile, and to the lowest point less thant 1.5 times the IQR from the lowest quartile. The remaining points are plotted as *outliers*.

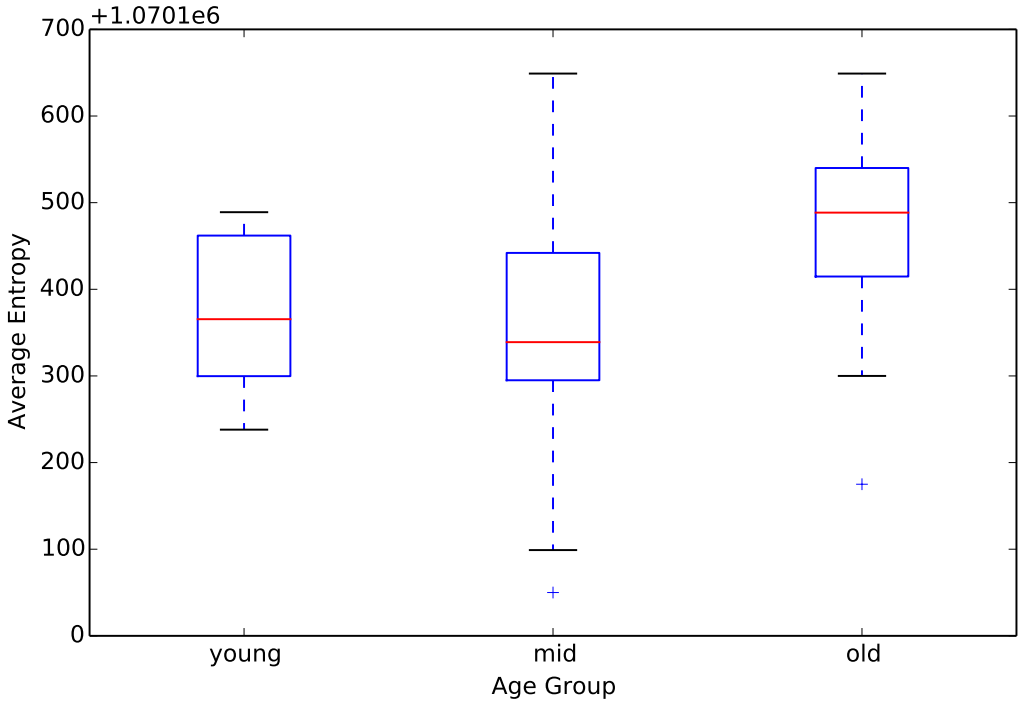


Figure 4.7: Average values of entropy for the three age groups.

We could try to act also on the second point, and analyse the network further by decomposing it in smaller sub-networks. An example of this kind of analysis can be found in the article "*Network Entropy measures applied to different systemic perturbations of cell basal state*",[7] where a cell network is decomposed in six sub-networks, each one with a specific function. In the case studied in the article some significant trends were identified.

4.2.2 Further perspectives

The algorithm we described is capable of extracting a set of constraints from a real network, finding amongst all the canonical network ensemble that satisfy these constraints the one with the maximum entropy, and eventually evaluating this entropy. We used the algorithm to analyse data coming from Biology. In the course of our work, however, we praised network theory for its wide applicability. Now it is the time to show the reader the power of such a theory, meeting a fundamental concept such as entropy whose

[7] Cf. [10]

applicability is no less wide.

The considerations we made were in fact based only on the concept of network and of entropy, and were in no way shaped for our biological data. This means that they are valid all the same for every other system with the required features, that is to say every system that can be modelled as a network whose nodes are correlated by another measure assimilable to a distance. The algorithm would need no modifications, we just have to provide it the data in the form of an adjacency matrix and a distance matrix, and it will return the value of the associated entropy. We could think of an infinity of models satisfying such requirements: a network of social relations correlated with the wealth of the subjects, or semantic networks correlated with a word-distance or even a model of a spin-lattice network, correlated with the difference between the values of the z-projection of spin. The possibilities coming from such an interesting perspective are promising.

Chapter 5

Conclusions

Let us summarize the main passages and the most important achievement of this work. After introducing the context of Systems Biology in order to give a meaning to the data we later analysed, we gave an overview of Network Theory, starting from the concept of graph, all the way up to the model of random graph by Erdős and Rényi. We drew an interesting viewpoint from this model, which led us to the concept of network ensemble.

We presented the concept of Entropy under different points of view, from Statistical Mechanics to Information Theory, and found an application to network ensembles. We were in fact able to define a measure of entropy for microcanonical and canonical network ensembles, and characterize it with respect to the constraints to which the ensemble is subject: the information this measure yields are related to the "strictness" or "mildness" of the constraints. In particular we analysed a case in which the constraints on the topology of the network ensemble are integrated with spatial constraint, concerning a generally defined distance relation between nodes.

Thanks to the wide applicability of network theory, this measure can be applied to a sizeable variety of systems. We used it to analyse data coming from biology, in particular protein-protein interaction networks correlated with data of gene expression, the latter providing us with the spatial constraints for our network ensemble. The extensiveness of the dataset made a computational approach necessary, and we developed a parallel algorithm capable of performing the following steps, starting from data concerning a real network: first it extracts relevant information from the real network, and turns it into constraints. Then it selects amongst all the canonical network ensemble that satisfy these constraints the one with the maximum entropy. Finally, it calculates the entropy of this ensemble.

We analysed the performance of this algorithm, comparing the precision and computation time of a non-parallel version of the algorithm with the same quantities obtained

from running a parallel version of the algorithm on a 8 cores machine and on a 32 cores hpc machines. A considerable improvement in computation time has been observed on the second case, showing a relevant enhancement from the previously available non-parallel code. This improvement made the analysis of our consistent biological dataset possible in a short amount of time. We illustrated the results of this analysis and suggested a possible interpretation for them.

As we already stated, the significance of the network analysis is by no means restricted to biological data only. We can perform the same analysis - without making any change to the algorithm - in every situation in which there is a network structure characterizing our system and a set of distance-alike measures associated to the nodes (i.e. the elements) of the network. The analysis could be applied to social and economical systems, or also physical systems (e.g. a spin glass or a random polymer). This opens the way to further interesting developments.

Bibliography

[1] Albert-László Barabási. *Scale-Free Networks: a Decade and Beyond.* Science, vol. 325, 24 July 2009.

[2] A. Barrat, M. Barthélemy, A. Vespignani. *Dynamical Processes on Complex Networks.* Cambridge University Press, 2008.

[3] G. Bianconi. *Entropy of a network ensemble.* Physical Review E 79, 036114, 2009.

[4] G. Bianconi. *The entropy of randomized network ensembles* EPL (Europhysics Letters), Vol. 81 n. 2. 2008.

[5] L. Chong, L. Bryan Ray. *Whole-istic Biology* Science, vol. 295, 1 March 2002.

[6] P. Erdős, A. Rényi. *On the evolution of random graphs.* Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5: 17–61, 1961.

[7] H. Kitano. *Systems biology: a brief overview.* Science, vol. 295, 1 March 2002.

[8] E. T. Jaynes. *Gibbs vs Boltzmann Entropies.* American Journal of Physics, Vol. 33, No. 5. pp. 391-398. 1965.

[9] E. Klipp, R. Herwig, A. Kowald, C. Wierling, H. Lehrach. *Systems Biology in Practice.* Wiley-VCH, 2005.

[10] G. Menichetti, G.Bianconi, E.Giampieri, G.Castellani e D.Remondini. *Network Entropy measures applied to different systemic perturbations of cell basal state* ArXiv, e-print n. 1305.5369. May 2013.

[11] G. Menichetti, *Statistical mechanics approaches to networks: the role of entropy with applications to gene expression time series data.* Master Degree graduation paper.

[12] J. Park and M. E. J. Newman. *The statistical mechanics of networks.* Phys. Rev. E 70, 066117. 2004.

[13] R. K. Pathria, Paul D. Beale. *Statistical Mechanics.* Butterworth-Heinemann, 2011.