Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Matematica

# A Latent Space Model Approach for Clustering Complex Network Data

Relatore: Prof.ssa Francesca IEVA
Correlatore: Prof. Piercesare SECCHI

Tesi di Laurea Magistrale di:
Alberto DONIZETTI
Matr. 837252

Anno Accademico 2015-2016

## Abstract

Complex networks are becoming increasingly important in many scientific domains, mostly due to the abundance of possible applications in mathematics, economy, biology or logistics. The mathematical community has been focusing on the modeling of networks as graphs objects since the late 1950s, producing an incredibly vast literature on the subject.
The first part of the dissertation covers the essential notions and definitions of graph theory and introduces the most prominent models by describing their positioning in the general state of the art and presenting both similarities and dissimilarities between them. In the second part, a special attention is given to the class of Latent Space Models, which are characterized by the presence of latent variables used to compute the likelihood of the observed networks. The goal of Latent Space Models is to map the observed network in the latent space by meeting precise probabilistic requirements: once the latent positions have been estimated, different kinds of clustering can be performed in the latent space to group similar nodes as an interesting alternative to algorithms designed for community detection.
Finally, throughout our work we provide many examples of analyses on benchmark datasets to assess the performances of Latent Space Models, as well as a final innovative application which also constitutes a bridge towards possible future developments.

*Keywords:* Latent Space Model, Complex Networks, Clustering, Community detection, MCMC Methods.

## Sommario

Lo studio delle reti complesse sta diventando sempre più importante in diversi campi scientifici, soprattutto grazie all'abbondanza di possibili applicazioni in matematica, economia, biologia o logistica. La comunità matematica si è dedicata alla modelizzazione delle reti sotto forma di grafi a partire dalla fine degli anni '50, portando allo sviluppo di una notevole letteratura al riguardo.
La prima parte della tesi è riservata alla definizione delle principali nozioni di teoria dei grafi ed alla trattazione dei modelli fondamentali, confrontati tra loro e collocati in precisi ambiti del più ampio contesto teorico. La seconda parte è invece dedicata alla presentazione dei cosiddetti Latent Space Models, una speciale classe caratterizzata dalla presenza di variabili latenti utilizzate per esprimere la verosimiglianza delle reti osservate. Partendo dai dati relativi ad una struttura di tipo grafo, l'obiettivo di questi modelli consiste nel mappare la rete nello spazio latente affinché siano rispettati determinati vincoli di natura probabilistica. Le posizioni latenti stimate possono essere successivamente utilizzate per individuare clusters e raggruppare quindi tra loro nodi strutturalmente simili, offrendo pertanto un'alternativa interessante ad algoritmi pensati per il rilevamento di comunità.
Nei capitoli sono contenuti esempi di analisi su datasets di riferimento, utilizzati per testare le performance dei Latent Space Models e preparare infine la discussione di un'applicazione innovativa tratta da un contesto particolarmente attuale.

*Parole chiave:* Latent Space Model, Complex Networks, Clustering, Community detection, MCMC Methods.

# Contents

# List of Figures

# List of Tables

# Introduction

"Networks are ubiquitous."
"Networks are essential to society."
"Networks play a vital role in our world."

According to the vast majority of dissertations and scientific papers from the field of complex networks, the importance of networks is undeniable. Indeed, nowadays many different examples can be found in arguably any domain: economy, information technology, mathematics, biology, but also social relationships and technical infrastructures.
However, most of the times we are so accustomed to the presence of networks in everyday matters that we do not even recognize them as fundamental structures to explain how things are connected. This is probably why scientific papers tend to be so clear about their role from the beginning: networks can be studied to explain a huge variety of phenomena, providing insights about possibly unknown subjects exclusively thanks to specific properties and features they possess.
Networks have always existed, but it is only recently that the mathematical community has devoted itself to the development of statistical network theories: starting from the 1950s, the related literature has grown exponentially thanks to many contributions about static or dynamic scenarios, directed or undirected connections, weighted or unweighted approaches.
The last century development can be explained by the fact that due to the complexity of some structures huge computational power was needed to tackle unanswered problems like the recognition of mesoscopic patterns or the definition of importance in terms of nodes and relative positioning in the network.
In recent years, computational tools have already been through some major improvements, yet they are still improving at an incredibly fast pace: needless to say, these are definitely good news for the flourishing of network theory models.

The purpose of our dissertation is to present Latent Space Models (LSM) and discuss their possible applications on real examples.
The term *latent* is a synonym of *unobserved*: latent variables are employed in mathematics to allow the expression of conditional probabilities based on some unobservable parameters. The reason why such formulations are particularly interesting is that performing inference on the latent variables usually gives meaningful a posteriori information about the observed data.
In the LSM framework, a specific latent position is associated to each node from the original network so that the probability of having a connection between two vertices is inversely proportional to their relative distance in the latent space. Our objective is to explain and discuss carefully how the observed network can be mapped into an unobserved space where the relative distances between positions hold this particular meaning, providing at the same time useful examples to facilitate the analysis and new scenarios to develop Latent Space Models.

The dissertation is structured as follows.
We begin our work by introducing the most fundamental notions in Ch. 1: we do not only state the formal definitions that are used as reference throughout the dissertation, but we also discuss more advanced matters like representation or clustering hoping to clarify the main concepts right from the start.
A large panorama of models from the literature is presented in Ch. 2: some of the examples are actually not used in the following chapters as we focus exclusively on a special model, but we thought it was necessary to have a clear overall perspective before diving into more specific details.
The focus on the core part of the dissertation is introduced in Ch. 3, where we present the Latent Space Model we use as main reference for our analyses. We provide detailed information about its technical features and give some results on benchmark datasets that serve multiple purposes, gradually moving from testing to full validation thanks to increasing levels of complexity.
Model issues and possible improvements are the target of Ch. 4, in which we propose our ideas to solve the problems encountered during simulations on benchmark datasets. We explain the possible methods and collect working solutions in a new model which is finally extended to weighted networks in Ch. 5, together with tests of performances and a real case-study.
Conclusions and possible future developments are eventually discussed.

# Chapter 1

# Preliminary notions

## 1.1 Overview

This chapter is meant as a short introduction of the most important concepts about graph topology, representation issues and differences between clusters and communities: as such, some parts may seem unrelated and focused on matters of varying complexity, but they all constitute a necessary knowledge basis for the rest of the dissertation. We hope that this initial effort will eventually pay off, preventing possible doubts and ambiguities regarding the analyses we will present about networks theory, clustering and community detection.

## 1.2 Generalities about graphs

In this section, we remind some generalities concerning graphs: the formalism and the notation that are introduced will be used throughout the rest of the dissertation.

**Definition 1.2.1** (Graph)**.** *A graph consists of a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (also called nodes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, with cardinalities $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$.*

An edge $(v_i, v_j)$ is said to be *incident* to vertices $v_i$ and $v_j$ and it is denoted $e_{ij}$.

**Definition 1.2.2** (Subgraph)**.** *A graph $\mathcal{H} = (\mathcal{V}_h, \mathcal{E}_h)$ is called a subgraph of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}_h \subseteq \mathcal{V}$ and $\mathcal{E}_h \subseteq \mathcal{E}$. An induced subgraph is a subgraph in which $\mathcal{E}_h$ is defined as $\mathcal{E}_h = \{e_{ij} \mid e_{ij} \in \mathcal{E},\ v_i \in \mathcal{V}_h,\ v_j \in \mathcal{V}_h\}$.*

As shown in the introduction, graphs are an efficient tool for the modeling of complex realities like social, biological or information networks: since so many different features need to be fully represented, both nodes and edges must be able to adapt to different specifications. Actually, most of the generalizations can be described by recurring to one of the following concepts, which basically involve additional edge related information like importance, structure or direction.

**Definition 1.2.3** (Weighted Graph)**.** *A weighted graph is a graph assigning a (typically) non-negative and real value $w_{ij}$ to each edge $e_{ij}$. A weighted graph can be described by the triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $W = \{w_{ij} \mid e_{ij} \in \mathcal{E}\}$.*

**Definition 1.2.4** (Multiple edges and loops)**.** *Multiple edges are two or more edges that connect the same two vertices. A loop is an edge linking a vertex to itself.*

**Definition 1.2.5** (Simple graph and multigraph)**.** *A graph defined to disallow (allow) multiple edges and loops is called simple graph (multigraph).*

**Definition 1.2.6** (Directed and undirected graphs)**.** *In a directed (undirected) graph the pairs belonging to the set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are ordered (unordered).*

Sometimes the pairs belonging to the set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are defined to be ordered: in this case, a graph is undirected if $\forall\, e_{ij} \in \mathcal{E}$ also $e_{ji} \in \mathcal{E}$, and vice versa.
The distinction between directed and undirected graphs is arguably one of the most important classifications in graph theory [50]: even basic definitions may greatly differ with respect to the considered framework, so it is preferable to introduce them separately in the directed and undirected case.

### 1.2.1 Undirected graphs

In an undirected graph, an edge spanning two vertices $v_i$ and $v_j$ is denoted $e_{ij}$. Since the order of the indices is not relevant, we can freely denote by $e_{ij}$ or $e_{ji}$ the link existing between the two nodes.

**Definition 1.2.7** (Adjacency and neighbors)**.** *Two vertices $v_i$ and $v_j$ are said to be adjacent if $e_{ij} \in \mathcal{E}$. The neighborhood $N(v_i)$ of $v_i$ corresponds to the set of nodes adjacent to $v_i$. Two edges sharing a common vertex are also called adjacent.*

**Definition 1.2.8** (Degree)**.** *In an undirected graph, the degree of a vertex $v_i$ is noted $k_i$ and is equal to the number of its neighbors.*

**Definition 1.2.9** (Regular graph)**.** *A regular graph is a graph where each vertex has the same degree.*

The degree probability distribution strongly affects the structure of a graph, so much that in the literature of random graphs many models are actually identified by its particular shape: for instance, a Poisson degree distribution characterizes graphs generated accordingly to the Erdös-Rényi procedure [11], while power laws are usually associated to the so called scale-free networks [5]. The degree is usually considered as a measure of the importance of a node: high values usually identify vertices that are well connected to the rest of the network, while small ones tend to spot peripheral vertices. Given this kind of implications, it is useful to make an immediate extension to the class of undirected weighted graphs, so that the different weights assigned to the edges can be taken into consideration for estimating the importance of a node.

**Definition 1.2.10** (Strength)**.** *In an undirected weighted graph, the strength of a vertex $v_i$ is noted $s_i$ and is equal to the total weight of the edges attached to $v_i$.*

Degrees and strengths provide information related to local properties, since they only consider the interaction between neighboring sites. In order to describe the topology of the graph on a larger scale, it is often necessary to analyze the possible connections between distant nodes and identify subgraphs that share some common features.

**Definition 1.2.11** (Connected graph)**.** *An undirected graph is connected when there is a path (sequence of adjacent edges) between every pair of distinct vertices. A connected component is a maximal connected subgraph of $\mathcal{G}$.*

**Definition 1.2.12** (Complete graph)**.** *A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by an edge.*

Completeness clearly requires both regularity and connectivity: given the strength of the requirements, it is pretty rare to observe this feature in real networks of significant size. However, a weaker form is extremely common and useful, as it can be used to identify sufficient statistics for the class of exponential random graphs models which we will introduce in Sec. 2.2.4.

**Definition 1.2.13** (Clique)**.** *A clique is a subset of vertices of an undirected graph such that its induced subgraph is complete. A clique of size $k$ is denoted as $k$-clique.*

### 1.2.2 Directed graphs

In a directed graph (or digraph), an edge connecting vertex $v_i$ to vertex $v_j$ is denoted $e_{ij}$. The order of the indices is fundamental: we know that $v_j$ (successor) is reachable from $v_i$ (predecessor), but we do not possess any information about the converse. With respect to $v_i$, $e_{ij}$ is labeled as outcoming edge, while it is an incoming one with respect to $v_j$.

**Definition 1.2.14** (Adjacency and neighbors). *Two vertices $v_i$ and $v_j$ are said to be adjacent if $e_{ij} \in \mathcal{E}$ or $e_{ji} \in \mathcal{E}$. The incoming (outcoming) neighborhood $N^-(v_i)$ ($N^+(v_i)$) of $v_i$ corresponds to the set of nodes such that $e_{ji} \in \mathcal{E}$ ($e_{ij} \in \mathcal{E}$). Two edges sharing a common vertex are also called adjacent.*

The distinction between incoming and outcoming neighbors provides useful information about dynamic processes over directed graphs and leads to an extension of the concepts of degree and strength:

**Definition 1.2.15** (In-degree and out-degree). *In an directed graph, the in-degree (out-degree) of a vertex $v_i$ is noted $k_i^-$ ($k_i^+$) and is equal to the number of its incoming (outcoming) neighbors. A source is a node $v_i$ with $k_i^- = 0$, while a vertex $v_j$ such that $k_i^+ = 0$ is called a sink.*

**Definition 1.2.16** (In-strength and out-strength). *In a directed weighted graph, the in-strength (out-strength) of a vertex $v_i$ is noted $s_i^-$ ($s_i^+$) and is equal to the total weight of the incoming (outcoming) edges attached to $v_i$.*

The notion of connectivity is also affected by the defined direction of the edges, since, as stated before, knowing that $v_j$ is reachable from $v_i$ does not imply anything related to $v_i$ being reachable from $v_j$. For this reason, connectivity is definitely more complex for directed networks and we need to discriminate between weakly and strongly connected graphs.

**Definition 1.2.17** (Weakly connected graph). *A directed graph is called weakly connected if replacing all of its directed edges with undirected ones produces a connected undirected graph.*

Before introducing strong connectivity, let us precise what we mean by *path*: as in the undirected case, it is a sequence of adjacent nodes, with the additional constraint that intermediate vertices must be both predecessors and successors for some edges. In other words, if a path from $v_i$ to $v_j$ passes through $v_k$, this means that there exist both links $e_{\cdot k}$ and $e_{k\cdot}$ belonging to the set of edges $\mathcal{E}$.

**Definition 1.2.18** (Strongly connected graph)**.** *A directed graph is called strongly connected if there is a directed path joining any pair of nodes.*

As for the definition of connected components, directed networks call for a particularly detailed classification. In particular, weakly connected components are defined as an extension of strongly connected ones by means of an addition of *in* and *out* components, *tendrils* and *tubes*. However, we will not give a precise definition of these concepts, since any further detail goes beyond the scope of our introduction to directed graphs.

## 1.3 Graph representation

In mathematics, a *topological graph* is a representation of a graph in the plane: the vertices are distinct points in the plane and the edges are simple continuous curves possibly intersecting but not passing through any other point representing a vertex. The spatial representation arguably offers a more intuitive viewpoint than the formal combinatorial definition given in the previous section [17], so we could wonder why we have adopted such an approach in the first place. The reason is actually very simple: given a graph $\mathcal{G}$, the combinatorial representation is unique, while the spatial configuration is arbitrary. For instance, we could apply a rotation to a given spatial definition to find another perfectly admissible representation of graph $\mathcal{G}$: the only exception is given by the special class of *spatial networks*, where the spatial positioning is uniquely set by the specific coordinates assigned to each node.

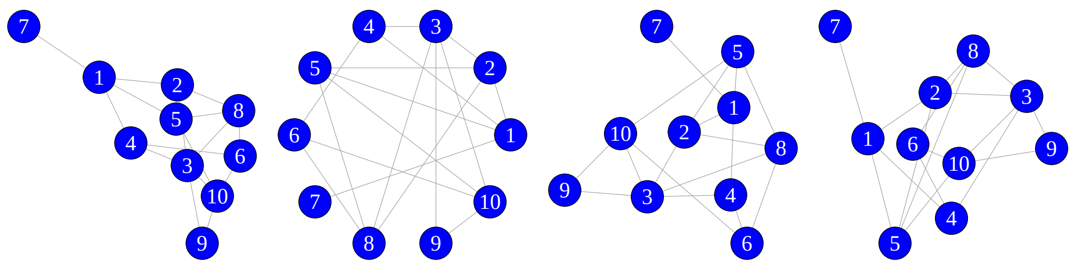

Figure 1.1: Different spatial representation for the same graph $\mathcal{G}$

Therefore, even if graphs are usually associated with a drawing, it is important to stress that in most of the cases they do not possess a precise spatial connotation. This lack of information is actually addressed by many mathematical models that try to estimate the positions and distances between the vertices by minimizing or maximizing specific objective functions.

Just to provide a couple of examples, force-directed graph drawing algorithms [25] rely on physical concepts like spring forces and energy to reach an equilibrium of the graph system which is aesthetically pleasing, while latent space models assign positions in multidimensional spaces to the vertices so that links are more likely to exist if the locations are close with respect to some metric defined in the latent space [20].
Spatial representations may facilitate the understanding of networks by conveying an intuitive overall idea, but graphs can actually be represented without considering additional information like position estimates. In such a spatially independent framework, we shall introduce the fundamental concepts of *incidence* and *adjacency* matrices [8].

**Definition 1.3.1** (Incidence matrix). *The incidence matrix of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, \dots, v_n\}$ and $\mathcal{E} = \{e_1, \dots, e_m\}$ is defined as*

$$B = (B_{ij})_{1 \leq i \leq n, 1 \leq j \leq m} \qquad \textit{where} \quad B_{ij} = \begin{cases} 1 & \textit{if } v_i \in e_j \\ 0 & \textit{otherwise} \end{cases}$$

*and $m, n$ denote the cardinality of sets $\mathcal{E}, \mathcal{V}$ respectively.*

We remark that the set of edges $\mathcal{E} = \{e_1, \dots, e_m\}$ has a different notation from the previously defined $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$: the edges are somehow ordered and we need to look at the incidence matrix in order to understand which nodes are connected by $e_i$.

**Definition 1.3.2** (Adjacency matrix). *The adjacency matrix of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, \dots, v_n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is defined as*

$$A = (A_{ij})_{1 \leq i,j \leq n} \qquad \textit{where} \quad A_{ij} = \begin{cases} 1 & \textit{if } e_{ij} \in \mathcal{E} \\ 0 & \textit{otherwise} \end{cases}$$

*For an undirected graph, the adjacency matrix is symmetric.*

The adjacency matrix $A$ is usually preferred to the incidence matrix $B$: in most of the cases we have to deal with many more edges that nodes (i. e. $m \gg n$), hence the adjacency approach has a great advantage in terms of matrix size and scalability. Furthermore, A is square and dense, while B is rectangular and sparse: performing computations on A is faster and allows to retrieve efficiently information related to connected paths and components.

## 1.4 Clusters and communities

Generally speaking, the term *clusters* refer to groupings of similar items. Closeness between statistical data is assessed through a quantitative scale that may take into account different features or methods: for instance, common examples are the Euclidean distance for numerical attributes in a $d$-dimensional space, the *Hamming distance* for strings or ad hoc functions for categorical values.
A *clustering algorithm* is a procedure that creates associations between objects so that the initial information is compressed into a smaller number of elements. The main goal of clustering methods is to extract significant features from the input data, neglecting as much as possible the micro-scale dynamics in order to focus on macro-scale patterns.
In network theory, the micro-scale considers individual vertices and their connections: describing a complex graph at the microscopic level is often uninteresting, because structure-defining features such as scale-free properties are actually the product of the interaction of multiple nodes. In this regard, the analysis at the mesoscopic level is way more meaningful, since groupings can be used to simplify the network and understand more clearly the patterns of communication across the graph. However, there is an important issue concerning the definition of clusters in this context: nodes do not necessarily possess specific attributes and both their labels and spatial configuration can be totally arbitrary (as discussed in Sec. 1.3), so it is more difficult to define a proper notion of similarity.
In many real networks we observe the existence of *communities*, subsets of highly interconnected nodes such as groups of related individuals in social networks, sets of Web pages dealing with the same topic or biochemical pathways in metabolic networks. Let us consider the social example: we usually assume that people belonging to the same group share some similarities, so it may seem appropriate to state that a community is a cluster of people. Unfortunately, this is a common misconception: in mathematical terms, communities are identified by considering the set of edges, while clusters are found by evaluating the similarity for each pair of nodes.
Even though there is a fundamental difference in their formulation, the concepts of *cluster* and *community* can actually be related by providing a suitable dissimilarity function. In the absence of any additional information, a vertex is characterized only by the set of its incident edges: intuitively enough, two nodes are considered *structurally similar* if they tend to have the same connections with the other vertices of the network [29]. For the sake of simplicity, let us consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a generic pair $\{v_1, v_2\} \in \mathcal{V}$. Then, let us define their relative distance $d(v_1, v_2)$ as the length of the shortest path

between them: if $v_1$ and $v_2$ are similar, than by definition

$$d(v_1, v^*) \approx d(v_2, v^*) \ \forall v^* \in \mathcal{V} \setminus \{v_1, v_2\} \tag{1.1}$$

Due to the high number of internal links, communities are connected components and so there exists an internal path between any couple of their nodes. Moreover, the length of these paths is generally much smaller than the average length of the shortest paths between the vertices of the whole graph: this leads to the conclusion that nodes belonging to the same community $\mathcal{C}$ are likely to be similar, since at least

$$d(v_i, v^*) \approx d(v_j, v^*) \ \forall v_i, v_j \in \mathcal{C}, \ \forall v^* \in \mathcal{V} \setminus \mathcal{C} \tag{1.2}$$

This kind of similarity constitutes the theoretical bridge between clusters and communities: even if some assumptions must be made regarding type of distance and approximation tolerance, the two concepts can indeed be related.
Unfortunately, such a bridge is actually the cause of the common misconception we mentioned before: the two terms are considered equivalent since clusters and communities usually refer to the same subsets of nodes, but as we pointed out this is true only within the right mathematical context. For this reason, we would like to stress once again the fact that the original formulations are different: in particular, communities do not need to be defined through dissimilarity values. Hoping to provide a clear example of correct formulation, we present a rigorous definition of community [12] based on the notions of *intra* and *extra* cluster densities. We consider a generic subgraph $\mathcal{C}$ of a graph $\mathcal{G}$, with $n_{\mathcal{C}} = |\mathcal{C}|$ and $n = |\mathcal{G}|$ vertices respectively.

**Definition 1.4.1** (Internal and external degree)**.** *The internal (external) degree of vertex $v \in \mathcal{C}$ is noted as $k_v^{int}$ ($k_v^{ext}$) and is equal to the number of edges connecting $v$ to other vertices belonging (not belonging) to $\mathcal{C}$. The internal (external) degree of $\mathcal{C}$ corresponds to the sum of the internal (external) degrees of its vertices.*

Les us think about two intuitive scenarios: if $k_v^{ext} = 0$ the vertex has neighbors only within $\mathcal{C}$, which is then a good community for $v$; instead, if $k_v^{int} = 0$ the vertex is disjointed from $\mathcal{C}$ and should be assigned to a different group.

**Definition 1.4.2** (Intra and extra cluster density)**.** *The intra (extra) cluster density $\delta_{int}$ ($\delta_{ext}$)of the subgraph $\mathcal{C}$ is the ratio between the number of internal (external) edges of $\mathcal{C}$ and the number of all possible internal (external) edges, i.e.*

$$\delta_{int}(\mathcal{C}) = \frac{k_{\mathcal{C}}^{int}}{n_{\mathcal{C}}(n_{\mathcal{C}} - 1)} \qquad \left(\delta_{ext}(\mathcal{C}) = \frac{k_{\mathcal{C}}^{ext}}{n_{\mathcal{C}}(n - n_{\mathcal{C}})}\right)$$

For the subgraph $\mathcal{C}$ to be a community we expect it to meet at least two fundamental requirements:

- $\delta_{int}(\mathcal{C})$ has to be appreciably larger than the average link density $\delta(\mathcal{G})$ of the graph $\mathcal{G}$, which can be easily computed as $\frac{m}{n(n-1)}$;
- $\delta_{ext}(\mathcal{C})$ has to be smaller than $\delta(\mathcal{G})$.

Thanks to these constraints, we are finally ready to give a formal definition:

**Definition 1.4.3** (Community)**.** *Given a graph $\mathcal{G}$ and a connected subgraph $\mathcal{C}$, we say that $\mathcal{C}$ is a community if $\delta_{ext}(\mathcal{C}) < \delta(\mathcal{G}) < \delta_{int}(\mathcal{C})$ and $\delta_{int}(\mathcal{C}) - \delta_{ext}(\mathcal{C}) > \epsilon$ for an opportunely chosen $\epsilon$.*

In a nutshell, we have reached our goal: Def. 1.4.3 is exclusively based on information coming from the set of edges. Unlike clusters, communities do not need to consider any kind of similarity between vertices.

We conclude the chapter by introducing the notion of *modularity*: this concept offers a new perspective on the community matter and will be necessary to provide a contextualization for modularity-based algorithms that will be briefly discussed later in the dissertation.

**Definition 1.4.4** (Modularity)**.** *Given a graph $\mathcal{G}$, the modularity of a partition $P = (P_1, \ldots, P_h)$ corresponds to the fraction of edges that falls within the given groups minus the expected fraction if edges were distributed at random:*

$$Q(P) = \frac{1}{m} \sum_{P_k} \sum_{i,j \in P_k} \left[ l_{ij} - \frac{k_i k_j}{m} \right]$$

*where $m = |\mathcal{E}|$, $k_i$ is the degree of node $v_i$ and $l_{ij}$ is the element $(i, j)$ of the adjacency matrix of $\mathcal{G}$.*

The importance of the modularity function lies in the fact that the graph is no longer considered in itself, but it is now compared to a null model: in other words, communities can be seen as striking anomalies in a random network and the modularity function can be used to identify communities and partitions of different levels.
Communities can have different modularities: we call $\alpha$-community a community whose modularity is bigger than a certain $\alpha$ value and $\alpha$-partition a partition in which all the communities are above the $\alpha$ threshold.

Thanks to these preliminary notions and discussions we are now ready to begin our comprehensive review of the main network models available in the literature. Some of the concepts we described in the chapter will appear quite late in the dissertation, so we will make explicit references to the introductory part whenever a quick refresh could be needed.

# Chapter 2

# Network Models

## 2.1 Overview

Many of the networks we observe in the real world exist in only one realization: for instance, given a set of actors in a social context, the graph describing the ties between them is uniquely defined. However, this does not mean that the observed realization is the only admissible network configuration: if we were to consider the same set of actors, common sense suggests that in other circumstances different ties could have been created. Better yet, dynamic networks are usually described through realizations at various times: during its evolution the graph goes through many configurations, proving that different structures are actually allowed.
Statistical models provide a rigorous mathematical framework for this intuition: considering specific instantiations as particular outcomes from some unknown stochastic process, their main goal is to propose a plausible and theoretically principled hypothesis for this process. Depending on the generative stochastic process, the appearance of a particular tie may be influenced by the presence or absence of other ties: in other words, the network is conceptualized as a self-organizing system of relational edges that could also depend on the similarity between node attributes.
Graph models can be mainly divided into two main categories: *static* and *dynamic*. Static models focus on estimating the parameters (or processes) that likely gave birth to the observed realization, while dynamic models try to follow as closely as possible the evolution over time of a given network. Even if the distinction is pretty clear, models belonging to static and dynamic classes are often closely related and most of the times dynamic models can be perceived as a generalization of their static counterparts.

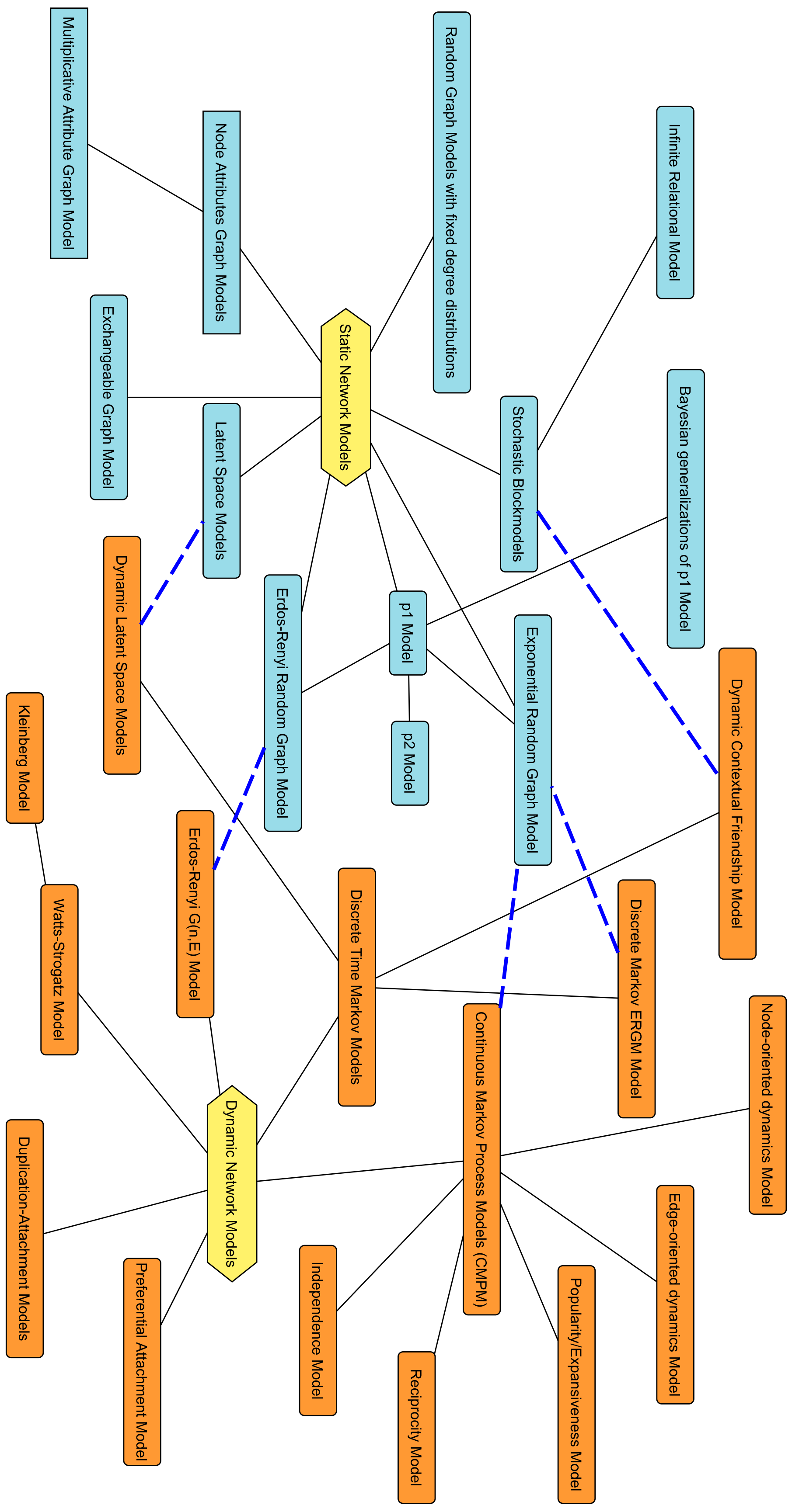
Static Network Models
Dynamic Network Models
Multiplicative Attribute Graph Model
Node Attributes Graph Models
Random Graph Models with fixed degree distributions
Infinite Relational Model
Exchangeable Graph Model
Latent Space Models
Stochastic Blockmodels
Bayesian generalizations of p1 Model
Dynamic Latent Space Models
Erdos-Renyi Random Graph Model
p1 Model
p2 Model
Exponential Random Graph Model
Dynamic Contextual Friendship Model
Kleinberg Model
Erdos-Renyi G(n,E) Model
Watts-Strogatz Model
Discrete Time Markov Models
Discrete Markov ERGM Model
Continuous Markov Process Models (CMPM)
Node-oriented dynamics Model
Duplication-Attachment Models
Preferential Attachment Model
Independence Model
Reciprocity Model
Popularity/Expansiveness Model
Edge-oriented dynamics Model

Figure 2.1: Literature panorama of network models

In the following sections, we will introduce some of the most prominent models in the literature and we will try to provide a contextualization for each of them by highlighting the relations and affinities with the other models. Due to the extent of the subject related literature, the focus will be on features deemed particularly relevant for the dissertation. The outline will closely follow the one provided by [16], with the addition of some models and explanation mainly from [5, 8, 9, 33, 34, 45, 50].
Concept maps like the one in Fig. 2.1 will be used to show synthetic representations of the literature panorama and highlight the conceptual links between models. The spatial positioning of the labels is only meant for the sake of visual clarity and does not wield any other conceptual meaning.

## 2.2 Static Network Models

Although static models vastly differ from one another both in terms of needed assumptions and complexity of the parameters, we can identify five fundamental steps corresponding to their general outline [40]:

1. each network tie is regarded as a random variable;
2. a dependence hypothesis is proposed;
3. the dependence hypothesis induces a particular form of the model;
4. the parameters are simplified by analyzing the constraints;
5. the remaining parameters are estimated and eventually interpreted.

The main purpose of this formulation is to give an overall idea of the possible variations that lead to the creation of different static models. Even if it might be interesting to detail the listed points for a known model from the literature, the analysis could actually be more troublesome than useful: first, we would need to interrupt the flow of the introduction to describe the model; second, the choice of a particular example would not be representative of the many possible configurations and could even give a biased idea of the common structure.
For these reasons, instead of focusing on a specific case we will provide examples that serve a twofold purpose: clarify the most critical points of the general outline and anticipate features from the models that will be described later.
The examples are ordered according to the previous list and the letters are simply used to denote different unrelated cases for each step:

1. in the literature, $Y_{ij} \in \{0, 1\}$ usually corresponds to the random binary variable related to $e_{ij}$: $Y_{ij} = 1$ if $e_{ij} \in \mathcal{E}$, $Y_{ij} = 0$ otherwise. During the

rest of the dissertation we will use $Y_{ij}$ and $e_{ij}$ as a general notation to formally distinguish between the random variable for the statistical model and the actual edge between vertices $v_i$ and $v_j$;

2. (a) $Y_{ij} \mid p \sim Be(p)$;
   (b) $Y_{ij} \mid x_i, x_j \sim Be\left(q(x_i, x_j)\right)$, where $x_i, x_j$ are vectors of features for vertices $v_i, v_j$ and $q(\cdot,\cdot)$ is a similarity measure;
   (c) $Y_{ij} \mid k_i, k_j \sim Be\left(h(k_i, k_j)\right)$, where $k_i, k_j$ are numbers denoting the groups to which vertices $v_i, v_j$ belong and $h(\cdot,\cdot)$ is a function defining the link probability between groups;
3. (a) exponential form;
   (b) hierarchical form;
   (c) configurations without a closed form;
4. if the model is too complex, some parameters may be grouped or removed by imposing some additional conditions such as:
   (a) equality (e.g., symmetry) constraints;
   (b) null constraints;
5. the parameters are usually estimated either by means of an optimization routine or a simulation procedure. For instance:
   (a) maximum likelihood;
   (b) MCMC algorithms.

The choices made at each step determine the final formulation of the model: in the following section, we will present the most significant examples from the literature starting from the simplest model and progressively increasing the complexity of both parameters and dependence assumptions.
We will discuss the relations between models and we will use Fig. 2.2 as reference for the overall view. In the concept map, yellow boxes contain the main models; their significant features are described by green labels, while blue ones provide insights related to the transitions from one model to another. Finally, red dotted lines denote the existence of both a certain similarity and a strong conceptual difference between two models: these links will be better explained and motivated in the following sections.

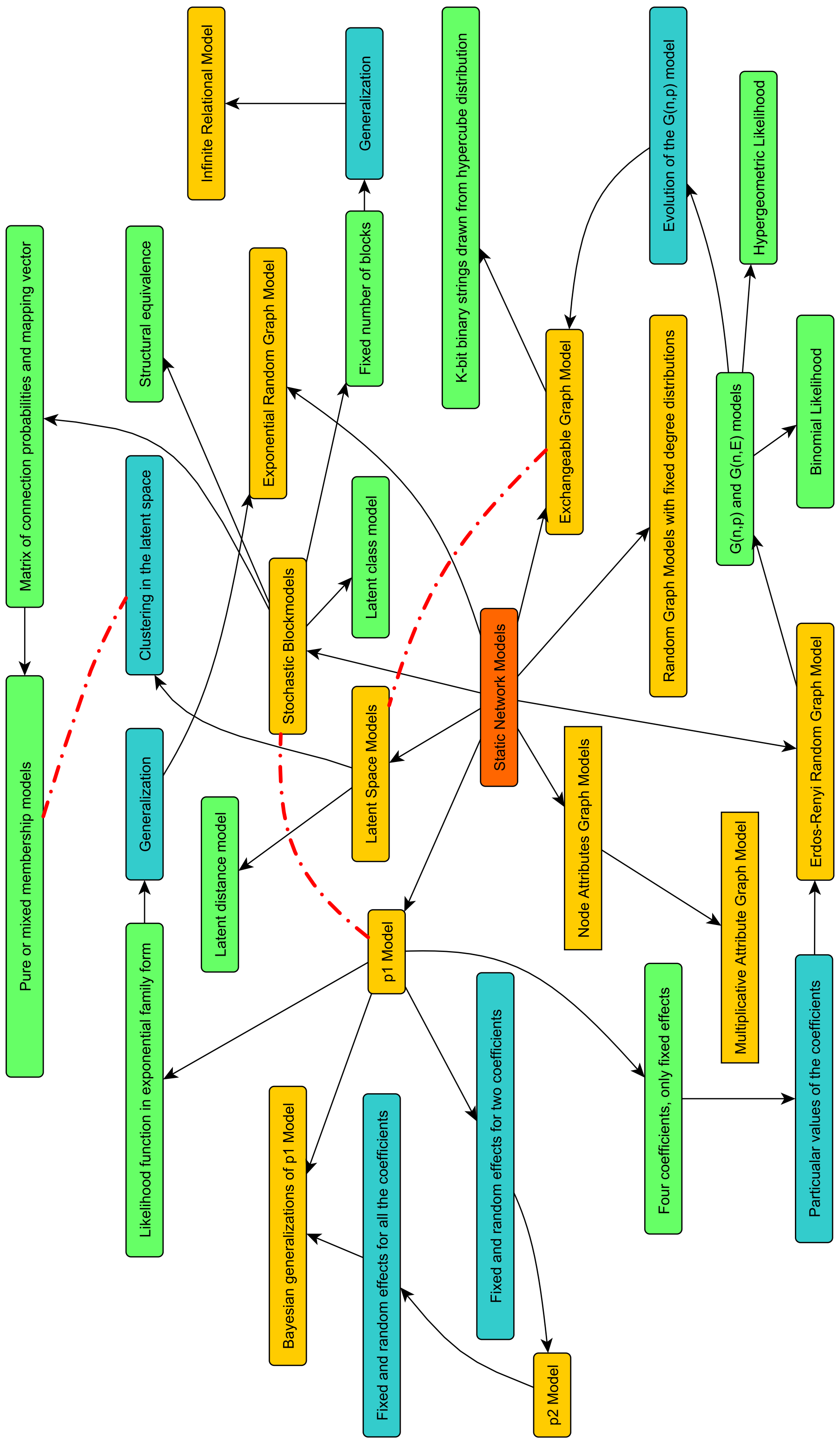
Static Network Models
Stochastic Blockmodels
Latent Space Models
p1 Model
Node Attributes Graph Models
Erdos-Renyi Random Graph Model
Random Graph Models with fixed degree distributions
Exchangeable Graph Model
Exponential Random Graph Model
Matrix of connection probabilities and mapping vector
Pure or mixed membership models
Structural equivalence
Clustering in the latent space
Generalization
Likelihood function in exponential family form
Latent distance model
Latent class model
Fixed number of blocks
Generalization
Infinite Relational Model
K-bit binary strings drawn from hypercube distribution
Evolution of the G(n,p) model
Hypergeometric Likelihood
G(n,p) and G(n,E) models
Binomial Likelihood
Multiplicative Attribute Graph Model
Particualar values of the coefficients
Four coefficients, only fixed effects
Fixed and random effects for two coefficients
Fixed and random effects for all the coefficients
Bayesian generalizations of p1 Model
p2 Model

Figure 2.2: Static Network Models concept map

### 2.2.1 Erdös-Rényi-Gilbert Model

In 1959, the American mathematician Edgar Gilbert published his findings [15] related to the probability of observing connected paths in a random graph built by following a specific procedure. According to his model, the graph was characterized by the two parameters $N$ and $p$: the first was referring to the graph size, while the second to the probability of existence for all the possible edges. Meanwhile, an independent yet closely related research [11] was being carried out by the couple of Hungarian mathematicians Paul Erdös and Alfred Rényi, who were investigating connectivity features and probabilities for a similar class of random graphs. Their model was actually considering the same size parameter $N$ as in Gilbert's formulation, but the probability $p$ was replaced by $E \in \mathbb{N}$, a constant specifying the number of desired edges.

**Definition 2.2.1** (Gilbert Model). *Given $N \in \mathbb{N}$ and $p \in [0,1]$, the graph $\mathcal{G}(N,p) = (\mathcal{V},\mathcal{E})$ is defined as the undirected graph with $N$ vertices such that the probability of having an edge between each pair of distinct nodes is equal to $p$. In symbols, $P(e_{ij} \in \mathcal{E}) = P(Y_{ij} = 1) = p \; \forall v_i, v_j \in \mathcal{V},\; i \neq j$.*

**Definition 2.2.2** (Erdös-Rényi Model). *Given $N \in \mathbb{N}$ and $E \leq \binom{N}{2}$ , the graph $\mathcal{G}(N,E) = (\mathcal{V},\mathcal{E})$ is defined as the undirected graph with $N$ vertices such that the $E$ edges are chosen randomly from the possible $\binom{N}{2}$ edges[1].*

The $\mathcal{G}(N,p)$ model has a binomial likelihood :

$$\mathcal{L}(\mathcal{G}(N,p) \text{ has } E \text{ edges}|p) \propto p^E(1-p)^{(\binom{N}{2}-E)}$$

The likelihood of the $\mathcal{G}(N,E)$ model is instead a hypergeometric distribution that induces a uniform distribution over the sample space of possible graphs.
The $\mathcal{G}(N,p)$ model controls the expected number of edges by specifying the probability of each link, while the $\mathcal{G}(N,E)$ model implies the expected probability of every edge by imposing a specific number of edges. Therefore, the link between the models is found by setting $E = p\binom{N}{2}$: given this relation, as $N$ goes to infinity $\mathcal{G}(N,p)$ behaves like $\mathcal{G}(N,E)$, meaning that the probability of obtaining a specific graph will be the same with the two models. An example of random network generated through the $\mathcal{G}(N,E)$ procedure for $N = 100$ and $E = 200$ can be observed in Fig. 2.3.
In the literature, the more commonly used model is $\mathcal{G}(N,p)$, since it is easier to analyze and has some computational advantages thanks to the independence of the edges. Furthermore, when considering its asymptotic behavior, the size of

[1]An equivalent interpretation is that all the $\binom{\binom{N}{2}}{E}$ graphs are equally likely.

the greatest connected component depends on a simple combination of the two parameters, $\lambda = Np$:

- it is at most $O(\log N)$ if $\lambda < 1$;
- it is $O(N^{(2/3)})$ if $\lambda = 1$;
- it is *giant* if $\lambda > 1$. The greatest connected component consists of almost all the nodes and others connected components do not contain more than $O(\log N)$.

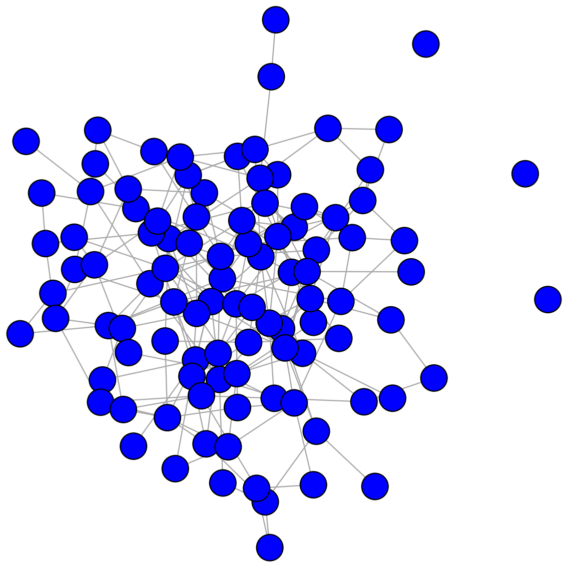

Figure 2.3: Example realization for the $\mathcal{G}(N, E)$ model

### 2.2.2 Exchangeable Graph Model

The exchangeable graph model provides the simplest possible extension of the Erdös-Rényi-Gilbert random graph model by introducing a weak form of dependence among the probability of sampling edges that is due to *non-observable* node attributes [16].

**Definition 2.2.3** (Finite and infinite exchangeability)**.** *A finite sequence of random variables $X_1, X_2, \ldots, X_n$ is finitely exchangeable with joint probability measure $P$ if, for any permutation $\pi$ of indices,*

$$P(X_1, X_2, \ldots, X_n) = P(X_{\pi(1)}, X_{\pi(2)}, \ldots, X_{\pi(n)})$$

*An infinite sequence $X_1, X_2, \dots$ is infinitely exchangeable if any finite subset of the sequence is finitely exchangeable.*

**Theorem 2.2.1** (De Finetti 0-1 Representation Theorem). *If $X_1, X_2, \dots$ is an infinite exchangeable sequence of binary variables with probability measure $P$, then there exists a distribution function $Q$ such that the joint mass function of $X_1, X_2, \dots, X_n$ has the form*

$$P(X_1, X_2, \dots, X_n) = \int_0^1 \left\{ \prod_{i=1}^{n} \theta^{X_i}(1 - \theta^{X_i}) \right\}$$

*where $Q(t) = \lim_{n\to\infty} P\left(\frac{\sum_{i=1}^n X_i}{n} \le t\right)$ and $\frac{\sum_{i=1}^n X_i}{n} \xrightarrow{a.s.} \theta$.*

In the sense of De Finetti, the edges of an exchangeable random graph are *conditionally independent* given the node attributes in the form of binary strings of size $k$. Let $\vec{b_i},\ \vec{b_j}$ be two binary strings related to vertices $v_i, v_j$ and $q$ a function mapping pairs of binary strings into the $[0, 1]$ interval:

$$\begin{aligned} Y_{ij}|q(\vec{b_i}, \vec{b_j}) &\sim Be\left(q(\vec{b_i}, \vec{b_j})\right) \\ \vec{b_i} &\sim \pi(\cdot) \end{aligned} \tag{2.1}$$

where $\pi(\cdot)$ is a generic prior distribution. For instance, we could follow a standard approach by choosing $\vec{b_i} \sim Unif$(vertex set of K-hypercube) or increase the model complexity by defining a hierarchical model with some hyperpriors. The shape of the graph is strongly affected by the number of bits $K$ and by the function $q$, which can be symmetric or not in order to model both undirected and directed graphs.
Even if we are already beyond the complexity of the $\mathcal{G}(N, p)$ and $\mathcal{G}(N, E)$ models, we have yet to reflect on the most important feature introduced by the exchangeable graph model. Let us consider a mapping function $q$ and some binary strings $\vec{b_1}, \vec{b_2}, \dots, \vec{b_n}$: since the value of the Bernoulli parameter depends on the pair of vertices, it is likely that there will be subsets of nodes behaving differently from others. Such *classes* of vertices were not allowed by the Erdös-Rényi model, since all nodes were characterized by a uniform affinity: by introducing a weak dependence among the edges, the exchangeable graph model leads to a range of connectivity patterns that are more structured and depend on the particular entities of the vertices.
The link between graph connectivity and node attributes is a fundamental step towards the analysis of more complex networks by means of multiplicative attribute models [28], stochastic blockmodels [27] or latent space approaches[20].

However, even if these formulations share some kind of common perspective with the exchangeable graph model, it is important to remark that the role played by the node attributes is substantially different. For instance, latent models for social networks provide information related to the positioning in a social space and stochastic blockmodels focus on the belonging to specific social groups, while binary strings from the exchangeable graph model are pure mathematical artifacts that do not carry semantic meaning [16]. Despite this limitation, they can be used as a tool to explore the space of connectivity patterns through a clear and principled semi-parametric procedure: this is actually very helpful, since it allows to identify the right number of parameters and induce an expressive parametric family of distributions that may serve as a basis for more complex models.

### 2.2.3 Attribute Graph Models

In many real world networks, each node is associated with a set of attributes: profile information for online users, chemical properties for biological elements or even acoustical features for musical tunes. Such additional data usually have a strong influence on the graph structure: for instance, some links can only exist when specific attribute requirements are met and cannot be explained otherwise. The mathematical challenge lies therefore in the definition of a model that can exploit this kind of information and capture the dynamics that lead to a particular network structure.
We have seen that the exchangeable graph model is already able to cope with different individual behaviors by assuming the existence of some non-observable node attributes: obviously enough, we should try to incorporate in the formulation of the dependence hypothesis any significant feature that is actually available.
In the framework of Attribute Graph Models, node characteristics are usually represented as numerical or categorical vectors and the link probability between vertices is based on a similarity measure defined on such vectors. For instance, the Multiplicative Attribute Graph Model (MAG) [28] considers networks in which each node has a set of categorical attributes associated with it. The model can simultaneously take into consideration attributes that reflect *homophily* as well as *heterophily*: since sharing the same feature does not necessarily imply an increase of link probability, we have to be able to deal with all sort of node attribute interactions that may occur in networks.
Formally, a vector of $L$ attributes $\vec{F_i} = [F_{i1}, F_{i2}, \ldots, F_{iL}]'$ is associated to each node and the affinity of the attribute $l$ to form a link is expressed by means of an *affinity matrix* $\Theta_l$, where the entry $\Theta[k, k']$ indicates the potential for vertices

$v_i$ and $v_j$ to form a link given their values of attribute $l$ (respectively $k$ and $k'$). The link probability between two nodes is finally defined as the product of the corresponding entries of the $L$ affinity matrices. Conditionally to attributes and affinity values, the random variables $Y_{ij}$ are independent:

$$Y_{ij} \mid \vec{F}_i, \vec{F}_i, \Theta_1, \Theta_2, \ldots, \Theta_L \sim Be\left(\prod_{l=1}^{L} \Theta_l[F_{il}, F_{jl}]\right) \tag{2.2}$$

As a final remark, let us observe that Attribute Graph Models are actually characterized by a great variety: for instance, MAG is centered on the affinity concept, but we could choose similarity functions based on scalar products or ad-hoc distances. We decided not to consider other possible examples simply because the mathematical core is more related to the exploitation of the covariates than to the analysis of the graph itself, which makes these formulations not so relevant for an overview of the state of the art. In order to keep our focus on the network topology, in the next section we will move onto a completely different class of graph models and we will start to investigate non trivial features like *reciprocity* or *transitivity*.

### 2.2.4 Exponential Random Graph Models

The exponential random graph model (also known as ERGM) defines a probability distribution over a specified set of possible graphs $\mathcal{G} = \{G\}$ such that the probability of a particular graph G has the following form[40]:

$$P(\mathcal{G} = G) = \frac{1}{k} \exp\left\{\sum_A \mu_A g_A(G)\right\} \tag{2.3}$$

where the summation is over all admissible configurations $A$, $k$ is a normalizing quantity, $\mu_A$ is a parameter corresponding to the configuration $A$ (we will later discuss its value in detail) and finally $g_A(G) = \prod_{y_{ij} \in A} y_{ij}$ is the network statistic related to configuration $A$ and it is equal to 1 if $A$ is observed in $G$, 0 otherwise. Depending on which configurations have a non-zero coefficient $\mu_A$, the structure of the graphs described by the probability distribution from the exponential family may vary a lot. Actually, we will see that the Erdös-Rényi model belongs to this category, as well as graphs that exhibit much more complicated conditional dependences among their edges. Following an order of increasing complexity, in the next sections we will present the most important models belonging to the class of exponential random graphs.

#### 2.2.4.1 p1 Model

The exponential random graph model was first proposed in the early 1980s by Holland and Leinhardt [22] in the framework of social networks and relationships analysis: the name *p1* was chosen because they considered this to be the first plausible and viable statistical model for directed graphs [20].
In this model *dyads* $(Y_{ij}, Y_{ji})$ are independent and each actor has two parameters $\alpha_i$ and $\beta_i$ responsible for the tendency of the actor to respectively send (*expansiveness*, influencing the out-degrees) and receive ties (*popularity*, influencing the in-degrees). In addition, $\theta$ and $\rho_{ij}$ influence the total number of ties and the tendency toward reciprocation between vertices $v_i$ and $v_j$.
Let $P(Y_{ij}, Y_{ji})$ be the probability distribution defined on a dyad and $P_{ij}(Y_{out}, Y_{in})$ the same probability with a slightly different notation. Since the edges are binary variables, the distribution can be identified by the following:

- $\log P_{ij}(0,0) = \lambda_{ij}$
- $\log P_{ij}(1,0) = \lambda_{ij} + \alpha_i + \beta_j + \theta$
- $\log P_{ij}(0,1) = \lambda_{ij} + \alpha_j + \beta_i + \theta$
- $\log P_{ij}(1,1) = \lambda_{ij} + \alpha_i + \beta_j + \alpha_j + \beta_i + 2\theta + \rho_{ij}$

In this representation of p1, $\lambda_{ij}$ is a normalizing constant to ensure that the probabilities for each dyad *(i,j)* add to 1.
Considering the independence between dyads, the p1 model has a likelihood function that is clearly in the exponential form. Special cases have even simpler likelihood functions, like the ones we can obtain by setting $\rho_{ij} = 0$, $\rho_{ij} = \rho$ or $\rho_{ij} = \rho + \rho_i + \rho_j$. The main problem of the p1 model is that the number of parameters strongly increases with the graph size: the most critical parameters are the $\rho_{ij}$, since due to their quadratic number there is a lack of identifiability. Furthermore, we do not have consistency results for the maximum likelihood estimates as $n \to \infty$.
For these reasons, the p1 model is often generalized by introducing additional constraints or grouping some coefficients into classes of nodes with the same behavior. This last solution is closely related to the stochastic blockmodels, since in both approaches the nodes belong to some classes: however, a fundamental difference lies in the fact that in stochastic blockmodels the partitioning of the vertices is meaningful and may even be the goal of the models (latent classes or a posteriori blockmodeling), while in the p1 framework the classes are fundamentally convenient tools to help reducing the number of parameters and decrease the risks of overfitting. Finally, there is also a distinction concerning

the model hierarchy: stochastic blockmodels follow a *top-down* direction, meaning that belonging to a class induce a similar behavior for the vertices, while p1 is more governed by a *bottom-up* process, meaning that individual nodes with similar behaviors tend to form subset of nodes with the same features.

#### 2.2.4.2 p2 Model and Bayesian extensions

In the statistical literature [37], the notion of *fixed effects* typically refers to a set of unknown constant quantities, while *random effects* are usually considered as unknown variable quantities that are drawn from a common underlying distribution. Fixed and random effects may actually serve the same purpose and, depending on the statistical model and on its goals, the distinction can sometimes be fuzzy [14].
In the p1 model, all the parameters are considered as *fixed effects*: their constant value needs to be estimate without considering any generative distribution for the coefficients. In the so called p2 model, the parameters of popularity and expansiveness switch from fixed to random: further generalizations and changes from fixed to random lead to the Bayesian extensions of the p1 model, which ultimately consist in additional hierarchic levels with prior distributions for all the model parameters $(\alpha, \beta, \theta, \rho, \lambda)$.

#### 2.2.4.3 Markov Graphs

In 1970, Davis [7] remarked that *transitivity of relations* is an outstanding feature that differentiates observed data from a pattern of random ties. Transitivity is expressed by *triad closure*: unfortunately, the presence of a number of short loops significantly higher than the one expected from a non transitive null model cannot be explained by models based on the crucial property of dyadic independence [33].
The main hope for progress in understanding the effects of transitivity lies in developing new statistical formulations based on some alternative ensemble of graph structures: in this regard, ERGMs seem to be a natural framework thanks to the adaptability provided by Eq. 2.3.
Some developments of the p1 model actually tried to relax the strong assumption of dyadic independence introduced by Holland and Leinhardt. For instance, the generalization proposed by Frank and Strauss [13] in 1986 focused on the analysis of the so called *Markov graphs*, particular structures in which only adjacent edges can be conditionally dependent.
Markov graphs can actually be seen as the fundamental step from models requiring independence among the edges to models allowing local and global forms of dependence, especially for exponential random graphs. In order to better ap-

preciate their fundamental contribution to the literature, we will start by briefly introducing some notation and concepts that will help us formulate the exponential likelihood of such graphs.
Let us start by considering a sequence $Z = (Z_1, Z_2, \dots, Z_m)$ of $m$ discrete random variables. Let $M = 1, 2, \dots, m$ be the index set, $R_i$ the range space of $Z_i$ and $R$ be the range space of $Z$.
Let us also denote the probability function of $Z$ by $Pr(Z = z) = P(z)$ for $z \in R$: this function can be defined by $P(Z) = c^{-1} \exp Q(z)$, where $c$ is a constant for normalization.
Conversely, $Q(z) = \log P(z) - \log P(z^*)$ for a fixed $z^* \in R$ and, by application of the *inclusion-exclusion principle*, it can be shown that $Q(z)$ can be expressed as $Q(z) = \sum_{A \subseteq M} \lambda_A(z_A)$, where $z_A$ is the subsequence of $z$ consisting of $z_i$ with $i \in A$. For the identifiability of each $\lambda_A$, which is called the $A$ interaction in $Z$, certain side conditions must be imposed: for further details, see [13].
The $\lambda_A$ interactions are useful to describe dependence structures among the random variables in the sequence Z: the necessary link between dependence structures and interactions is provided by the Hammersley-Clifford theorem [2].
A dependence graph $D$ is a graph that defines the conditional dependencies present among the pairs of random variables extracted from a given sequence $Z$.
According to the theorem, $\lambda_A(z_A) = 0 \forall z \in R$ unless A is a clique of $D$. Thus in the expansion of $Q(z)$ we need only those $A$ interactions that correspond to cliques of $D$.
We are finally ready to use these notions for the analysis of Markov graph models: dependence graphs $D$ can be built over the set of edges of a random graph $G$ and through the Hammersley-Clifford theorem we are able to obtain the following characterization of the probability function of a general random graph $G$:

$$Pr(G) = c^{-1} exp \sum_{A \subseteq G} \alpha_A \tag{2.4}$$

where $\alpha_A$ is an arbitrary constant if $A$ is a clique of $D$ and is equal to 0 otherwise.
As we stated at the beginning of the section, a graph is said to be a Markov graph if only incident dyads can be conditionally dependent. We can now provide an alternative formulation: a graph is a Markov graph or has Markov dependence if $D$ contains no edge between disjoint sets $s, t$ and $u, v$ in $M$, the index set of nodes.
For general Markov graphs the cliques of D correspond to sets of edges such that any pair of edges within the set must be incident: such sets are just the triangles and $k$-stars.

Thanks to these *sufficient subgraphs*[2], any undirected Markov graph has probability

$$Pr(G) = c^{-1} exp \left( \tau t + \sum_{k=1}^{n-1} \sigma_k s_k \right) \tag{2.5}$$

where $t$ is the number of triangles, $s_k$ the number of $k$-stars and $\tau$ and $\sigma$ are the respective values of the interactions. The reason why we do not need to consider the specific vertices belonging to the sufficient subgraphs is that we are assuming an *homogeneity* condition: all isomorphic graphs have the same probability.
The equation can be slightly modified in order to express two other parameters, $\rho = \sigma_1$ and $r = s_1$ :

$$Pr(G) = c^{-1} exp \left( \tau t + r\rho + \sum_{k=2}^{n-1} \sigma_k s_k \right) \tag{2.6}$$

This formula leading to the formulation of the so called $\rho\sigma\tau$ or *triad model*, which can be seen as a simplified version where the only sufficient statistics are the number of edges, 2-stars and triangles.
Let us now consider once again the Erdös-Rényi $\mathcal{G} = (N, p)$ model: by taking $\tau = 0$ and $\sigma = 0$ in the triad model we can obtain the Erdös-Rényi probability distribution for a graph $G$, where $p = \frac{\exp \rho}{1+\exp \rho}$ (or, conversely, $\rho = \log \frac{p}{1-p}$).
Starting from simple models like $\mathcal{G} = (N, p)$ and thanks to the contribution by Frank and Strauss, we have seen that graphs with a non trivial dependence structure can be correctly represented and analyzed while keeping a convenient likelihood formulation. The importance of Markov graph models also lies in the fact that, unlike many other models, they do not focus exclusively on node attributes or positioning as responsible for the edge probabilities, but instead they consider the role of dependencies in the self-organizing stochastic system of relational edges.
Finally, let us conclude the presentation of exponential random graphs with a remark regarding notation: p1, p2 and Markov graph models can actually be extended with the class of the so called $p^*$ models. This further generalization basically coincide with the whole panorama of exponential random graphs, which is why the two labels of ERGM and $p^*$ are often interchanged.

Let us stop for a moment before introducing the last two static network models from our introduction. Up until now we have considered statistical approaches

[2]A statistic is said to be sufficient with respect to a statistical model and its associated unknown parameter if it does not exist another statistic based on the sample data that provides additional information related to the parameter. Triangles and $k$-stars are sufficient subgraphs [48] since their number provide all the necessary information to estimate the parameters of a given graph.

that were based on different dependence assumptions and were designed to deal with specific features of the observed data:

- everything started with Erdös-Rényi, which was only capable of dealing with simple and regular structures;
- additional node-level information led to the development of the so-called attribute models;
- issues coming from lack of observable data were partially solved by considering the models allowing reciprocity, transitivity and, more generally, specific behaviors for each vertex.

However, some problems are still left unsolved: for instance, we came across mathematical tools that can only be used as an initial approximation or exploration (exchangeable graph) or too many parameters that do not scale well with networks of bigger size (p1 model).
Furthermore, the fundamental concepts of community and clusters still need to be properly explored and analyzed: we will advance in this direction thanks to the last two examples, stochastic blockmodels and latent space models. Their description will complete the overview of static network models and provide the final foundations needed for the dissertation.

### 2.2.5 Stochastic Blockmodels

A stochastic blockmodel [46, 47] is a model for sociometric data obtained from a network characterized by *block structure* (Fig. 2.4), meaning that the nodes are partitioned into subgroups called blocks. Such modules may be known a priori or inferred from the observed ties: in both cases, the fundamental hypothesis is that the links distribution is dependent on the blocks to which the vertices belong. More formally, the usual assumption is $Y_{ij} \mid k_i, k_j \sim Be\,(h(k_i, k_j))$, where $k_i, k_j$ are numbers denoting the blocks to which vertices $v_i, v_j$ belong and $h(\cdot, \cdot)$ is a function defining the link probability between groups. The blocks are designated by a set of numbers $\{1, 2, ..., K\}$ whose values may also be labeled as positions, classes or colors.
In its most basic version, the stochastic block model (SBM) of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ vertices is defined by a scalar value and two simple data structures:

- $K$ : a scalar value denoting the number of communities in the network;
- $z$ : a $n$-dimensional vector containing the group indices of the vertices;
- $M$ : a $(K \times K)$ stochastic block matrix, where $M_{ij}$ gives the probability that a vertex of class $i$ is connected to a vertex of class $j$.

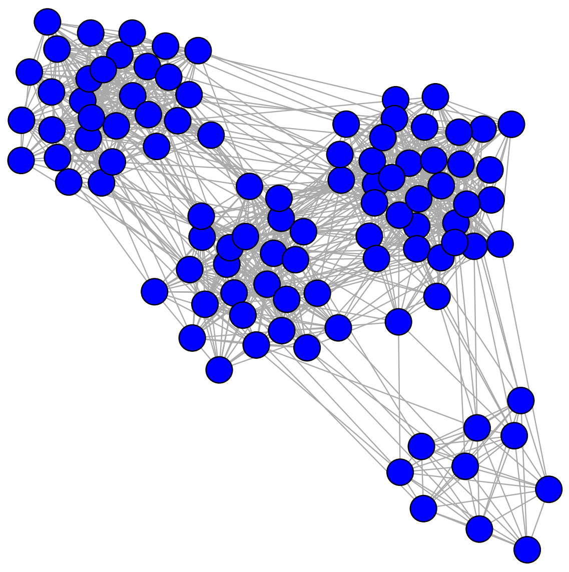

Figure 2.4: Example realization for a stochastic blockmodel

It is important to remark that $K$ must be chosen before either $z$ or $M$ can be assigned: in other words, the number of classes or partitions can not be inferred by the data and must therefore be known a priori. Since each vertex in a given group connects to all other vertices in the same way, vertices with the same label are called *stochastically equivalent* [23]: the information regarding the nodes needed by the model is then kept to a minimum.
In Section 2.2.4.1 we provided a quick comparison wit the p1 model and we mentioned the fact that the generative approach of the two models is actually different. Let us now consider the creation process of SBMs on a deeper level:

1. we suppose that the nodes of a given set $\mathcal{V}$ belong to some classes;
2. the link probability between two vertices depends on the blocks they belong to: for instance, if the two nodes are in the same module, the probability is greater, otherwise it is smaller (homophily);
3. links are created accordingly to such probabilities.

Going backward, once we have an actual graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the purpose of communities detection is to characterize the groups that have supposedly influenced the graph structure through the above generative process. For the statistical application of blockmodels, an important distinction must be made: if the blocks are known a priori or can be derived from the node attributes, then the inference problem is reduced to the estimation of some group parameters; if the blocks are

latent or non observable, the situation becomes much more complicated since the partition itself must be inferred from the observed edges of the network.
The latter problem, known as *a posteriori blockmodeling*, is way more interesting and does not have to be restricted to the case where $K$ is fixed: models like the Infinite Relational Model [27] are efficient even when the number of classes is unknown or may evolve in time due to the addition of new nodes to the network.

### 2.2.6 Latent Space Models

Exchangeable Graph Models introduced additional feature strings in order to better distinguish the behaviors followed by the vertices, while stochastic blockmodels assumed the existence of latent classes to help partitioning the nodes. Latent Space Models (LSM) bring the latent approach a step further, allowing for a more general probability conditioning.
These models assume the independence of $Y_{ij} \mid z_i, z_j, x_i, x_j, \theta$, where $x_i, x_j$ are some features specific for each node (like in the MAG model), $\theta$ is an unknown parameter and $z_i, z_j$ are latent positions in a multidimensional space. Therefore, the overall probability of a network configuration in matrix notation is given by

$$P(Y|Z, X, \theta) = \prod_{i \neq j}^{n} p(Y_{ij} = 1|z_i, z_j, x_i, x_j, \theta)^{Y_{ij}} p(Y_{ij} = 0|z_i, z_j, x_i, x_j, \theta)^{1-Y_{ij}} \quad (2.7)$$

where the probability $p$ depends on the distance between latent positions and $Y_{ij}$ are the usual binary variables denoting the existence of edge $e_{ij}$.
For instance, considering $\theta = (\alpha, \beta)$ we can define $p$ through the *logit* function:

$$\text{logit } (p(Y_{ij} = 1|z_i, z_j, x_i, x_j, \alpha, \beta)) = \alpha + \beta g(x_i, x_j) - \|z_i - z_j\| =: \eta_{ij} \quad (2.8)$$

where $g(\cdot)$ is a generic function used to evaluate the similarity between the observable features and $\|\cdot\|$ is the Euclidean distance. By injecting the probability value from Eq. 2.8 into Eq. 2.7, we finally get

$$p(Y|\eta) = \prod_{i \neq j}^{n} \frac{e^{\eta_{ij} Y_{ij}}}{1 + e^{\eta_{ij}}} \quad (2.9)$$

Latent Space Models are used to estimate the set of parameters and distances such that $\eta_{ij}$ maximize the likelihood of the observed graph: the latent coordinates are more than simple mathematical tools, since they are the actual objective of the statistical model. The reason is simple: the inferred spatial representation can offer new insights related to the graph topology, for instance by providing a clear representation of the unobservable node space and a quantitative evaluation of the structural similarity of the vertices (see Sec. 1.4).

Given their importance for our dissertation, Ch. 3 will be entirely dedicated to Latent Space Models, with analyses of further details related to space dimension, type of distance and estimation techniques.
For the time being, let us conclude this brief introduction with an interesting remark: the pairwise distances determined by the model allow us to identify some set of "similar" nodes through clustering algorithms. In this regard, latent space models may be used to achieve the same goals of the a posteriori blockmodeling we have introduced for stochastic blockmodels: actually, a comparison between the two methods can be truly interesting, since the partitions are inferred by exploiting completely different methods. However, while stochastic blockmodels are only able to split the nodes into suitable clusters, LSM can also provide valuable information related to the cluster's shape and density, since to each vertex is associated a precise location in the latent space.

## 2.3 Dynamic Network Models

In the models we have seen so far, the number of vertices and edges are fixed: the network is observed only once, thus they cannot change over time, as well as the parameters that must be estimated. However, since many real world scenarios (e.g., the World Wide Web or friendship relations) are time dependent, it is absolutely important to learn how to deal with dynamic networks. In graph terms, dynamics can be translated into the creation and destruction of both edges and nodes.
Static models have proved to be extremely efficient in understanding *how* a graph and its form can be described, but they failed in giving a complete explanation of *why* a network should have such a form in the first place [34].
Dynamic models are able to accomplish this task by means of generative processes: the statistical analysis of time-dependency provide new insights that may add great value to the considerations coming from the inspection of nodes characteristics and edges spatial distributions. The generative processes introduced for some models from the previous section need to be extended, so that the continuity of evolution can overcome the limits imposed by a particular realization. Actually, whenever a generative process is used to describe a static network, the underlying assumption is that the observed data can be seen as a specific configuration in time of an evolving network: in this regard, generative processes for static models are a hybrid class that exploits some dynamic features for a better understanding of the observed reality.

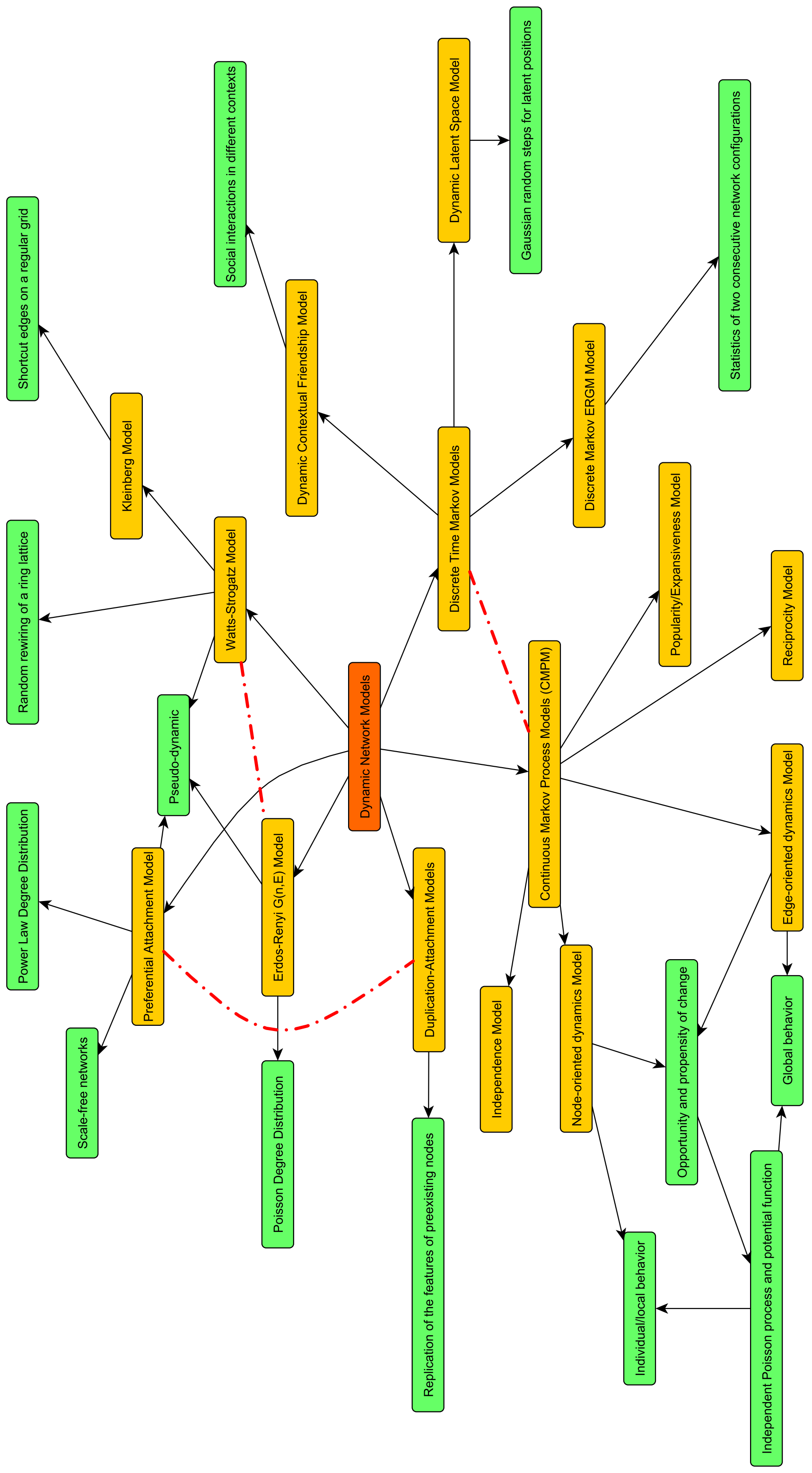
Dynamic Network Models
Watts-Strogatz Model
Kleinberg Model
Shortcut edges on a regular grid
Random rewiring of a ring lattice
Pseudo-dynamic
Preferential Attachment Model
Power Law Degree Distribution
Scale-free networks
Erdos-Renyi G(n,E) Model
Poisson Degree Distribution
Duplication-Attachment Models
Replication of the features of preexisting nodes
Discrete Time Markov Models
Dynamic Contextual Friendship Model
Social interactions in different contexts
Dynamic Latent Space Model
Gaussian random steps for latent positions
Discrete Markov ERGM Model
Statistics of two consecutive network configurations
Continuous Markov Process Models (CMPM)
Popularity/Expansiveness Model
Reciprocity Model
Edge-oriented dynamics Model
Global behavior
Independence Model
Node-oriented dynamics Model
Opportunity and propensity of change
Individual/local behavior
Independent Poisson process and potential function

Figure 2.5: Dynamic Network Models concept map

In the following sections, we will present the most important dynamic networks models, focusing especially on those that present a conceptual link with the examples from Sec. 2.2. Such links are represented by blue dotted lines in Fig. 2.1, while a focus exclusively on dynamic models is given by the map in Fig. 2.5 with the same graphical notation we used for static models.

### 2.3.1 Erdös-Rényi $\mathcal{G}(N, E)$ Model

As we did for static models, due to simplicity reasons we will begin our analysis by looking at the well-known Erdös-Rényi model: in particular, we will pay attention to its $\mathcal{G}(N, E)$ version (which is the properly called Erdös-Rényi Model). Let us remind the formal definition from Sec. 2.2.1:

**Definition 2.3.1** (Erdös-Rényi Model). *Given $N \in \mathbb{N}$ and $E \leq \binom{N}{2}$ , the graph $\mathcal{G}(N, E) = (\mathcal{V}, \mathcal{E})$ is defined as the undirected graph with $N$ vertices such that the $E$ edges are chosen randomly from the possible $\binom{N}{2}$ edges.*

Unlike Gilbert's model (also known as the Erdös-Rényi $\mathcal{G}(N, p)$ version), the $\mathcal{G}(N, E)$ one can be reinterpreted as a dynamic process:

- at time $t = t_0$ we have $N$ unconnected nodes;
- at each subsequent time step we randomly select a couple of vertices that have yet to be linked and add the corresponding edge to $\mathcal{E}$ until $|\mathcal{E}| = E$.

We could argue that also Gilbert's model could become dynamic by considering each possible edge at a different time instant: however, since the link creation probability would remain equal to $p$, this procedure would not change the static nature of the model. On the other hand, while the a priori probability of observing an edge between a given couple of nodes in the $\mathcal{G}(N, E)$ model is equal to $p_0 = E/\binom{N}{2}$, during the creation process this value becomes $p_t = (E - t)/(\binom{N}{2} - t)$ where $t$ denotes the number of edges already formed. The probability value changes dynamically over time, which is why it makes sense to consider such a formulation.
The $\mathcal{G}(N, E)$ model is surely simple and easy to study, but it does not address many issues present in real network dynamics. Actually, this formulation has two huge limitations:

- the number $E$ is fixed a priori, meaning that the dynamic graph will actually become static once all the edges have been added;
- situations where the number of nodes could change over time cannot be considered, since $N$ is assumed to be constant.

Furthermore, the model cannot produce graphs characterized by a power law degree distribution, which is a necessary feature for scale-free networks. For this reason, much subsequent research has focused on developing a model which could well represent power law distributions and relax the $\mathcal{G}(N, E)$ constraints.

### 2.3.2 Preferential Attachment Model

Barabási and Albert developed a model specifically designed to generate scale-free networks: starting from an initial graph, they kept adding new nodes and creating links between them and the most connected ones.
Actually, this "rich-get-richer" phenomenon had already been studied by the economist Herbert Simon and translated into a network context by Price in the 1970s [39]. Even if the mechanism was called *cumulative advantage*, the new name *preferential attachment* (PA) proposed by Barabási and Albert became the standard reference in the literature due to their outstanding contributions in the field.
More precisely, the PA generative process creates $c$ edges every time a new node $v^*$ is added to the network: the $c$ vertices that will be linked to $v^*$ are chosen accordingly to a multinomial distribution whose probabilities are $p_i = \frac{k_i}{\sum_j k_j}$ for each node $v_i$.
Thanks to the above procedure, the output (Fig. 2.6) is actually a network whose resulting degree distribution satisfy the equation of a power law [10], $P(k) = \alpha k^{-c}$.

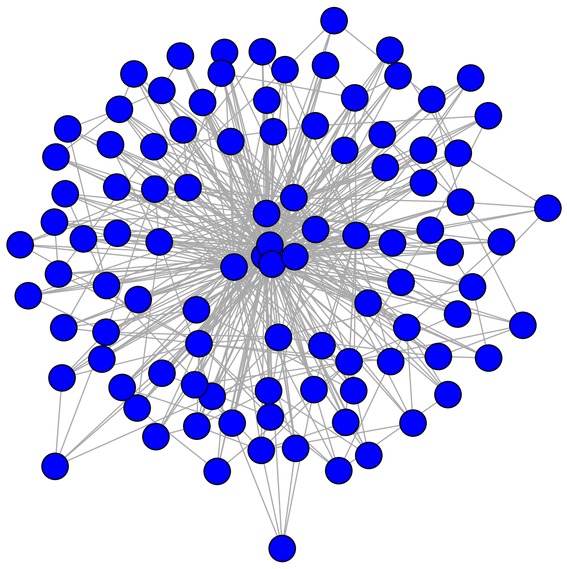

Figure 2.6: Example realization for the PA model

Some drawbacks are caused by the strong assumption related to the fixed number of connections $c$ created for each new node and the absence of specific features for each node: the model is not able to cope with scale-free networks that present slightly different features, which is why different generalizations or extensions of the model allow for flexible power-law exponents, edge modifications or non-uniform dependence on the node degree.

### 2.3.3 Duplication Attachment Model

A simple extension of the PA model is given by the Duplication Attachment model, which is based on a different link creation assumption. Instead of creating $c$ random connections following a multinomial distribution, each new node $v^*$ mimics the behavior of a randomly selected vertex $v'$.
The imitation (or duplication) is stochastic: assuming that in the starting network each node has at least $c$ edges, the $i$-th link from $v^*$ will reach either a random node (with probability $\alpha$) or the same destination of the $i$-th link from the model vertex $v'$ (with probability $1 - \alpha$).
Thanks to this mechanism, the model is able to reproduce the behaviors observed in networks like the web graph, where each new web page usually copies at least partially the hyperlinks of similar preexisting pages.

### 2.3.4 Small-World Models

As we have already introduced in Sec. 2.2.4, one of the least well-understood features of real-world networks is transitivity [34]: two neighbors of a vertex tend to be neighbors of one another, a behavior that significantly differs from a non-transitive null model like Erdös-Rényi $\mathcal{G}(N, p)$. Since transitivity translates into triad closures, the comparison between a real observed network and the data generated by a specific model is usually based on statistics quantifying the proportion of triad closures in the set of edges.

**Definition 2.3.2** (Clustering coefficient). *The local clustering coefficient of vertex $v_i$ is defined by:*

$$c_i = \frac{\textit{triangles connected to } v_i}{\textit{2-stars centered on } v_i} \tag{2.10}$$

High values of the clustering coefficient means that transitivity has a huge effect in the neighborhood of vertex $v_i$, while low values correspond to the opposite behavior. Real social networks often have values $c_i \in [0.4, 0.6]$: even if models like Erdös-Rényi or PA only produce networks with lower clustering coefficients, it is not difficult to come up with formulations that output the desired values

(e.g., a regular grid or a ring lattice). However, PA was able to reproduce important features of real graphs like their scale-free aspect, while a regular grid is usually far from reality: Small-World models try to find a compromise and output networks with high clustering coefficients that are also close to reality.
The Watts-Strogatz models begins with a ring lattice with $k$ edges per node and progressively rewires each edge with a fixed probability $p$: the output is some kind of mediation between the regular starting lattice and an Erdös-Rényi graph. The reason why this model is actually interesting is that its clustering coefficients are close to the starting ones, while other statistics like the *diameter* (i.e., the length of the longest shortest path) are relatively small, exactly as observed in many real networks.
The well known theory of the "Six degrees of separation" [26] provides an excellent example of what is called *small world property*: despite the size of the network, the length of the shortest path between most couple of vertices is usually very small.
Also Kleinberg's formulation exhibits this property: in his model, the basis is given by a fixed grid to which some random edges are added. As for the previous model, this process guarantees suitable values for both the clustering coefficient and the length of shortest paths.
Finally, more complex extensions of these basic models allow a better definition of the rewiring process leading to the creation of the so called *navigable networks* [43]: by choosing the right parameters, power law degree distributions may arise while maintaining the desired properties for the above mentioned statistics [6].

### 2.3.5 Continuous Markov Chain Models

Holland and Leinhardt [21] were the first to propose the use of continuous Markov processes to model dynamic networks. Thanks to their contribution, evolution dynamics began to appear as something much more complicated than simple generative algorithms based on nodes addition and edges rewiring.
Indeed, Markov assumptions allow graphs to evolve by adopting a new configuration with a probability that depends on some specifically defined transition mechanism and on the network current structure.

**Definition 2.3.3** (Continuous Markov process)**.** *Let $\{Y_t \mid t \in \mathcal{T}\}$ be a stochastic process, where $Y_t$ may take values in a countable (finite) space $\mathcal{Y}$ and $\mathcal{T}$ is a continuous time interval. $Y_t$ is a continuous Markov process if it satisfies the Markov property*

$$P\left(Y_{t_b} = y \mid Y_t = y_t, \forall t \leq t_a\right) = P(Y_{t_b} = y \mid Y_{t_a} = y_{t_a}) \tag{2.11}$$

*for any possible outcome $y \in \mathcal{Y}$ and any couple of time instants $t_a, t_b$ such that $(t_a < t_b \; | t_a, t_b \in \mathcal{T})$.*

Whenever the *transition probability* from state $y_t$ to state $y$ only depend on the difference $\delta_t = t_b - t_a$, the Markov process is said to be time-homogeneous. In this particular case, the matrix of probability transitions can be written as $P(t) = e^{tQ}$, where $Q$ is called *intensity matrix.* Its elements correspond to the rate of change of the probability of state transition.
Before introducing the main dynamic models, let us make a remark concerning the mathematical notation adopted to translate a stochastic process: the outcome space $\mathcal{Y}$ contains all the possible network configurations, thus any instance $y \in \mathcal{Y}$ will be characterized by a vector of $N(N-1)$ elements[3]. As for the intensity matrix $Q$, things get a bit complicated: each element $Q_{ij}$ should be related to the transition probability from state $s_i$ to state $s_j$, but this would produce a $N(N-1) \times N(N-1)$ matrix. Actually, the majority of dynamic Markov models solve this computational problem by recurring to specific assumptions that help reduce the number of needed probability transitions, so that usually there is no need for the whole intensity matrix.
However, it is extremely useful to denote the probability transitions, because a generalized notation will provide precious insights related to the differences between dynamic models. Following [16], let us focus on a single edge $e_{ij}$ and its associated random variable $Y_{ij}$: the propensity for $Y_{ij}$ to switch its value between 0 and 1 will be specified by a probability transition function $q_{ij}(\cdot)$. Being in a Markov time-homogeneous framework, the graph evolution only depends on the current configuration $y$, leading to $q_{ij} = q_{ij}(y)$. Thanks to this notation, we can express the intensity matrix as a set of N(N-1) intensity matrices of size $N \times N$. This formulation is not useful in practice since it has even more coefficients than the previous case, but if offers huge advantages in terms of problem formulation and may be significantly more tractable if specific assumptions are made.
For instance, let us consider the following examples:

- *independent arc model*: all edges are assumed independent, so $q_{ij}(y) = \lambda_{ij}$. The independence hypothesis leads to only one $N \times N$ intensity matrix, since the $N(N-1)$ possible configurations do not have any influence on the probabilities of transition. A slightly more complex model may distinguish between the type of transition, either $0 \rightarrow 1$ or $1 \rightarrow 0$: in this case, $q_{ij}(y) = \lambda_{y_{ij}}$ and $2N^2$ coefficients are needed;

- *reciprocity model*: $q_{ij}(y) = \lambda_{y_{ij}} + \mu_{y_{ij}} y_{ij}$. In this case, the existence of an edge between two nodes has an effect (positive or negative) on the change

[3]Directed graph, no loops allowed.

probability for the symmetric edge. Unlike the previous case, the intensity matrix does not exist in a closed form based on the above quantities;

- *popularity model*: $q_{ij}(y) = \lambda_{y_{ij}} + \pi_{y_{ij}} k_j^+$. The bigger the in-degree of vertex $v_j$, the higher the switch probability for $Y_{ij}$: the model induces a simple dependence on the node popularity;

- *edge-oriented and node-oriented dynamics*: the intensity matrix is actually split into two components, respectively accounting for *opportunity* and *propensity* of change (see [44] for further details).

### 2.3.6 Discrete Markov Chain Models

Time-homogeneous Markov processes on a countable state space can also be defined with respect to a discretized time: in this framework, the graph structure at time $t + 1$ depends only on its configuration at time $t$.
A simple dynamic extension of the ERGM is given by the following:

$$P(y_{t+1} \mid y_t) = \frac{1}{k} \exp\left\{ \sum_i \beta_i s_i(y_{t+1}|y_t) \right\} \tag{2.12}$$

where $s_i$ are network statistics accounting for features like reciprocity and transitivity, $\beta_i$ are weighting coefficients and $k$ is a normalizing constant.

### 2.3.7 Dynamic Latent Space Models

In a static context, latent variables are used to define better formulations for the edge probabilities, so that conditioning on hidden or unobserved values may facilitate the analysis of particular graph structures. Different network realizations are likely to produce different estimations for the latent variables, which means that a dynamic network must be characterized by a sequence of latent variables values.
Dynamic models are usually focused on identifying the evolution mechanism responsible for the transition of the graph structure $Y$ from $Y_t$ to $Y_{t+1}$: the change may be due to some specific rewiring rule or to more complex equations like Eq. 2.12, but the general idea is that the probability of observing a particular evolution depends on the current structure of the graph.
Dynamic Latent Space Models introduce an interesting idea: instead of focusing on the graph itself, they consider the probability of observing a certain shift in the node positions in the latent space. Therefore, the update of the hidden values depends both on the newly observed data and on the previous latent space positions [42]. Given a particular latent variable $h$, its value changes accordingly

to a probability distribution which usually takes into consideration the difference between $h_t$ and $h_{t+1}$.
The greater the difference, the more unlikely the change: the "continuity" of evolution of the graph structure is therefore reflected by a "continuity" in the latent space positions, which means that we can reconstruct the network dynamics by looking at the consecutive transformations in the latent space.
This is especially useful when the graph size does not allow an easy understanding of the changes occurring between two time instants. As we discussed in Sec. 1.3, we can associate different graphical representations to the same graph: we usually adopt the one which seems the most effective by an arbitrary choice, but there is no guarantee that similar networks will have similar representations.
It is enough to consider two connected graphs and their union by the addition of a single edge shown in Fig. 2.7: unless the new edge connects the two closest nodes (B) (with respect to the initially chosen graphical representation (A)), the addition of one link could drastically change the spatial positions of the nodes, since simply drawing a new link would result in a poor image (C). Moreover, the initial positions are arbitrary, so sticking to the initial representation is actually meaningless.

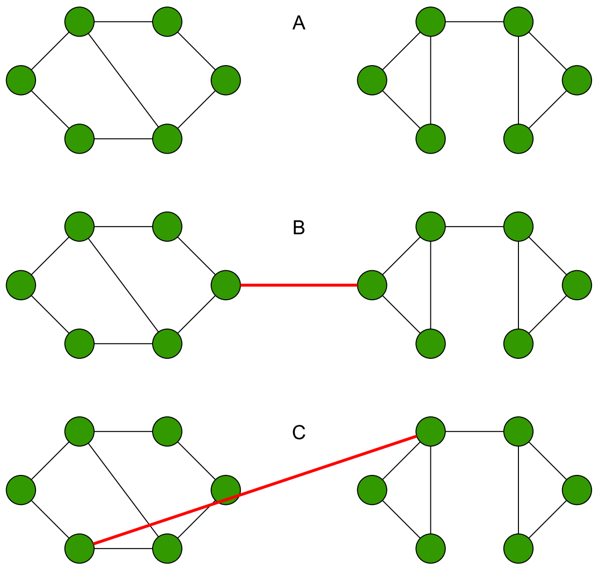

Figure 2.7: Illustrative example

Multiple evolutions could produce an incomprehensible output, while gradual and smaller changes controlled by the displacement probability in the latent space can offer a much more efficient description of the network dynamics: besides the advantages already presented for in the static framework, Latent Space Models can also provide useful tools for the visualization of dynamic data.

# Chapter 3

# Latent Space Models

## 3.1 Link probability and latent variables

Throughout Ch. 2 we considered different hypotheses concerning dependence among edges, gradually increasing the complexity of the assumptions and observing their impact on the graph structure. We remarked that whenever a network possesses non-trivial topological features the interaction between nodes cannot be explained by means of complete randomness: Gilbert's model is way too simple to adapt to real world scenarios, which is why we have to relax the requirement of completely independent edges and formulate instead models relying on hypotheses of conditional independence.
Following the usual notation, let us consider a couple of vertices $v_i$ and $v_j$: the conditioning of $Y_{ij}$ does not need to be limited to observable variables, but can also be extended to latent features. Furthermore, information may be node or edge specific, meaning that besides $x_i$ and $x_j$ we may also have to deal with $x_{ij}$, a term taking into account additional pair related information. More formally, let $x$ and $z$ denote respectively observable and latent variables: with the usual notation, $Y_{ij} \mid x_i, x_j, x_{ij}, z_i, z_j, z_{ij}, \theta$ are assumed independent, where $\theta$ is an unknown parameter (or vector of parameters) that must be determined. In the framework of Latent Space Models (LSM), the conditioning is usually restricted to $Y_{ij} \mid z_i, z_j, x_i, x_j, \theta$, where $x_i, x_j$ are observable features and $z_i, z_j$ are the latent positions in a multidimensional space:

$$P(Y|Z,X,\theta) = \prod_{i\neq j}^{n} p(Y_{ij}=1|z_i,z_j,x_i,x_j,\theta)^{Y_{ij}} p(Y_{ij}=0|z_i,z_j,x_i,x_j,\theta)^{1-Y_{ij}} \quad (3.1)$$

As we mentioned in Sec. 2.2.6, the main goal of LSMs consists in performing inference on the $z$ coordinates, since the inferred spatial representation can offer new insights related to the graph structure. For instance, once the network nodes have been properly mapped into the latent space, we can group nodes with similar behaviors thanks to clustering algorithms or identify regions that may carry relevant meaning with respect to a specific dataset. A potential weakness lies in the fact that without additional knowledge the spatial outputs may prove very difficult to interpret, but, even in this case, LSMs still preserve their greatest strength: unlike other approaches, they provide a visual and interpretable spatial representation of data relations and allow the statistical uncertainty to be quantified and represented graphically.

In the following sections, we will discuss common choices in terms of latent variables type and dimensionality and explain how *proximity data* can be inferred from graph information and converted into positions in the latent space (Sec. 3.2). We will also focus on the matter of equivalent spatial representations (Sec. 3.3) and present initialization techniques and Markov Chain Monte Carlo algorithms for the estimation of latent positions (Sec. 3.4).
We will follow the work by Hoff, Raftery and Handcock presented in [20] as principal reference throughout the chapter and we will show the results obtained by implementing their model on four different benchmark datasets (Sec. 3.5). Finally, since the code retrieved from `http://www.stat.washington.edu/people/pdhoff/Code/hoff_raftery_handcock_2002_jasa/` was partially rewritten, we will also provide a brief description of the major changes that we have introduced.

## 3.2 Multidimensional scaling and initialization

Multidimensional scaling (MDS) is quite a popular technique for exploratory data analysis and visualization [3, 4]. Rather than focusing on different features from multivariate data, MDS typically takes as input a measure of the global similarity or dissimilarity of the stimuli or objects under investigation. The primary outcome of the MDS analysis is a spatial configuration [49] in which the objects are represented as points in $\mathbb{R}^k$. These points are arranged in such a way that their distances correspond to the similarities of the objects: to put it simply, similar objects are represented by points that are close to each other, while dissimilar objects by points that are far apart.
Since any kind of similarity can be translated into dissimilarity through an appropriate monotone decreasing function, we can restrict our analysis to the so called *proximity data* without loss of generality. Such data consist of dissimilarity values for pairs of objects: assuming their labels are $i = 1, 2, \dots, N$, we will denote each proximity value by $D_{ij}$. Thanks to MDS, the objects can be mapped into a $k$-dimensional space where the given dissimilarities $D_{ij}$ are well-

approximated by the distances $\|x_i - x_j\|$ [1] (usually Euclidean). The choice of the embedding dimension $k$ is arbitrary in principle, but low in practice because points mainly serve as an intuitive visual representation of the objects.
Latent Space Models follow essentially the same idea: given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we are interested in finding the latent positions that minimize the discrepancies between dissimilarity values and distances in the latent space. However, there is a huge difference between MDS and LSM: the former takes proximity data as input, while the latter deals with network data. It is therefore necessary to define a method or function that "extracts" proximity values from information related to edges and connected components. For instance, we can assign to each pair of vertices the length of the shortest path between them (this is actually a very common choice, since in this way neighbors will have the lowest dissimilarity).
Even if different methods can be used to determine proximity values from network data, we should always check two requirements:

- dissimilarity values $D_{ij}$ have to satisfy the triangle inequality, because we want to find positions $x_i$ and $x_j$ such that $\|x_i - x_j\|$ is as close as possible to $D_{ij}$. Indeed, due to the fact that $\|x_i - x_l\| \leq \|x_i - x_j\| + \|x_j - x_l\|$ for every norm defined on the latent space, the contradiction determined by assuming $D_{il} > D_{ij} + D_{jl}$ would produce poor approximation results.

- the network must be *representable*, meaning that for each vertex $v_i$ there exist a point $x_i$ in the latent space $\mathbb{R}^k$ such that

$$\|x_i - x_j\| > 1 \ \forall i, j \text{ s.t. } Y_{ij} = 0 \text{ and } \|x_i - x_j\| < 1 \ \forall i, j \text{ s.t. } Y_{ij} = 1 \quad (3.2)$$

  More generally, depending on the parametrization used in a model we could replace 1 with any constant simply by rescaling the $x_i$ points: whatever the value $\alpha$, the parameter acts as a threshold allowing connections only between nodes whose relative latent distance is smaller than $\alpha$.

Whenever these two conditions hold, we remark that the model is inherently reciprocal and transitive thanks to symmetry of distances and closeness of triads in the latent space: this is absolutely important, since reciprocity and transitivity are both features observed in real world networks.
As we stated in the previous section, the main goal of LSM is to perform inference on latent coordinates: the log-likelihood of a conditional independence model is relatively simple, therefore likelihood-based estimation methods such as maximum likelihood and Bayesian inference are feasible [20]. However, since

[1]These positions may actually be found by many different methods such as Classical MDS, ALSCAL or Bayesian MDS [35], but their discussion goes beyond the scope of the dissertation.

the log-likelihood is not generally concave, the identification of a global maximum likelihood estimator is quite problematic: a possible approach consists in identifying a set of distances $d_{ij}$ not necessarily Euclidean that maximizes the likelihood and then apply MDS to find a set of positions in $\mathbb{R}^k$ that well approximate those distances. The set of estimated positions in the latent space can then be used a starting point for a nonlinear optimization routine. The model by Hoff, Raftery and Handcock we chose to present uses an even simpler approach: since determining $d_{ij}$ as a convex minimization problem may still be computationally heavy, dissimilarities values $D_{ij}$ are used in place of $d_{ij}$. We only have to pay attention to the two constraints listed before, but generally measures like the *geodesic distance* (or shortest path length in the network case) provide us with good estimation for the starting points in the latent space.

## 3.3 Equivalent representations

In the LSM framework, the link probability between two vertices depends on the distance between the corresponding positions in the latent space (Sec. 2.2.6):

$$\text{logit } (p(Y_{ij} = 1 | z_i, z_j, x_i, x_j, \alpha, \beta)) = \alpha + \beta g(x_i, x_j) - \|z_i - z_j\|$$

The distance in the $k$-dimensional Euclidean space is invariant under reflection, translation and rotation: given a configuration estimated by the model, we can thus find an infinity of equivalent mappings simply by applying some linear transformation to our set of points. Picking a representation out of all the possible ones may only seem like a minor problem of convention given that the set of distance is always the same, but this equivalence actually causes some critical issues. Indeed, since LSM uses a Markov Chain Monte Carlo (MCMC) algorithm to perform Bayesian inference, for each iteration we could have similar distances but completely different mappings, which would bring us to overestimate the variability of the unknown positions. Credible regions based on the posterior distribution could actually lose their meaning, because even by centering the latent points we could not eliminate the random rotation.
As an example, let us consider five points labeled $A, B, C, D, E$ placed on a ring so that the letters are alphabetically ordered with respect to the clockwise rotation (Fig. 3.1).
Due to their positions, if these points were actually the outcome of a Markov chain iteration from a LSM, then the originally observed graph should have the following set of undirected edges: $(A, B), (B, C), (C, D), (D, E), (E, A)$. Then, as a result of the LSM inference we would like to have five separated credible

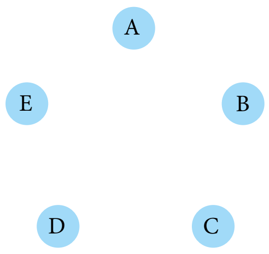

Figure 3.1: Clockwise ordered points

regions, suggesting that each node has a well defined behavior with respect to the other vertices. We would have to keep in mind that any rotation or translation of the output latent representation could still represent the initial graph, but in any case the relative behaviors among nodes would be correctly captured by the model. Unfortunately, due to the equivalence of rotated sets of points, the MCMC algorithm randomly chooses a different rotation for each iteration, so that the credible regions for the five points eventually coincide and look like the blue area in Fig. 3.2.

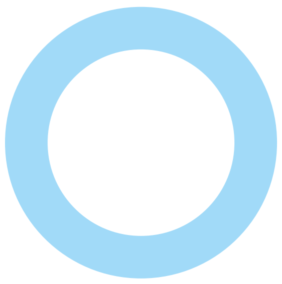

Figure 3.2: Overlapping of credible regions

Such regions do not allow us to recognize any specific behavioral pattern: not only the alphabetical order is clearly lost, but we do not even have any information related to the pairs that are more likely to form a connection. Truth to be told, in the algorithm we implemented the rotation is due to a more complicated process, but the resulting effect is essentially the same: as a proof, we will later see a simulated output in Fig. 3.6.
In order to avoid such a poor conclusion we need to introduce a fundamental constraint, so that we will be able to perform inference on particular configurations that are comparable. Each MCMC iteration introduces some node-specific stochastic updates in terms of latent positions with respect to the previous coordinates: considering vertex $v_i$ and iteration $k+1$, we can write $z_i^{k+1} = z_i^k + \delta_i^{k+1}$. The last term $\delta_i^{k+1}$ can be further decomposed into $\delta_{i,n}^{k+1} + \delta_{i,s}^{k+1}$, where the indices

$n$ and $s$ refer to *node* and *set*: $\delta_{i,n}^{k+1}$ is truly node-specific, while $\delta_{i,s}^{k+1}$ accounts for a change affecting the whole set of coordinates, like a common translation or rotation. Since we are only interested in the variability regarding the relative positions, we have to neglect the $\delta_{i,s}^{k+1}$: in order to do so, we need to introduce some new concepts.
Let us denote by $Z$ a spatial representation and by $|Z|$ its equivalence class, whose members are the spatial representations equivalent to $Z$ under reflection, rotation and translation. Let us define a fixed set of positions $Z_0$: then, at each iteration of the MCMC, we will consider as representation the element $Z^* \in |Z|$ which is the closest to $Z_0$ in terms of sum of *squared positional difference.* In other words, we choose

$$Z^* = \left(z^*_{\pi(1)}, z^*_{\pi(2)}, \dots, z^*_{\pi(n)}\right) = \arg\min_{\pi, Z'} \sum_{i=1}^{n} \|z_{0,i} - z'_{\pi(i)}\|^2 \tag{3.3}$$

where $z_{0,i}$ denotes the coordinates of the $i$-th point of $Z_0$, $(z'_1, z'_2, \dots, z'_n) = Z' \in |Z|$ and $\pi$ is a permutation of the $n$ indices. This permutation is needed since the ordering of the $n$ locations is merely a convention: for this reason, the sum of squared distances must be minimized with respect to the most suitable re-ordering, so that the initial labeling of the points will not produce biased results. Going back to the previous example, this means that we can fix an initial set of positions (e. g., the one in Fig. 3.1) and then identify for each MCMC output the element from its equivalence class that is the closest to our fixed configuration, avoiding the randomness of rotation and thus the poor credible region shown in Fig. 3.2.
More formally, we say that $Z^*$ is the result of a *Procrustean transformation* of $Z$ whose objective is $Z_0$ [20], where the term Procrustean is used to designate a procedure that only involves rotation, reflection and uniform scaling [2].
A formulation equivalent to Eq. 3.3 is given by

$$Z^* = \arg\min_{T} tr\big((Z_0 - TZ)'(Z_0 - TZ)\big) \tag{3.4}$$

where $T$ ranges over the set of rotations, reflections, translations. If $Z_0 Z'$ is nonsingular then $Z^*$ is unique and relatively easy to compute: assuming all configurations are centered at the origin, we obtain $Z^*$ as

$$Z^* = Z_0 Z'(Z Z_0' Z_0 Z')^{-1/2} Z \tag{3.5}$$

To conclude the section, let us make a remark concerning $Z_0$. The configuration we use as reference only provides an orientation to the MCMC outputs, especially

[2]To be precise, following [20] we do not allow scaling.

since we are not allowing any rescaling. For this reason, we could virtually use any (meaningful) configuration such that $Z_0 Z'$ is nonsingular: however, if $Z_0$ is sufficiently close to the actual MCMC realizations in terms of sum of squared positional differences then the Procrustean procedure is more robust. A common practice consists in setting $Z_0$ equal to the initial estimator of latent positions we discussed at the end of Sec. 3.2, since the starting point for the nonlinear optimization routine should be at least an educated guess of the final configuration.

## 3.4 Sampling process

Once the starting point of the Markov chain has been identified and the constraints in terms of reflection, translation and rotation have been imposed, the sampling from the posterior distribution is usually performed by a Metropolis-Hastings (MH) algorithm.
The general outline (see [19, 24] for details) can be described as follows:

```
Algorithm 1: Metropolis-Hastings
  Initialize $x^{(0)} \sim f(x)$
  for $i = 1, 2, \ldots, N$ do
      Propose $x^* \sim J_i(x \mid x^{(i-1)})$
      Acceptance Probability :
          $\alpha(x^*|x^{(i-1)}) = \min\left(1, \frac{J_i(x^{(i-1)}|x^*)p(x^*|\mathbf{y})}{J_i(x^*|x^{(i-1)})p(x^{(i-1)}|\mathbf{y})}\right)$
      Sample $u \sim \text{Unif}(0, 1)$
      if $u < \alpha$ then
          accept $x^*$ : $x^{(i)} = x^*$
      else
          reject $x^*$ : $x^{(i)} = x^{(i-1)}$
```

Let us consider the variables and probability distributions introduced by the MH algorithm:

| | |
|---|---|
| $x^{(i)}$ : | simulated value at iteration $i$; |
| $f(x)$ : | distribution for the sampling of $x_{(0)}$. The starting value can also be chosen through a deterministic procedure; |
| $J_i$ : | proposal distribution at iteration $i$. Usually, $J_i = J$, meaning that such distribution is stationary. The Metropolis algorithm is a particular case where $J$ is also symmetric, i.e. $J_i(a\|b) = J_i(b\|a)$; |

**y** : observed data, usually written as $Y$ when referring to the adjacency matrix of the observed network;

$\alpha$ : acceptance probability, defined as the minimum between 1 and the ratio $\frac{p(x^*|\mathbf{y})}{p(x^{(i-1)}|\mathbf{y})}$ weighted by the effect of the proposal distribution. This value is used to assess the plausibility of candidate point $x^*$ relatively to $x^{(i-1)}$.

In the following sections, we will provide specific settings for the Metropolis-Hastings and illustrate how the acceptance probability can be rewritten in a more suitable form for the considered model.

## 3.5 LSM by Hoff, Raftery and Handcock

Up to this point, the chapter was meant to introduce and explain the main matters related to initialization, representation issues and sampling process: we will now proceed by giving a complete yet synthetic overview of the model described in [20] and showing some results of the model application on four benchmark datasets.

**Conditional independence** The general LSM assumption $Y_{ij} \mid z_i, z_j, x_i, x_j, \theta$ is simplified and re-labeled into $Y_{ij} \mid z_i, z_j, \alpha$, leading to

$$P(Y|\alpha, Z) = \prod_{i \neq j}^{n} p(Y_{ij} = 1|\alpha, z_i, z_j)^{Y_{ij}} p(Y_{ij} = 0|\alpha, z_i, z_j)^{1-Y_{ij}} \tag{3.6}$$

which expresses the overall likelihood of a network configuration conditioned on the latent space positions.
The probability $p$ is actually defined through the *logit* function:

$$\text{logit } (p(Y_{ij} = 1 \mid \alpha, z_i, z_j)) = \alpha - \|z_i - z_j\| \tag{3.7}$$

where the latent positions are limited to the bidimensional or tridimensional case. Eq. 3.7 can be rewritten as

$$p(Y_{ij} = 1 \mid \alpha, z_i, z_j) = \frac{e^{-\|z_i - z_j\|}}{e^{-\alpha} + e^{-\|z_i - z_j\|}} \tag{3.8}$$

where the conditional dependence is expressed even more clearly. We remark that in these equivalent formulations the differences in terms of link probability between vertices are only due to the Euclidean distances in the latent space, since $e^{-\alpha}$ is a common term. Following the short discussion about Eq. 3.2, this parameter can be interpreted as the threshold below which two vertices should be connected and disconnected otherwise: $\alpha$ must not be regarded as something either right or wrong, since it only serves as a qualitative index for the relative positions and does not provide all the information needed for the computation of the overall likelihood.

**Initialization** The model defines the dissimilarities between vertices by creating a proximity matrix $D$ in which element $D_{ij}$ corresponds to the length of the shortest path between $v_i$ and $v_j$ (geodesic distance). Then classical MDS is applied in order to determine the starting positions in the latent space.

**MCMC algorithm** The a posteriori distributions of the latent positions are obtained by means of a MCMC algorithm: the final estimates of the latent coordinates are inferred by taking the average over all the realizations.
Metropolis-Hastings is used for both $\alpha$ and the latent positions $Z$: their update is based on two symmetric distributions, respectively a Uniform and a Gaussian proposal. For instance, we have $J(z^*|z) = \mathcal{N}(z^* - z; 0, \sigma) = \mathcal{N}(z - z^*; 0, \sigma) = J(z|z^*)$ : an algorithm based on similar proposals ($J$ symmetric, $J(a|b) = J(a - b)$) is called random walk MH.
The $\alpha$ parameter has an exponential prior, while diffuse independent normal priors are used for the components of $Z$, so that the latent coordinates will not scatter too much.
Finally, whenever the proposed update is accepted, a Procrustean transformation is performed.
Let us take a closer look at the algorithm:

1. at iteration $k$, sample a value $\hat{Z}$ from $J(\hat{Z}|Z_k)$, where $J$ is a symmetric stationary proposal distribution;

2. accept $\hat{Z}$ as $Z'_{k+1}$ with probability $\min(1, P_k)$, otherwise set $Z'_{k+1}$ equal to $Z_k$. Using the usual notation in matrix form, $P_k$ is given by

$$P_k = \frac{p(\hat{Z}|Y, \alpha_k) J_k(Z_k|\hat{Z})}{p(Z_k|Y, \alpha_k) J_k(\hat{Z}|Z_k)} = \frac{p(Y|\hat{Z}, \alpha_k)\pi(\hat{Z})}{p(Y|Z_k, \alpha_k)\pi(Z_k)} \tag{3.9}$$

   where $\pi(\cdot)$ is the prior for the latent positions (assumed independent from $\alpha$);

3. if $Z'_{k+1} \neq Z_k$ then $Z_{k+1} = \arg\min_T tr\left((Z_0 - TZ'_{k+1})'(Z_0 - TZ'_{k+1})\right)$, being $T$ the same matrix of Eq. 3.4;

4. $\alpha$ is updated with a classic Metropolis-Hastings algorithm.

Once again, the whole process follows a MH procedure, with the exception that every accepted proposal must be processed by a Procrustean transformation.

**Final estimation** Each position is defined as the average over all the realizations from the Markov chain. In the paper, credible regions are simply assumed to be represented by the simulated positions, without any further analysis.

### 3.5.1 R Code

The original code provided by Hoff[3] had to be slightly modified:

- the initial optimization phase presented some problems related to methods relying on finite differences, so we changed approach and switched from BFGS (an approximation of Newton's method) to Nelder-Mead [32] (also known as *downhill simplex*);
- we improved the running time of the algorithm by resorting to the R package *Rcpp* and converting some essential functions in C++, especially those frequently called during the MCMC procedure. We used the C++ template library *Eigen* to deal with matrix computations;
- we added a new part concerning the final visualization, since the preexisting code simply showed the points simulated by the Markov chain.

Concerning the second point, we actually went further and rewrote all the core functions in C++: however, since the advanced optimization is not necessary to run the code on the examples from the following sections, we decided to postpone the description of our final code in App. A. The code presented and commented corresponds to the functions we are currently releasing as part of the R package BLSM (Bayesian Latent Space Model).

### 3.5.2 Benchmark datasets

The basic optimization of the code (the closest to the H-LSM original version) was run on four different datasets, progressively increasing the size and the complexity of the graph. The first two examples are quite comparable, since their incidence matrices are respectively $16 \times 16$ and $34 \times 34$, while the third one considers 379 nodes corresponding to the biggest connected component of a coauthorship network. Finally, the fourth dataset has an intermediate size ($115 \times 115$) and it is a well-known example from the field of network communities. The first three examples will only serve as a qualitative validation of the model, while the last test will allow us to quantitatively measure the performance of LSM on a community detection task.

[3]`http://www.stat.washington.edu/people/pdhoff/Code/hoff_raftery_handcock_2002_jasa/`

#### 3.5.2.1 Florentine Families

In 1993, Padgett and Ansell compiled data on marriage and business relations between 16 historically prominent Florentine families [36]. Hoff, Raftery and Handcock used a slightly modified version of this dataset (they removed a family disconnected from all the others) to test their model, which is actually focused on social networks analysis. The small size of the network (which we will refer to as FF) allows for a detailed inspection of the behavior of each vertex, so that we can check by hand the goodness of the spatial representation. The aim of this trial is to compare the results obtained from the modified code with the ones presented in [20] as a mean to verify the correct implementation of the LSM.
We present the results obtained by running $2 \times 10^6$ iterations of the sampling process described in Sec. 3.5, setting the *burn-in* period to $10^5$ and the *thinning* to 1:1000.
Some diagnostics concerning the MCMC algorithm are shown in Fig. 3.3: the parameter space is well explored by the $\alpha$ value and its autocorrelation function (ACF) plot confirms the fact that the *mixing* of the Markov chain is satisfactory. Finally, the *traceplots* of the whole set of $X$-coordinates can be observed in Fig. 3.4.

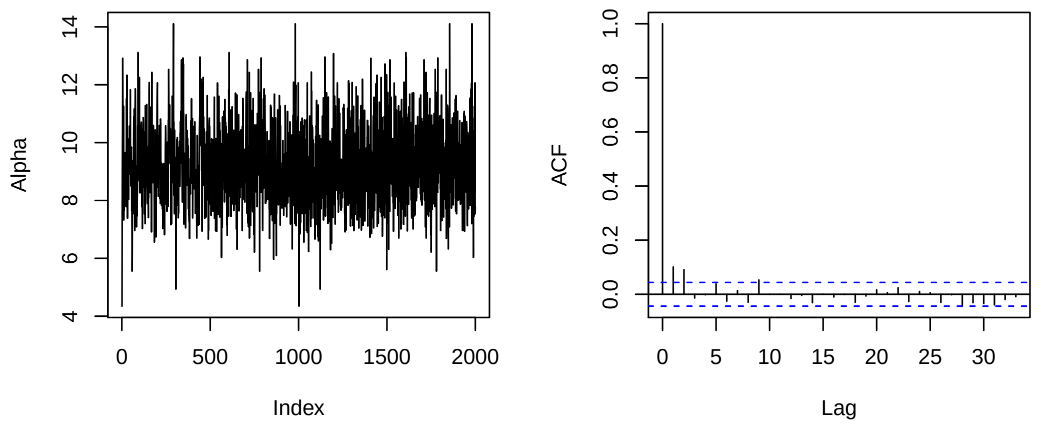

Figure 3.3: Traceplot and autocorrelation function for the $\alpha$ parameter

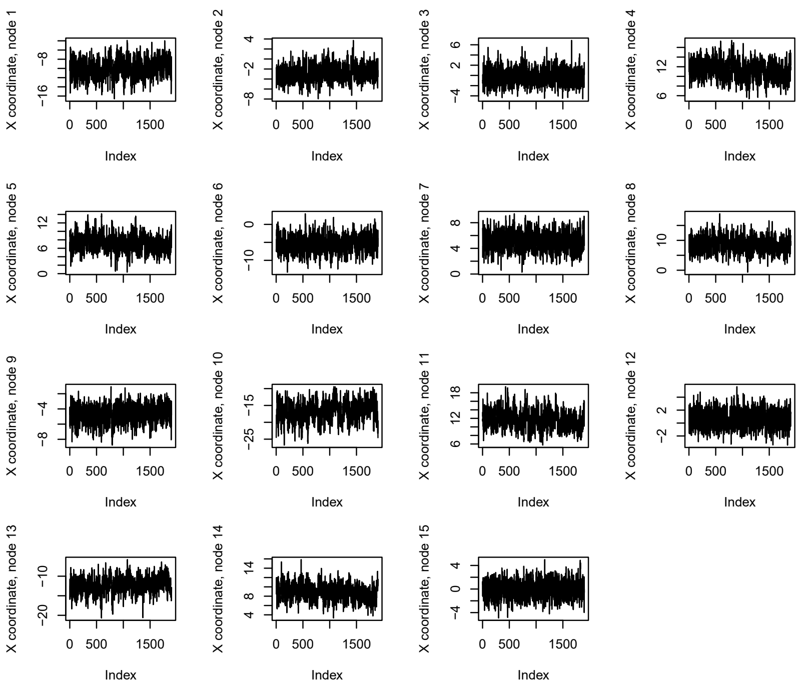

Figure 3.4: Traceplots for the $X$-coordinates of the 15 nodes from FF data

Fig. 3.5 shows the 2000 realizations in the latent space, where each color is related to a different node and each colored point corresponds to one of its stochastic positions. We can actually see that there are some regions that present an higher density for points related to a specific node: this is possible thanks to the Procrustean transformation, since without its "regularization" we would get the incomprehensible output represented in Fig. 3.6 (as argued in Sec. 3.3).
The main purpose of Latent Space Models consists in locating the vertices in the latent space so that their relative distances reflect their structural similarity: the closer the nodes, the higher the probability of having them linked. The shape of the colored regions is defined by the a posteriori distributions, so we can infer the unobservable position of each vertex by taking the average over all the MCMC realizations (as illustrated by the yellow dots).
Before continuing with our analyses, let us discuss a crucial feature of Fig. 3.5: blue lines represent the observed network and are only used as an intuitive tool to prove the relation between distance in the latent space and link probability. The observed network does not possess any intrinsic spatial configuration (see discussion from Sec. 1.3): the only reason why we are plotting the graph structure is that otherwise we would not be able to appreciate the strength of the latent mapping. Indeed, Latent Space Models allow us to find a meaningful representation: we can remark that vertices connected in the observed graph tend to be close to each other in the latent space, while non-neighboring nodes have greater relative distances.

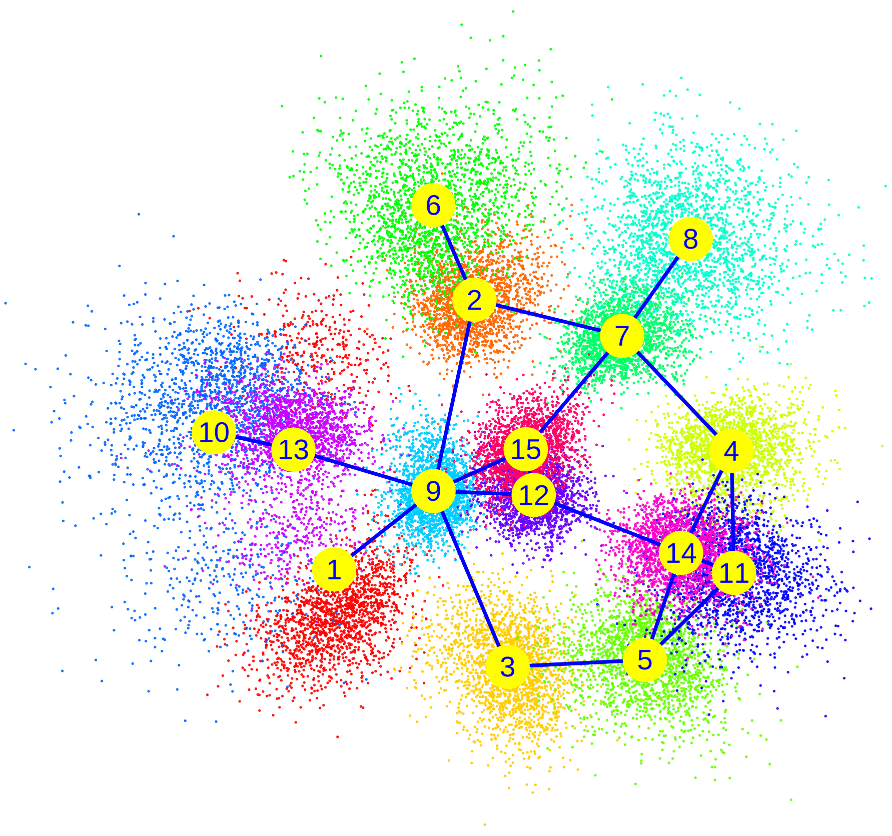

Figure 3.5: MCMC realizations for the Florentine Families dataset. Colored points correspond to stochastic positions and yellow dots indicate the average position for each node. Blue lines represent the observed network.

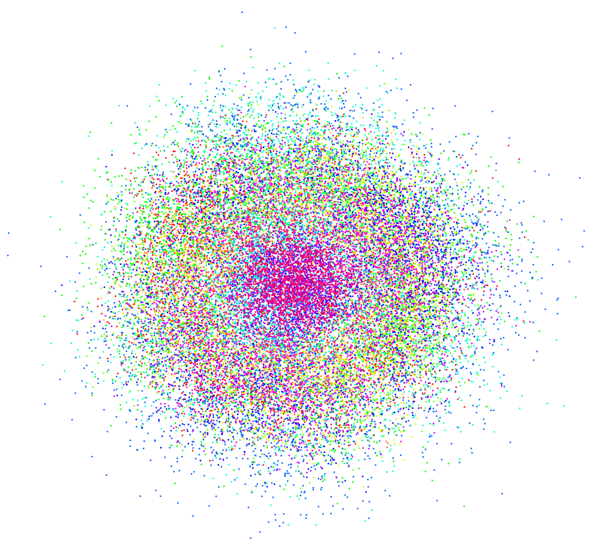

Figure 3.6: MCMC realizations for the Florentine Families dataset without the additional step of Procrustean transformation: the well defined colored regions from Fig. 3.5 are clearly lost.

As we discussed before, the latent reference configuration $Z_0$ used as the objective of the Procrustean procedure was determined by applying Classical Multidimensional Scaling (CMDS) on the proximity data based on the length of the shortest paths.
Fig. 3.7 seems to confirm the validity of this approach: the starting positions are qualitatively similar to the final estimates, meaning that they can be successfully used as the objective of the Procrustean transformation (see the last paragraph from Sec. 3.3 for further details).
In the picture, the red dotted lines highlight the difference between initial (right after CMDS) and final (at the end of the MCMC algorithm) position for each node, while the blue and black lines are simply used to represent the Florentine Families network respectively in the initial and final configuration.
We remark that the final estimates do not coincide with the starting positions: CMDS is useful but not sufficient as a mean to represent the network in the latent space. Indeed, the LSM nonlinear optimization following the initialization is necessary to identify the best configuration in terms of the graph likelihood.

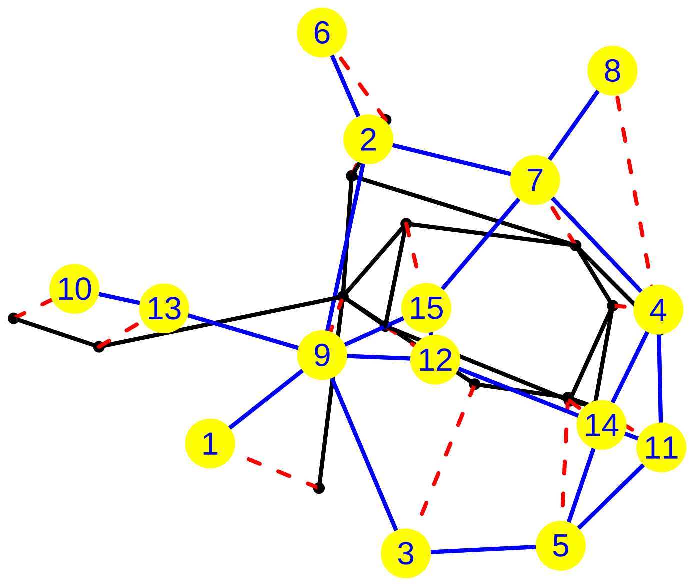

Figure 3.7: Comparison between initial and final latent positions: red dotted lines highlight the difference between initial and final position for each node, while the blue and black lines are simply used to represent the Florentine Families network respectively in the initial and final configuration.

Finally, let us take a closer look at the density distributions of the latent positions in $\mathbb{R}^2$: as we can observe from the *contour* plot in Fig. 3.8, the distributions seem to be all unimodal except for the one related to node 1 in Fig. 3.5 (red points on the middle/left side).

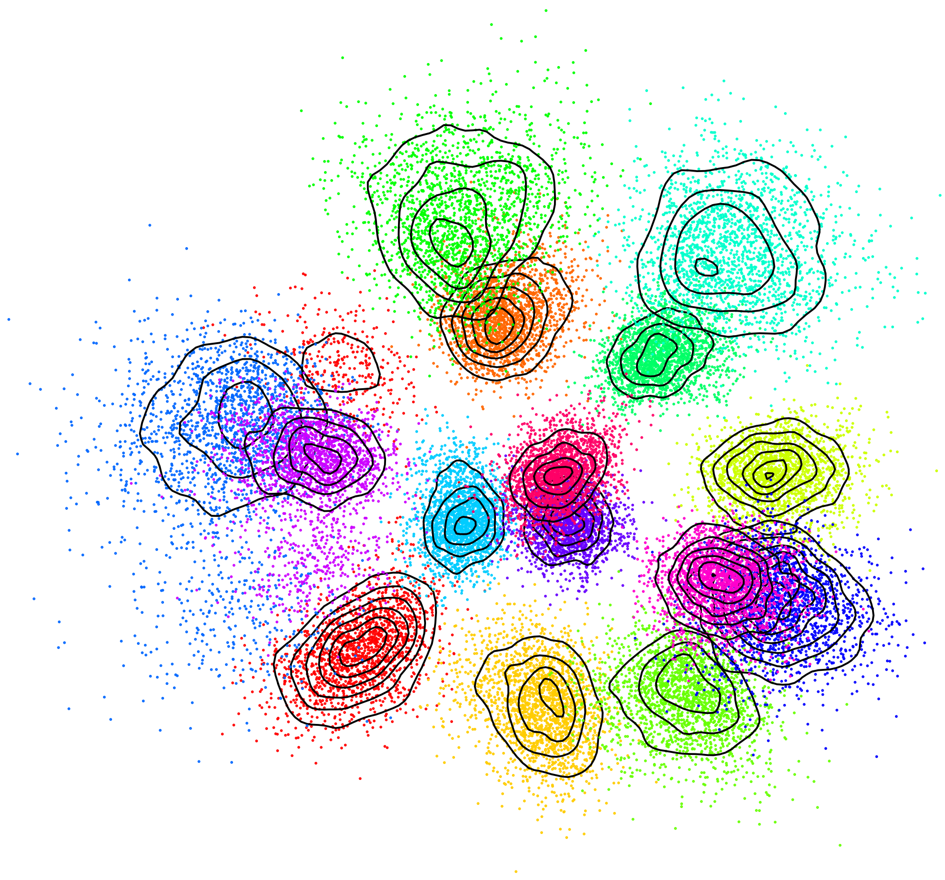
Figure 3.8: Contour plot for the Florentine Families dataset

This may appear like an error, but there is actually a reason: as explained in [20], if we were to swap the position of 1 with those of 10 and 13 in Fig. 3.5 we would get a configuration that still represents the network, meaning that the points $x_i$ satisfy $\|x_i - x_j\| > \alpha\ \forall i, j$ s. t. $Y_{ij} = 0$ and $\|x_i - x_j\| < \alpha\ \forall i, j$ s. t. $Y_{ij} = 1$. Therefore, there are two admissible configurations: the density is bimodal and the difference in terms of peak intensity is due to the fact that a configuration is way more likely than the other.
The contour plot does not capture the fact that also the distributions for 10 and 13 are bimodal, but this is only because their second peak is below the threshold set for the contour plot.

Let us conclude the analysis on the Florentine Families dataset with a remark concerning the final estimates of the latent positions (yellow numbered dots in Fig. 3.5). Those estimates were computed by taking the average values over all the Markov chain outputs: this approach seems to perform well when the distribution is actually unimodal, but it comes up short in the other situations. For this reason, we decided to follow a more robust method and identify the final latent positions by means of a *maximum a posteriori* (MAP) estimation. The result is shown in Fig. 3.9: there are no substantial differences for most of the locations, even if MAP estimates appear more meaningful for nodes affected by bimodality. However, given that these matters are quite complicated, we will postpone their discussion to Sec. 4.4, where we will specifically address multimodality issues.

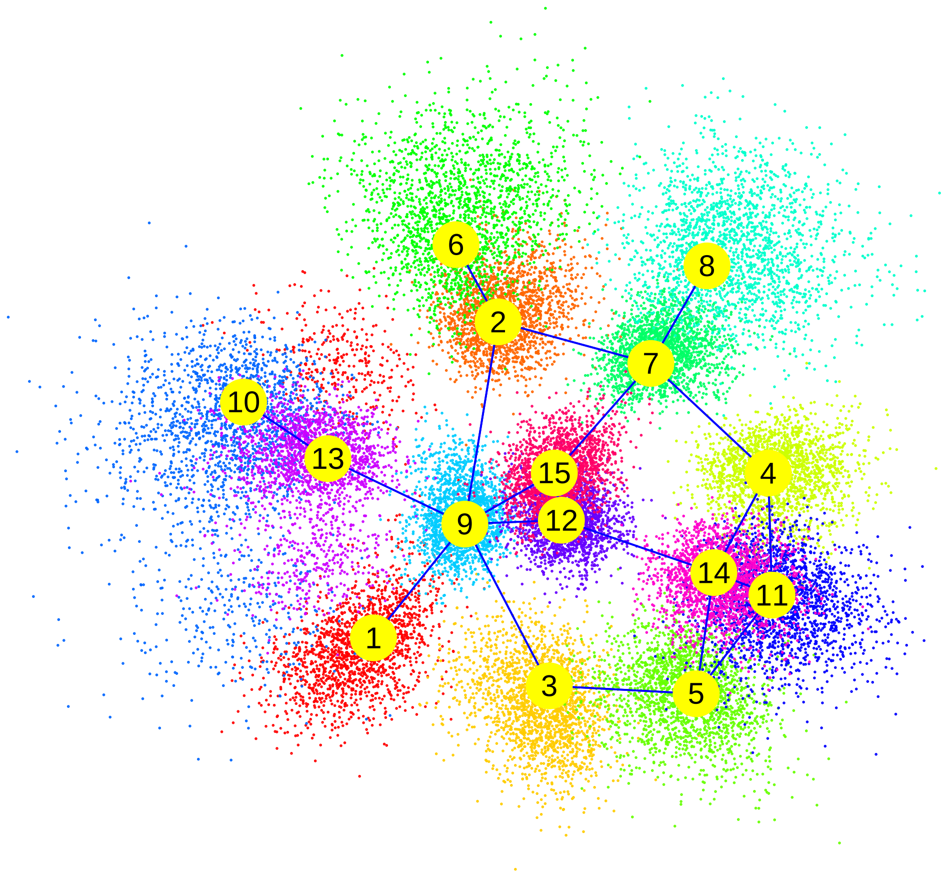

Figure 3.9: MAP estimates of the latent positions for the Florentine Families dataset

#### 3.5.2.2 Zachary's karate club

From 1970 to 1972, W. Zachary collected data related to a karate club: the network contains 78 undirected links between 34 members who interacted outside the club [52]. Even if the original purpose was to study the network evolution after a conflict between two important members, the graph can still be considered as a second experiment to assess the visual representation of our model. Indeed, the first dataset was essentially used to verify the coherence between our modified code and the results from [20], but we could not prove that the latent configuration was actually meaningful. By comparing the LSM output for the karate with a representation recognized by the literature we can at least show that the information displayed by the latent positions is qualitatively similar to other known results.
We decided to use the Kamada-Kawai (KK) algorithm [25], which is a well-known example of *force-directed* graph drawing. The reason why we chose this method is that it is quite simple, effective and completely unrelated to LSM: KK defines a set of attractive and repulsive forces interacting between nodes, then simulates the corresponding physical system to find its mechanical equilibrium. The features of the karate club are well captured by KK, since for instance the information related to *centralities* and *clustering coefficients* is properly represented. These two quantities are often used in network analysis to describe the "importance" of a vertex: high values usually indicate that a node belongs to a cluster characterized by a great number of both internal and external connections.
LSM relies on completely different assumptions than KK: however, by comparing Fig. 3.10 and 3.11 we can observe that the results are extremely similar, especially in terms of relative positions of the vertices. This means that also LSM manages to capture the essential features of the karate club, proving that the displayed information is actually meaningful.
Furthermore, we can get a better a grasp of the usefulness of the latent space: while in the KK case we are only focusing on finding a configuration that is aesthetically clear, thanks to the LSM we can actually quantify the behavioral differences of the vertices.

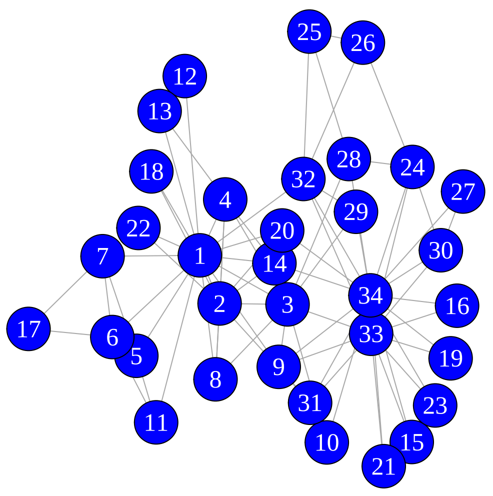
Figure 3.10: Kamada-Kawai representation of Zachary's karate club network

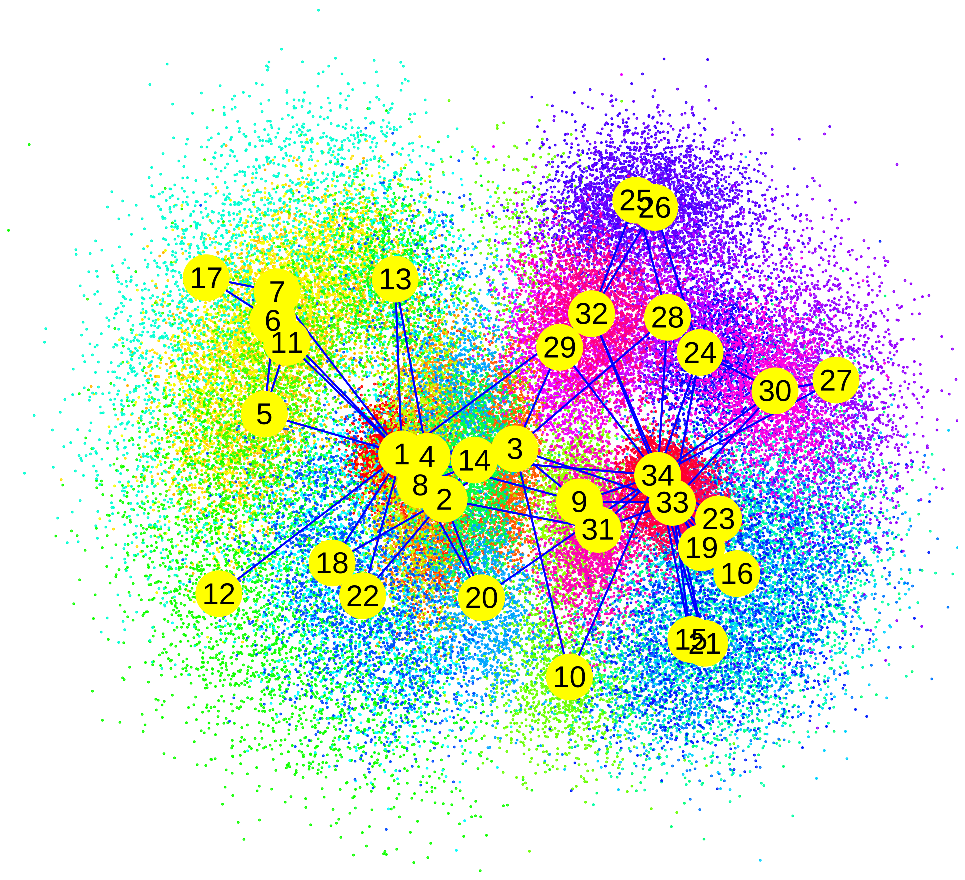
Figure 3.11: MCMC realizations for Zachary's karate club dataset. Colored points correspond to stochastic positions and yellow dots are used to denote the average position for each node. Blue lines represent the observed network.

#### 3.5.2.3 Science coauthorship network

In order to test our model on an even harder benchmark, we also analyzed the biggest connected component (379 nodes) of the coauthorship network of scientists active in the field of network theory and experiment (as compiled by M. Newman in 2006). This dataset greatly differs from the two previous examples because of its size and the dynamics of interaction between nodes.
Probably due to the high number of vertices, the graph is not fully representable in $\mathbb{R}^2$, so the spatial distribution of the latent positions cannot provide enough information to describe the entirety of the graph structure. Nevertheless, the model output can still be meaningful: the estimated coordinates are the ones maximizing the likelihood of the observed network, so our interest lies in evaluating the suitability of the bidimensional representation.
For instance, we can define a ratio based on the latent estimates $Z_i$ and the concept of representability (Eq. 3.2): first, we create an adjacency matrix $A'$ where $A'_{ij} = 1$ if $\|Z_i - Z_j\| < 1$ and $A'_{ij} = 0$ otherwise; then, we choose a dissimilarity function and we calculate its value with respect to the true adjacency matrix $A$ of the network.
For example, we could define our own *representability ratio* as

$$RR = \frac{\sum_{i,j} \mathbb{1}_{A'_{ij}=A_{ij}} - n}{n(n-1)} \tag{3.10}$$

where $n$ is the matrix size and is subtracted from both numerator and denominator because we are not allowing loops in our graphs.
We could argue that the approach seems to be too simple, but this is actually not true since we can consider different penalizations depending on whether a link is missing or created by mistake and even adapt the ratio to weighted networks. However, since statistical models consider observed data as particular outcomes from some unknown stochastic process, a non trivial issue is given by the robustness of such a dissimilarity indicator. Indeed, assuming that the generative process of the network is well defined, slight variations in the set of observed edges could produce different values for the ratio, which would be troublesome when comparing the model performance with other algorithms.
Another problem is given by the fact that the defined ratio does not allow us to quantify the severity of the error: if an edge $e_{ij}$ is missing, the difference in terms of adjacency matrices will not tell us if the error was huge or relatively little, like $\|x_i - x_j\| = 1.001$ or $\|x_i - x_j\| = 1000$.
Due to these reasons, we decided to adopt an approach more focused on determining if the model has successfully grasped the significant features of the network. Communities are usually defined as groups of nodes with an almost

identical behavior (see Sec. 1.4), meaning that they tend to have a similar number of edges directed towards the same vertices: if our model is able to retrieve the significant information from the data, then the positions related to such nodes should be located in a specific region of the latent space (even if we are considering only a bidimensional space).
This is theoretically possible, since we already mentioned that LSMs are able to reproduce the transitivity observed in many real world networks thanks to their assumption of a distance-based conditional probability. Therefore, we are interested in checking the model efficiency through a figure of merit which is defined as the percentage of nodes correctly assigned to the network communities.
Of course, even with this approach there are some issues:

- the clustering method clearly affects the final results, which is why we have to consider different alternatives before reaching a conclusion;
- the "real" communities are not known, meaning that we have to accept as "likely" the ones identified by a community detection algorithm like *Walktrap* [38] or *Infomap* [41] (we used the R package *igraph*, which contains the full implementations of the two methods). In this regard, this dataset serves as a bridge between the analysis of the meaningfulness of the visual representation and the exclusive focus on community detection which will be more rigorously tested through our fourth benchmark network;
- comparing two different partitions may be troublesome, since some assumptions must be made in order to determine the best correspondence between their sets. Indeed, the partitions are not uniquely ordered, which means that the first set of partition $P$ could perfectly coincide with the third set of partition $P'$: if we want to compare two partitions, we need to start by finding the best matching between them.

Concerning the last point, we tried to identify a method that would allow us to find the best matching or *overlap* between two partitions. We eventually devised an algorithm based on the *Jaccard index*, which is defined as

$$J = \frac{|A \cap B|}{|A \cup B|} \tag{3.11}$$

where $|\cdot|$ denotes the cardinality of a set and $A$ and $B$ belong respectively to partitions $P$ and $P'$.

The algorithm is structured as follows:

**Algorithm 2:** BestOverlap

```
Input: $P$ = first partition, $P'$ = second partition
Output: $P^*$ = reordering of $P'$ that has the best correspondence with $P$

Set : $n_P = |P|$, $n_{P'} = |P'|$, $ind = \{1, 2, \dots, n_{P'}\}$

for $i = 1, 2, \dots, n_P$ do
    $max = 0$
    $cont = i$
    if $|ind| > 0$ then
        for $j \in ind$ do
            Evaluate Jaccard index : $temp = \frac{|P[i] \cap P'[j]|}{|P[i] \cup P'[j]|}$
            if $temp \geq max$ then
                $max = temp$
                $cont = j$
        $P^*[i] = P'[cont]$
        $ind \leftarrow ind \setminus \{cont\}$

if $|ind| > 0$ then
    $cont = n_P + 1$
    for $j \in ind$ do
        $P^*[cont] = P'[j]$
        $cont \leftarrow cont + 1$
```

We are finally ready to present the results of the community detection by LSM, obtained by using agglomerative hierarchical clustering in the latent space with *complete linkage* and setting the number of clusters equal to the cardinality of the partition defined by the Infomap algorithm.
The Infomap communities are shown in Fig. 3.12, while the LSM output can be observed in Fig. 3.13: the graph layout is given once again by the Kamada-Kawai algorithm, which is independent from the two models and it is only used as a mean to inspect impartially the local differences in terms of community assignment.
By comparing the vertices in the same position there seems to be an overall agreement, but the colors may sometimes be misleading: 56.7% of the vertices actually belong to the "same" groups once the partitions have been reordered through Algorithm 2.

There may be a loss of precision due to the bidimensionality constraint and some uncertainty due to the fact that a node may possibly be close enough to two different communities, but overall LSM proves to be quite efficient, especially considering that the graph itself does not have a significant community structure. Proof of this “weakness” is given by the fact that three independent algorithms are giving different results, since we already mentioned the fact that the KK procedure follows yet another approach compared to LSM and Infomap. Indeed, by looking at both Fig. 3.12 and Fig. 3.13 we can remark that some communities do not seem so meaningful with respect to the proposed layout.
The results so far can only be seen as “promising”, since their potential is yet to be fully assessed. Furthermore, the percentage value has been evaluated by referring to the Infomap partition, which could have some flaws itself. For this reason, in the following section we will consider our final benchmark dataset and present a more accurate analysis of the community detection task based on a true known partition.

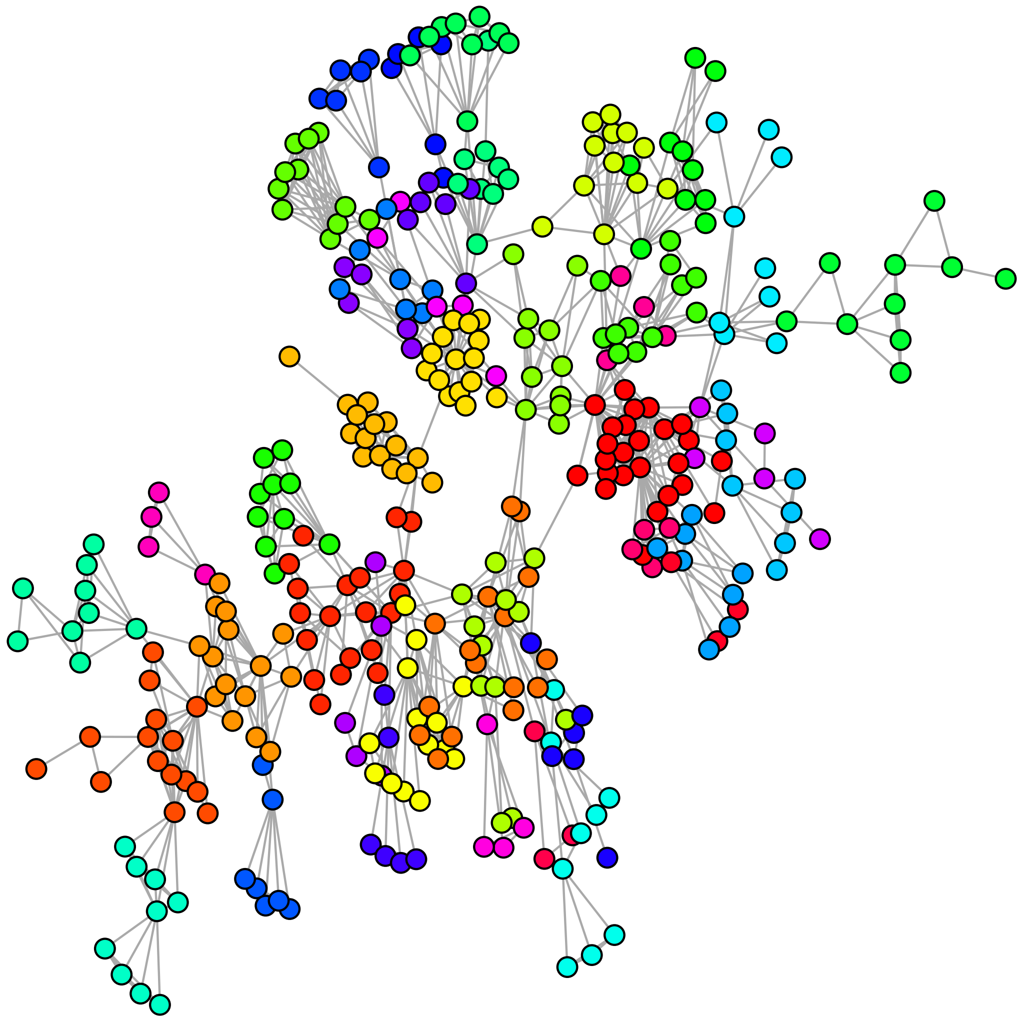

Figure 3.12: Communities detected by the algorithm Infomap in the science coauthorship network

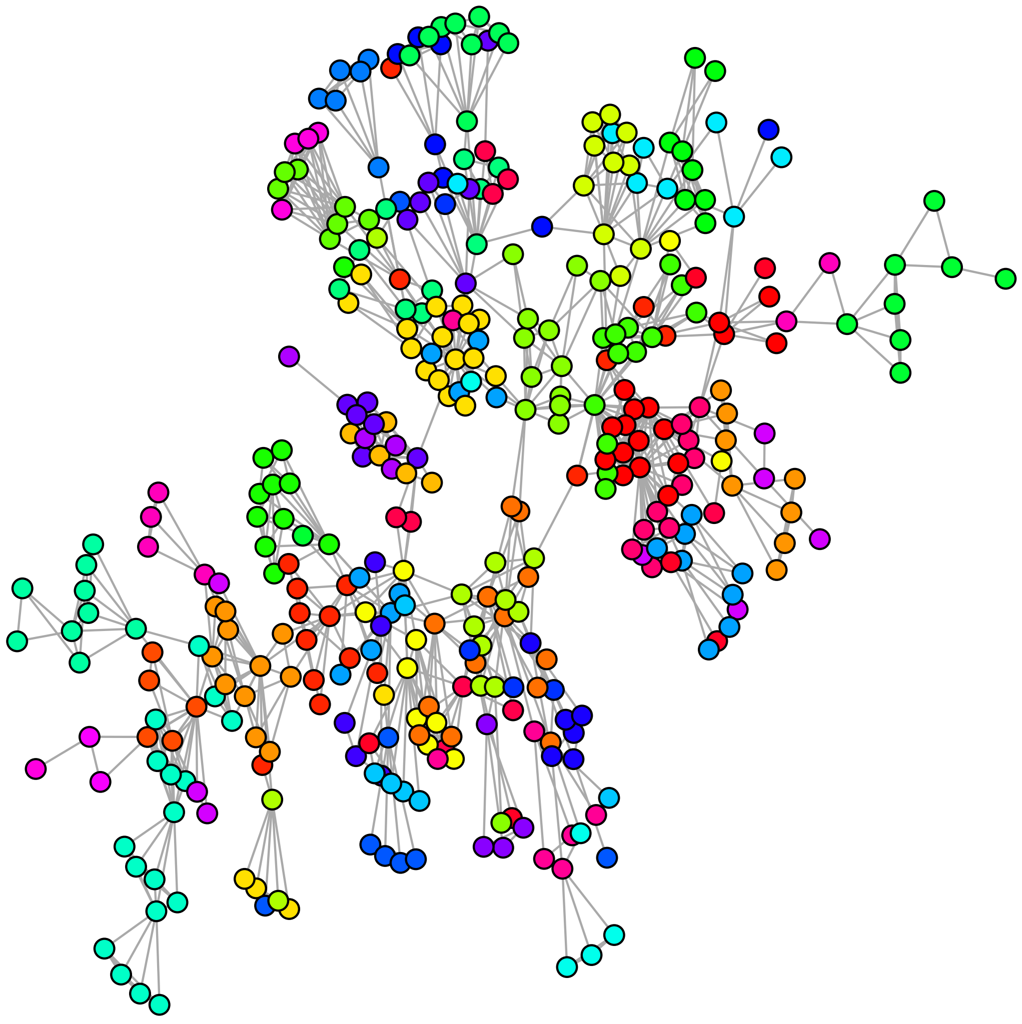
Figure 3.13: Communities detected through clustering of LSM estimates for the science coauthorship network

#### 3.5.2.4 NCAA network

As the final step for model validation we decided to analyze the network of National Collegiate Athletic Association (NCAA) American football games between Division I-A colleges during the regular season of Fall 2000. Most colleges participate in athletic conferences, which are groups of teams that play competitively against each other. Teams typically play at least half of their games within their conferences, so there are some strong communities to which the vast majority of the 115 vertices belong to. For this reason, due to the knowledge available a priori, researchers usually consider the athletic conferences as the 12 real communities that should be identified (see Fig. 3.14).

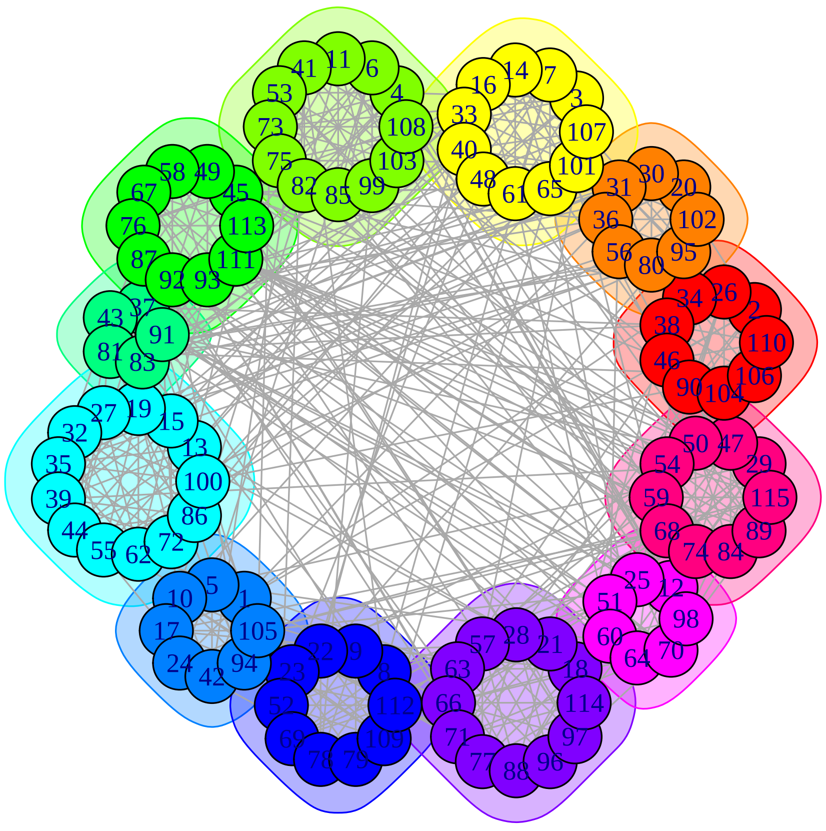

Figure 3.14: NCAA athletic conferences

The size of the network is definitely smaller than the previous case, so we no longer have the big issues related to the graph representability: Fig. 3.15 shows the LSM output, which seems to possess the desired features in terms of mutual distances and clusters. The yellow dots correspond to the average positions of the nodes, while the colored points are used to represent all the stochastic realizations from a Markov chain with $3 \times 10^6$ iterations ($10^6$ burn-in, thinning 1:1000).

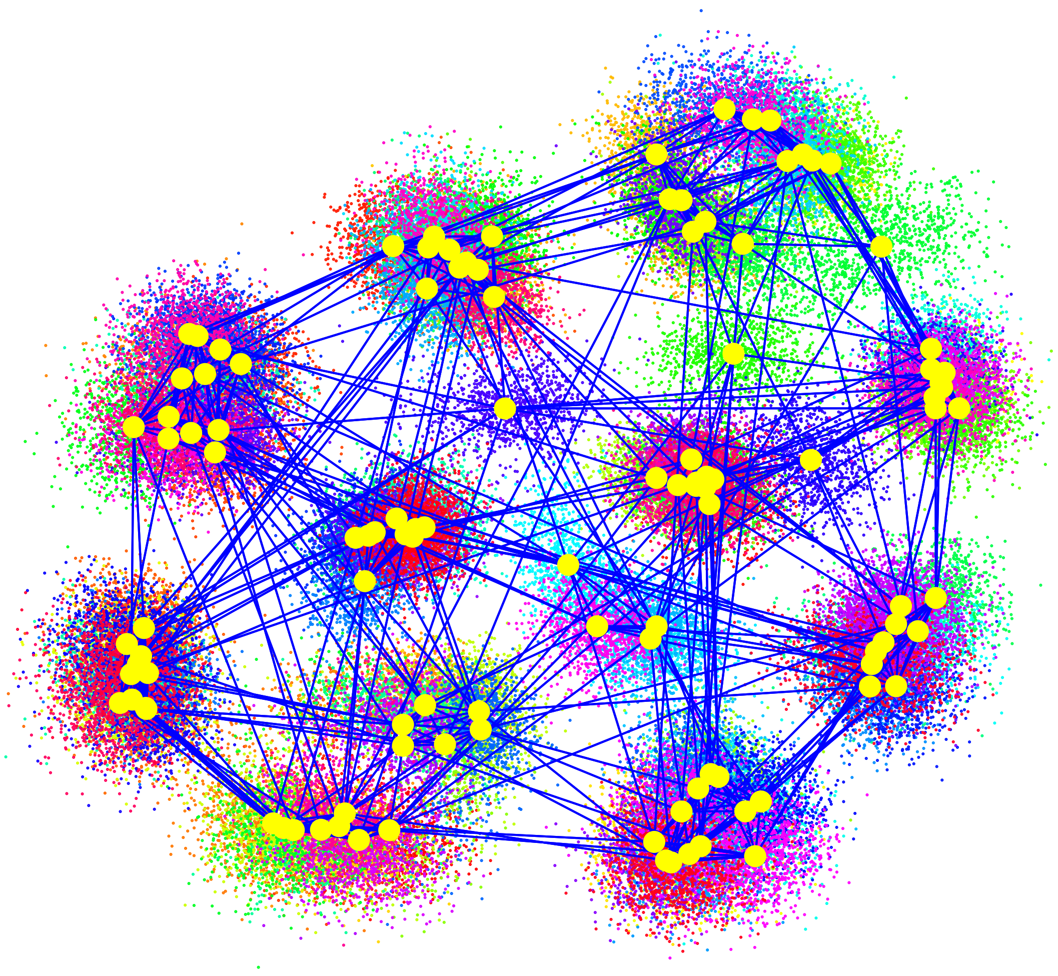

Figure 3.15: MCMC realizations for the NCAA dataset. Colored points correspond to stochastic positions and yellow dots are used to denote the average position for each node. Blue lines represent the observed network.

Fig. 3.16 shows the result obtained by using hierarchical clustering with complete linkage and setting the number of clusters equal to 12, while Fig. 3.17 highlights the nodes that have been misclassified according to the true partition.

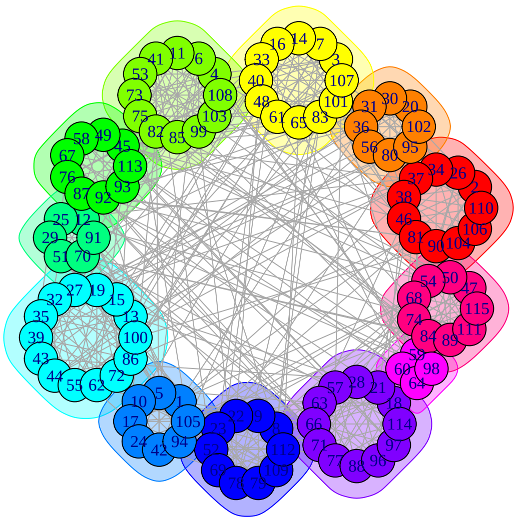

Figure 3.16: Communities detected by LSM in the NCAA network

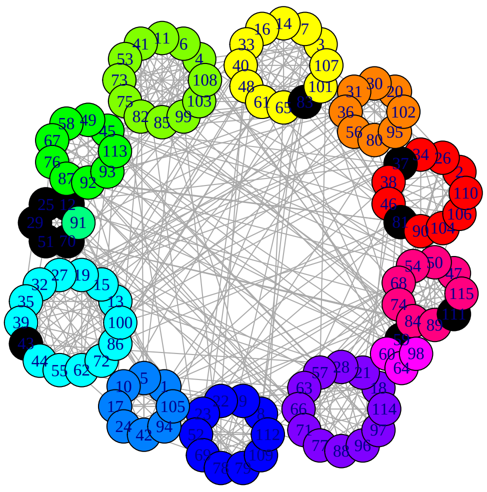

Figure 3.17: LSM communities vs athletic conferences

The absolute score is definitely positive: 90.43% of the vertices have been correctly classified. However, since the value alone is not so meaningful, we pushed our analysis a bit further and evaluated the performances of other four algorithms in order to make a comparison. As we can see from the results displayed in Tab. 3.1, LSM actually proved to be the best in terms of classification rate.

| Method | Type | Classification rate |
|---|---|---|
| LSM | Stochastic | 0.9043 |
| Infomap | Stochastic | 0.8098 |
| Spinglass | Stochastic | 0.7130 |
| Optimal | Deterministic | 0.7043 |
| Walktrap | Stochastic | 0.6521 |

Table 3.1: Comparison of five methods for community detection

The labels used for the listed algorithms are the ones provided by the R package *igraph*, since all of the simulations were made by using its functions. In particular, the method labeled as "Optimal" maximizes the modularity of the partition (see Eq. 1.4.4): since it is computationally heavy (exponential complexity) and usually guaranteed to provide good results, the better performance by LSM seems all the more impressive. As for the algorithms with at least some stochastic components, we considered their rate to be the average over 100 community detection tasks.

Fig. 3.22 shows examples of partition identified by the considered methods: we remark that cardinality and group associations can actually be pretty different, meaning that the community identification is not trivial. Fig. 3.21 offers yet another proof of its difficulty, since a force-drawing method like Kamada-Kawai is meaningless with respect to the true partition.

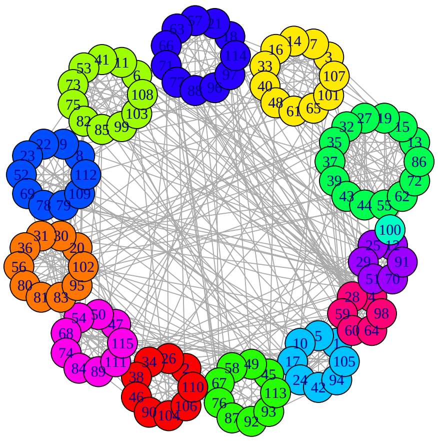
Figure 3.18: Infomap

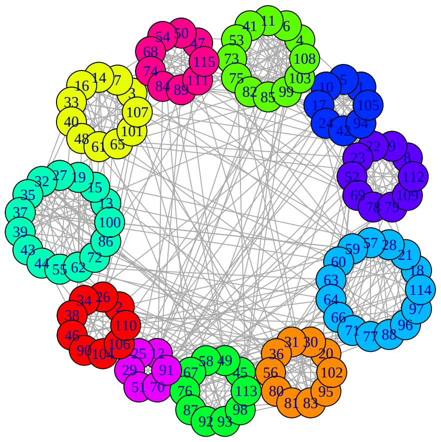
Figure 3.19: Spinglass

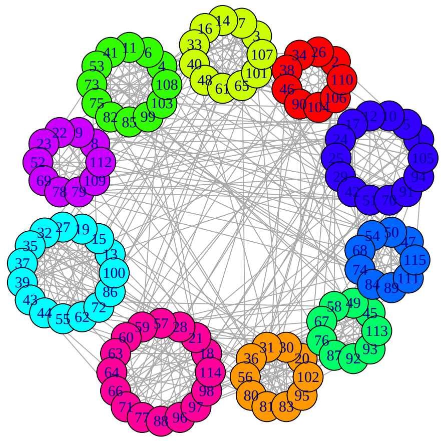
Figure 3.20: Optimal

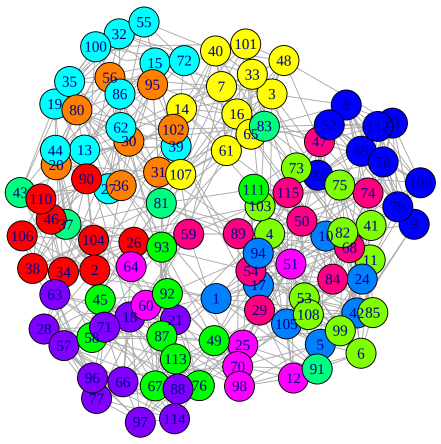
Figure 3.21: KK layout

Figure 3.22: A comparison between the communities detected by the algorithms Infomap, Spinglass and Optimal can be observed in Figs. 3.18,3.19,3.20, while Fig. 3.21 shows the true communities with the Kamada-Kawai layout.

The striking score obtained by LSM can be partially explained by the fact that the number of communities was set thanks to our previous knowledge of the dataset, which is obviously an advantage. However, if we inspect the dendrogram and follow a procedure like the *elbow method* for the estimation of the number of clusters, we realize that such a number is actually meaningful: the total within-sum-of-squares (WSS) stops decreasing for bigger values of the cardinality of the partition, showing that further cluster divisions would not improve the overall representation.

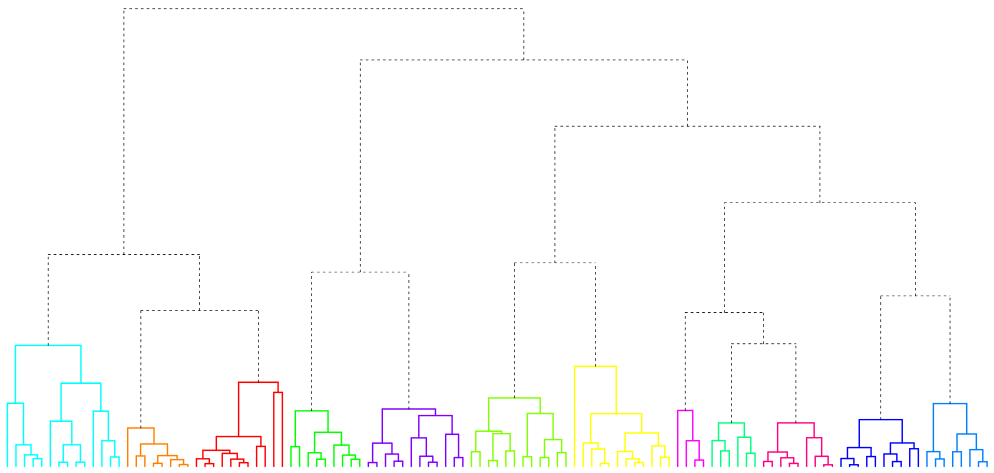

Figure 3.23: Dendrogram of the agglomerative clustering with complete linkage performed on the final estimates of latent positions for the NCAA network

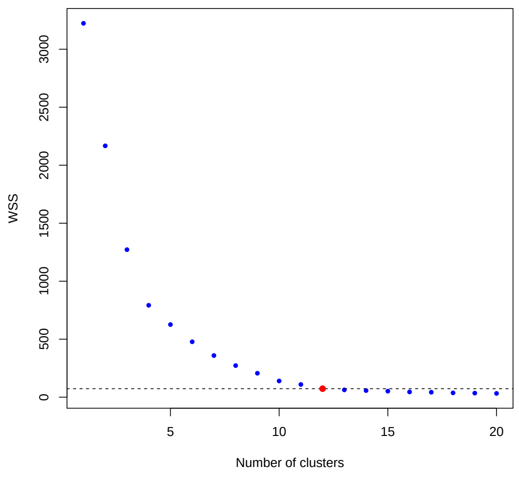

Figure 3.24: Elbow method for estimating the number of clusters

# Chapter 4

# Modified LSM

## 4.1 LSM strengths and weaknesses

The results presented in Ch. 3 showed that the LSM described in [20] is a valuable model for both visualization and community detection tasks.
We observed that the model assumption of a conditional probability based on the distance between latent positions is definitely suitable for the representation of networks with transitive features and non-trivial clustering. We also remarked that structurally equivalent nodes tend to be grouped together, which is a fundamental requirement for performing community detection. Even if the graph is not fully representable in the chosen multidimensional space, communities are well identified and mapped into specific latent regions, so that through clustering methods we are able to capture the mesoscopic structure of the network.
However, despite the positive results, the LSM algorithm devised by Hoff, Raftery and Handcock has some critical weaknesses: some of them are caused by the code implementation, while others are directly linked to mathematical aspects of the model. In the following sections, we will discuss the main problems and propose a new model crafted by considering possible solutions: in order to distinguish between the two LSMs, we will label Hoff's model and the modified one as H-LSM and M-LSM respectively.
In particular, Sec. 4.3 and Sec. 4.4 will cover code implementation issues: the first will focus on matters related to the convergence of the MCMC such as mixing and speed, while the second will offer some analyses related to the posterior densities obtained through the MCMC algorithm. Sec. 4.2 and Sec. 4.5 will consider the mathematical aspects and describe the critical issues arising from dimensionality, representability and uniqueness of the representation. Finally, we will make a quick recap in Sec. 4.6 in order to summarize the most relevant features from M-LSM before considering our most advanced development in the next chapter.

## 4.2 Representability and dimensionality

A primary goal of data visualization is to convey information clearly and efficiently, meaning that the representation should be nonambiguous and focused on the most relevant concepts. In H-LSM, the distance between two nodes is directly related to the probability of observing a link: the resulting visualization is clear because the notion of distance is relatively simple and familiar, but also efficient since the distance is actually the fundamental quantity used in the statistical model itself. However, we cannot take these properties for granted: the information is correctly communicated only if the graph is representable.
We already gave the formal definition of representability in Sec. 3.2, so let us consider a more intuitive approach and restate our problem: given a set of distances between objects, we have to locate the objects in the space so that all the requirements are met. Of course, this is only feasible if the set of distances is "admissible" and the space actually allow us to place the objects "where we need": in other words, the distances have to satisfy the properties of a distance measure and the space dimensionality has to be large enough to let the positions satisfy all the distance constraints. In our framework, a network is representable if the vertices can be located in the latent space in such a way that the distances maximizing the overall likelihood are well respected. Indeed, if such a set of coordinates exists then the $\alpha$ threshold from our model (see Eq. 3.2) should be coherent with the latent positions, since otherwise the likelihood value could clearly be improved.
Any graph is representable if the space dimensionality is high enough, but unfortunately large networks usually require more than two or three dimensions to be fully representable. For this reason, it is necessary to project them on a lower dimensional visualizing space, possibly making the representation of the latent structure less clear. Actually, we can force the algorithm to estimate locations in a smaller space as we did for the science coauthorship network, but this introduces errors and approximations in the model.
In this regard, *star*-like structures (Fig. 4.1) are especially critical: peripheral nodes need to be far from each other but at the same time near to the two centers, which is clearly not feasible in a bidimensional space.
One of the biggest issues related to representability is that it is actually difficult to know a priori if a network is representable within a $n-$dimensional space. H-LSM initializes the Markov chain through Classical Multidimensional Scaling, so we can at least get an approximated idea of the overall representability: if the graph is fully representable, than even CMDS should manage to estimate distances such that the value of our representability ratio (Eq. 3.10) is not too small. However, since the locations have yet to be optimized through the MCMC

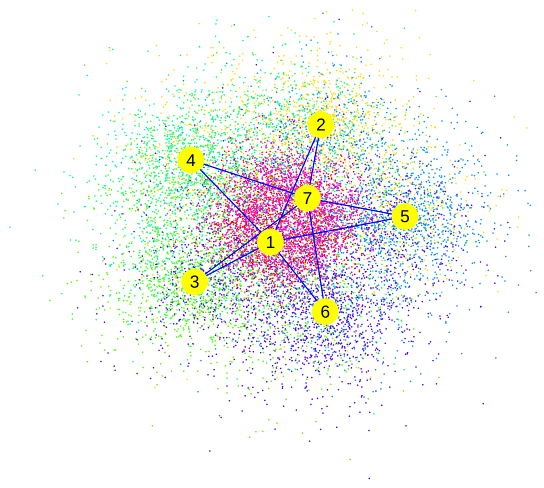

Figure 4.1: Representability issues of a double star network

algorithm, we do not know how much the score can be improved (Tab. 4.1 will provide some real examples): this is troubling because we need to choose the space dimensionality before running the algorithm, so we have to find a balance between computational complexity and representability even though we do not possess enough information. Only at the end of a simulation we can check whether the results are satisfying and, if they are not, we have no choice other than re-run the whole model by increasing the dimensionality.
Due to these complications, representability issues can be extremely difficult to address: figure of merit like the representability ratio may help us quantify the overall goodness of a latent configuration, but we will see that they can also be misleading. Moreover, the strong dependence on the $\alpha$ threshold makes it even harder to understand a priori if a graph is representable, since our threshold is actually a model variable and needs to be correctly estimated.
In the next sections we will have to focus on other matters, but we will also illustrate different contexts where both the representability concept and the $\alpha$ parameter play a fundamental role: our hope is that their importance and close relationship with the core of latent space models will become even clearer.

## 4.3 MCMC convergence

As we have already said, the objective of our model is to estimate the $Z$ positions in the latent space: the reason why the MCMC algorithm plays a fundamental role in LSM is that we need to know the posterior distribution of each location in order to estimate the final latent configuration. Computationally speaking, the MCMC procedure is particularly demanding, so we should definitely try to ensure that the algorithm runs as smoothly as possible.

Let us consider once again the algorithm from Sec. 3.5:

1. at iteration $k$, sample a value $\hat{Z}$ from $J(\hat{Z}|Z_k)$, where $J$ is a symmetric stationary proposal distribution;
2. accept $\hat{Z}$ as $Z'_{k+1}$ with probability $P_k = \frac{p(\hat{Z}|Y,\alpha_k)J_k(Z_k|\hat{Z})}{p(Z_k|Y,\alpha_k)J_k(\hat{Z}|Z_k)} = \frac{p(Y|\hat{Z},\alpha_k)\pi(\hat{Z})}{p(Y|Z_k,\alpha_k)\pi(Z_k)}$, otherwise set $Z'_{k+1}$ equal to $Z_k$;
3. if $Z'_{k+1} \neq Z_k$ then apply a Procrustean transformation to $Z'_{k+1}$ with $Z_0$ as objective ;
4. $\alpha$ is updated with a classic Metropolis-Hastings algorithm.

Each $Z_k$ corresponds to a specific set of states for the Markov chain: ideally, the chain should quickly converge to the best configuration without getting stuck into local optima. In other words, we are interested in obtaining a sequence of $Z_k$ such that the state space is explored as much as possible and the final configuration is also rapidly attained, so that the posterior distribution of the latent position will not resent from the initial phase.
Many different choices can influence the convergence speed and mixing of the chain: for instance, a major role is played by the proposal distribution, since the probability of observing an update will depend on the quality of the candidate. A proposal is efficient if it generates candidates with high acceptance probabilities: the chain is more likely to switch between different states, resulting in a good mixing and low autocorrelation values.
H-LSM proposes a simultaneous batch update of the latent positions by using Gaussian proposals with the same parameters for each node. The random variables sampled as candidates are therefore i.i.d., but this does not mean that they will have similar acceptance probabilities: since the specificity of each location is not considered, there may actually be multiple losses in terms of efficiency. In order to better illustrate this point, Fig. 4.2 shows the candidates accepted during a simulation with $2 \times 10^6$ iterations for a very simple V-shaped graph $\mathcal{G} = (\{v_1, v_2, v_3\}, \{(v_1, v_2), (v_2, v_3)\})$: we can remark that the colored regions have definitely different shapes, proving that a strategy considering the same proposal for each node has definitely room for improvement.
The green points correspond to the central vertex $v_2$: this node has more constraints than the other two, so the variations for its candidates are more controlled and result in a cloud which is more concentrated than the others.
Looking back at the LSM outputs from the previous chapter (e.g., Fig. 3.5), we can easily observe the same behavior: peripheral positions tend to have a larger variability. As a final remark concerning Fig. 4.2, the colored clouds are not aligned due to the effect of the diffuse normal priors used for the latent

coordinates: these distributions are used to penalize points far from the origin, so that the final result is not too scattered.

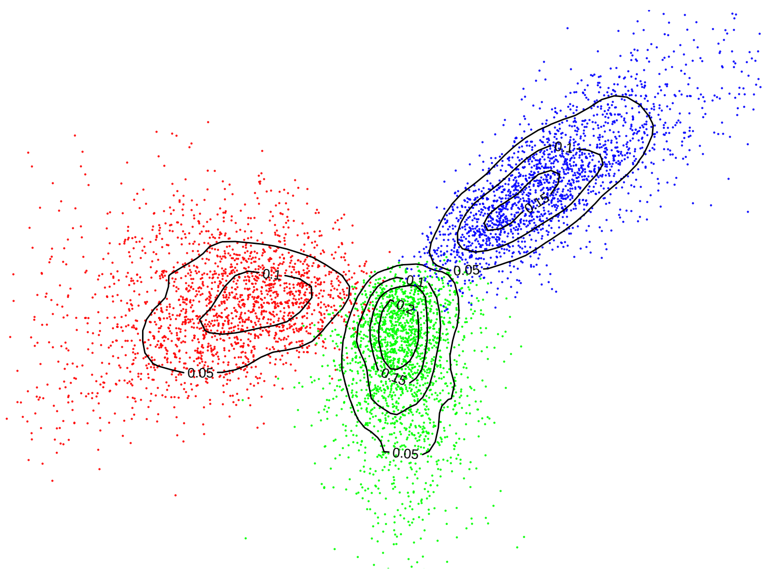

Figure 4.2: Candidates accepted by the MH algorithm: the colored regions are clearly different, despite having the same proposal distribution for every node

A second problem caused by the batch update in H-LSM is that the acceptance probability considers all the stochastic updates at the same time, meaning that we look at the overall goodness instead of evaluating the quality of each update. Let us delve deeper into the matter and suppose we have a graph with 10 vertices and a non trivial link structure. Then, let us consider two possible scenarios during the MCMC simulation:

1. the sampling from the proposals returns 9 candidates with high acceptance probabilities and 1 candidate with a really low acceptance probability;
2. the sampling from the proposals returns 9 candidates with low acceptance probabilities and 1 candidate with a really high acceptance probability.

In the first case the global update is likely to be accepted since there is only one poor candidate, while in the second situation the outcome would probably be the opposite: individual updates could perform a lot better, accepting only the good candidates from both scenarios.
More generally, we could avoid any kind of compromise by evaluating efficiently every single update. This is particularly relevant during the transition phase of the simulation, when the latent positions are evolving from the initial to the final configuration: accepting only good candidates means improving the overall likelihood without being affected by the negative impact of poor updates, which

is obviously helpful if we are interested in speeding up the convergence process. In the last example we focused on a graph with 10 nodes, but if we scale our problem to much bigger networks the advantages obtained by using individual updates become even greater. Indeed, we remarked that highly connected nodes seem to be more constrained than the others, making it more difficult for them to accept a new position candidate. Whenever the acceptance probability of such candidates is too low also the overall probability is strongly affected: as a result, even nodes that could update their position will have to stick to the current configuration, losing information in terms of simulated points and slowing considerably the convergence of the algorithm. We can actually improve the acceptance rate by reducing the variance of the proposals, but this is not a suitable solution because of the existing trade-off between mixing and space exploration.

Considering our analyses about batch and local updates it is clear that individual proposals should perform better, but unfortunately this is useless unless we are able to determine exactly the type of distributions we should sample from. More specifically, we need to know the full conditional density of each variable, which is not necessary in the global approach where all the variables are simultaneously sampled from the joint density. However, given that in H-LSM the updates from the proposals are assumed independent, in our current framework the choice between a local and a global update is actually arbitrary: the only difference between the two is that simultaneous changes lead to a higher variance for the accepted candidates, which should be clear considering they are the product of multiple local interactions. Unfortunately, the variance is increased through some complicated mechanisms, so it is extremely hard to find the exact settings that could lead the two different approaches to the same final configuration. Indeed, we could expect the variance to be the sum of variances from independent proposals, but this is actually false because of the filter on the candidates applied by the MH algorithm. Due to the complexity of the matter we will better discuss this analysis in Sec. 4.4, so that we can presently keep our focus on the convergence and mixing of the chain.

In M-LSM, we propose individual updates for all the latent locations, which means that candidates have to be sequentially evaluated at each iteration. The order we choose does not influence the final outcome of the algorithm: indeed, given two candidates we may have different acceptance probabilities depending on which one is selected first, but this effect is completely balanced by the following iterations. Moreover, we are interested in the relative distances between nodes: given vertices $v_a$ and $v_b$, due to the symmetry of the distance measure it does not matter whether we are updating first $v_a$ or $v_b$.

The improvement offered by M-LSM can be quantified and observed thanks to Fig. 4.3 and Fig. 4.4, where we show some diagnostics related to two simulations run for the NCAA dataset ($2 \times 10^6$ iterations, thinning 1:2000) using H-LSM and M-LSM respectively. Since in M-LSM we are accepting individual updates even in those cases where the batch update was globally rejected, the acceptance rate is much higher than in H-LSM: as a consequence, the mixing of the chain is faster and M-LSM traceplots are clearly better. Autocorrelation values confirm the improvement obtained through the individual updates. Moreover, it is interesting to remark that even the $\alpha$ parameter benefits from the increased acceptance rate of the latent position candidates. The value of $\alpha$ can be seen as a distance threshold and has a strong impact on the overall likelihood: if the chain is stuck into a particular configuration then it may be more difficult to accept updates for $\alpha$, which is why a better mixing for the latent position actually leads to lower autocorrelation values also for the parameter.

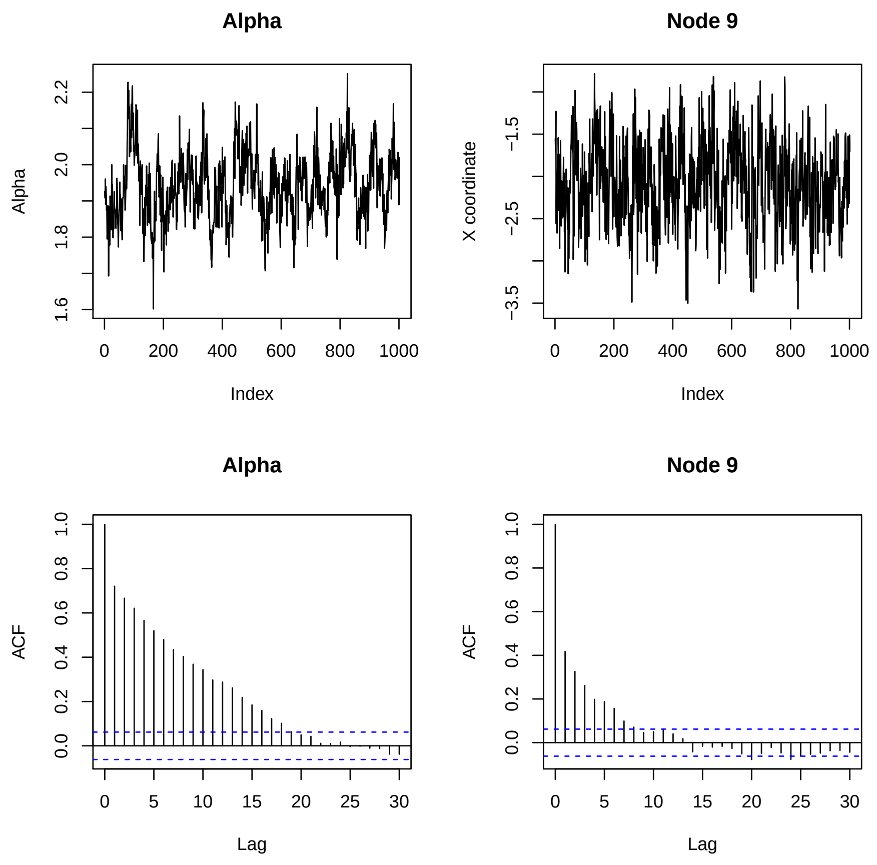

Figure 4.3: H-LSM: Traceplots and ACF plots with batch update

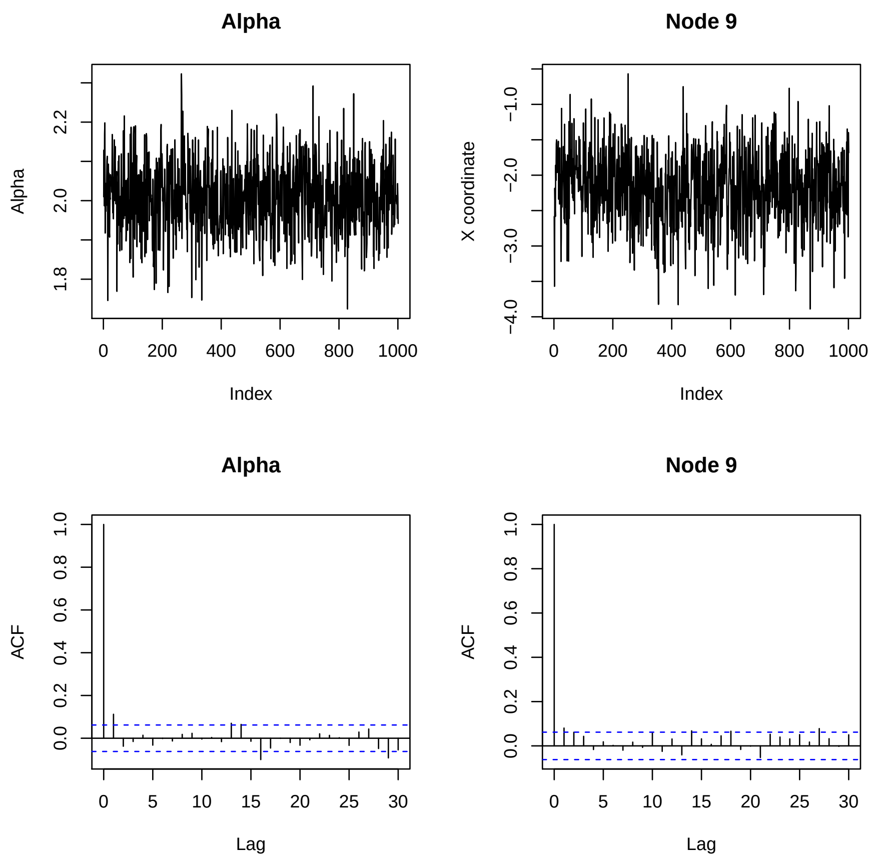

Figure 4.4: M-LSM: Traceplots and ACF plots with individual updates

Let us conclude the section with a take-home message: switching from batch to individual updates allow us to obtain a better mixing of the chain and a faster convergence without having to introduce further hierarchical levels in the model. Moreover, node-specific constraints can be individually considered and respected without globally affecting and slowing the MCMC updates.

## 4.4 Posterior multimodality

During our analysis of the Florentine Families (FF) dataset we briefly discussed the problem of *multimodality* of the posterior distribution and we showed its effect on the final estimates thanks to Figs. 3.5, 3.8, 3.9.
The existence of equivalent representations is particularly troublesome when we want to estimate the latent positions through their average values. For instance, Fig. 4.5 shows two equivalent configurations (left and middle panels) for a simple network and their spatial averages (right panel): we can remark that in the last picture the information concerning the relative distance between red and green nodes is completely lost. As it happens for the FF dataset, we can observe equivalent configurations during the simulation: the latent positions are

meaningful per se, but completely useless for making inference through their average values.

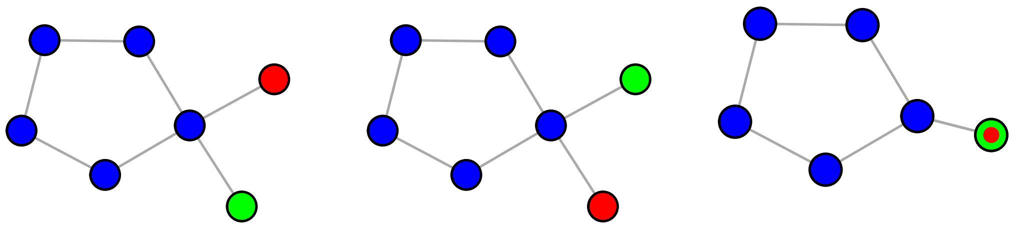

Figure 4.5: Equivalent representations of a network in the latent space (left and central panel) and misleading average estimates (right panel)

Instead of taking the average values, we could partially avoid this issue by recurring to the MAP estimates (see Fig. 3.9), which are more robust yet inefficient: they ignore the smaller peaks of the posterior density, so the corresponding observations do not contribute to the final estimation. In other words, it is a bit like we were removing some iterations from our final sample, but not rigorously nor uniformly for each node. For instance, the MAP estimate for the red and green positions in Fig. 4.5 would either be the left or the central panel depending on their frequency of occurrence in the simulation: in any case, the information coming from the excluded configuration would be wasted.
As we will explain soon, a better solution consists in avoiding multimodality by reducing the variance of the proposals, even if we always have to pay attention to the trade-off with the exploration of the parameter space. Unfortunately, it is extremely difficult to find a variance value that guarantees the absence of multimodality, especially in the batch update performed by H-LSM where it is already hard enough to define the variance associated to each node. Hoping to clarify the matter, let us take a closer look at the scenarios that can lead to multimodality and finally tackle the “variance problem” anticipated in the previous section.
Up to this point of the dissertation, we almost took the presence of equivalent representations for granted: if we look more carefully into the matter, it is true that there may be different configurations characterized by the same overall likelihood, but it should not be so easy to pass from one another. Indeed, if we think about the updates that must be accepted by the MH algorithm in order to switch from a configuration to an equivalent one, during the transition the likelihood has to decrease, so in most of the cases the algorithm should reject the update due to Eq. 3.9.

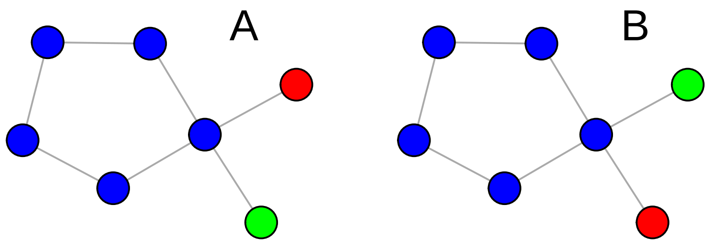

Figure 4.6: Equivalent representations

Let us look again at the network from Fig. 4.5 and let us focus on the two equivalent configurations (A, B in Fig. 4.6): red (R) and green (G) are disconnected, so any representation in which R and G are too close should be discarded by the model. Assuming we start with R above G, this means that in the same iteration we have to sample a candidate R' which falls in the region occupied by G and a candidate G' which falls in the region occupied by R: this can happen if we perform a batch update like in H-LSM, provided that the proposal variance is large enough. Indeed, we can accept both candidates and perform the switch from A to B because the final representation is equivalent to the initial one and the relative distance between R and G stays above the $\alpha$ threshold (see Eq. 3.2). Reducing the variance of the proposals definitely help in lowering the probability of having multimodality in the posteriors, but decreasing its value could also lead the algorithm to get stuck into local optima. To top it off, in complex networks we cannot determine a priori if the variance of the proposal will allow the latent positions to change between equivalent configurations, so we can only follow a slow trial and error procedure.
Individual updates lead to better performances even from this perspective: we cannot invert directly the position of R and G since we have to evaluate sequentially the two candidates, so the crucial step of accepting R' very close to G should be rejected because the corresponding likelihood is surely worse than the initial one.
Unfortunately, even if individual updates lower significantly the probability of switching between equivalent configuration, they do not completely prevent these transitions: for instance, we could observe an evolution from A to B if the candidates were sampled according to the following schema (fully illustrated in Fig. 4.7).

1. R' is sampled in the area above R and sufficiently far from the uppermost blue node. R' is accepted because the acceptance probability should be close enough to 1;

2. G' is sampled in the area previously occupied by R. G' is also accepted because the acceptance probability should still be reasonably high;

3. R" is sampled (and accepted) way below, sufficiently far from G' but also near the region previously occupied by G.

4. G" is sampled (and accepted) above G', near the initial coordinates of R.

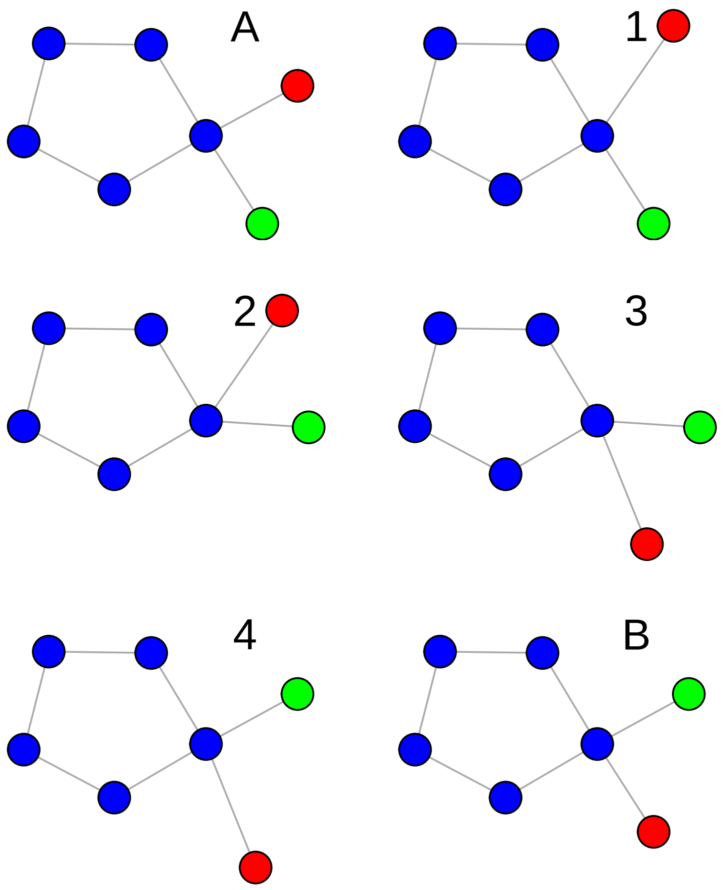

Figure 4.7: Example of possible transition between two equivalent configurations A and B through sequential individual updates

All the listed candidates can only be generated if the variance of the proposal is large enough, but it should seem pretty clear that individual updates are definitely less likely to reproduce the same behaviors created by batch updates. Furthermore, within the sequential framework of M-LSM it is way easier to predict which variance values should prevent these changes from happening. If we were to give a rule of thumb, we could probably follow this reasoning:

- we know that the $\alpha$ parameter can be regarded as a meaningful threshold: the distance between disconnected vertices should be greater than $\alpha$ and smaller otherwise, since this should at least give us a configuration for which the graph is representable;
- during the transition from one equivalent configuration to another the likelihood value clearly decrease if the distance between disconnected vertices becomes smaller than $\alpha$, so the update will have a very low acceptance probability;
- the proposals allow us to propose a candidate $\hat{z}$ for position $z_i$ such that $\|\hat{z} - z_i\| = \beta$, where the probability of observing large $\beta$ values grows with the variance of the proposals;
- if $\beta \geq 2\alpha$ we could "invert" two positions in one fell swoop, since the likelihood could remain basically the same.

Therefore, if we only propose candidates that are within range $2\alpha$ from their current position we should at least ensure that the probability of accepting the new value will be very low: since we deal with Gaussian proposals, we could pick a standard deviation $\sigma$ such that most of the candidates should fall within the $2\alpha$ radius. Unfortunately, the $\alpha$ value is only known a posteriori, so we would still need to run the algorithm at least once in order to make educated guesses about the standard deviation: if anything, the trial and error procedure should be quite smooth compared to the one necessary for the global approach of H-LSM.
As we did in the previous section, let us conclude with a take-home message: the variance for the proposals must be chosen by considering the trade-off between state space exploration and multimodality issues. Regarding the latter, M-LSM is more robust than H-LSM thanks to its individual updates: however, even if we can establish a relationship between the standard deviation and the $\alpha$ parameter, this does not ensure the total absence of multimodality. Indeed, the probability of sampling a candidate far enough from its current iteration value can be arbitrarily lowered, but will always be strictly greater than zero.

### 4.4.1 M-LSM validation on benchmark datasets

We ran M-LSM on the very same graph we used as example in Fig. 4.6: we wanted to prove that a low standard deviation can actually prevent multimodality, so we considered three very different values ($\times 10$ factor: 0.1,1,10) just to make sure that their impact was significant. Moreover, we decided to use the smallest value to observe the behavior of H-LSM and make an interesting comparison.

The results are displayed in Fig. 4.12: using individual updates from M-LSM, we observe multimodal distributions for $\sigma \geq 1$ and unimodal densities for $\sigma = 0.1$. We remark that the average estimates are only meaningful in the last scenario, proving once again the necessity of solving multimodality issues. Finally, the bottom-right figure shows that H-LSM and its batch update are indeed less performing than M-LSM: in order to avoid multimodality we would need to reduce even more the standard deviation of the proposals, causing the algorithm to worsen the exploration on the state space.

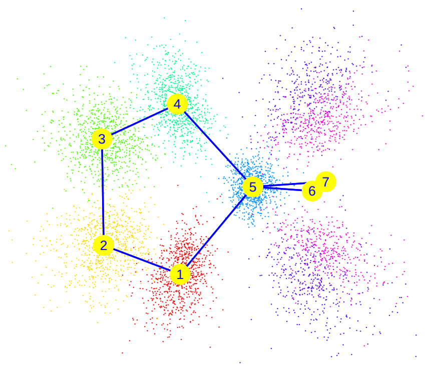

Figure 4.8: M-LSM, $\sigma = 10$

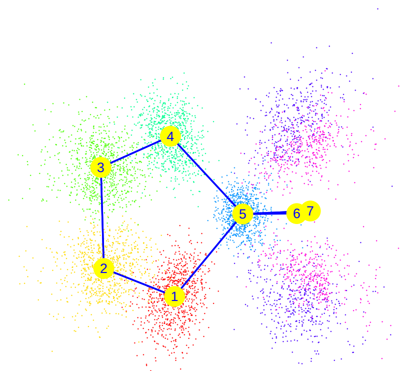

Figure 4.9: M-LSM, $\sigma = 1$

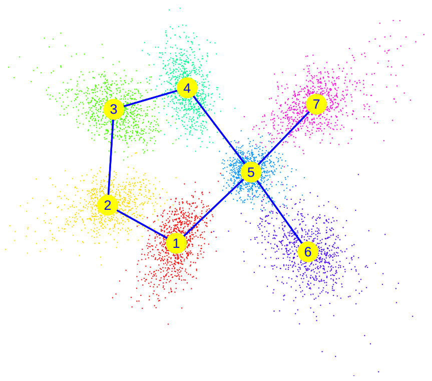

Figure 4.10: M-LSM, $\sigma = 0.1$

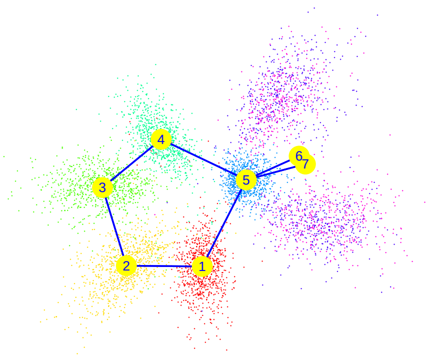

Figure 4.11: H-LSM, $\sigma = 0.1$

Figure 4.12: Analysis of multimodality in the posteriors for different levels of standard deviation and update type (individual or batch).

Even if they are a bit concealed and thus more difficult to observe, there are also two other fundamental features we want to highlight:

- the acceptance rates are totally different. We allow many more updates when the variance is smaller, so the clouds have a higher density and the estimates can be more accurate. We restrict our comparison to H-LSM simulations just to make sure that the variance is the only parameter changing from one test to another: the acceptance rates for $\sigma = 10, 1, 0.1$ are respectively $0.003, 0.65, 0.98$;
- the frequency of switch between equivalent configurations is definitely higher for H-LSM. This is actually very critical: we could force a rejection whenever we have a switch during a simulation, but this could only work if the Markov chain is pretty stable. We already said that we cannot completely guarantee unimodality: as it turns out, for $\sigma = 0.1$ we can have multimodal posteriors even with M-LSM individual updates. According to our tests, if we run the simulation for the example graph for long enough we can end up with equivalent configurations as we would have with H-LSM: however, considering $3 \times 10^6$ iterations, the number of times when the chain switches between such states is equal to 4 and 86 for M-LSM and H-LSM respectively. This huge difference can be seen as the final proof to validate the fact that individual updates are indeed more robust than batch ones with respect to multimodality issues.

As an additional benchmark analysis, we chose the same parameters we had used for the study on the Florentine Families dataset in Sec. 3.5.2.1 and we ran a simulation with M-LSM and individual updates. We obtained the outputs showed in Fig. 4.13, where all the posterior distributions are clearly unimodal. Finally, we followed the approach described in Sec. 3.5.2.3 and evaluated a simple representability ratio (Eq. 3.10) so that we could compare H-LSM and M-LSM. As we can observe in Tab.4.1, both models improved the initial configuration, meaning that they succeed in optimizing the latent positions so that the distances are compliant with the observed data. However, M-LSM is always better than H-LSM, proving that individual updates are indeed more suitable for our goals: a faster MCMC convergence and improved robustness with respect to multimodality allow us to improve the final estimates and therefore the overall goodness of representation. Moreover, the improvement is actually way more significant than it may seems, since these representability ratios do not give all the insights we need to make a proper quality assessment for the two models. Indeed, given that the adjacency matrices are quite sparse especially for large networks like the scientific coauthorship graph, a figure of merit as simple as

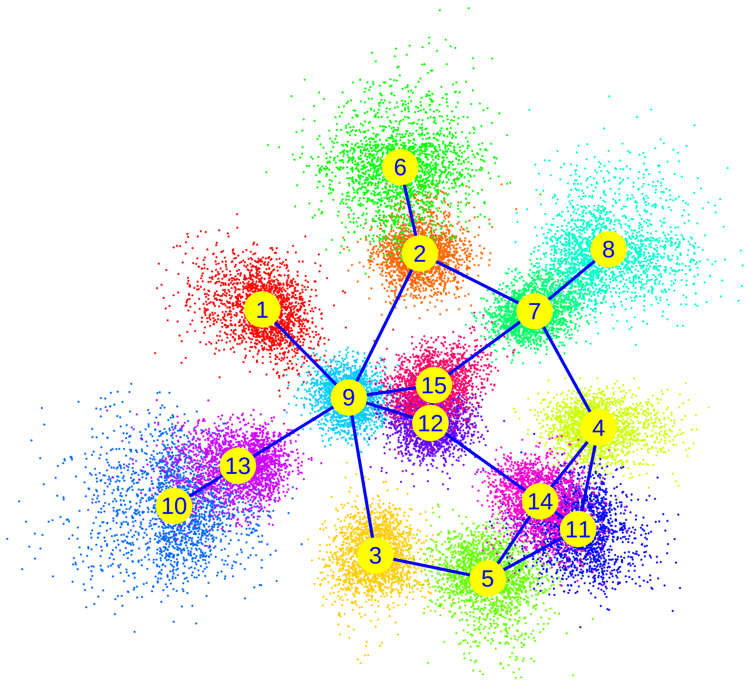

Figure 4.13: M-LSM applied to Florentine Families dataset: using the same parameters, individual updates produce unimodal posteriors, while batch updates lead to multimodality

the one we proposed ($RR = \frac{\sum_{i,j} \mathbb{1}_{A'_{ij}=A_{ij}} - n}{n(n-1)}$) may be misleading: its value for the initialized latent configuration is already around 0.95, but that is just because most of the nodes have only a few links. If we look at the performance on existing links (i.e., we remove the terms with $A_{ij} = 0$ from the summations) then the results are definitely more interesting: the starting configuration obtained through Classical Multidimensional Scaling has a very low ratio of 0.039, while the final estimates through H-LSM reach the value of 0.445. M-LSM is better even from this perspective: its final score is 0.637.

| Dataset | Starting conf. | Final H-LSM conf. | Final M-LSM conf. |
|---|---|---|---|
| FF | 0.848 | 0.961 | 0.971 |
| Zachary | 0.857 | 0.882 | 0.890 |
| Science | 0.947 | 0.989 | 0.991 |
| NCAA | 0.905 | 0.947 | 0.955 |

Table 4.1: Representability ratios

## 4.5 Procrustean transformation

One of the main strengths of H-LSM is that the model gives a clear picture of the MCMC simulations: the clouds of colored points showed in Ch. 3 can be used to perform inference on the latent estimates and are particularly useful in visualizing the different shapes and variances associated to the nodes. Unfortunately, in order to be able to represent all the points we need to introduce an external constraint, since without the Procrustean transformation the points can wander so much in the latent space that a final averaged estimate is completely meaningless. As we have already explained, the intermediate Procrustean step minimizes the distance between the updated configuration and a reference one: even though it allows us to put together all the MCMC outputs, it impacts the variances by altering the coordinates of accepted updates, so that in the end we cannot consider the variances observed as proper ones.
If we leave aside the visualization goal, many other important tasks like clustering in the latent space can be performed without having to know the exact coordinates at each iteration. Since we are only interested in the relative distances between nodes, any kind of information related to the spatial positioning is redundant and we do not need any longer to realign our data through Procrustean transformations, which is an advantage both in terms of computational load and absence of mathematical alterations introduced in the model.
Unfortunately, the final latent estimates in H-LSM are computed by taking the average on all the simulated positions, so we have to change our estimation means if we want to implement an algorithm not relying on the Procrustean approach. For this reason, we chose to develop two different versions of M-LSM, one focused on visualization and one dedicated to the essential description of networks through relative latent distances. Tab. 4.2 shows a summary of the main features of the models described, where P-M-LSM and NP-M-LSM stand respectively for Procrustean M-LSM and Non-Procrustean M-LSM.

| Model | Update type | Multimodality | Procrustean |
|---|---|---|---|
| H-LSM | Batch | Likely | Yes |
| P-M-LSM | Individual | Unlikely | Yes |
| NP-M-LSM | Individual | Unlikely | No |

Table 4.2: Features of the different LSM considered in the analysis

A comparison of representability ratios is displayed in Tab. 4.3: we can remark that the overall performances from the Non-Procrustean version are better, even though the improvement is usually quite contained. The truth is that, especially for simple networks, the final averaged estimates are already well performing,

| Dataset | M-LSM with Proc. transf. | M-LSM without Proc. transf. |
|---|---|---|
| FF | 0.971 | 0.971 |
| Zachary | 0.890 | 0.920 |
| Science | 0.991 | 0.992 |
| NCAA | 0.955 | 0.960 |

Table 4.3: Comparison of representability ratios for M-LSM with or without Procrustean transform

so it is difficult to observe significant differences. However, it is also true that the approach exclusively based on the relative distances between latent positions seems to be always better than the one based on the averaged estimates. The reason is pretty straight-forward: H-LSM defines as final relative distances those between the average positions, while Non-Procrustean M-LSM considers the average of the distances between nodes. The latter approach is definitely preferable and does not need the Procrustean step nor the visualization of each representation, so it is particularly suitable for applications only interested in community detection.

To be honest, we could keep the Procrustean transform as a mean to represent the points coherently and save both the coordinates and the relative distances at each iteration: however, this is computationally not recommended and also useless for the final estimates. We have also seen that the variances are somehow altered, which means that they cannot be used properly for making inference. Therefore, it seems reasonable to exclude the Procrustean step from our simulations, also because we can still represent graphically the final estimates by performing a CMDS on the final relative distances and choosing a space dimensionality such that the distances can be fully respected. This is actually a huge advantage compared to H-LSM, since we can somehow limit the constraints of dimensionality and choose a satisfying value a posteriori so that at least the average distances can be visualized without any information loss.

Fig. 4.14 and Fig. 4.15 show the final configurations of the FF network for the P-M-LSM and NP-M-LSM cases respectively: even though the output has a different orientation, we can remark that the relative positions are actually the same, proving the coherence and validity of both procedures.

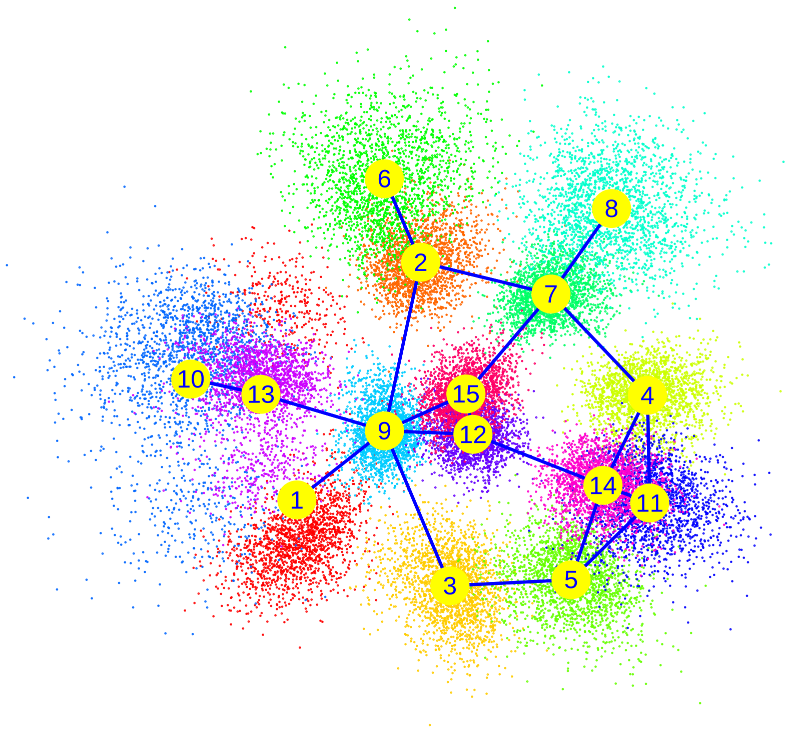

Figure 4.14: P-M-LSM: MCMC realizations and average estimates (yellow points) for the FF dataset

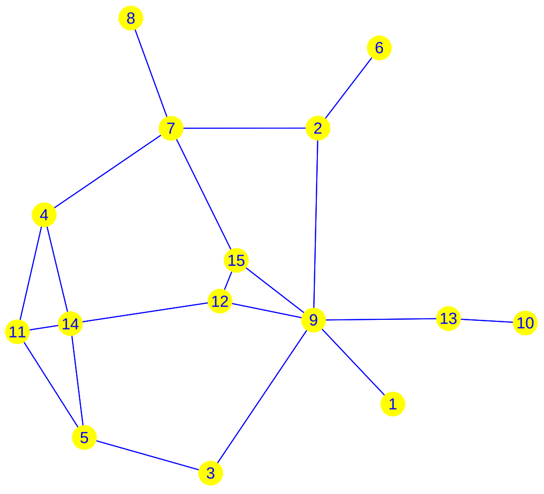

Figure 4.15: NP-M-LSM: Estimates obtained through a posteriori CMDS based on the average relative distances from the MCMC realizations

Finally, the representability issues presented in the Sec. 4.2 are also partially solved by following the Non-Procrustean approach: instead of taking the average of the displayed positions we are now considering the average of the distances, which is more robust to multimodality.
Let us look again at the example we illustrated in Fig. 4.5 and let us focus on the green (G) and red (R) vertices.

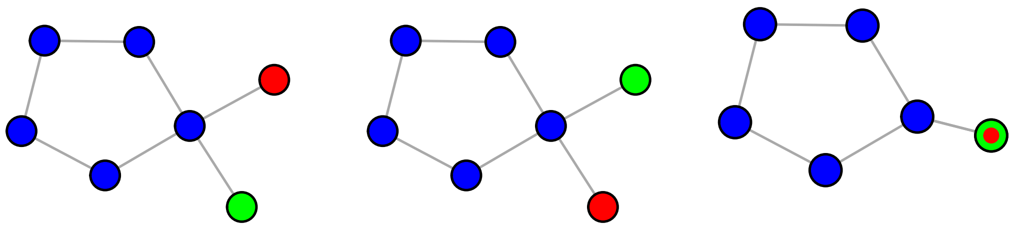

Figure 4.16: Equivalent representations of a network in the latent space (left and central panel) and misleading average estimates (right panel)

In this case, we have already argued that the average on displayed positions is actually meaningless, since we would have a null distance between two disconnected nodes: according to the latent configuration the probability of observing a link between R and G would then be equal to 1, which is obviously not coherent with the observed data. On the other hand, the average on the relative distances will output a meaningful positive value for the distance between R and G and consistent values even for the other vertices, since the relative distance from blue nodes and R/G does not change too much from one configuration to the other.

## 4.6 M-LSM: quick review

In Sec. 4.1 we claimed that M-LSM had been crafted by considering possible solutions to the main problems we found in H-LSM: during the rest of the chapter, we showed this was really the case by analyzing the most critical issues and proving that our changes could actually solve or at least mitigate them.
Our main contributions were related to the implementation of individual updates to replace global ones and to the removal of the Procrustean transformation, which before was deemed as absolutely necessary. Thanks to these new features, our model has a faster convergence towards the a posteriori distributions and handles much better multimodality issues. Moreover, it offers an improved approach to representability matters and also reduces the "noise" created by equivalent configurations, especially thanks to the fact that we are now esti-

mating the final distances by looking at their true values during the simulation instead of computing them from the final average positions.
In the next chapter we will present our final contribution to Latent Space Models, hoping to provide both an interesting example of application and a bridge towards network theory fields like weighted graphs and dynamic models, which we did not consider too much up to this point of the dissertation.

# Chapter 5

# LSM for weighted graphs

## 5.1 Overview

The real networks we have considered so far were all belonging to the same class: the edges were both undirected and unweighted and the nodes did not have any kind of additional feature. The last point is hardly limiting in the context of social networks, where the model was originally developed: we can easily add a covariate term to include additional node-level information while keeping the same conditional distance-based approach.
Unfortunately, the first two constraints are more difficult to relax. Distance is symmetric in itself, thus directed networks cannot be handled properly; weighted edges, depending on how we decide to implement them, could change our model to the point that the current assumptions would no longer be meaningful.
We will provide a detailed discussion about the previous statement in Sec. 5.2, where we will motivate our choices for the proposed extension of M-LSM to weighted networks. To the best of our knowledge, this is a completely new feature that allows us to improve the mapping in the latent space by taking into account the strength of the edges: the likelihood needs to evolve from its previous formulation, so we will detail the changes that we had to introduce.
We will refer to the new model as Weighted LSM (W-LSM in short) and display its good performances on a slight variation of a well-known dataset in Sec. 5.3: the Florentine Families network will be transformed into a simple weighted graph by changing the value of one edge at a time, allowing us to observe focused structural modifications in the latent representation.
Finally, the last section will be dedicated to an innovative study on a particular dataset: Amazon reviews and metadata from May 1996 to July 2014, already organized and used for academic purposes as in [30, 31]. We used the data[1] about digital music ratings and reduce the initial number of 836,006 reviews

[1]Retrieved from http://jmcauley.ucsd.edu/data/amazon/, *Digital music - ratings only.*

until we had only 94 unique users. We will later explain and justify the process, but the important thing is that we could properly explore and represent the latent space of the users and observe clusters of people with similar musical tastes.

## 5.2 W-LSM

A link between two objects simply tells us that there exists an observable relationship, but we do not know how strong it can be. Depending on the context we may actually need to have this kind of additional knowledge, as it could be necessary to differentiate node behaviors and obtain meaningful insights about the graph structure. Weighted networks carry exactly the information we are looking for, since they provide us with details related to the strength of each edge.

Statistical models for weighted networks can reach quite a variety of goals: for instance, they can focus on forecasting link values or estimating statistics like centralities and clustering coefficients to analyze the variations with respect to the corresponding unweighted networks. However, what we are trying to obtain by extending latent space models to weighted networks is completely different, so we will discuss some relevant possibilities during the rest of the chapter.

Let us consider a graph with node-specific observable features. We know we can define a similarity function based on the attributes such that the link probability can be based upon it: if we possess information about the strength of the edges we can improve our analysis and estimate not only the link probability but also its weight. If we think about our experience we can probably remark that existence and strength are often correlated: for example, relationships are generally stronger between people with a similar background. However, this is not a general rule and decoupling strength and existence should allow us to model extreme behaviors as unlikely yet strong bonds or very likely links with low strength.

These considerations are particularly relevant in the framework of Latent Space Models: throughout the dissertation we have seen that the link probability depends on the relative distance in the latent space, but the link strength is not taken into account. Therefore, if we want to generalize the model to weighted networks we have to identify the most suitable type of extension.

We can actually follow very different approaches:

- simply neglect the weights, falling again into the category of unweighted networks: this is clearly not the best option as we are ignoring some information;

- define a new threshold $\beta$ such that only nodes with a strong connection will be placed at a distance smaller than $\beta$;
- integrate the weights in the likelihood so that stronger connections will translate into closer positions in order to maximize the likelihood;
- change the type of variable used to describe the edges from binary to discrete or even continuous and let it assume the values corresponding to the weights. In this case, we obviously need to redefine the hypothesis of conditional dependence on the latent distance in order to properly create a distribution over positive values for the edge variables.

We can easily observe that not all the strategies have the same level of complexity: the first two are basically workarounds to apply the same model we used for unweighted networks, while the last ones need to introduce real changes in the model formulation. We may think that creating a new type of variable is the best choice since it is the most complete option, but unfortunately there is a big issue: we cannot model both existence and strength by recurring to the same latent distance. We are forced to make some arbitrary choices to find a good compromise: our goal is to determine latent positions such that the configuration maximizes the graph likelihood, but in the new scenario its value is influenced by the two possibly unrelated estimations of existence and strength. Therefore, we are trying to give the same explanation to two different phenomena: the only way we can succeed in this task is by linking the two aspects so that we can establish some kind of priority. For instance, we could model the probability of observing a specific weight in terms of the latent distance and choose the relative positions that optimize our usual likelihood multiplied by this new extra factor.
Let us suppose we have four nodes and some of the possible connections between them with the properties described in Table 5.1:

| Edge $e_{ij}$ | $P(e_{ij}=1)$ | $P(w_{ij}=1)$ | $P(w_{ij}=10)$ |
|---|---|---|---|
| $e_{12}$ | 0.95 | 0.9 | 0.01 |
| $e_{13}$ | 0.5 | 0.4 | 0.4 |
| $e_{14}$ | 0.1 | 0.05 | 0.9 |

Table 5.1: Critical scenario of unrelated strength and existence propabilites

In the example we have node $v_1$ serving as center or reference and we want to understand which should be the best positioning for the other vertices in terms of distance from $v_1$:

- $v_2$ is very likely to form a bond, but this will probably have a low intensity;

- $v_3$ is less likely to form a connection, but the weight will be greater compared to $w_{12}$;
- $v_4$ is unlikely to create a link with $v_1$, but if this happens it will almost surely have a strong intensity.

If we stick to the normal assumption that latent distance should be inversely proportional to the weights of the edges and directly proportional to the probability of link existence so that both strongly connected and structurally similar nodes should be located in the same region of the latent space, then we can remark that we should represent the nodes $v_2, v_3 and v_4$ at almost the same distance from $v_1$. However, by doing so we would lose the ability to reconstruct their behavior starting from the latent positions. We did not have this problem for unweighted networks, since we could tell that two nodes had to be structurally similar in the observed graph if they were close in the latent space: this was precisely the reason why the model and the latent representation were meaningful and effective, while in the current scenario we lose track of the original node features and we only have an idea about the mixed effect of link existence probability and weight intensity. This is quite a letdown, since the intuitiveness of the latent configuration is lost, but the real drawback is that clustering would regroup nodes without letting us have a clear understanding of their common features, which could be either structural similarity or strong connections.
Given these issues, we decided to ignore the decoupling between existence and strength and limit our model to those networks in which it is safe to assume that bigger weights are exclusively related to edges that are likely to exist. Thanks to this hypothesis we can simply interpret weights as additional constraints to further reduce the distance between nodes: if we increase the weight of a specific edge, then the two incident nodes should become closer in the latent space.
The logit function of the link probability in H-LSM was given by :

$$\text{logit}\ (p(Y_{ij} = 1 \mid \alpha, z_i, z_j)) = \alpha - \|z_i - z_j\| \tag{5.1}$$

In order to take the weights into consideration, we changed the formulation to:

$$\text{logit}\ (p(Y_{ij} = 1 \mid \alpha, z_i, z_j, \tilde{w}_{ij})) = \alpha - \tilde{w}_{ij}\|z_i - z_j\| \tag{5.2}$$

To carry out the weighted extension properly, we need to define the variables $\tilde{w}_{ij}$, which are different from the weights $w_{ij}$ of the observed network. Indeed, if we were to consider the original weights we would have null values multiplying the latent distance in all the cases where the link is missing: in other words, we would not be taking into account the latent distances for disconnected vertices. Neglecting part of the latent distances clearly leads to a biased estimation of

the optimal latent configuration, meaning that our model would no longer be useful.
Let us compare Eq. 5.1 and Eq. 5.2: since the two formulations have to be equivalent for unweighted networks, $\tilde{w}_{ij} = 1 \; \forall i, j$ every time we use H-LSM. Therefore, the coefficients $\tilde{w}_{ij}$ we introduce to model weighted networks have to satisfy two constraints:

- $\tilde{w}_{ij} \propto w_{ij}$ : as long as the requirement is met, the relationship can be arbitrary and allow us to stress or reduce the effect caused by the weights;

- $\tilde{w}_{ij} \geq 1$ : the unity threshold is simply given as a coherent value with respect to what we observed for unweighted networks. More generally, we could say that all values should be positive and greater than the coefficient fixed for non existing links.

Finally, from a computational point of view the introduction of a weight matrix to be passed along other arguments to the core functions of the MCMC procedure contribute to slow the R code whose speed was already reduced by the adoption of individual updates. For this reason, we rewrote entirely in C++ the core functions of the MCMC procedure and changed the structure of many of them in order to reduce the computational time as much as we could.
All the results we will present in the following sections have been obtained by running the code presented in App. A, which is perfectly consistent with the results from H-LSM on the benchmark datasets in Ch. 3.

## 5.3 W-LSM test on FF dataset

The Florentine Families is the most suitable dataset to test our different variations of H-LSM: the graph structure is simple yet definitely not trivial, so we can take full advantage of the small number of nodes to have fast simulations and clear representations while correctly investigating the performance of our model extension to weighted networks.
We started by running W-LSM with $\tilde{w}_{ij} = 1 \; \forall i, j$ to ensure the coherence of its results with those obtained by H-LSM: as we can see from the usual output in Fig. 5.1, we obtained one of the two equivalent configurations we observed previously, meaning that the new model is indeed suitable for unweighted networks. We also remark that we had only one equivalent configuration thanks to the fact that we are avoiding multimodality by recurring to individual updates.

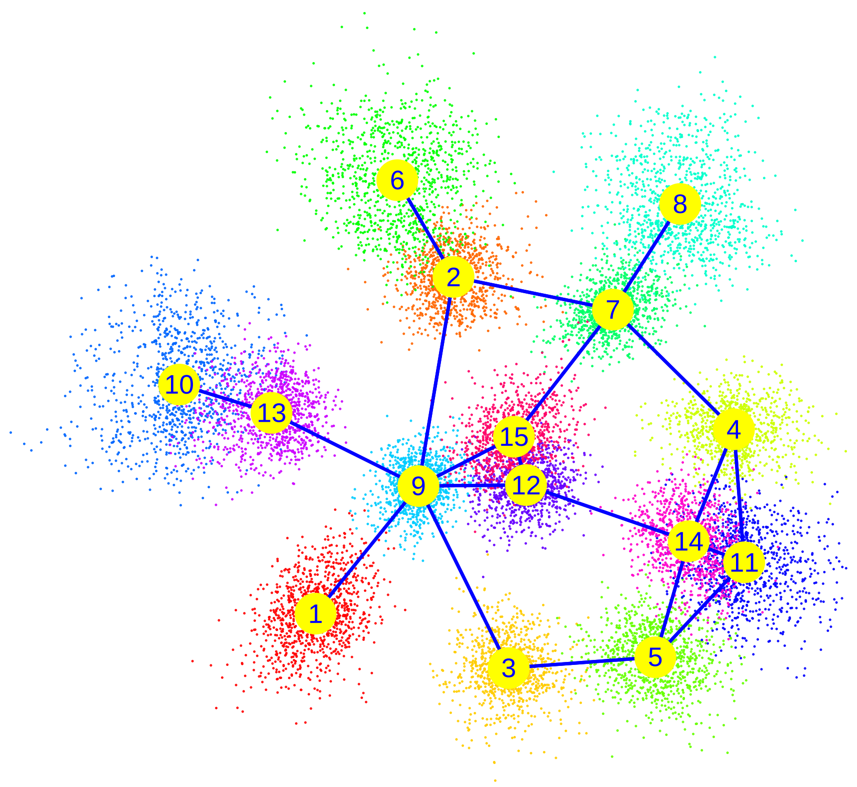

Figure 5.1: MCMC realizations for the FF dataset using W-LSM. Colored points correspond to stochastic positions and yellow dots indicate the average position for each node. Blue lines represent the network.

The second step we took in order to validate W-LSM consisted in creating slightly different versions of the FF network by altering the weight of one edge at a time. In other words, the network remains essentially unweighted except for one link, allowing us to experiment in a very simple and controlled scenario. Our immediate goal is to verify that the model output meets our expectations, meaning that increasing the weight from the default $w_{ij} = 1$ to $w_{ij} = 2$ should reduce the distance between nodes $v_i$ and $v_j$.
Figg. 5.2, 5.3 and 5.4 show modifications in edge weights between nodes 9 and 13, 2 and 7, 9 and 14 respectively. These figures actually serve a twofold purpose: first, the outputs are clearly pointing in the right direction, since increasing an edge weight leads to a reduction of the latent distance between its incident vertices; secondly, even though we are limited to a qualitative remark, we can see that the graph structure does not allow all the nodes to react in the same way. The latter point is particularly interesting, as it proves that the configuration is more or less robust to variation in terms of weights depending on the importance of some positions for the overall likelihood. Peripheral nodes tend to be more subject to such variations, while central ones need to properly balance strengthened connections with all the other relative distances.

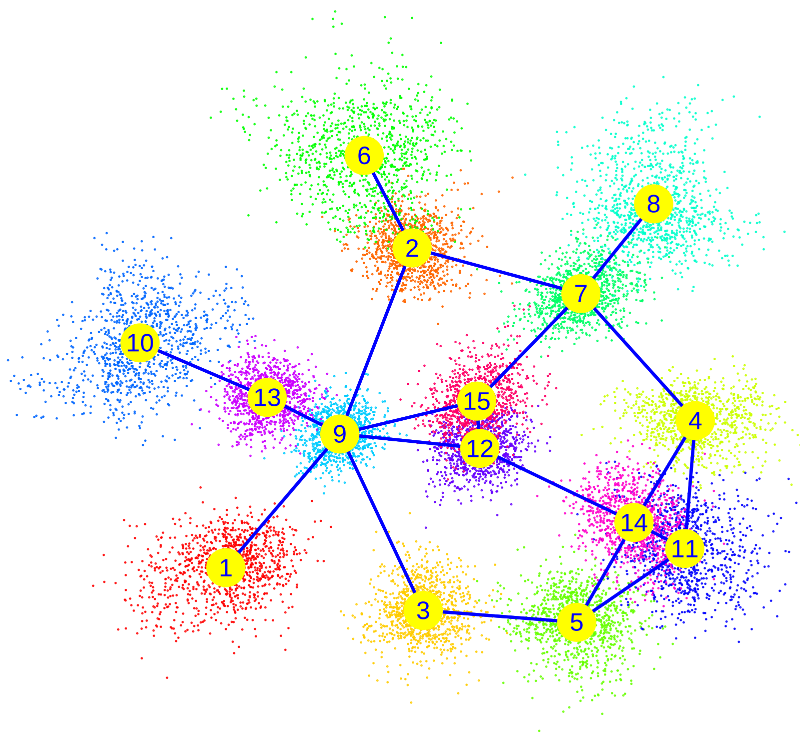

Figure 5.2: $w_{ij} = 1 \; \forall i, j$ except for the connection between nodes 9 and 13, whose weight is doubled.

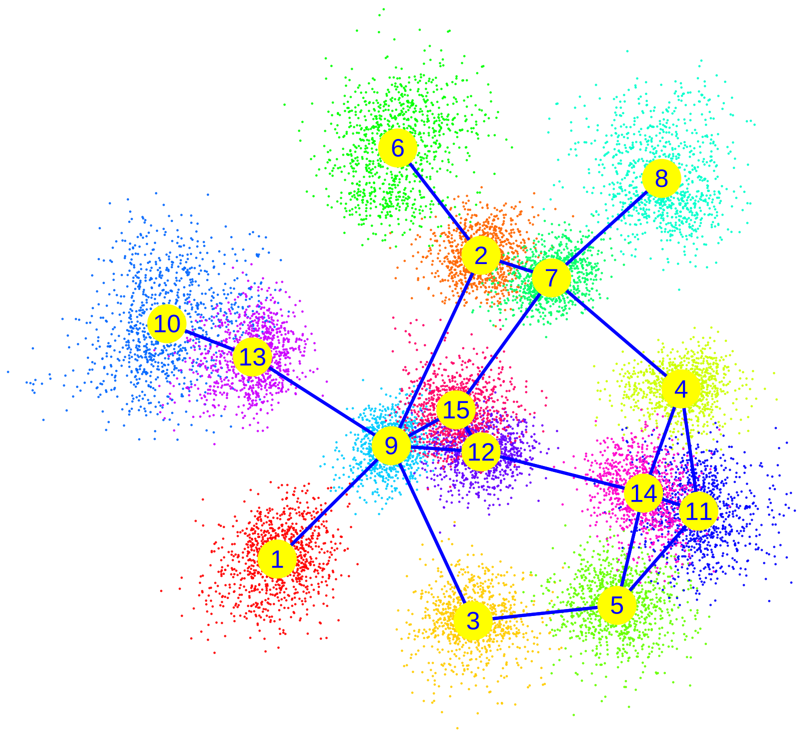

Figure 5.3: $w_{ij} = 1 \; \forall i, j$ except for the connection between nodes 2 and 7, whose weight is doubled.

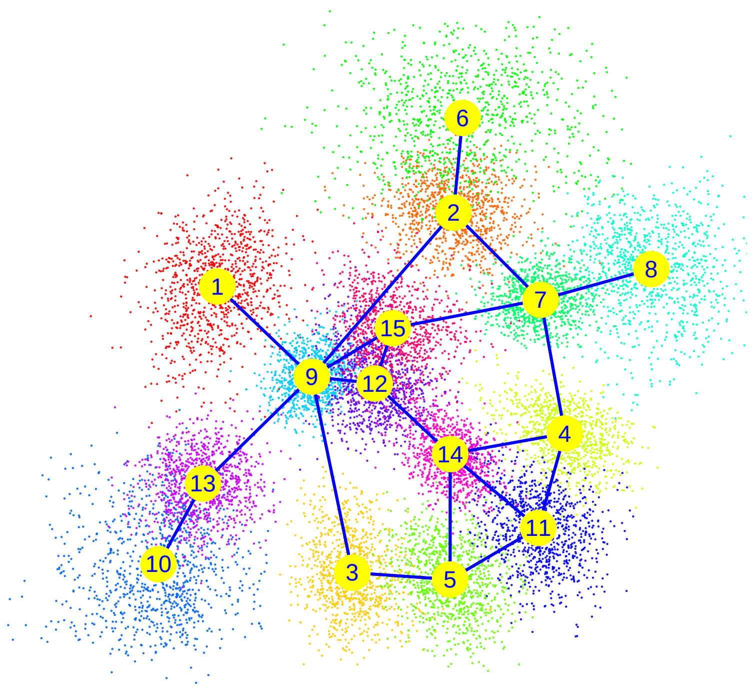

Figure 5.4: $w_{ij} = 1 \ \forall i, j$ except for the connection between nodes 12 and 14, whose weight is doubled.

Finally, Fig. 5.5 is truly important because it provides us with the following information:

- multiple changes are introduced: the model is effective and the final output is a sort of compromise of the three configurations recently shown for individual changes;
- individual updates are still able to prevent multimodality issues, which is something we could not take for granted as the reformulation of the likelihood could have increased the probability of switching between equivalent configurations;
- the $\alpha$ threshold (corresponding to the radius of the colored circles) keeps his "guardian" role: we can verify that all connected vertices fall within an $\alpha$ radius from each other, while the opposite happens for disconnected nodes.

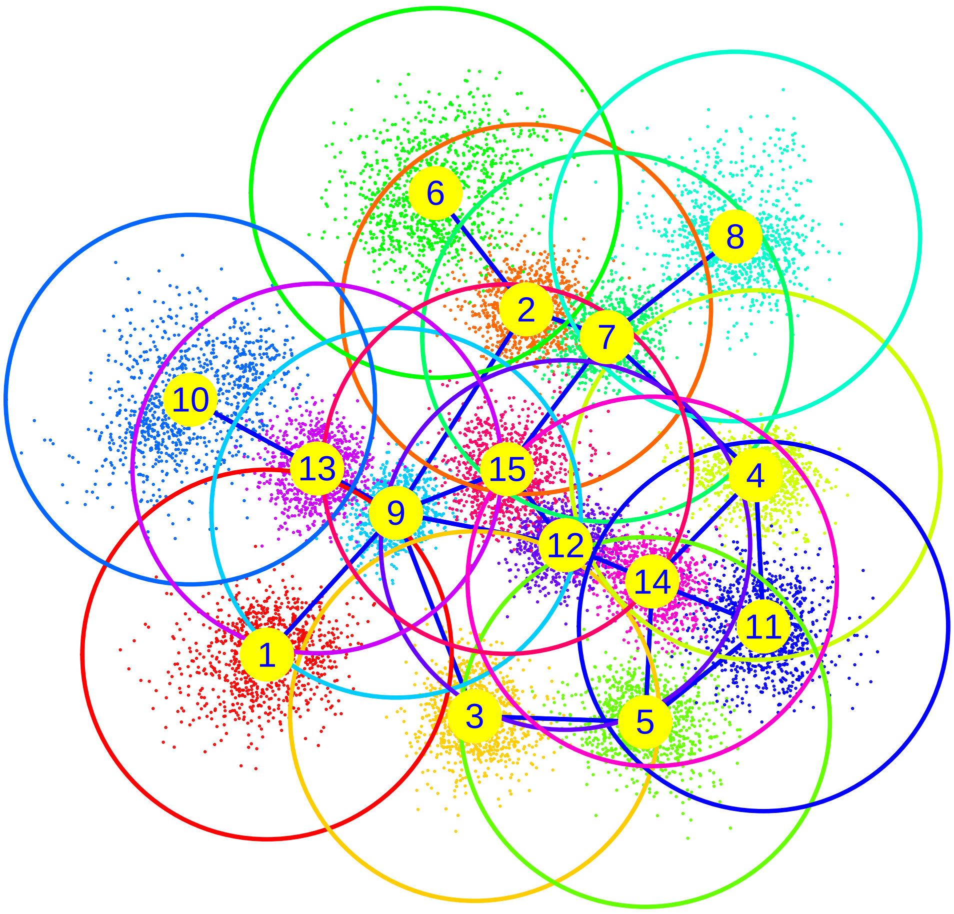

Figure 5.5: $w_{ij} = 1 \; \forall i, j$ except for the connection between nodes 9 and 13, 2 and 7, 12 and 14, whose weights are doubled. Colored circles are used to enhance the role of the $\alpha$ threshold.

The conclusion of this section is simple and satisfactory: W-LSM works well with both unweighted and weighted networks, provided that we are dealing with graphs where it is safe to assume that bigger weights are associated with highly probable links. If such an hypothesis is verified, the final latent configuration is meaningful and allows us to cluster nodes that are structurally similar in the network.

## 5.4 Amazon music ratings

We are finally ready to present the application of W-LSM to the Amazon dataset, even though we still need to highlight some crucial points before going into details. For this reason, in the next two sections we will motivate some of the choices we will discuss by looking at the constraints we had to face. We will also introduce the concept of bipartite networks, which will be fundamental to understand our analysis on Amazon's data.

### 5.4.1 Motivation

Latent Space Models were initially meant for applications on social networks: not only the subjects were interesting, but they also possessed some remarkable feature compared to other kind of networks.
Albert-László Barabási, one of the most renowned active researchers in the field of network theory, named a chapter from his book “Linked: The New Science of Networks” in a very suggestive way: “A web without a spider”, which is an incredibly simple and powerful image to describe the self organizing nature of many complex networks. Most of the times, social graphs belong to this category: the absence of a central authority allows actors to create very heterogeneous structures, usually decentralized and characterized by significant clusters.
In the LSM framework, these features are actually requirements: we already know that star-like graphs are critical for representability issues and also uninteresting as they do not possess a proper community structure. Therefore, social networks seem to be the perfect candidate for latent space models, also because they are generally independent from measurable distances: if we observed a network clearly based on distance (traffic, power grids, transmissions), then LSM should output a configuration which is more or less similar to the positions of the nodes in the real space. Usually we possess this kind of information, so the model estimate is a bit uninteresting as it could only be used as a tool to check if the dependence from distance is truly as imagined: however, LSM can also prove to be a great resource in this context if the real distances are difficult to measure precisely or cannot be observed directly.
The brief considerations presented so far were meant to give an idea of the constraints we face when searching for an application outside the field of social networks: we need to keep in mind that not all models are equally effective for a specific set of observations, so we carefully have to choose a relevant network type for our analyses.

### 5.4.2 Bipartite networks

Bipartite graphs are a special class of complex networks: the nodes are divided into two sets and internal connections are not allowed. A traditional example is given by the links between actors and movies, where an actor-movie link stands for "actor A played in movie M", but many more situations can be easily described in terms of bipartite networks. A practical representation is given by the so called *incidence matrix*, a rectangular matrix whose dimension is given by the cardinalities of the two connected sets: the definition is very similar to the one we provided in Def. 1.3, except that the we are now considering nodes both for rows and columns as it is more understandable. In other words, let $B$ denote the incidence matrix: $B_{ij} = 1$ if we observe a link between the $i$-th element of the first set and the $j$-th element of the second, $B_{ij} = 0$ otherwise.
The reason why we are suddenly interested in bipartite networks is that they can be used as a basis to obtain weighted graphs through a simple operation called *projection*: we can either compute $B \times B^T$ or $B^T \times B$, depending if we are interested in obtaining a weigth matrix related to the relationships among the objects belonging to either the first or the second set. Let us considering once again the actor-movies example and let us call the two sets A and M: by projecting the incidence matrix we will get a square weight matrix whose size could be $|A| \times |A|$ or $|M| \times |M|$. Such matrix can be thought as a normal adjacency matrix, except for the fact that its non-zero elements are weights and they do not correspond to real links since the information they provide is related to some similarity in terms of common "neighbors" from the other set. In other words, if we project our bipartite network of actors and movies onto the actors' space then the $(i, j)$-th element of the adjacency matrix will tell us the number of movies in which actors $i$ and $j$ played together.
Since we are dealing with weight values, we can rescale the matrix as much as we like and this is especially useful if we want to create weights that are suitable to represent the recently created network of actors. We could trim outliers, discretize the coefficients or group them into categories if we want to allow our edges to assume only a limited number of possible values.
Even more generally, the weight matrix can represent user-defined similarities: if we assume we have a binary incidence matrix for a bipartite network than we can rescale the weight matrix so that the $(i, j)$-th element will be given by the dot product between rows $i$ and $j$. Actually, the initial matrix does not need to be binary, since we can compute similarities between any pair of vectors: the matrix obtained through projection can be seen as a weighted network, where the weight associated to each edge is related to the similarity between the incident vertices.

### 5.4.3 W-LSM analysis on Amazon dataset

In recent years, *recommendation systems* and *collaborative filtering* have definitely been a hot topic: given a group of users and a set of products, a great economic interest lies in understanding which kind of suggestions we should make to a specific user in order to lead him to buy a product. From a mathematical perspective, the challenge consists in defining efficient algorithms to optimize recommandations based on the most relevant features we can extract from specific users: the task is really difficult, since it is often hard to define behaviors and tastes from the available characteristics. For this reason, recent approaches to the matter shifted from individual to group analyses, meaning that the users are now clustered thanks to similarity values: the essential idea is that we should recommend to each user only items appreciated by similar people, so that the chances to meet the user's tastes should increase.
The related literature is already full of well performing models, but almost all of them are focused on what we could label as "local comparisons": given user $u_1$, they find the most similar users $u_2, u_3, u_4, \dots$ and use them to refine the recommendations. Our goal is completely different, since we are interested in mapping the whole users space thanks to W-LSM so that the scope may become global by taking into account all the relative distances that are usually neglected. Moreover, we change the nature of clusters by starting from latent positions instead of similarities. This is clearly an approximation of the initial ratings data, but it offers many great advantages:

- we can merge ratings and reviews with real network information about users, which could prove to be extremely powerful if we consider that we are currently experiencing a huge increase in data collection about users from many actors like social networks or advertising companies;

- we can store information much more efficiently by mapping and updating the users latent space thanks to dynamic models as in Sec. 2.3.7. The alternatives consist in either keeping in memory all the reviews in order to re-evaluate similarities for every update or introducing other approximations to deal with storage issues for large amount of data;

- if the network is representable, we can investigate visually the users space and possibly gain some insights about their common behaviors;

- we could apply a whole set of network analyses and measures (centralities, for instance) to our data and use them to target differently the most critical users.

Given their potential, we decided to apply latent space models to the weighted network obtained from Amazon data. As we stated in the introduction, the dataset of music ratings (values from 1 to 5) consisted in 836,006 reviews and ratings made by 478,235 users, but we decided to narrow it down to the ratings belonging to a special list of unique users. We wanted to have an understandable representation of the users latent space, so we started our screening by setting some arbitrary thresholds and choosing only people that had reviewed at least 3 common songs, where "common" arbitrarily stands for "reviewed at least 55 times". Then, we evaluated the distribution of number of reviews per user and kept only people who were between the $30th$ and $70th$ percentiles, so that we finally ended up with 94 unique users who made a similar (or at least comparable) number of reviews. As a final step, we normalized the weigths so that they were all included in the $[1, 3]$ interval and we created the network corresponding to the binary adjacency matrix whose non-zeros elements had the same indices of the non-zero elements of the weight matrix.
Most of the data processing we made was meant to avoid the scenario where the majority of the weights are positive: if we consider the extreme case of only positive values, then LSM should collapse all the latent positions as the likelihood would be easily maximized since there are no disconnected nodes in the corresponding graph. We could probably find a solution by setting some thresholds to redefine connection between vertices, but we preferred not to introduce additional arbitrary parameters.
Fig. 5.6 shows the latent estimates obtained by W-LSM without Procrustean transformation.

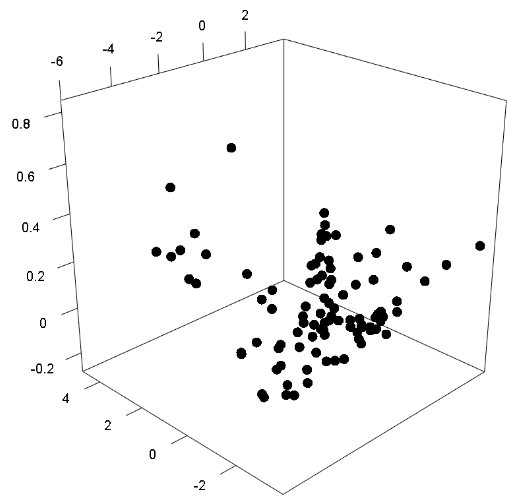

Figure 5.6: W-LSM application on Amazon music ratings: the network nodes are mapped into a three-dimensional latent space by following the non Procrustean approach with individual updates.

The first obvious remark is that we are in a three-dimensional space, proving that W-LSM can indeed work with higher dimensionalities: however, what is important to notice is that the space is actually fully occupied by the points, meaning that due to the complexity of the graph structure we would probably have representability issues within a two-dimensional representation.
We know the network has weights related to the similarity between users, where similarity is an index of how closely they gave the same ratings in music reviews. We also know that stronger similarities should lead the algorithm to regroup users, so in order to investigate if particular tastes truly correspond to specific regions of the users latent space we decided to consider one song at a time and highlight the users that reviewed it.
We adopted five colors to easily identify the possible ratings:

1 - red
2 - orange
3 - yellow
4 - green
5 - blue

where 1 denotes a poor rating and 5 is the maximum value allowed.
Fig. 5.7 and Fig. 5.8 illustrate different scenarios: in the first page we collected outputs where the music was reviewed and appreciated equally by users, while in the second page we focused on songs with mixed reviews.
As we can onbserve, similar ratings were successfully grouped together, while the behavior is clearly different when the ratings are mixed.

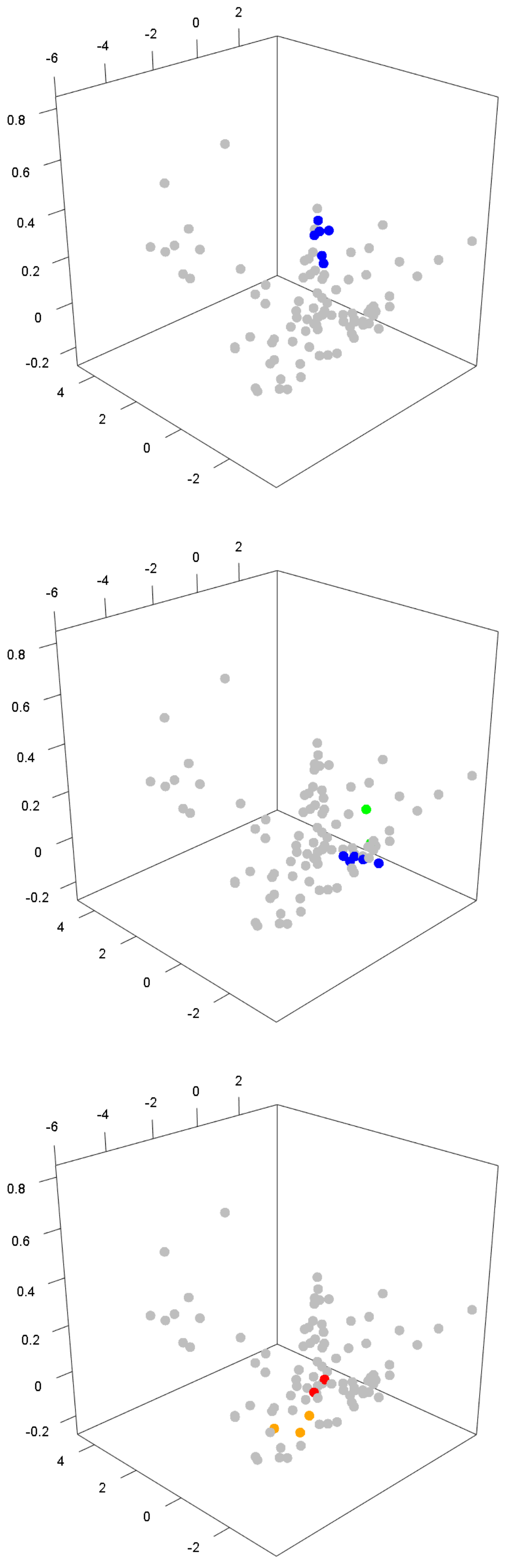

Figure 5.7: Each plot shows the ratings made by users for one specific song at a time: we observe that users who made similar reviews (either positive or negative) are clustered together.

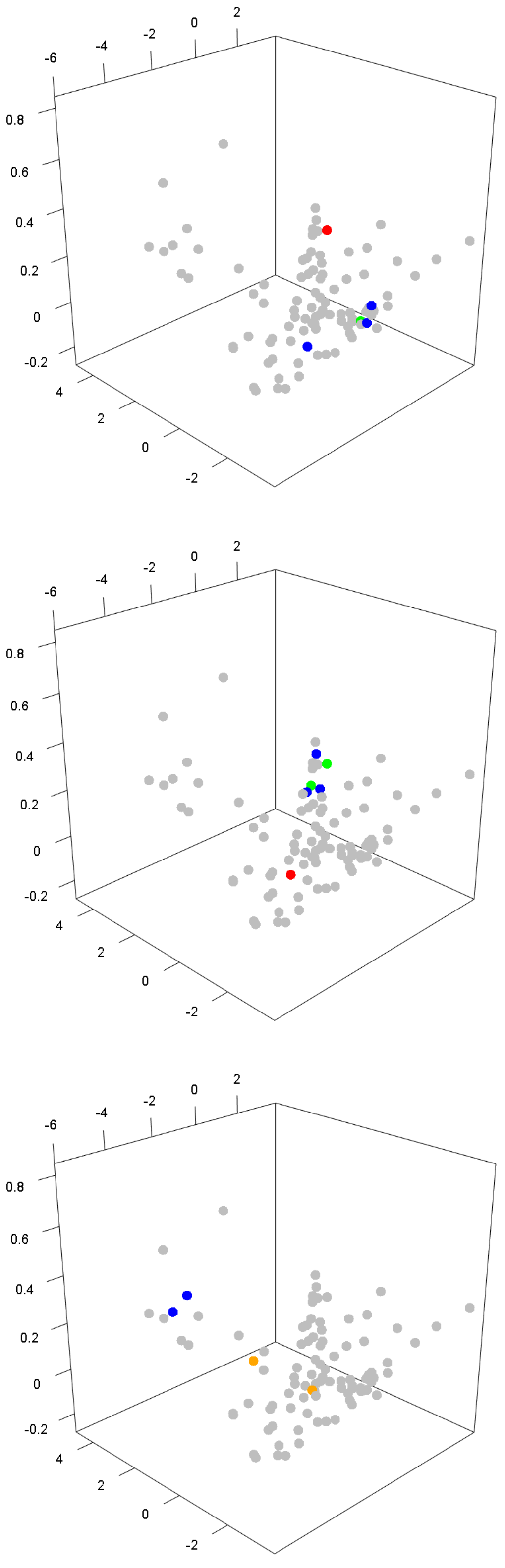

Figure 5.8: Each plot shows the ratings made by users for one specific song at a time: we observe that users who made dissimilar reviews (either positive or negative) are not clustered together.

Thanks to these outputs we can finally claim that W-LSM was actually able to map the users in a latent space which appears to be meaningful with respect to musical tastes.
As previously explained, we now possess a representation of the global scope of similarities between users which could be improved by merging additional information like true interactions between people (blogs, chats, followers). Going back to the framework of recommender systems, we could choose a specific user and propose a new song by looking only at those appreciated by the closest users in the latent space: unfortunately, we do not have such data at our disposal, but we believe it would be very interesting to compare the performances of "traditional" recommender systems versus network-based analyses from latent space models.

We will present very shortly our final considerations about the whole work presented in the dissertation, focusing more on high level concepts than analytical matters. For this reason, let us conclude this section with a last technical confirmation, so that we will hopefully leave no doubts as to the performances of W-LSM on complex weighted networks.
Fig. 5.9 shows the usual MCMC diagnostics we considered for our past simulations: we can observe that we have a good mixing of the chain and low autocorrelation coefficients ($2 \times 10^6$ iterations, $10^6$ burn-in, $10^3$ thinning).
Finally, individual updates and the removal of Procrustean transformation seem to be once again worthy of our attention, since we do not observe multimodality in the traceplot and we are able to visualize a three-dimensional latent configuration without the noise created by all the MCMC realizations.

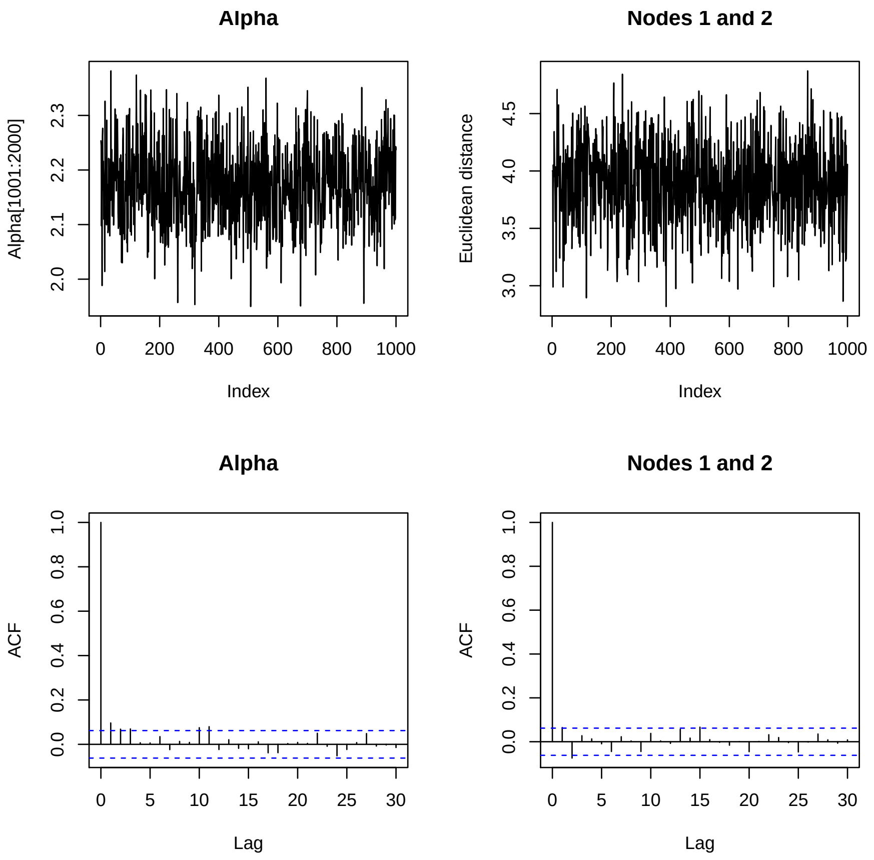

Figure 5.9: M-LSM: Traceplots and ACF plots for W-LSM on Amazon music ratings.

# Conclusion

Throughout the dissertation we have essentially developed our work along two main axes: the statistical analysis of theoretical features of LSM and the optimization of the algorithms needed for our simulation.
Regarding the former, we have introduced some significant changes in the original H-LSM formulation: the proposed W-LSM extension is able to deal with weighted complex networks and has a better behavior in terms of MCMC convergence and robustness.
Concerning the latter, we have entirely rewritten the core functions in C++ in order to speed up the process. We also modified the functions used to propose the candidates during the MCMC routine so that they are now minimizing the number of matrix computations needed for the updates and thus allowing us to implement efficiently the individual proposals.
In the following sections we will give some more details about what has been accomplished and we will eventually conclude our work by presenting possible future developments.

## 5.5 Model improvements

Tab. 5.2 shows the final comparison between all the different versions of LSM we have discussed: P and NP are used respectively for Procrustean and Non-Procrustean, while the rest of the notation is the same from the previous chapters.

| Model | Update type | Multimodality | Procrustean | Weights |
|---|---|---|---|---|
| H-LSM | Batch | Likely | Yes | No |
| P-M-LSM | Individual | Unlikely | Yes | No |
| NP-M-LSM | Individual | Unlikely | No | No |
| P-W-LSM | Individual | Unlikely | Yes | Yes |
| NP-W-LSM | Individual | Unlikely | No | Yes |

Table 5.2: Comparison between LSM versions

As we remarked in Ch. 4, our main contributions were initially related to the implementation of individual updates and to the removal of the Procrustean transformation. Thanks to these new features, our M-LSM model has a faster convergence towards the a posteriori distributions and handles much better multimodality issues. Moreover, we proved that M-LSM also offers a better approach to representability matters by reducing the "noise" created by equivalent configurations, since we are now estimating the latent distances by looking at their values during the MCMC simulation instead of computing them from the final average positions.
Due to the good performances of M-LSM we decided to keep the modified features also in our model extension to weighted networks, which is already quite significant per se as it allows us to generalize our analyses to a whole new class of networks. In Ch. 5 we first proved the validity of W-LSM by testing it on the Florentine Families and observing good results, then we used the model to inspect a weighted network obtained by projection of the matrix corresponding to Amazon music ratings. The mapping of the users space we found seemed pretty interesting, especially because it offered an overall representation of the different tastes and behaviors. Finally, we explained that there can be many possible advantage by merging similarity and network information, so that the model can become a compromise between recommendations based exclusively on ratings and suggestions coming from network analyses.

### 5.5.1 BLSM Package

We adapted the core functions to the new features and used C++ to speed up the algorithms so that they could be scaled to bigger networks. To the best of our knowledge, even though there exist other R packages for latent models, the BLSM package (Bayesian Latent Space Model) will be the only one allowing for the Non-Procrustean choice and applications to weighted networks.
Concerning its performances, Tab. 5.3 offers a quick comparison of running times in seconds with respect to the initial version of H-LSM:

| Network | Size | W-LSM | H-LSM |
|---|---|---|---|
| Florentine Families | 15 | 13.30 | 54.01 |
| Zachary's karate club | 34 | 64.05 | 112.81 |
| NCAA Network | 115 | 752.80 | 722.37 |

Table 5.3: Comparison between running times for H-LSM and W-LSM

The results are obtained by using the same parameters for $10^5$ iterations. The running times may seem to suggest that H-LSM scale much better, but the truth is that the algorithm is slow especially when it has to accept an update and perform the Procrustean transformation: due to the batch update, H-LSM is actually accepting around 50% of the updates accepted by W-LSM in the Zachary's test and only 5% in the last example. In other words, if we wanted to have the same effective sample size from W-LSM and H-LSM than we should multiply H-LSM values and obtain around 14000 seconds for the NCAA network, which is 20 times higher than the value from W-LSM.

## 5.6 Further developments

The model we labeled as H-LSM has actually evolved since its creation by new contributions from the original authors (Hoff, Raftery and Handcock) as in [18]. However, they followed a different approach from what we proposed, focusing on coupling clustering and estimation of the latent positions for social networks so that the model could benefit from an additional hierarchical level for the priors and include the belonging to a latent cluster in the likelihood. Even if it is undoubtedly an interesting extension that can improve the clustering process, this kind of evolution leads unfortunately to known issues of identifiability and label switching, which can only be solved by imposing further constraints (a bit like the Procrustean transformation) or analyzing a posteriori the simulated iterations of the MCMC.

During our work we were in pursue of different objectives, mainly the thorough discussion of some important modeling aspects (individual updates, multimodality, MCMC convergence, Procrustean transformation) and the extension to weighted networks in order to apply the latent space model even to networks not belonging to the "social sphere". Following this direction, we think it would be worthy to investigate furhter the analysis of weighted networks by considering scenarios where the link existence probability and strength are actually unrelated, so that new types of networks could be mapped into the latent space. Also bipartite networks could offer new possibilities in terms of biclustering or different types of projection and weight transformation, since these operations need to be specifically evaluated for each network observation.

Another development could be related to the analysis of the a posteriori distributions observed for the relative distances, which we used in the Non-Procrustean approach without really investigating the possible differences between them. For instance, a clustering based on such distributions could result in a soft assignment procedure and thus offer new insights related to the nature of the identified

groups.
Finally, we could observe the latent representation from yet another perspective and try to find a way to characterize different network observations by comparing their latent structures: for instance, we could define distances over the a posteriori distributions to quantify the dissimilarity between two networks. Given a series of i.i.d. observed graphs, we could also identify outliers starting from the average latent configuration of the series, or at least quantify the dissimilarity from the expected network realization.
All these directions are promising and testify the important role of Latent Space Models in the statistical approach to network data analysis. More generally, thanks to the great potential coming from the coupling of intuitive representation and strong adaptability to complex scenarios, these models are definitely worthy of additional investigation and improvement.

# Appendix A

# R code

## Main script (except for visualization functions)

```
############################################
# Preliminary settings and Initialization #
############################################

rm(list = ls())
graphics.off()

directory"custom_directory"
setwd(directory)

# Load packages
library(Rcpp)
library(GMD)
library(igraph)
library(igraphdata)

# Some auxiliary functions
source("WAux1.R")
sourceCpp("WAux2.cpp")


# Custom settings
procrusteanFALSE
dynamic_plotFALSE
dynamic_initdynamic_plot
dynamic_circlesTRUE
save_plot_fileFALSE


# MCMC parameters
# Proposal distribution
adelta = 0.4
zdelta = 0.1

# Iterations
burnin = 10^6
nscan = 10^6+burnin

# Thinning
odens = 10^3

# Maximum number of clusters
mnc15

# Set random seed
set.seed(1)

```

```
# Data import
N.B.:  we show only the karate's example as the data can be loaded from the igraphdata package

data("karate")
karate_mat = as.matrix(as_adjacency_matrix(karate))
colnames(karate_mat) = c()
rownames(karate_mat) = c()
Y = karate_mat
name = "/Images/Zach_"


# Data processing
# Take out disconnected actors
zrowhich(rowMeans(Y)==0)
if(length(zro)>0) {Y = Y[-zro,-zro]}

# Remaining parameters
N.B.:  the dimensionality is not fully custom yet, so we cannot choose it at the beginning

n = dim(Y)[1]     # number of nodes
k = 2             # latent space dimensionality
mnc = min(mnc,n) # update of maximum number of clusters

# Graphical settings
my_colors = rainbow(n)
graph_params = c(1,5,1)


# Starting values for Z and alpha
alpha = 2

# Classical MDS
D = dst((Y>0)+0)
Z = cmdscale(D,k)

# Centering values
Z[,1] = Z[,1]-mean(Z[,1])
Z[,2] = Z[,2]-mean(Z[,2])


# Non linear optimization routine
avZ = c(alpha,c(Z))

# Simulated annealing
avZ = optim(avZ,mlpY,Y=Y,W=W,method="SANN")$par # mlpY: negative value of the graph log-likelihood

# Downhill simplex
avZ = optim(avZ,mlpY,Y=Y,W=W,method="Nelder-Mead")$par

avZ = avZ*2/(avZ[1])  #MLE too extreme, keep shape but rescale
alpha = avZ[1]

# Z.mle: objective value of the Procrustean transformation in the for loop
Z.mle = matrix(avZ[-1],nrow=n,ncol=k)
Z = Z.mle

# Distance matrix
lpz = lpz_dist(Z)


# Representability of Y at the end of the initialization
RecMat = (((alpha+lpz)>=0)+0)
diag(RecMat) = 0
repr_init(sum (YRecMat) - n)/ (n*(n-1))




```

```
########
# MCMC #
########

aca = acz = 0  # acceptance rates
Alpha = alpha  # creating alpha vector
Lik = lpY(Y,lpz,alpha,W)  # alpha and likelihood

Zp = list()      # positions
# The position of node i at iteration k is given by column i of matrix Zp[[k]]
for(i in 1:k){
  Zp[[i]] = t(Z[,i])
}


# Two alternatives (with/without Procrustean transformation)
if (procrustean){
  if (dynamic_plot){
    x11(xpos=0,ypos=0)
  }

# Posterior mean: matrix initializion
  Z.pm = Z

# MCMC iterations
  for(ns in 1:nscan){
    tmp = Z_up(Y,Z,W,alpha,zdelta)  # Update z's

    if(any(tmp!=Z)){
      acz = acz+sum(tmp!=Z)/(2*n*odens)

# Procrustean transformation with objective Z.mle
      Z[,1] = Z[,1]-mean(Z[,1])
      Z[,2] = Z[,2]-mean(Z[,2])
      Z = proc.crr(tmp,Z.mle)
    }
    lpz = lpz_dist(Z)
    tmp = alpha_up(Y,lpz,W,alpha,adelta,a_a=2,a_b=1)  # Update alpha

    if(tmp!=alpha) {
      aca = aca+1/odens
      alpha = tmp
    }

# Thinning
    if (ns%%odens==0){
      lik = lpY(Y,lpz,alpha,W)
      Lik = c(Lik,lik)
      cat(ns,aca,acz,alpha,lik,"\n")
      acz = aca = 0

# Display burn-in message
      if (dynamic_init){
        dev.hold()
        plot.new()
        plot.window(xlim=c(-10,10),ylim=c(-3,3))
        text(0,0,paste0("Waiting for the end of the burn-in period : ",round
            (100*(ns)/burnin,2),"%"),col = "black")
        dev.flush()
      }

# Start saving thinned iterations after burn-in
      if( ns>burnin){
# Close burn-in message and create windows for new plots
        if (dynamic_init){
          dev.off()
          dynamic_init = FALSE
          x11(xpos=0,ypos=0)
        }

        Alpha = c(Alpha,alpha)
        for(i in 1:k){
          Zp[[i]] = rbind(Zp[[i]],t(Z[,i]))
        }

        for(i in 1:k){
          Z.pm[,i] = colMeans(Zp[[i]])
        }
        if (dynamic_plot){
          dev.hold()
          plot.new()
          plot.window(xlim=range(Zp[[1]]),ylim=range(Zp[[2]]))
          box()
          for(i in 1:n) {
            points( Zp[[1]][,i],Zp[[2]][,i],pch=20,cex=0.1,col=my_colors[i])
          }

          for(i in 1:n){
            for(j in 1:n){
              lines(  Z.pm[c(i,j),1],Z.pm[c(i,j),2],lty=Y[i,j],col="blue",lwd1)
            }
          }
          points(Z.pm[,1],Z.pm[,2],xaxt="n",yaxt="n",xlab="",ylab="",col="
              yellow",pch=20,cex=2)
          if (dynamic_circles){symbols(Z.pm,circles=rep(mean(Alpha),n),add=TRUE
              ,fg=my_colors,inches=F)}
          dev.flush()
        }
      }
    }
  }

# Remove starting values if burnin>0
  if (burnin>0){
    for(i in 1:k){
      Zp[[i]] = Zp[[i]][-1,]
    }
    Lik = Lik[-1]
    Alpha = Alpha[-1]
  }


# Representability of Y at the end of the MCMC simulation
  RecMat = (((mean(Alpha)+lpz_dist(Z.pm))>=0)+0)
  diag(RecMat) = 0
  repr_fin = (sum (YRecMat) - n)/ (n*(n-1))
  print (paste0("Starting representability percentage : ",repr_init))
  print (paste0("Final representability percentage : ",repr_fin))

# Z.mle needs to be rescaled in order to be "compared" with Z.pm : alpha
    changes,so does the overall scale.
  Z.mle = Z.mle*sqrt(sum(diag(Z.pm%*%t(Z.mle)))/sum(diag(t(Z.mle)%*%Z.mle)))

# Final Procrustean transfrom
  Z.mle = proc.crr(Z.mle,Z.pm)


# Clustering
  dist_Zpm = dist(Z.pm)
  dend = hclust(dist_Zpm,method="complete")
  wss = rep(0,mnc)
  for (i in 1:mnc){
    temp = cutree(dend,i)
    eval = css(dist_Zpm,temp)
    wss[i]eval$totwss
  }
}

```

```
# Non Procrustean alternative
if (!procrustean){
  MD = array(0,dim=c(n,n,(nscan-burnin)/odens))

  for(ns in 1:nscan){
    Z_tmp = Z_up(Y,Z,W,alpha,zdelta)  # Update z's
    if(any(Z_tmp!=Z)){
      acz = acz+sum(Z_tmp!=Z)/(2*n*odens)
      Z = Z_tmp
    }
    lpz = lpz_dist(Z)
    tmp = alpha_up(Y,lpz,W,alpha,adelta,a_a=2,a_b=1)  # Update alpha
    if(tmp!=alpha) {
      aca = aca+1/odens
      alpha = tmp
    }
    if (ns%%odens==0){
      lik = lpY(Y,lpz,alpha,W)
      Lik = c(Lik,lik)
      cat(ns,aca,acz,alpha,lik,"\n")
      acz = aca = 0

      if( ns>burnin){
        Alpha = c(Alpha,alpha)
        MD[,,(ns-burnin)/odens] = lpz
      }
    }
  }

# Remove starting values if burnin>0
  if (burnin>0){
    for(i in 1:k){
    Zp[[i]] = Zp[[i]][-1,]
  }
    Lik = Lik[-1]
    Alpha = Alpha[-1]
  }

# Relative distance average
  M_MD = rowMeans(MD,dims=2)

# Multidimensional scaling
  est_coord = cmdscale(M_MD,2)

# Representability of Y
  RecMat = (((mean(Alpha)+M_MD)>=0)+0)
  diag(RecMat)0
  repr_fin(sum (YRecMat) - n)/ (n*(n-1))
  print (paste0("Starting representability percentage : ",repr_init))
  print (paste0("Final representability percentage : ",repr_fin))

# Clustering
  dist_MDas.dist(-M_MD)
  dendhclust(dist_MD,method="complete")
  wss = rep(0,mnc)
  for (i in 1:mnc){
    tempcutree(dend,i)
    evalcss(dist_MD,temp)
    wss[i]eval$totwss
  }
}
```

## Auxiliary R functions

Most of the R functions were actually used for data processing from sources other than the igraphdata package or for visualization purposes, so we decided not to display them here in order to avoid details that may create unnecessary confusion. We report uniquely the implementation of the Procrustean transformation since it was fundamental for the whole H-LSM algorithm.

```
# Procrustean transformation
proc.crr=function(Z,Z0,k=2){

  for( i in 1:k) {
    Z[,i]=Z[,i]-mean(Z[,i]) + mean(Z0[,i])
  }

  A=t(Z) %*% (Z0%*%t(Z0)) %*% Z
  eA=eigen(A,symmetric=T)
  Ahalf=eA$vec[,1:k]%*%diag(sqrt(eA$val[1:k]))%*%t(eA$vec[,1:k])

  t(t(Z0) %*% Z %*% solve(Ahalf) %*% t(Z))
}
```

## Auxiliary C++ functions (Rcpp)

In this last section, we report the core functions from the MCMC algorithm rewritten in C++. We also provide some brief explanations in order to clarify their role and give the necessary contextualization.

```
#include <RcppEigen.h>
#include <numeric>

//[[Rcpp::depends(RcppEigen)]]

using Eigen::Map;                  // maps (references) instead of copying
using Eigen::MatrixXd;
using Eigen::VectorXd;
using Eigen::SelfAdjointEigenSolver;


dst function: evaluates the geodesic distance between nodes

// [[Rcpp::export]]
MatrixXd dst(const Map<MatrixXd> M, bool infd = 0){
  int g=M.rows();
  MatrixXd Yr=M;
  MatrixXd Dst=M;
  for (int i(0); i<g;i++){
    for (int j(0); j<g;j++){
      if (M(i,j)==1){
        Dst(i,j)=1;
      }
      else {
        Dst(i,j)=g;
      }
    }
  }
  for (int r(2);r<g;r++){
    Yr=Yr*M;
    for (int i(0); i<g;i++){
      for (int j(0); j<g;j++){
        if (Yr(i,j)>0 && Dst(i,j)==g){
          Dst(i,j)=r;
        }
      }
    }

  }
  return Dst;
}
```

*lpz_dist function: returns the negative square root of the latent Euclidean distances*

```
// [[Rcpp::export]]
MatrixXd lpz_dist(MatrixXd Z){
  MatrixXd ZtZ=Z*(Z.adjoint());
  int k=Z.rows();
  MatrixXd temp;
  temp.setOnes(1,k);
  MatrixXd temp2=ZtZ.diagonal()*temp;
  MatrixXd mg=temp2;
  mg=mg+temp2.adjoint()-ZtZ*2;

  for (int i(0);i<k;i++){
    for (int j(0);j<k;j++){
      mg(i,j)=-sqrt(mg(i,j));
    }
  }
  return mg;
}
```

*lpY function: evaluates the likelihood for the observed graph*

```
// [[Rcpp::export]]
double lpY (MatrixXd Y, MatrixXd lpz, double alpha, MatrixXd W){
  double val(0);
  double lpg;
  int k=lpz.rows();

  for (int i(0);i<k;i++){
    for (int j(0);j<k;j++){
      if (i!=j){
        lpg=W(i,j)*lpz(i,j)+alpha;
        val=val+Y(i,j)*lpg-log(1+exp(lpg));
      }
    }
  }
  return val;
}
```

*mlpY function: small variation of lpY function*

```
// [[Rcpp::export]]
double mlpY (VectorXd avZ, MatrixXd Y, MatrixXd W){
  int k=Y.rows();
  double val(0);
  double alpha=avZ(0);

  MatrixXd Z=avZ.tail(avZ.size()-1);
  Z.resize(k,2);
  MatrixXd lpz=lpz_dist(Z);
  double lpg;

  for (int i(0);i<k;i++){
    for (int j(0);j<k;j++){
      lpg=W(i,j)*lpz(i,j)+alpha;
      if (i!=j){
        val+=Y(i,j)*lpg-log(1+exp(lpg));
      }
    }
  }
  return -val;
}
```

*lpz_distNODE function: lpz_dist optimized for individual updates*

```
// [[Rcpp::export]]
VectorXd lpz_distNODE(MatrixXd Z, int node, VectorXd diag){
  int k=Z.rows();
  VectorXd ZtZ=Z.row(node)*(Z.adjoint());
  VectorXd temp(k);
  temp.fill(diag(node));
  VectorXd mg;
  mg=temp+diag-ZtZ*2;
  for (int i(0);i<k;i++){
    mg(i)=-sqrt(mg(i));
  }
  return mg;
}
```

*lpzYNODE function: lpY optimized for individual updates*

```
// [[Rcpp::export]]
double lpYNODE (MatrixXd Y, MatrixXd Z, double alpha, int node, VectorXd diag,
    MatrixXd W){
  int k=Z.rows();
  double val(0);
  VectorXd lpz=lpz_distNODE(Z, node, diag);
  double lpg;
  for (int i(0);i<k;i++){
    if (i!=node){
      lpg=W(i,node)*lpz(i)+alpha;
      val+=Y(i,node)*lpg-log(1+exp(lpg));
    }
  }
  return val;
}
```

*Z_up function: proposes and accepts updates for latent positions (here the individual version)*

```
// [[Rcpp::export]]
MatrixXd Z_up (MatrixXd Y, MatrixXd Z, MatrixXd W, double alpha, double zdelta,
     double mu_z=0, double sd_z = 10){
  Rcpp::RNGScope scope;
  int k=Z.rows();
  int h=Z.cols();
  MatrixXd Znew = Z;
  double lnew, lold, hr ;
  Rcpp::NumericVector UP(h), D1(h), D2(h), OLD(h);
  VectorXd ZtZ=(Z*(Z.adjoint())).diagonal();
  for (int i(0);i<k;i++){
    for (int j(0);j<h;j++){
      OLD[j] = Z(i,j);
      UP[j] = Rcpp::rnorm(1,OLD[j],zdelta)[0];
      Znew(i,j) = UP[j];
    }
    D1 = Rcpp::dnorm(UP, mu_z, sd_z, TRUE);
    D2 = Rcpp::dnorm(OLD, mu_z, sd_z, TRUE);
    lold=lpYNODE(Y,Z,alpha,i, ZtZ, W);
    ZtZ(i) = Znew.row(i)*Znew.row(i).adjoint();
    lnew=lpYNODE(Y,Znew,alpha, i, ZtZ, W);
    hr = 2*(lnew-lold) + std::accumulate(D1.begin(),D1.end(),0) - std::
        accumulate(D2.begin(),D2.end(),0);
    if (Rcpp::runif(1).at(0)>exp(hr)) {
      Znew.row(i)=Z.row(i);
      ZtZ(i) = Znew.row(i)*Znew.row(i).adjoint();
    }
    else {
      Z.row(i)=Znew.row(i);
    }
  }
  return Znew;
}


*alpha_up function: proposes and accepts updates for the alpha threshold*

// [[Rcpp::export]]
double alpha_up (MatrixXd Y, MatrixXd lpz, MatrixXd W, double alpha, double
    adelta, double a_a=1, double a_b = 1){
  Rcpp::RNGScope scope;
  double lnew, lold, hr;
  Rcpp::NumericVector alphanew(1),alphaV(1);
  alphanew[0] = std::abs(alpha+Rcpp::runif(1,-adelta,adelta)[0]);
  alphaV[0] = alpha;
  lnew=lpY(Y,lpz,alphanew[0], W);
  lold=lpY(Y,lpz,alpha, W);
  hr=lnew-lold + Rcpp::dgamma(alphanew,a_a,a_b,TRUE)[0] - Rcpp::dgamma(alphaV,
      a_a,a_b,TRUE)[0];

  if(Rcpp::runif(1).at(0)>exp(hr)){
    alphanew[0]=alpha;
  }

  return alphanew[0];
}
```

# Bibliography

[1] A. L. Barabási, R. Albert, *Emergence of scaling in random networks*, Science, Vol. 286, (1999), pp. 509-512.

[2] J. Besag, *Spatial Interaction and the Statistical Analysis of Lattice Systems*, Journal of the Royal Statistical Society. Series B (Methodological), (1974), pp. 192-236.

[3] I. Borg, P. Groenen, *Modern Multidimensional Scaling*, Springer Series in Statistics, New York, NY, USA, (1997).

[4] A. Buja et al., *Data Visualization with Multidimensional Scaling*, Journal of Computational and Graphical Statistics, Vol. 17, No. 2, (2008), pp. 444-472.

[5] G. Caldarelli, *Scale-Free Networks: complex webs in nature and technology*, Oxford University Press, (2007).

[6] A. Clauset, C. Moore, *How Do Networks Become Navigable?*, arXiv:cond-mat/0309415, (2003).

[7] J. A. Davis, *Clustering and hierarchy in interpersonal relations: testing two graph theoretical models on 742 sociomatrices*, American Sociological Review, Vol. 35, No.5, (1970), pp. 843-851.

[8] R. Diestel, *Graph Theory*, Springer-Verlag Heidelberg, New York, NY, USA, (2006), pp. 23-28.

[9] R. Durrett, *Random Graph Dynamics*, Cambridge University Press, (2007).

[10] D. Easley, J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, Cambridge University Press, (2010), pp. 543-560.

[11] P. Erdös, A. Rényi, *On Random Graphs*, Publicationes Mathematicae Debrecen, Vol. 6, (1959), pp. 290-297.

[12] S. Fortunato, *Detecting the Overlapping and Hierarchical Community Structure in Complex Networks*, Physics Reports, Vol. 486, No. 5, (2010), pp. 75-174.

[13] O. Frank, D. Strauss, *Markov Graphs*, Journal of the American Statistical Association, Vol. 81, No. 385, (1986), pp. 831-842.

[14] A. Gelman, *Analysis of variance—why it is more important than ever*, The Annals of Statistics s, Vol. 33, No. 1, (2005), pp. 1-31.

[15] E. N. Gilbert, *Random Graphs*, The Annals of Mathematical Statistics, Vol. 30, No. 4, (1959), pp. 1141-1144.

[16] A. Goldenberg, A. X. Zheng, S. E. Fienberg, E. M. Amaldi, *A survey of statistical network models*, Foundations and Trends in Machine Learning, Vol. 2, No. 2, pp. 129-233.

[17] J. L. Gross, T. W. Tucker, *Topological Graph Theory*, Wiley-Interscience, New York, NY, USA, (1987), pp. 1-23.

[18] M.S. Handcock, A. E. Raftery, J. Tantrum, *Model-based clustering for social networks*, Journal of the Royal Statistical Society, Vol. 170, No. 2, pp. 301-354.

[19] P. D. Hoff, *A First Course in Bayesian Statistical Methods*, Springer Science, New York, NY, USA, (2009).

[20] P. D. Hoff, A. E. Raftery, M. S. Handcock, *Latent Space Approaches to Social Network Analysis*, Journal of the American Statistical Association, Vol. 97, No. 460, (2002), pp. 1090-1098.

[21] P. W. Holland, S. Leinhardt, *A Dynamic Model for Social Networks*, The Journal of Mathematical Sociology, Vol. 5, No. 1, (1977), pp. 5-20.

[22] P. W. Holland, S. Leinhardt, *An exponential family of probability distributions for directed graphs*, Journal of the American Statistical Association, Vol. 76, No. 373, (1981), pp. 33-50.

[23] P. W. Holland, K. B. Laskey and S. Leinhardt, *Stochastic blockmodels: first steps*, Social Networks, Vol. 5, (1983), pp. 109-137.

[24] S. Jackman, *Bayesian Analysis for the Social Sciences*, John Wiley & Sons Ltd., Chichester, UK, (2009).

[25] T. Kamada, S. Kawai, *An algorithm for drawing general undirected graphs*, Information processing letters, Vol. 31, No. 1, (1989), pp. 7-15.

[26] F. Karinthy, *Chains*, Everything is different, (1929).

[27] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada and N. Ueda, *Learning Systems of Concepts with an Infinite Relational Model*, Artificial Intelligence, Proceedings of the National AAAI Conference, (2006).

[28] M. Kim, J. Leskovec, *Multiplicative attribute graph model of real-world networks*, Internet Mathematics, Vol. 8, No. 2, (2012), pp. 113-160.

[29] F. Lorrain, H. C. White, *Structural equivalence of individuals in social networks*, The Journal of Mathematical Sociology, Vol. 1, No. 1, (1971), pp. 49-80.

[30] J. McAuley, Q. Shi, C. Targett, A. Van Den Hengel, *Image-based recommendations on styles and substitutes*, Proceedings of the $38^{th}$ International ACM SIGIR Conference on Research and Development in Information Retrieval, (2015), pp. 43-52.

[31] J. McAuley, R. Pandey, J. Leskovec, *Inferring Networks of Substitutable and Complementary Products*, Proceedings of the $21^{th}$ ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2015), pp. 785-794.

[32] J. A. Nelder, R. Mead, *A simplex method for function minimization*, The Computer Journal, Vol. 7, No. 4, (1965), pp. 308-313.

[33] M. E. J. Newman, *The Structure and Function of Complex Networks*, SIAM Review, Vol. 45, No. 2, (2003), pp. 167-256.

[34] M. E. J. Newman, *Networks: an introduction*, Oxford University Press, (2010).

[35] M. Oh, A. E. Raftery, *Bayesian Multidimensional Scaling and Choice of Dimension*, Journal of the American Statistical Association, Vol. 96, No. 455, (2001), pp. 1031-1044.

[36] J. F. Padgett, C. K. Ansell, *Robust Action and the Rise of the Medici*, American Journal of Sociology, Vol. 98, No. 6, (1993), pp. 1259-1319.

[37] J. C. Pinheiro, D. M. Bates, *Mixed-effects models in S and S-plus*, Springer-Verlag New York, New York, NY, USA, (2000).

[38] P. Pons, M. Latapy, *Computing Communities in Large Networks Using Random Walks*, Journal of Graph Algorithms and Applications, Vol. 10, No. 2, (2006), pp. 191-218.

[39] D. Price, *A General Theory of Bibliometric and Other Cumulative Advantage Processes*, Journal of the American Society for Information Science, Vol. 27, No. 5, (1976), pp. 292-306.

[40] G. Robins, P. Pattison, Y. Kalish, D. Lusher, *An introduction to exponential random graph ($p^*$) models for social networks*, Social Networks, Vol. 29, No. 2, (2007), pp. 173-191.

[41] M. Rosvall, C. Bergstrom, *Maps of random walks on complex networks reveal community structure*, Proceedings of the National Academy of Sciences, Vol. 105, No. 4, (2008), pp. 1118-1123.

[42] P. Sarkar, A. W. Moore, *Dynamic Social Network Analysis using Latent Space Models*, ACM SIGKDD Explorations Newsletter, Vol. 7, No. 2, (2005), pp. 31-40.

[43] O. Sandberg, I. Clarke, *The evolution of navigable small-world networks*, Technical Report, Chalmers University of Technology, (2007).

[44] T. Snijders, *Statistical methods for network dynamics*, Proceedings of the XLIII Scientific Meeting, Italian Statistical Society, (2006), pp 281-296.

[45] T. Snijders, *Statistical Models for Social Networks*, Annual Review of Sociology, Vol. 37, (2011), pp. 131-153.

[46] T. Snijders, K. Nowicki, *Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure*, Journal of Classification, Vol. 14, (1997), pp. 75-100.

[47] T. Snijders, K. Nowicki, *Estimation and Prediction for Stochastic Blockstructures*, Journal of the American Statistical Association, Vol. 96, No. 455, (2001), pp. 1077-1087.

[48] T. Snijders, P. E. Pattison, G. L. Robins, M. S. Handcock, *New specifications for exponential random graph models*, Sociological Methodology, Vol. 36, No. 1, (2006), pp. 99-153.

[49] W. S. Torgerson, *Multidimensional Scaling: Theory and Method*, Psychometrika, Vol. 17, No. 4, (1952), pp. 401-419.

[50] M. van Steen, *Graph Theory and Complex Networks: An Introduction*, Maarten van Steen, (2010).

[51] S. Wasserman, P. Pattison, *Logit models and logistic regressions for social networks: an introduction to Markov graphs and $p^*$*, Psychometrika, Vol. 61, No. 3, (1996), pp. 401-425.

[52] W. W. Zachary, *An Information Flow Model for Conflict and Fission in Small Groups*, Journal of Anthropological Research, Vol. 33, No. 4 , (1977), pp. 452-473.